الجمهورية الجزائرية الديمقراطية الشعبية
وزارة التعليم العالي و البحث العلمي

Université Abou Bekr Belkaid
Tlemcen Algérie

جامعة أبي بكر بلقايد
تلمسان الجزائر

REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

MINISTRE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE ABOU BEKR BELKAID-TLEMCEN-

FACULTE DE TECHNOLOGIE

DEPARTEMENT DE TÉLÉCOMMUNICATIONS

Projet de fin d'études pour l'obtention du diplôme de master

En

Télécommunications

Option

Systèmes de télécommunications

Présenté par

CHEIKH RIHAB ZOHRA

Et

BOUFATAH MOHAMMED HICHAM

THÈME

SURVEILLANCE INTELLIGENTE DES ECOSYSTEMES VERTS

Soutenu en

Devant le jury :

Président :	Mr BORSALI Ahmed Riadh	Professeur, Université de TLEMCEN
Encadreur :	Mr BENADDA Belkacem	Professeur, Université de TLEMCEN
Examinateur :	Mr MERZOUGUI Rachid	Professeur, Université de TLEMCEN

Année Universitaire 2020-2021

DÉDICACE

A mes très chers parents

Quoi que je fasse ou que je dise, je ne saurai point vous remercier comme il se doit, pour tous vos efforts et votre soutien, votre affection et votre bienveillance. Aucun mot ne pourra exprimer mon amour pour vous, j'espère vous satisfaire avec une réussite permanente InchaaALLAH.

J'adresse mes remerciements à mes tantes Hakima, Sihem, Wassila, mon oncle Houssem ainsi qu'à leurs conjoints Mohammed, Abdessamade, Amine et Meriem.

Mes cousins Feriel, Chihab, Sirine, Maria, Racim, Charaf, Abderrezak et Achraf puisse Allah vous donner santé, bonheur et réussite.

Et à mes copines Nassima, Zoubida, Wafaa, Farah, Nesrine et Samra qui n'ont jamais cessé de me soutenir, je leur souhaite toute la réussite dans leurs vies.

Sans oublier toute la famille CHEIKH en particulier mon oncle et mes tantes paternels.

Finalement à mon binôme HICHAM, merci pour ta collaboration. Nous avons formé une bonne équipe.

« *Au final c'est toujours les mêmes, toujours les vrais qui me soutiennent ceux-là mêmes, que je pleure de rire ou de peine alors laissez-moi rendre hommage à ceux et celles qui m'encouragent les seuls qui peuvent prétendre faire partie de mon entourage*. »

CHEIKH RIHAB ZOHRA

Un mot à mes parents

chers parents, ce que je fais et ce que je ferai ne sera jamais comparé à ce que vous avez fait en premier lieu, j'espère que je vais vous rendre fiers et j'espère vous redonner un jour le quart de ce que vous m'avez offert, sans oublier ma petite sœur ma joie de vivre.

Mots à mes amis

Ryad, Lotfi, Azzeddine vous êtes les amis que chaque personne rêve d'avoir dans sa vie, je vous souhaite tout le succès et je vous remercie d'être de tels amis fidèles.

A Monsieur l'encadreur B.BENADDA

Vous avez eu un tel impact sur moi en tant qu'étudiant, votre ambition et vos accomplissements semblent toujours me surprendre, je suis ravi de vous avoir comme professeur et encadreur.

Finalement à ma binôme RIHAB

Vous avez été une vraie bosseuse dès le premier jour, je suis ravi de vous avoir comme une amie et collègue.

BOUFATAH MOHAMMED HICHAM

REMERCIEMENTS

En tout premier lieu, nous remercions ALLAH, le tout puissant, de nous avoir donné la force et la volonté pour dépasser toutes les difficultés et terminer nos études et permis de mener à terme ce modeste travail.

Nous tenons à exprimer toute notre reconnaissance et notre gratitude à notre encadreur Monsieur BENADDA Belkacem, professeur au département de Télécommunications, que nous remercions vivement de nous avoir encadré, orienté, aidé et conseillé durant ce projet de fin d'étude.

Nos vifs remerciements vont aussi à Monsieur BORSALI Ahmed Riadh, professeur au département de Télécommunications pour avoir accepté de nous faire l'honneur de présider le jury de ce mémoire et de bien vouloir l'examiner.

Nous remercions profondément Monsieur MERZOUGUI Rachid, professeur au département de Télécommunications, qui a pris la peine d'examiner et juger notre modeste travail.

Enfin nous tenons à remercier tous les enseignants du département de Télécommunication qui ont contribué à notre formation.

SOMMAIRE

CHAPITRE 3 : Machine Learning et réseaux de neurones

CHAPITRE 4 : Analyses d'images satellitaires par TensorFlow

LISTES DES FIGURES

CHAPITRE 1

CHAPITRE 2

CHAPITRE 3

CHAPITRE 4

NOTATIONS

SAR : synthétique Aperture radar

AI : artificial intelligence

IA : intelligence artificielle

RSO : radars a synthèse d'ouverture

RCS : section transversale radar

σ : Grandeur physique = rcs

$I_{reçu}$: L'intensité du signal reçu

$I_{incidente}$: L'intensité du signal incident

R : rayon

SM : strip map

UMM : unité minimale de cartographie

BBC : british broadcasting chanel

MIT : Massachusetts Institute of Technology

IAG : intelligence artificielle générale

SVM : support vector machine

NDAS : solution national d'analyse de données

Compas : correctional offender management profiling for alternative sanctions

GPS : Global Positioning System

AA : apprentissage automatique

ADN : acide désoxyribonucléique

CNN : réseau de neurones à convolution

OBD : optimale brain damage

OBS : optimale brain surgeon

Cpu : central processing unit

Gpu : Graphics processing unit

CUDA : Compute Unified Device Architecture

Gpgpu : General-Purpose Graphics Processing Unit

TPU : Tensor Processing Units

RÉSUMÉ

Les écosystèmes verts font une très grande partie de la surface des terres en Algérie, pour cela il est nécessaire de les surveiller des dangers naturels que ce soit la désertification, des pâturages non contrôlés ou d'autres dangers. Le but de notre travail est de trouver un moyen pour surveiller ces écosystèmes en construisant un modèle basé sur l'intelligence artificielle permettant la surveillance et la classification des images des différentes zones au fil du temps.

Mots clés : Radar, SAR, réseaux neuronaux, intelligence artificielle, imagerie par satellite, TensorFlow, Python.

ABSTRACT

Green ecosystems make up a very large part of Algerian lands thus it is necessary to monitor them from natural hazards such as desertification, overgrazing and other dangers. The proposed monitoring solution is based on Tensorflow AI model to classify and monitor satellite images all over the time.

keywords : Radar, SAR, neural networks, artificial intelligence, satellite imaging, TensorFlow, Python.

ملخص

الجزائر هي أكبر بلد افريقي وبيئتها غنية نباتيا وحيوانيا بتنوعها. في ظل التغيرات المناخية و التحديات البيئية الحالية من الضروري حماية البيئة المحلية من الأخطار سواء كانت طبيعية أو بشرية التصحر أو الرعي الغير الخاضع للمراقبة و غيرها من الأخطار. المراقبة المقترحة طي هذا المشروع تتم عن طريق بناء نموذج الذكاء الاصطناعي لتصنيف صور من الاقمار الصناعية ورصد الاختلاف المسجل عبر الزمن مناطق بيئية مختلفة.

كلمات مفتاحية: الأقمار الصناعية، الرادار ذي الفتحة الاصطناعية، الشبكات العصبية الذكاء الاصطناعي، التصوير بالأقمار الصناعية، تينسورفلو بايثون.

INTRODUCTION GENERALE

En général, chaque ensemble formé par une communauté d'êtres vivants en interaction avec leur environnement est considéré comme étant un écosystème.

En Algérie, on s'intéresse plutôt aux écosystèmes verts. La surveillance de ces écosystèmes verts est d'une grande importance puisqu'elle envisage de veiller sur les terres agricoles et surtout celles qui sont difficiles à atteindre.

Ces derniers occupent une grande surface des terres algériennes donc on se doit de les préserver et les surveiller, il est difficile de garder une surveillance présentielle à long terme pour ce il est recommandé d'utiliser des technologies avancées qui n'exigent pas la présence sur terrains, mais plutôt à distance, en envisageant qu'elles soient plus accessibles. Parmi lesquelles :

- La surveillance aérienne en utilisant un drone équipé par un ou plusieurs capteurs d'images qui les envoie simultanément à une base de contrôle et surveillance, ou même utiliser un hélicoptère qui fait le même travail du drone.
- Imagerie satellitaire soit en utilisant des satellites équipés par capteurs photo (le même principe du drone), ou bien des satellites radar, qui utilisent les ondes électromagnétiques émises et réfléchies sur une surface précise, qui permettent de reconstruire une image. Cette technologie est très utilisée dernièrement vu que ces satellites procurent une vue vaste des terres ciblées, beaucoup plus que les images prises par des drones ou des hélicoptères.

Le but de notre travail à la base, est la surveillance des écosystèmes verts en utilisant une base de données qui offre un accès gratuit à des images satellitaires pour créer un classificateur d'images en utilisant l'intelligence artificielle, plus précisément les réseaux de neurones.

Ce mémoire s'articule sur quatre chapitres :

Dans le premier chapitre, nous allons présenter l'imagerie par satellite, son principe de fonctionnement et la notion de la télédétection qui est le principe fondamental de l'imagerie satellitaire ; en même temps on définit le radar à ouverture synthétique (SAR) ainsi que ses domaines d'utilisation, ses modes de fonctionnement et ses limites.

Le second chapitre présente en général l'intelligence artificielle (AI) qui a facilité notre tâche pour la création d'un classificateur d'images. On abordera par la suite l'intelligence artificielle forte et l'intelligence artificielle faible tout en précisant leurs limites, risques et avantages et leurs domaines d'applications.

Dans le troisième chapitre on a introduit le Machine Learning (apprentissage automatique), ses principes, ses modes d'apprentissage et ses applications. Ce dernier utilise un principe fondamental pour son fonctionnement intitulé réseaux de neurones, dont on a tenu à préciser sa définition et sa structure tout en introduisant notre outil d'implémentation TENSORFLOW.

Le dernier chapitre est consacré à JUPYTER et COLAB qui furent très importants pour la réalisation de notre classificateur d'images dépendant de l'intelligence artificielle implémentée dans Colab. Ce chapitre est axé précisément sur notre programme d'apprentissage, tout en expliquant chaque étape et ses détails afin de garantir un résultat optimal.

CHAPITRE 1

Les Imageries par Satellite et RADAR à Synthèse d'ouverture

S.A.R

1. Introduction

L'imagerie par satellite trouve actuellement d'innombrables applications pour la télédétection et la supervision des écosystèmes verts. Ce chapitre présente un état de l'art sur les principes fondamentaux de l'imagerie par satellite, une attention particulière est accordée à l'imagerie à base des RADAR à synthèse d'ouverture (SAR Synthetic aperture RADAR).

2. Imagerie par satellite et télédétection

2.1 Imagerie par satellite

Les satellites artificiels ont beaucoup d'utilités qui sont très importantes au sein du développement scientifique. Parmi ces utilités, les captures de l'image soit de la planète terre ou d'autres planètes.

L'imagerie par satellite consiste à prendre des images de la terre ou d'autres planètes grâce à des satellites artificiels, tenant compte de méthodes utilisées pour construire une image satellitaire. Il s'agit bien de l'imagerie et non pas la photographie car parfois on utilise des méthodes de reconstruction de l'image avec des fausses couleurs juste pour une représentions bien précise. [1] Il existe des satellites qui embarquent des capteurs radar. Parmi ces capteurs, les plus courants sont appelés Radars à Synthèse d'Ouverture (RSO ou Synthétique Aperture Radar, SAR en anglais). Différents capteurs optiques, qui sont appelés « actifs » car ils n'utilisent pas la lumière du soleil pour observer la Terre, mais émettent des ondes électromagnétiques à la surface du globe pour l'illuminer. C'est la puissance avec laquelle un objet réfléchit le signal qui est alors mesurée (aussi appelée rétrodiffusion). [2]

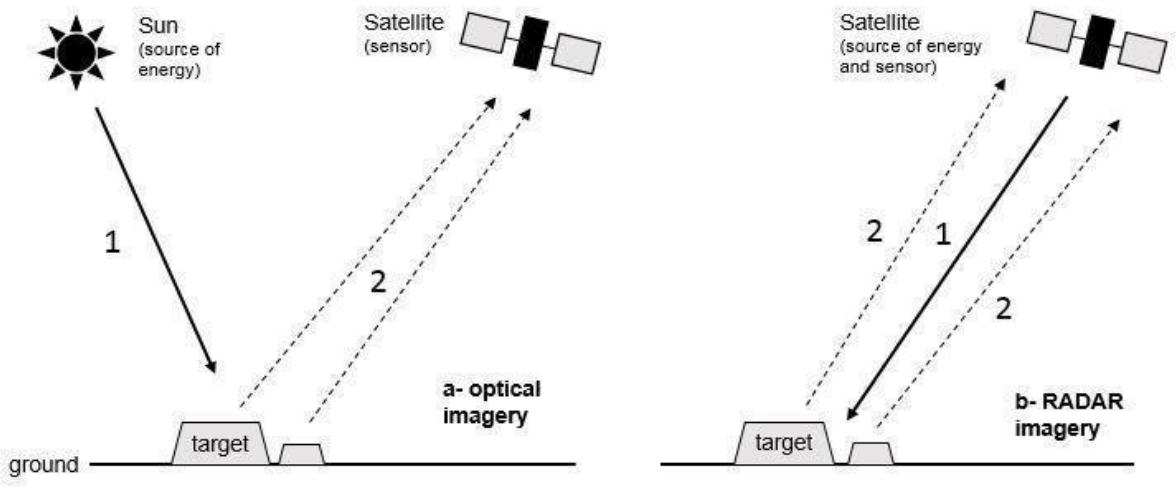

Figure 1.1 : La différence entre les capteurs optiques et radar.

2.2 Télédétection

Le terme télédétection (« remote sensing ») désigne, dans son acception générale, un ensemble de techniques permettant de mesurer à distance (c'est-à-dire sans contact) des grandeurs physiques caractéristiques des objets ou des phénomènes étudiés. Toutefois, en pratique, il est surtout utilisé pour les applications spatiales et aéroportées, et concerne essentiellement les techniques mises en œuvre pour l'observation de la surface de la Terre et d'autres planètes. Ces systèmes nécessitent l'utilisation de capteurs qui sont embarqués sur des ballons, des avions, des navettes ou des satellites. Ils exploitent essentiellement la mesure du rayonnement électromagnétique émis ou réfléchi par les objets. Ce rayonnement est analysé soit dans les domaines visible – du bleu au rouge – et infrarouge (les capteurs sont alors dits optiques), soit dans le domaine des micro-ondes millimétriques ou centimétriques (les capteurs sont alors appelés radars), en utilisant les principes de la télédétection on peut avoir une image capturée de notre planète de plus de 800 km. [3]

Dans la plupart des cas, la télédétection implique une interaction entre l'énergie incidente et les cibles. Le processus de la télédétection au moyen de systèmes imageurs comporte les sept étapes, présentées dans la figure 1.2 :

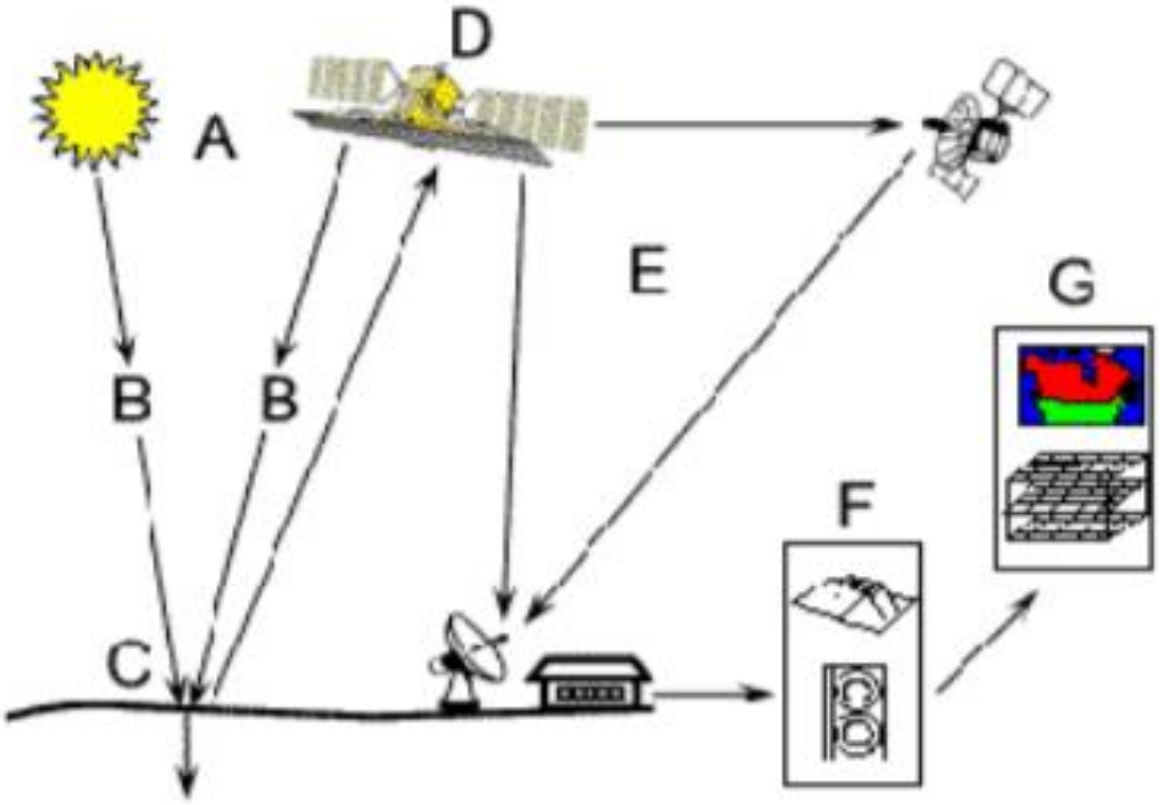

Figure 1.2 : Les sept étapes couvrant le processus de la télédétection.

1. Source d'énergie ou d'illumination (A) :

À l'origine de tout processus de télédétection se trouve nécessairement une source d'énergie pour illuminer la cible.

2. Rayonnement et atmosphère (B) :

Durant son parcours entre la source d'énergie et la cible, le rayonnement interagit avec l'atmosphère. Une seconde interaction se produit lors du trajet entre la cible et le capteur.

3. Interaction avec la cible (C) :

Une fois parvenue à la cible, l'énergie interagit avec la surface de celle-ci. La nature de cette interaction dépend des caractéristiques du rayonnement et des propriétés de la surface.

4. Enregistrement de l'énergie par le capteur (D) :

Une fois l'énergie diffusée ou émise par la cible, elle doit être captée à distance (par un capteur qui n'est pas en contact avec la cible) pour être enfin enregistrée.

5. Transmission, réception et traitement (E) :

L'énergie enregistrée par le capteur est transmise, souvent par des moyens électroniques, à une station de réception où l'information est transformée en images (numériques ou photographiques).

6. Interprétation et analyse (F) :

Une interprétation visuelle et/ou numérique de l'image traitée est ensuite nécessaire pour extraire l'information que l'on désire obtenir sur la cible.

7. Application (G) :

La dernière étape du processus consiste à utiliser l'information extraite de l'image pour mieux comprendre la cible, pour nous en faire découvrir de nouveaux aspects ou pour aider à résoudre un problème particulier.

3. Radar à ouverture synthétique

3.1 Définition

Le radar à ouverture synthétique est prononcé en Anglais Synthetic Aperture Radar d'où provient son acronyme connu « S.A.R », c'est un radar imageur qui fait le traitement des données qu'il reçoit pour pouvoir améliorer la résolution en azimut qui est l'angle dans le plan horizontal entre la direction d'un objet et une direction de référence. Il survole les terrains à bord d'un avion ou d'un satellite et crée des images détaillées, qui donnent des informations sur plusieurs paramètres. [4]

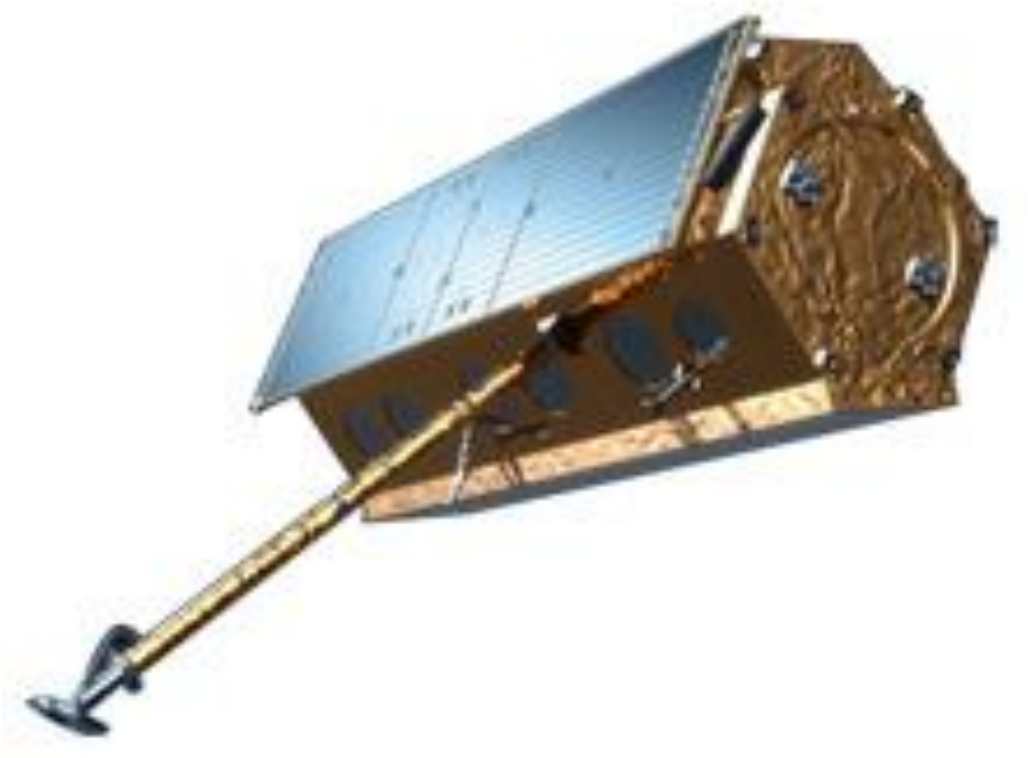

Figure 1.3 : Radar à synthèse d'ouverture TerraSAR-X.

3.2 Le fonctionnement du SAR

Les SAR transmettent des signaux micro-ondes à un angle oblique comme montré dans la figure 1.4, et mesurent la partie rétrodiffusée (dans le sens du capteur) de ce signal afin d'analyser les caractéristiques à la surface. Mathématiquement, cette mesure (étalonnée) est décrite à l'aide du terme section transversale radar (RCS) σ, qui est définie comme étant le rapport entre l'intensité du signal incident et l'intensité du signal reçu :

$$\sigma = \frac{I_{reçu}}{I_{incidente}} * 4\pi * R^2 \left[m^2\right] \quad (1.1)$$

σ : section transversale radar

$I_{reçu}$: L'intensité du signal reçu

$I_{incidente}$: L'intensité du signal incident

R : rayon

Le RCS enregistré par un SAR pour une caractéristique de surface spécifique n'est pas toujours facile à interpréter, car il est affecté à la fois par plusieurs caractéristiques de scène ainsi que par les paramètres du capteur d'imagerie.

Ces capteurs fournissent leur propre source d'énergie et peuvent ainsi fonctionner jour et nuit. [5]

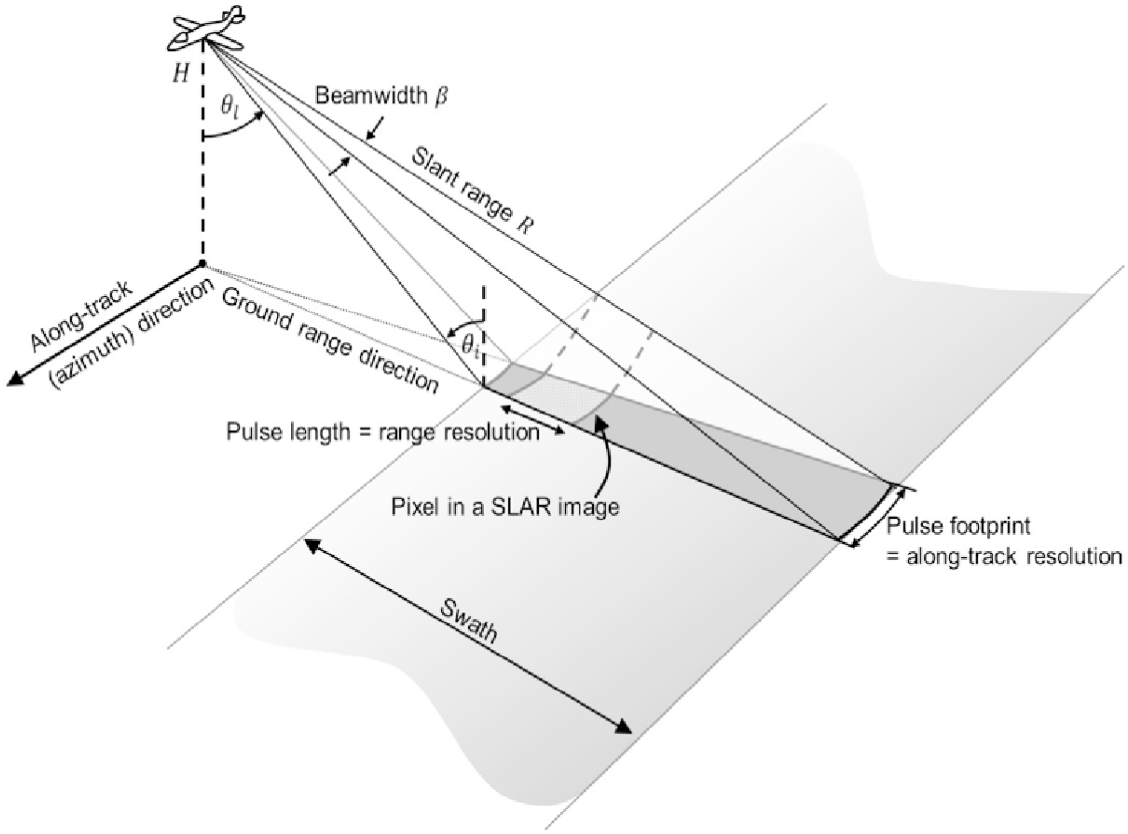

Figure 1.4 : Fonctionnement d'un SAR placé à bord d'un avion.

3.3 Propriétés d'acquisition des données SAR

Quel que soit le capteur satellitaire, les images sont acquises selon deux modes de fonctionnement :

3.3.1 Mode Stripmap

Le mode Stripmap (SM) acquiert des données avec un andain de 80 km avec une résolution spatiale légèrement supérieure à 5 m par 5 m (vue unique). La figure 1.5 montre l'andain au sol qui est éclairé par une séquence continue d'impulsions tandis que le faisceau de l'antenne pointe vers un angle azimut fixe et un angle hors nadir approximativement fixe (ce qui est sujet à de petites variations en raison de la direction par roulis). [6]

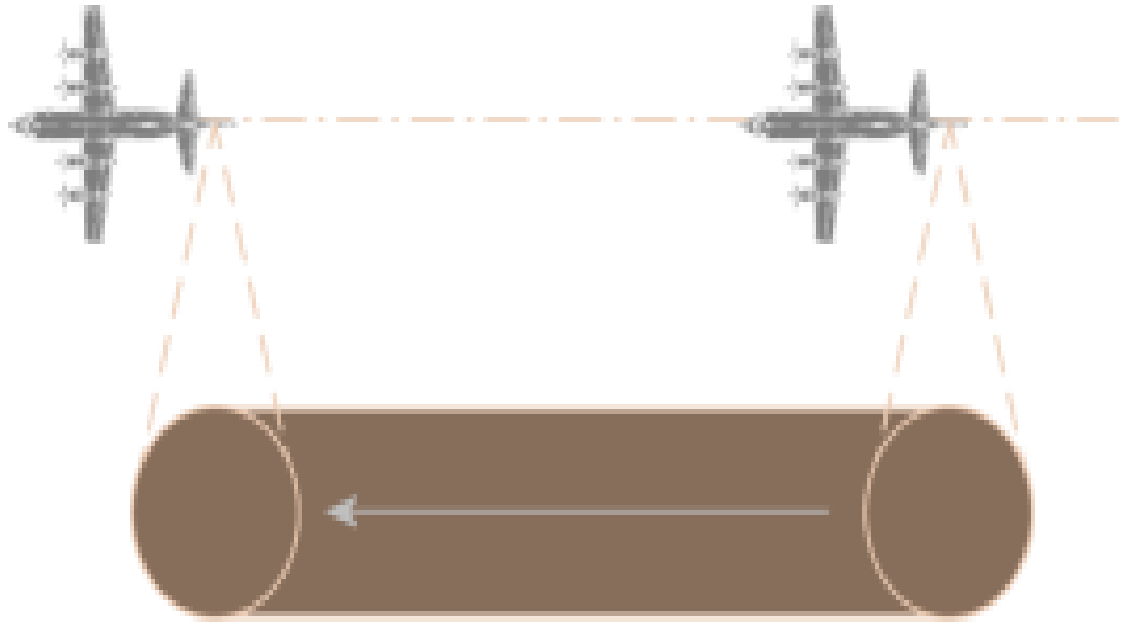

Figure 1.5 : Le mode de fonctionnement stripmap.

Les images SM ont une qualité d'image continue le long de la piste à un angle d'incidence approximativement constant. Il est présent sur les satellites opérants dans les micro-ondes depuis le satellite ERS-1. Il sert à l'acquisition des images TerraSAR-X, Radarsat-2 et Alos.

3.3.2 Mode Spotlight

Spotlight-SAR est un mode de fonctionnement SAR permettant d'obtenir une haute résolution en pilotant le faisceau radar pour garder la cible à l'intérieur du faisceau plus longtemps et ainsi former une ouverture synthétique plus longue comme montré dans la figure 1.6.

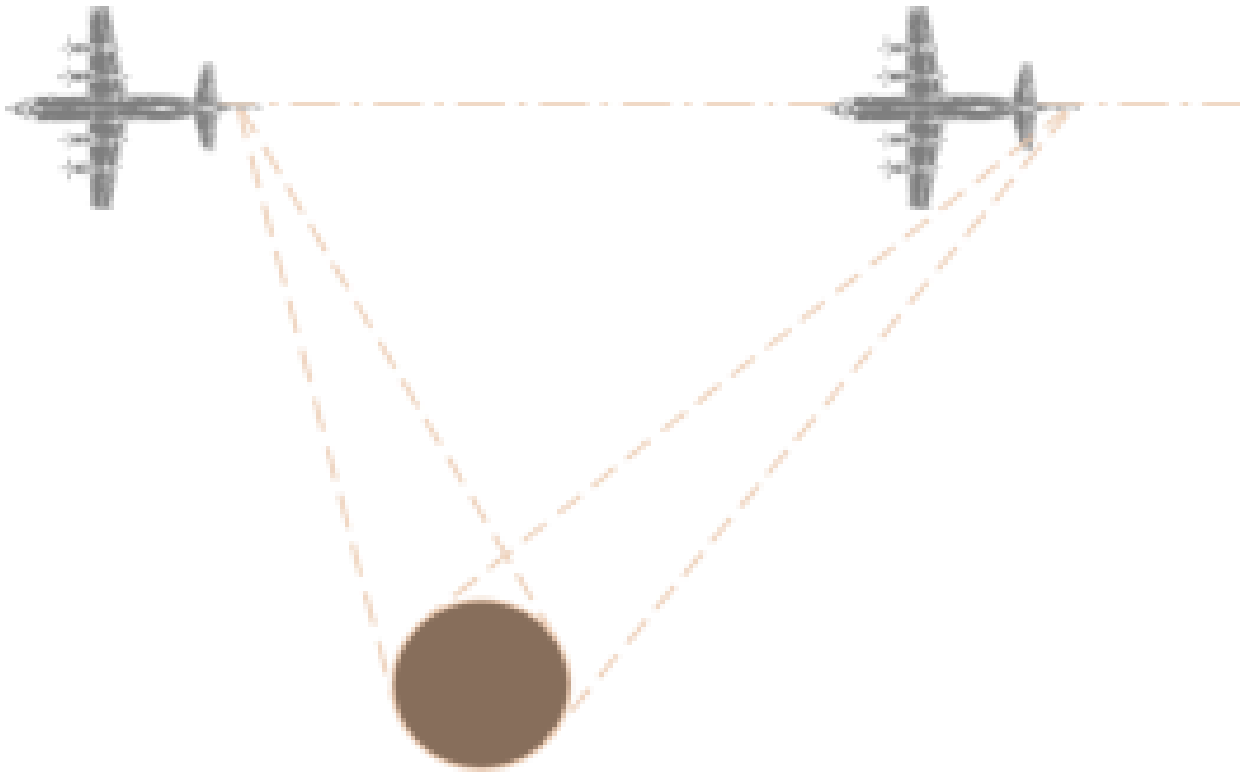

Figure 1.6 : Le mode de fonctionnement spotlite.

Spotlight SAR est capable d'étendre considérablement la capacité d'imagerie SAR à haute résolution. À mesure que plus d'impulsions sont utilisées, la résolution azimut augmente. Pour ce faire, il faut maintenir une cible plus longtemps dans l'éclairage du projecteur du faisceau radar grâce à la direction électronique du faisceau, ce qui donne une ouverture synthétique plus longue. Le mode de fonctionnement de Spotlight SAR se fait habituellement au détriment de la couverture spatiale, car les autres zones d'une zone d'accessibilité donnée du SAR ne peuvent pas être éclairées pendant que le faisceau radar éclaire une zone cible particulière. Il est utilisé pour l'acquisition d'une série d'images TerraSAR-X, à haute résolution spatiale. [7]

3.4 Domaine d'utilisation

Le SAR est utilisé dans plusieurs domaines, voici les plus importants :

3.4.1 Agriculture

Il mesure Les différences dans la rugosité de la surface du champ :

En milieu agricole il est fort utile de superviser le résultant du travail du sol, ceci correspond au relief formé par les éléments périodiques (sillons, buttes…) et apériodiques (mottes de terres, graviers…). En télédétection SAR, l'influence du sol sur les signaux est variable, dépendante de l'état de rugosité et d'autres paramètres : longueur d'onde, d'état de polarisation, angle d'incidence et direction de visée. (Callens et al., 2006; Davidson et al., 1998; Davidson et al., 2000; Jackson et al., 1997; Mattia et al., 1997; Ulaby et al., 1978; Zribi et al., 2005c). [8]

3.4.2 Inondations

Les différences de réflexion de surface peuvent aider à distinguer les inondations importantes, les inondations légères, les zones urbaines et les plans d'eau permanents.

3.4.3 Affaissement des terres

Les différences dans les mesures au fil du temps peuvent révéler des déplacements de terres, comme l'enfoncement du sol causé par l'extraction de ressources naturelles souterraines.

3.4.4 Couverture de neige

Les différences dans la réflexion de surface peuvent aider à prévoir la fonte des neiges en distinguant la neige mouillée, la neige sèche et les zones exemptes de neige.

3.4.5 Feux de forêt

La pénétration de la fumée épaisse peut fournir des renseignements plus précis sur l'étendue d'un incendie de forêt et peut aider à quantifier la perte de végétation.

3.4.6 Milieux humides

La pénétration dans les zones humides peut révéler une végétation inondée où les terres sont recouvertes d'eau peu profonde. [8]

4. Les raisons du choix du radar à ouverture synthétique

Pour la collecte des images de terrains nous avons opté pour les images procurées par les S.A.Rs et cela en raison de plusieurs facteurs :

- ➢ Les nuages n'affectent pas sur ses ondes propagées, car les micro-ondes à basse fréquence utilisées par les SARs ne sont pas affectées par les conditions atmosphériques, ce qui permet l'utilisation dans des régions nuageuses de la nature intrinsèque hétérogène et dynamique de l'agro écosystème étudié.
- ➢ L'étendue géographique à étudier.
- ➢ L'Unité minimale de cartographie (UMM) nécessaire pour résoudre les champs individuels et d'autres unités écologiques significatives (terres humides, boisés, etc.)
- ➢ Les contraintes techniques des systèmes de télédétection (taille de l'herbe ; résolutions spatiales, temporelles, spectrales et radiométriques ; couverture nuageuse des systèmes optiques).
- ➢ Disponibilité des données (ouvertes ou payantes). [5]

5. Limitation d'utilisation du S.A.R

L'estimation de biomasse en utilisant les S.A.Rs est bien limitée par certains paramètres dont on site :

- Selon la longueur d'onde, saturation rapide du signal à un stock de biomasse hors sol bas (~50 à 100 t C/ha).
- Augmentation des erreurs dues au terrain.
- Effets de pluie et de l'humidité du sol.
- Manque de cohérence des estimations en fonction des paramètres des capteurs.

6. conclusion

Nous avons présenté dans ce chapitre une vue globale de l'utilité de la télédétection par satellite, plus précisément la technologie SAR, dans la perspective d'exploiter cette technologie dans la lutte contre la désertification en Algérie. Le chapitre suivant va présenter l'intelligence artificielle et ses domaines d'utilisations.

CHAPITRE 2

Un aperçu sur L’intelligence artificielle

1. Introduction

L'utilisation des images satellitaires seules nécessite un travail long s'il est utilisé par des opérateurs humains. De même l'analyse de ces images a atteint un degré de maturité et un savoir-faire qui peut être exploité pour l'apprentissage et la création d'un système intelligent. Utiliser l'intelligence artificielle, pour le traitement et l'apprentissage de l'information venant du satellite en usant du savoir-faire acquis. Ce chapitre va introduire la notion de l'intelligence artificielle et des modèles associés.

2. Intelligence artificielle (AI)

Ce nouvel axe de la technologie a pour but de simuler l'intelligence humaine à un certain degré en utilisant différentes techniques et théories afin de réaliser des machines de simulation d'intelligence d'un cerveau humain.

Comme étant une branche de la science cognitive, l'intelligence artificielle est basée sur la neurobiologie computationnelle (particulièrement aux réseaux neuronaux), à la logique mathématique (partie des mathématiques et de la philosophie) et à l'informatique. Elle recherche des méthodes de résolution de problèmes à forte complexité logique ou algorithmique. Par extension, elle désigne, dans le langage courant, les dispositifs imitant ou remplaçants l'homme dans certaines mises en œuvre de ses fonctions cognitives.

3. Historique

Historiquement, l'idée d'intelligence artificielle semble émerger dans les années 1950 quand Alan Turing se demande si une machine peut « penser ». Dans l'article « Computing Machinery and Intelligence » (Mind, octobre 1950), Turing explore ce problème et propose une expérience (maintenant dite test de Turing) visant à trouver à partir de quand une machine deviendrait « consciente ». Il développe ensuite cette idée dans plusieurs forums, dans la conférence « L'intelligence de la machine, une idée hérétique », dans la conférence qu'il donne à la BBC 3e programme le 15 mai 1951 « Les calculateurs numériques peuvent-ils penser ? » ou la discussion avec M.H.A. Newman, Sir Geoffrey Jefferson et R.B. Braithwaite les 14 et 23 janvier 1952 sur le thème « Les ordinateurs peuvent-ils penser ? ».

Une autre origine probable est la publication, en 1949, par Warren Weaver d'un mémorandum sur la traduction automatique des langues qui suggère qu'une machine puisse faire une tâche qui relève typiquement de l'intelligence humaine.

Le développement des techniques informatiques (augmentation de la puissance de calcul) aboutit ensuite à plusieurs avancées :

- Dans les années 1980, l'apprentissage automatique se développe, notamment avec la renaissance du connexionnisme. L'ordinateur commence à déduire des « règles à suivre » en analysant seulement des données

- Parallèlement, des algorithmes « apprenants » sont créés qui préfigurent les futurs réseaux de neurones (l'apprentissage par renforcement, les machines à vecteurs de support, etc.). Ceci permet par exemple en mai 1997 à l'ordinateur Deep Blue de battre Garry Kasparov au jeu d'échecs lors d'un match revanche de six parties.

- l'intelligence artificielle devient un domaine de recherche international, marquée par une conférence au Dartmouth College à l'été 1956, à laquelle assistaient ceux qui vont marquer la discipline. Depuis les années 1960, la recherche se fait principalement aux États-Unis, notamment à l'université Stanford sous l'impulsion de John McCarthy, au MIT sous celle de Marvin Minsky, à l'université Carnegie-Mellon sous celle de Allen Newell et Herbert Simon et à l'université d'Édimbourg sous celle de Donald Michie ; en Europe et en Chine, ainsi qu'au Japon avec le projet « ordinateurs de cinquième génération (en) » du gouvernement. En France, l'un des pionniers est Jacques Pitrat.

- Dans les années 2000, le Web 2.0, le big data et de nouvelles puissances et infrastructures de calcul permettent à certains ordinateurs d'explorer des masses de données sans précédent ; c'est l'apprentissage profond (« deep learning »), dont l'un des pionniers est le français Yann Le Cun.

Les bornes de ce domaine varient, ainsi optimiser un itinéraire était considéré comme un problème d'intelligence artificielle dans les années 1950 et n'est plus considéré aujourd'hui que comme un simple problème d'algorithmie.

Vers 2015, le secteur de l'intelligence artificielle cherche à relever quatre défis : la perception visuelle, la compréhension du langage naturel écrit ou parlé, l'analyse automatique du langage et la prise de décision autonome. Produire et organiser des données nombreuses et de qualité, c'est-à-dire corrélées, complètes, qualifiées (sourcées, datées, géoréférencées…), historisées est un autre enjeu. La capacité déductive et de généralisation pertinente d'un ordinateur, à partir de peu de données ou d'un faible nombre d'évènements, est un autre objectif, plus lointain.

Entre 2010 et 2016, les investissements auraient été décuplés, atteignant une dizaine de milliards de dollars en 2016. [9]

4. Type d'intelligence artificielle

4.1 L'intelligence artificielle forte

L'intelligence artificielle forte (IA) est une forme de machine intelligente qui équivaut à l'intelligence humaine. Les principales caractéristiques de l'IA forte comprennent la capacité de raisonner, de résoudre des énigmes, de porter des jugements, de planifier, d'apprendre et de communiquer. Battre Kasparov aux échecs ne relève pas forcément d'une telle IA forte, coder correctement des dispositifs informatiques classiques peut suffire pour cela. Faire participer des robots humanoïdes à un réseau d'intelligence collective mené par des ingénieurs serait en revanche une autre paire de manches. Une IA faible, telle que nous la connaissons aujourd'hui, sera insuffisante. [10]

Le plus important peut-être est en effet qu'une IA forte devrait également avoir une conscience, des pensées objectives, une conscience de soi, une sensibilité et ce que l'on appelle la sapience, ou sagesse abstraite, typique de l'homme qui a beaucoup lu et beaucoup appris. L'intelligence artificielle forte est aussi appelée intelligence véritable ou intelligence artificielle générale (IAG).

- Risques et avantages de l'IA forte :

La possibilité d'une IA forte présenterait des avantages potentiels gigantesques. Mais elle serait porteuse aussi de cauchemars dignes de Kubrick ou de Spielberg. Certaines personnes craignent que, si les machines d'IA forte deviennent une

réalité, elles puissent devenir plus intelligentes que l'homme, phénomène connu sous le nom de singularité.

Ses facultés seraient alors telles que l'IA pourrait se modifier elle-même et poursuivre ses propres objectifs sans interventions humaines, éventuellement de manière préjudiciable pour l'homme (pensez à des robots tueurs comme dans le film I, Robot). L'IA forte pourrait-elle être développée avec des contraintes pour empêcher de tels résultats ? L'IA forte pourrait-elle être programmée avec des valeurs morales souhaitables et l'humanité pourrait-elle s'accorder sur ce que seraient ces valeurs souhaitables ? Des recherches plus poussées sur ces questions pourraient aider à prévenir la possibilité que des robots se retournent contre nous ou à déterminer s'ils pourraient même exister. [10]

4.2. L'intelligence artificielle faible

L'intelligence artificielle faible ou l'IA étroite est une machine intelligente limitée à une zone spécifique ou étroite. L'intelligence artificielle (IA) faible simule la cognition humaine et aide l'utilisateur en automatisant des tâches fastidieuses et en procédant à des analyses de données qu'un homme ne pourrait réaliser lui-même avec le cerveau humain.

On peut avoir des villes intelligentes (smart cities), des robots tueurs, de la domotique, de l'aide à la décision. Cela n'a encore rien à voir avec la puissance de l'esprit d'un seul mathématicien. On est encore loin de savoir programmer un réseau de neurones équivalents à ceux de l'espèce humaine.

La startup qui produira le premier robot humanoïde, qu'il s'appelle Rob, Sido, Sumo ou Vilani, capable de résoudre un problème de mathématique de classe de 4ème tout en effectuant de la reconnaissance faciale (et de la reconnaissance vocale) pour identifier son propriétaire et, par exemple, surveiller un gamin sur une aire de jeu, cette start up-là est loin d'être née. [11]

- Les limites de l'intelligence artificielle faible :

Outre ses capacités limitées, l'IA faible pose d'autres problèmes. Notamment la possibilité de causer des dommages en cas de défaillance du système – imaginez un véhicule sans conducteur qui calcule mal l'emplacement d'un véhicule en approche et provoque une collision mortelle. Ou encore la possibilité de causer

des dommages si ces systèmes autonomes sont utilisés par quelqu'un qui souhaite faire du mal (comment contrer un terroriste qui utilise une voiture autonome pour déployer des explosifs dans une zone surpeuplée ?). [11]

5. Conception de systèmes intelligents

Au fil du temps, certains langages de programmation se sont avérés plus commodes que d'autres pour écrire des applications d'intelligence artificielle.

Aujourd'hui, ce sont Python et R qui fournissent les outils les plus riches dans ce domaine. Des plateformes comme TensorFlow et ses bibliothèques haut niveau ont démocratisé et accéléré le développement d'intelligences artificielles. [9]

6. Distinction entre intelligence artificielle Machine Learning et Deep Learning

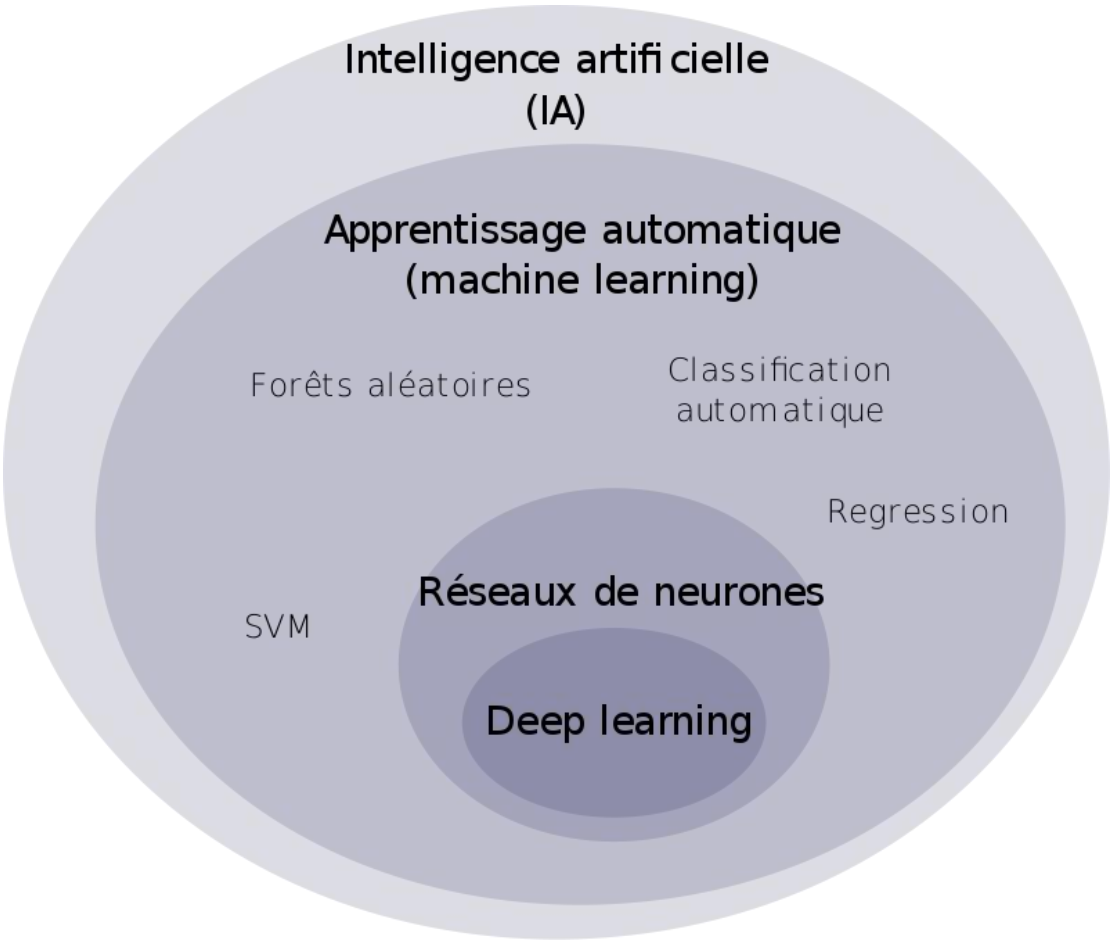

Figure 2.1 : Schéma montrant le positionnement des notions d'IA.

Il y’a une confusion fréquente dans le débat public entre « intelligence artificielle », apprentissage automatique (machine Learning) et apprentissage profond (deep Learning). Pourtant, ces notions ne sont pas équivalentes, mais sont imbriquées :

- l'intelligence artificielle englobe le « machine Learning », qui lui-même englobe le deep Learning.
- l'intelligence artificielle peut aussi englober plusieurs autres types de briques logicielles, comme les moteurs de règles. [9]

7. Domaines d’application

L’intelligence artificielle touche plusieurs domaines parmi ces domaines on peut citer :

7.1 Finance et banques

Certaines banques font appel à des systèmes experts d'évaluation de risque lié à l'octroi d'un crédit notamment en utilisant ces systèmes pour la vérification des informations fournies, ou leur récupération et traitement de façon automatisée. A l’instar des autres secteurs du droit, la réglementation bancaire et financière repose sur le postulat d’une supervision humaine, éventuellement aidée par la machine ; l’IA inverse cette proposition, induisant des bouleversements majeurs dans la banque de détail, la banque d’investissement et en matière de conformité.

7.2 Militaire

Le domaine militaire utilise des systèmes tels que les drones, les systèmes de commandement et d'aide à la décision et même des armes autonomes appelé aussi robot militaire comme l’illustre la figure 2.2.

L’utilisation des intelligences artificielles dans le domaine militaire est devenue de plus en plus importante. Les États-Unis ont dépensé 18 milliards de dollars pour trois années de recherches dans tous les domaines requis à l’automatisation de l’armement militaire.

Une course aux armements à base d'IA est en cours, telle qu'illustrée par le projet Maven aux États-Unis.

L'IA automatise certaines opérations courantes de la Défense et délivre des informations permettant de mieux comprendre certaines situations. Si l'IA met en lumière des problèmes techniques et permet de prévoir des dysfonctionnements dans le secteur militaire, seul un esprit humain est capable d'apporter une réponse à un problème. [9]

Figure 2.2 : Un robot militaire utilisé par l'armée de la Corée du sud.

7.3 Médecine

La médecine a aussi vu de grands progrès grâce à :

- L'utilisation de systèmes d'aide au diagnostic.
- Diagnostic automatisé.
- La Téléchirurgie.
- La télémédecine.
- l'utilisation de capteurs connectés pour amener une personne à modifier son comportement dans le cadre d'une action d'éducation thérapeutique.

7.4 Renseignement policier

Un usage de l'IA se développe dans le domaine de la prévention des crimes et délits. La police britannique, par exemple propose une IA opérationnelle en mars 2019, baptisée National Data Analytics Solution (Solution nationale d'analyse de données ou NDAS), elle repose sur l'IA et des statistiques et vise à estimer le risque qu'une personne commette un crime ou en soit elle-même victime, pour orienter les services sociaux et médicaux qui peuvent la conseiller.

L'usage d'outils de prédiction des crimes à partir des données préalablement existantes est toutefois l'objet de controverses, compte tenu des biais sociaux (notamment raciaux) qu'il comporte. En effet, la logique d'identification de schémas propre à ces technologies joue un rôle de renforcement des préjugés déjà existants.

7.5 Droit

Le droit fait appel à l'IA dans la perspective de prédire les décisions de justice, d'aider à la décision et de trancher les cas simples. L'Estonie a par exemple développé une intelligence artificielle capable de prendre des décisions de justice sur des délits mineurs. Les États-Unis utilisent par ailleurs dans certaines juridictions le système COMPAS (Correctional Offender Management profiling for Alternative Sanctions), un système d'aide de prise à la décision pour les juges. Plusieurs startups se sont spécialisées dans ce créneau, créant le domaine de la legaltech.

7.6 Logistique et transports

Le domaine de la logistique a vu certains projets utilisant de l'intelligence artificielle se développer notamment pour la gestion de la chaîne logistique (supply chain) ou des problématiques de livraison.

L'intelligence artificielle est également fortement utilisée dans le domaine des transports en commun, car elle permet de faciliter la régulation et la gestion du trafic au sein de réseaux de plus en plus complexes, comme le système UrbanLoop actuellement en cours d'étude dans la ville de Nancy.

Même si les problèmes d'optimisation de temps de trajet ou de transports font partie des plus anciennes applications de solutions à base d'intelligence artificielle (voir le problème du voyageur de commerce ou l'algorithme de Dijkstra), les avancées récentes, notamment en apprentissage profond, ont permis des progrès significatifs en matière de précision. Certains projets comme Google Maps utilisent par exemple des systèmes d'IA en milieu urbain pour compenser la réflexion du signal GPS sur les immeubles avoisinant, ou pour cartographier des zones où peu d'informations sont disponibles.

Plusieurs entreprises ont par ailleurs annoncé avoir développé des programmes de recherche en voiture autonome, notamment Google à travers sa filiale Waymo, l'entreprise française Navya ou encore Tesla.

7.7 Industrie

Les systèmes intelligents deviennent monnaie courante dans de nombreuses industries. Plusieurs tâches peuvent leur être confiées, notamment celles considérées comme trop dangereuses pour un humain. Certaines applications se concentrent sur les systèmes de maintenance prédictive, permettant des gains de performance grâce à une détection des problèmes de production en amont.

7.8 Robotique

La robotique a recours à l'intelligence artificielle à plusieurs égards. Notamment pour la perception de l'environnement (objets et visages), l'apprentissage et l'intelligence artificielle développementale.

L'interaction homme-robot manque encore souvent de naturel et est un enjeu de la robotique. Il s'agit de permettre aux robots d'évoluer dans le monde dynamique et social des humains et d'échanger avec eux de façon satisfaisante. L'échange nécessite également, à l'inverse, une évolution du regard que les humains portent sur les robots ; selon Véronique Aubergé, chercheuse à l'Université Grenoble-Alpes « la vraie révolution n'est pas technologique, elle est culturelle ». D'ores et déjà, à travers les robots dotés d'intelligence artificielle, tels Google Home, les utilisateurs combleraient un isolement social.

7.9 Jeux vidéo

L'intelligence artificielle est par exemple utilisée pour animer les personnages non-joueurs de jeux vidéo, qui sont conçus pour servir d'opposants, d'aides ou d'accompagnants lorsque des joueurs humains ne sont pas disponibles ou désirés. Différents niveaux de complexité sont développés.

Le 24 janvier 2019, Google DeepMind présente sur son blog AlphaStar, une intelligence artificielle dédiée au jeu de stratégie en temps réel StarCraft II qui a affronté deux joueurs humains lors d'un match retransmis en direct sur Internet. Durant cet évènement, AlphaStar bat deux joueurs professionnels, dont Grzegorz « MaNa » Komincz, de l'équipe Team Liquid, l'un des meilleurs joueurs professionnels au monde. Le développement de cette intelligence artificielle a été permis par un partenariat entre Google DeepMind et Blizzard Entertainment, l'éditeur du jeu. [9]

7.10 Art

Dès la fin des années 1980, des artistes s'emparent de l'intelligence artificielle pour donner un comportement autonome à leurs œuvres. Les Français Michel Bret, Edmond Couchot et Marie-Hélène Tramus sont des pionniers, ainsi qu'en témoignent des œuvres comme La Plume et Le Pissenlit (1988)132, puis La Funambule (2000), animée par un réseau de neurones. L'Américain Karl Sims, en partenariat avec la société Thingking Machines, créée en 1993 Genetic Images, machines incorporant des algorithmes génétiques. Le couple franco-autrichien Christa Sommerer et Laurent Mignonneau crée depuis le début des années 1990 de nombreuses œuvres dans le champ de la vie artificielle, parmi lesquelles Interactive plant growing (1992) ou A-Volve (1994). Le Français Florent Aziosmanoff propose quant à lui de considérer que l'emploi de l'intelligence artificielle dans l'art conduit à l'émergence d'une nouvelle discipline d'expression, qu'il nomme le Living art.

En mars 2018, l'artiste Joseph Ayerle publie la vidéo d'art intitulée Un'emozione per sempre 2.0, dans laquelle il met en scène une Ornella Muti virtuelle, recréée par une intelligence artificielle. Après seulement quelques jours d'entraînement, l'intelligence artificielle est capable d'animer le visage de l'actrice italienne pour réaliser des scènes qu'elle n'a jamais jouées.

Le 23 octobre 2018, la société de vente aux enchères Christie's met en vente le tableau Portrait d'Edmond de Belamy réalisé par une intelligence artificielle à l'aide de réseaux antagonistes génératifs. La peinture est signée par la formule mathématique à l'origine de sa création (« Min (G) max (D) Ex [log (D(x))] + Ez [log (1-D(G(z)))] »). Cette vente soulève de nombreux débats sur son statut de création artistique et sur l'auteur de l'œuvre : il peut être l'intelligence artificielle elle-même ou les trois créateurs qui l'ont programmée. L'œuvre est achetée pour 350 000 dollars. Cette vente peut être considérée comme une reconnaissance du GAN-isme (l'abréviation de Generative Adversarial Networks, « réseaux antagonistes génératifs » en français), un mouvement artistique qui utilise l'intelligence artificielle dans la création d'une œuvre picturale.

L'artiste numérique Solimán López utilise l'intelligence artificielle comme outil pour créer des interactions inédites avec d'autres médias, outils et concepts. En 2019, dans High Meshes, il invente des micro-communautés de personnes réelles scannées en 3D par photogrammétrie. Ces données alimentent un logiciel d'intelligence artificielle qui rassemble les corps en fonction de leurs informations purement numériques sans tenir compte des questions raciales, sexuelles, religieuses, politiques ou culturelles. Dans le projet D.A.I, en 2018, des cartes d'identités de multiples pays sont analysées par une intelligence artificielle et aboutissent à de nouveaux papiers, symbolisant un monde sans frontières. [9]

8. L'intelligence artificielle aujourd'hui

De nos jours l'IA est partout comme montré dans la figure 2.3, et il ne fait aucun doute que ses technologies seront d'ordre transformationnel. Des avancées spectaculaires seront réalisées, une richesse extraordinaire sera créée et bon nombre de nos structures sociales et institutionnelles seront transformées. De nos jours, seul l'Intelligence Artificielle faible existe, par opposition à l'IA forte qui est une machine dotée de conscience, de sensibilité et d'esprit.

Figure 2.3 : Domaines touchés par l'intelligence artificielle.

En effet, l'IA faible se contente de reproduire un comportement humain sans conscience. Cet outil est déjà puissant puisqu'il est capable d'automatiser les tâches, comme le fait notamment un algorithme de Machine Learning.

Voici quelques exemples qui illustrent les capacités actuelles de l'IA :

- reconnaître ce qu'il y a dans une image,
- reproduire le style de n'importe quel artiste reconnu,
- comprendre les requêtes en langage naturel,
- générer des images qui semblent réelles,
- reconnaître vos enfants sur une image mieux que vous-mêmes,
- diagnostiquer le cancer mieux qu'un médecin, etc.

9. Conclusion

Dans ce chapitre on a présenté l'intelligence artificielle ses modèles, principes, types ainsi que ses vastes domaines d'utilisations. Dans le prochain chapitre on parlera de l'apprentissage automatique et les réseaux de neurones ainsi que l'outil que nous utiliserons pour l'implémentation à savoir TensorFlow.

CHAPITRE 3

Machine Learning et réseaux de neurones.

1. Introduction

En général, l'intelligence artificielle consiste à apprendre à une machine des techniques mises en œuvre afin qu'elle puisse simuler une intelligence presque identique à celle de l'humain. L'un des champs d'étude de l'IA le plus fonctionnel est l'apprentissage, qui se fonde sur la création de réseau de neurones.

Toutefois, le processus d'acquisition des données s'est avéré être très difficile donc il a fallu trouver une solution qui puisse aider à faciliter l'entraînement des modèles, c'est ainsi que TensorFlow a été créé.

Dans ce chapitre on va faire une présentation et description des réseaux de neurones en même temps on fera une description de la solution utilisée pour intégrer les objectifs de notre projet.

2. Machine Learning

2.1 Définition

L'apprentissage automatique ou Machine Learning en anglais, est une science moderne permettant de découvrir des répétitions (des patterns) dans un ou plusieurs flux de données et d'en tirer des prédictions en se basant sur des statistiques. En clair, le Machine Learning se base sur le forage de données, permettant la reconnaissance de patterns pour fournir des analyses prédictives.[12]

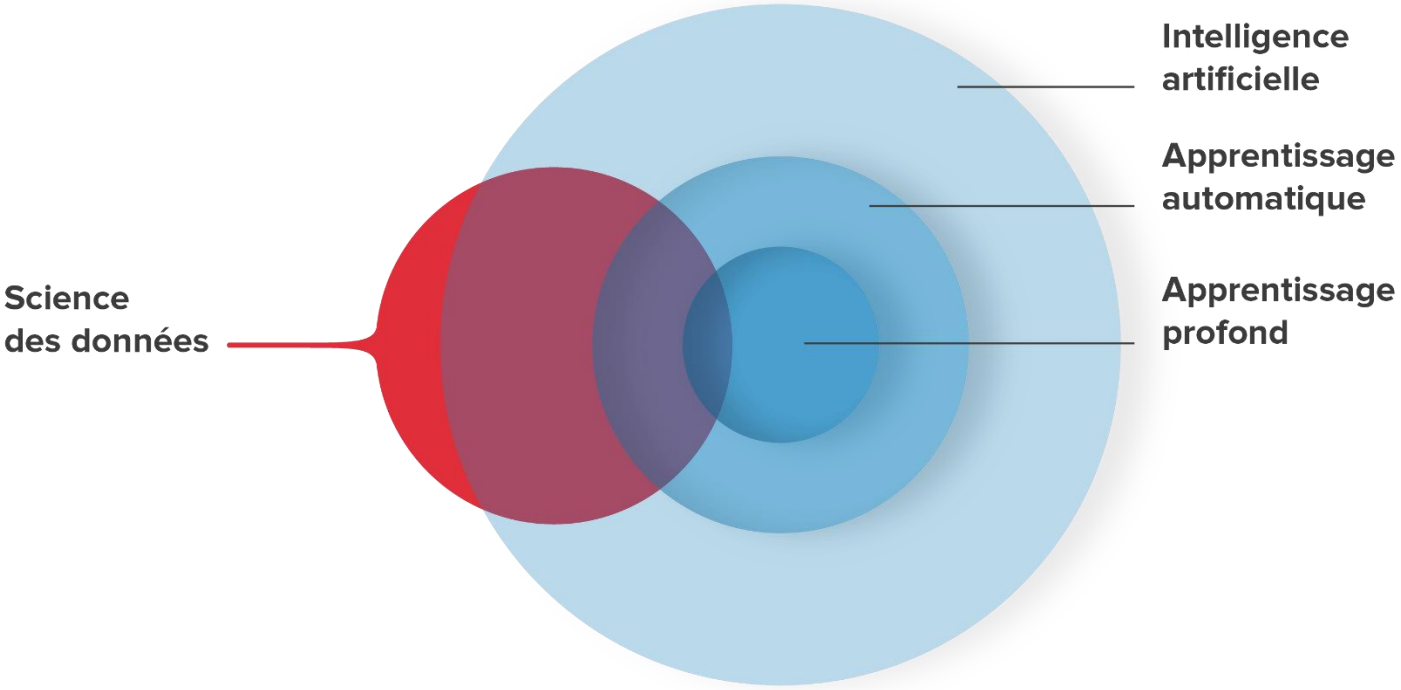

Figure 3.1 : La place de l'apprentissage automatique par rapport à l'intelligence artificielle.

2.3 Principes

L'apprentissage automatique (AA) permet à un système piloté ou assisté par ordinateur comme un programme, une IA ou un robot, d'adapter ses réponses ou comportements aux situations rencontrées, en se fondant sur l'analyse de données empiriques passées issues de bases de données, de capteurs, ou du web.

L'AA permet de surmonter la difficulté qui réside dans le fait que l'ensemble de tous les comportements possibles compte tenu de toutes les entrées possibles devient rapidement trop complexe à décrire et à programmer de manière classique (on parle d'explosion combinatoire). On confie donc à des programmes d'AA le soin d'ajuster un modèle pour simplifier cette complexité et de l'utiliser de manière opérationnelle. Idéalement, l'apprentissage visera à être non supervisé, c'est-à-dire que les réponses aux données d'entraînement ne sont pas fournies au modèle.

Ces programmes, selon leur degré de perfectionnement, intègrent éventuellement des capacités de traitement probabiliste des données, d'analyse

de données issues de capteurs, de reconnaissance (reconnaissance vocale, de forme, d'écriture…), de fouille de données, d'informatique théorique… [12]

2.4 Modes d'apprentissage

Il existe plusieurs modes d'apprentissage comme illustrés dans la figure 3.1 :

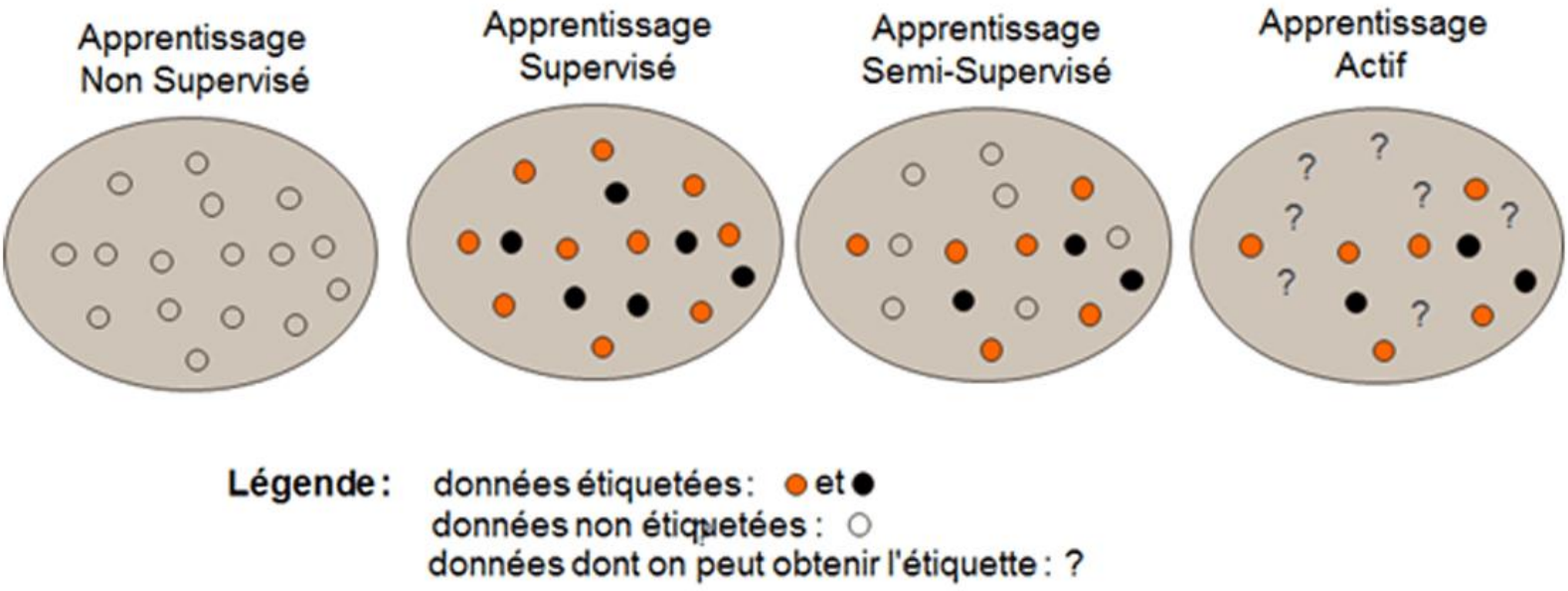

Figure 3.2 : quelques modes d'apprentissage.

- Apprentissage supervisé :

Si les classes sont prédéterminées et les exemples connus, le système apprend à classer selon un modèle de classification ou de classement ; on parle alors d'apprentissage supervisé (ou d'analyse discriminante). Un expert (ou oracle) doit préalablement étiqueter des exemples. Le processus se passe en deux phases. Lors de la première phase (hors ligne, dite d'apprentissage), il s'agit de déterminer un modèle à partir des données étiquetées. La seconde phase (en ligne, dite de test) consiste à prédire l'étiquette d'une nouvelle donnée, connaissant le modèle préalablement appris. Parfois il est préférable d'associer une donnée non pas à une classe unique, mais une probabilité d'appartenance à chacune des classes prédéterminées (on parle alors d'apprentissage supervisé probabiliste). [12]

- Apprentissage non supervisé :

Quand le système ou l'opérateur ne dispose que d'exemples, mais non d'étiquette, et que le nombre de classes et leur nature n'ont pas été prédéterminés, on parle d'apprentissage non supervisé ou clustering en anglais. Aucun expert n'est requis. L'algorithme doit découvrir par lui-même la structure plus ou moins cachée des données. Le partitionnement de données, data clustering en anglais, est un algorithme d'apprentissage non supervisé.

Le système doit ici — dans l'espace de description (l'ensemble des données) — cibler les données selon leurs attributs disponibles, pour les classer en groupes homogènes d'exemples. La similarité est généralement calculée selon une fonction de distance entre paires d'exemples. C'est ensuite à l'opérateur d'associer ou déduire du sens pour chaque groupe et pour les motifs (patterns en anglais) d'apparition de groupes, ou de groupes de groupes, dans leur « espace ». Divers outils mathématiques et logiciels peuvent l'aider. On parle aussi d'analyse des données en régression (ajustement d'un modèle par une procédure de type moindres carrés ou autre optimisation d'une fonction de coût). Si l'approche est probabiliste (c'est-à-dire que chaque exemple, au lieu d'être classé dans une seule classe, est caractérisé par un jeu de probabilités d'appartenance à chacune des classes), on parle alors de « soft clustering » (par opposition au « hard clustering »). Cette méthode est souvent source de sérendipité.

- Apprentissage semi-supervisé :

Effectué de manière probabiliste ou non, il vise à faire apparaître la distribution sous-jacente des exemples dans leur espace de description. Il est mis en œuvre quand des données (ou « étiquettes ») manquent… Le modèle doit utiliser des exemples non étiquetés pouvant néanmoins renseigner.

- Apprentissage partiellement supervisé :

Probabiliste ou non, quand l'étiquetage des données est partiel. C'est le cas quand un modèle énonce qu'une donnée n'appartient pas à une classe A, mais peut-être à une classe B ou C.

- Apprentissage par renforcement :

L'algorithme apprend un comportement étant donné une observation. L'action de l'algorithme sur l'environnement produit une valeur de retour qui guide l'algorithme d'apprentissage.

- Apprentissage par transfert :

L'apprentissage par transfert peut être vu comme la capacité d'un système à reconnaître et appliquer des connaissances et des compétences, apprises à partir de tâches antérieures, sur de nouvelles tâches ou domaines partageant des similitudes.

2.5 Applications

L'apprentissage automatique est utilisé dans un large spectre d'applications pour doter des ordinateurs ou des machines de capacité d'analyser des données d'entrée comme perception de leur environnement dans plusieurs domaines dont on site :

- ❖ Vision
- ❖ Reconnaissance de formes telles des visages
- ❖ Schémas

- segmentation d'image
- langages naturels
- caractères dactylographiés ou manuscrits
- moteurs de recherche
- analyse et indexation d'images et de vidéo en particulier pour la recherche d'image par le contenu
- aide aux diagnostics médicaux notamment, bio-informatique, chémoinformatique ; interfaces cerveau-machine.
- détection de fraudes à la carte de crédit
- cybersécurité
- analyse financière, dont analyse du marché boursier
- classification des séquences d'ADN
- jeu
- génie logiciel
- adaptation de sites Web

- ❖ robotique (locomotion de robots, etc.)

- ❖ analyse prédictive dans de nombreux domaines (financière, médicale, juridique, judiciaire).

3. Réseau de neurones

3.1 Définition

Un réseau de neurones artificiels, ou réseau neuronal artificiel, est un système dont la conception est à l'origine schématiquement inspirée du fonctionnement des neurones biologiques comme nous le montre la figure 3.2, et qui par la suite s'est rapproché des méthodes statistiques.

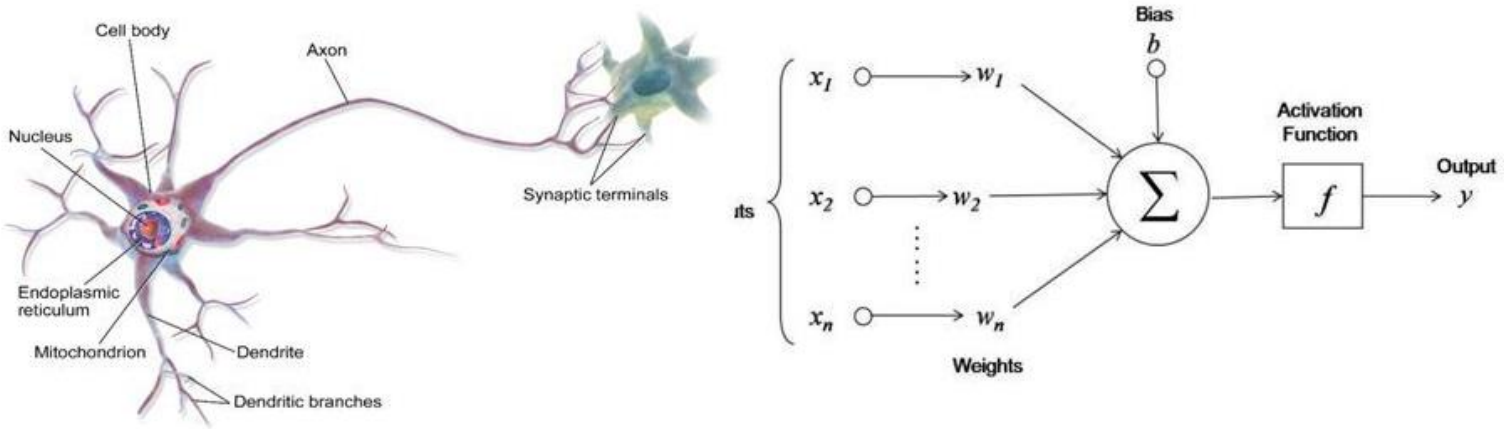

Figure 3.3 : Ressemblance entre un réseau neuronal artificiel et les neurones biologiques.

Les réseaux de neurones sont généralement optimisés par des méthodes d'apprentissage de type probabiliste, en particulier bayésien. Ils sont placés d'une part dans la famille des applications statistiques, qu'ils enrichissent avec un ensemble de paradigmes permettant de créer des classifications rapides (réseaux de Kohonen en particulier), et d'autre part, dans la famille des méthodes de l'intelligence artificielle auxquelles ils fournissent un mécanisme perceptif

indépendant des idées propres de l'implémenteur, et des informations d'entrée au raisonnement logique formel (voir Deep Learning).

En modélisation des circuits biologiques, ils permettent de tester quelques hypothèses fonctionnelles issues de la neurophysiologie, ou encore les conséquences de ces hypothèses pour les comparer au réel. [13]

3.2 Historique

Les réseaux neuronaux sont construits sur un paradigme biologique, celui du neurone formel (comme les algorithmes génétiques le sont sur la sélection naturelle). Ces types de métaphores biologiques sont devenus courants avec les idées de la cybernétique et biocybernétique. Selon la formule de Yann Le Cun, celui-ci ne prétend pas davantage décrire le cerveau qu'une aile d'avion, par exemple, copie celle d'un oiseau. En particulier le rôle des cellules gliales n'est pas simulé.

3.2.1 Neurone formel

Les neurologues Warren McCulloch et Walter Pitts publiaient dès la fin des années 1950 les premiers travaux sur les réseaux de neurones, avec un article fondateur : What the frog's eye tells the frog's brain (Ce que l'œil d'une grenouille dit à son cerveau). Ils constituent ensuite un modèle simplifié de neurone biologique communément appelé neurone formel. Ils montrèrent que des réseaux de neurones formels simples peuvent théoriquement réaliser des fonctions logiques, arithmétiques et symboliques complexes.

Le neurone formel est conçu comme un automate doté d'une fonction de transfert qui transforme ses entrées en sortie selon des règles précises. Par exemple, un neurone somme ses entrées, compare la somme résultante à une valeur seuil, et répond en émettant un signal si cette somme est supérieure ou égale à ce seuil (modèle ultra-simplifié du fonctionnement d'un neurone biologique). Ces neurones sont par ailleurs associés à des réseaux dont la topologie des connexions est variable : réseaux proactifs, récurrents, etc. Enfin, l'efficacité de la transmission des signaux d'un neurone à l'autre peut varier : on parle de « poids synaptique », et ces poids peuvent être modulés par des règles d'apprentissage (ce qui mime la plasticité synaptique des réseaux biologiques).

Une fonction des réseaux de neurones formels, à l'instar du modèle vivant, est d'opérer rapidement des classifications et d'apprendre à les améliorer. À l'opposé des méthodes traditionnelles de résolution informatique, on ne doit pas construire un programme pas à pas en fonction de la compréhension de celui-ci. Les paramètres importants de ce modèle sont les coefficients synaptiques et le seuil de chaque neurone, et la façon de les ajuster. Ce sont eux qui déterminent l'évolution du réseau en fonction de ses informations d'entrée. Il faut choisir un mécanisme permettant de les calculer et de les faire converger si possible vers une valeur assurant une classification aussi proche que possible de l'optimale. C'est ce qu'on nomme la phase d'apprentissage du réseau. Dans un modèle de réseaux de neurones formels, apprendre revient donc à déterminer les coefficients synaptiques les plus adaptés à classifier les exemples présentés. [13]

3.2.2 Perceptron

Les travaux de McCulloch et Pitts n'ont pas donné d'indication sur une méthode pour adapter les coefficients synaptiques. Cette question au cœur des réflexions sur l'apprentissage a connu un début de réponse grâce aux travaux du physiologiste canadien Donald Hebb sur l'apprentissage en 1949 décrits dans son ouvrage The Organization of Behaviour. Hebb a proposé une règle simple qui permet de modifier la valeur des coefficients synaptiques en fonction de l'activité des unités qu'ils relient. Cette règle aujourd'hui connue sous le nom de « règle de Hebb » est presque partout présente dans les modèles actuels, même les plus sophistiqués. [13]

À partir de cet article, l'idée se sema au fil du temps dans les esprits, et elle germe dans l'esprit de Frank Rosenblatt en 1957 avec le modèle du perceptron. C'est le premier système artificiel capable d'apprendre par expérience, y compris lorsque son instructeur commet quelques erreurs (ce en quoi il diffère nettement d'un système d'apprentissage logique formel).

En 1969, un coup grave fut porté à la communauté scientifique gravitant autour des réseaux de neurones : Marvin Lee Minsky et Seymour Papert publièrent un ouvrage mettant en exergue quelques limitations théoriques du perceptron, et plus généralement des classifieurs linéaires, notamment l'impossibilité de traiter des problèmes non linéaires ou de connexité. Ils étendirent implicitement ces limitations à tous modèles de réseaux de neurones artificiels. Paraissant alors dans une impasse, la recherche sur les réseaux de neurones perdit une grande

partie de ses financements publics, et le secteur industriel s'en détourna aussi. Les fonds destinés à l'intelligence artificielle furent redirigés plutôt vers la logique formelle. Cependant, les solides qualités de certains réseaux de neurones en matière adaptative (e.g. Adaline), leur permettant de modéliser de façon évolutive des phénomènes eux-mêmes évolutifs, les amèneront à être intégrés sous des formes plus ou moins explicites dans le corpus des systèmes adaptatifs; utilisés dans le domaine des télécommunications ou celui du contrôle de processus industriels.

En 1982, John Joseph Hopfield, physicien reconnu, donna un nouveau souffle au neuronal en publiant un article introduisant un nouveau modèle de réseau de neurones (complètement récurrent). Cet article eut du succès pour plusieurs raisons, dont la principale était de teinter la théorie des réseaux de neurones de la rigueur propre aux physiciens. Le neuronal redevint un sujet d'étude acceptable, bien que le modèle de Hopfield souffrît des principales limitations des modèles des années 1960, notamment l'impossibilité de traiter les problèmes non linéaires.

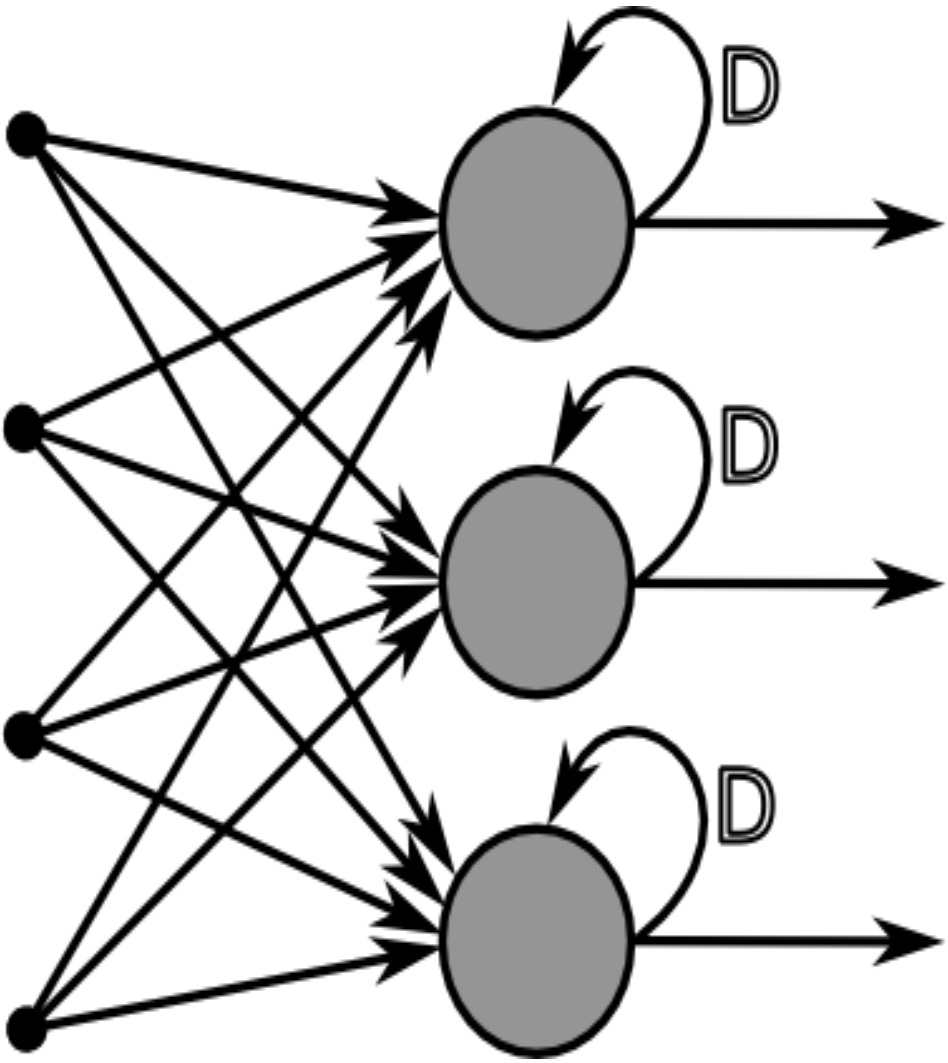

Figure 3.4 : Réseau de neurones avec rétroaction.

3.2.3 Perceptron multicouche

À la même date, les approches algorithmiques de l'intelligence artificielle furent l'objet de désillusion, leurs applications ne répondant pas aux attentes. Cette désillusion motiva une réorientation des recherches en intelligence artificielle vers les réseaux de neurones (bien que ces réseaux concernent la perception artificielle plus que l'intelligence artificielle à proprement parler). La recherche fut relancée et l'industrie reprit quelque intérêt au neuronal (en particulier pour des applications comme le guidage de missiles de croisière). En 1984, c'est le système de rétropropagation du gradient qui est le sujet le plus débattu dans le domaine.

Une révolution survient dans le domaine des réseaux de neurones artificiels : une nouvelle génération de réseaux de neurones, capables de traiter avec succès des phénomènes non linéaires : le perceptron multicouche ne possède pas les défauts mis en évidence par Marvin Minsky. Proposé pour la première fois par Paul

Werbos, le perceptron multi-couche apparait en 1986 introduit par David Rumelhart, et, simultanément, sous une appellation voisine, chez Yann Le Cun. Ces systèmes reposent sur la rétropropagation du gradient de l'erreur dans des systèmes à plusieurs couches, chacune de type Adaline de Bernard Widrow, proche du perceptron de Rumelhart.

Les réseaux de neurones ont par la suite connu un essor considérable, et ont fait partie des premiers systèmes à bénéficier de l'éclairage de la théorie de la « régularisation statistique » introduite par Vladimir Vapnik en Union soviétique et popularisée en Occident depuis la chute du mur. Cette théorie, l'une des plus importantes du domaine des statistiques, permet d'anticiper, d'étudier et de réguler les phénomènes liés au surapprentissage. On peut ainsi réguler un système d'apprentissage pour qu'il arbitre au mieux entre une modélisation pauvre (exemple : la moyenne) et une modélisation trop riche qui serait optimisée de façon illusoire sur un nombre d'exemples trop petit, et serait inopérant sur des exemples non encore appris, même proches des exemples appris. Le surapprentissage est une difficulté à laquelle doivent faire face tous les systèmes d'apprentissage par l'exemple, que ceux-ci utilisent des méthodes d'optimisation directe (e.g. régression linéaire), itératives (e.g., l'algorithme du gradient), ou itératives semi-directes (gradient conjugué, espérance-maximisation...) et que ceux-ci soient appliqués aux modèles statistiques classiques, aux modèles de Markov cachés ou aux réseaux de neurones formels. [13]

3.2.4 Réseau neuronal convolutif

Les réseaux de neurones évoluent avec un nouveau type de réseau non complètement connecté, pour alléger les modèles en nombre de paramètres, et améliorer les performances et leur capacité de généralisation. Une des premières applications a été la reconnaissance automatique des codes postaux US, avec le réseau LeNet-5. En apprentissage automatique, un réseau de neurones convolutifs ou réseau de neurones à convolution (en anglais CNN ou ConvNet pour Convolutional Neural Networks) est un type de réseau de neurones artificiels acycliques (feed-forward), dans lequel le motif de connexion entre les neurones est inspiré par le cortex visuel des animaux. Les neurones de cette région du cerveau sont arrangés de sorte qu'ils correspondent à des régions qui se chevauchent lors du pavage du champ visuel. Leur fonctionnement est inspiré par les processus biologiques, ils consistent en un empilage multicouche de

perceptrons, dont le but est de prétraiter de petites quantités d'informations. Les réseaux neuronaux convolutifs ont de larges applications dans la reconnaissance d'image et vidéo, les systèmes de recommandation et le traitement du langage naturel. [13]

3.3 La structure d’un réseau de neurones

Un réseau de neurones est en général composé d'une succession de couches dont chacune prend ses entrées sur les sorties de la précédente. Chaque couche (i) est composée de Xi neurones, prenant leurs entrées sur les Xi-1 neurones de la couche précédente. À chaque synapse est associé un poids synaptique, de sorte que les Xi-1 sont multipliés par ce poids, puis additionnés par les neurones de niveau i, ce qui est équivalent à multiplier le vecteur d'entrée par une matrice de transformation. Mettre l'une derrière l'autre les différentes couches d'un réseau de neurones reviendrait à mettre en cascade plusieurs matrices de transformation et pourrait se ramener à une seule matrice, produit des autres, s'il n'y avait à chaque couche, la fonction de sortie qui introduit une non linéarité à chaque étape. Ceci montre l'importance du choix judicieux d'une bonne fonction de sortie : un réseau de neurones dont les sorties seraient linéaires n'aurait aucun intérêt.

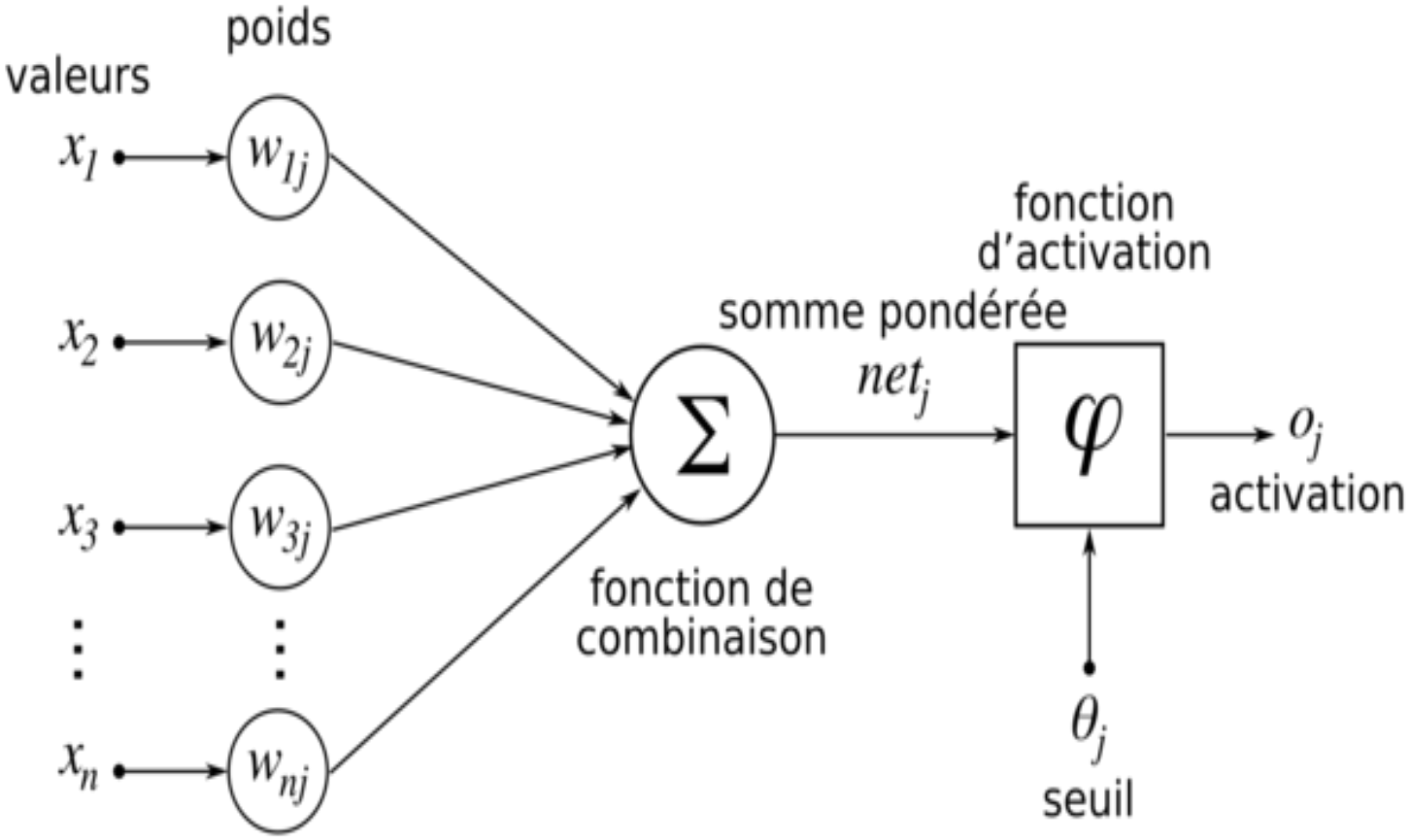

Figure 3.5 : Structure d'un neurone artificiel ou neurone formel.

Au-delà de cette structure simple, le réseau de neurones peut également contenir des boucles qui en changent radicalement les possibilités mais aussi la complexité. De la même façon que des boucles peuvent transformer une logique combinatoire en logique séquentielle, les boucles dans un réseau de neurones transforment un simple dispositif de reconnaissance d'entrées en une machine complexe capable de toutes sortes de comportements. [13]

4. Apprentissage

4.1 Base théorique

La notion d'apprentissage, bien que connue déjà depuis Sumer, n'est pas modélisable dans le cadre de la logique déductive : celle-ci en effet procède à partir de connaissances déjà établies dont on tire des connaissances dérivées. Or il s'agit ici de la démarche inverse : par observations limitées, tirer des généralisations plausibles : c'est un procédé par induction.

La notion d'apprentissage recouvre deux réalités souvent traitées de façon successive :

- Mémorisation : le fait d'assimiler sous une forme dense des exemples éventuellement nombreux.
- Généralisation : le fait d'être capable, grâce aux exemples appris, de traiter des exemples distincts, encore non rencontrés, mais similaires.

Dans le cas des systèmes d'apprentissage statistique, utilisés pour optimiser les modèles statistiques classiques, réseaux de neurones et automates markoviens, c'est la généralisation qui est l'objet de toute l'attention.

Cette notion de généralisation est traitée de façon plus ou moins complète par plusieurs approches théoriques. [13]

- ❖ La généralisation est traitée de façon globale et générique par la théorie de la régularisation statistique introduite par Vladimir Vapnik. Cette théorie, développée à l'origine en Union soviétique, s'est diffusée en Occident depuis la chute du mur de Berlin. La théorie de la régularisation statistique s'est diffusée très largement parmi ceux qui étudient les réseaux de neurones en raison de la forme générique des courbes d'erreurs résiduelles d'apprentissage et de généralisation issues des procédures d'apprentissage itératives telles que les descentes de gradient utilisées pour l'optimisation des perceptrons multi-couches. Ces formes génériques correspondent aux formes prévues par la théorie de la régularisation statistique ; cela vient du fait que les procédures d'apprentissage par descente de gradient, partant d'une configuration initiale des poids synaptiques explorent progressivement l'espace des poids synaptiques possibles ; on retrouve alors la problématique de l'augmentation progressive de la capacité d'apprentissage, concept fondamental au cœur de la théorie de la régularisation statistique.

- ❖ La généralisation est aussi au cœur de l'approche de l'inférence bayésienne, enseignée depuis plus longtemps. Le théorème de Cox-Jaynes fournit ainsi une base importante à un tel apprentissage, en nous apprenant que toute méthode d'apprentissage est soit isomorphe aux probabilités munies de la relation de Bayes, soit incohérente. C'est là un résultat extrêmement fort, et c'est pourquoi les méthodes bayésiennes sont largement utilisées dans le domaine. [13]

4.2 Algorithme

La grande majorité des réseaux de neurones possède un algorithme d'entraînement qui consiste à modifier les poids synaptiques en fonction d'un jeu de données présentées en entrée du réseau. Le but de cet entraînement est de permettre au réseau de neurones d'apprendre à partir des exemples. Si l'entraînement est correctement réalisé, le réseau est capable de fournir des réponses en sortie très proches des valeurs d'origine du jeu de données d'entraînement. Mais tout l'intérêt des réseaux de neurones réside dans leur capacité à généraliser à partir du jeu de test. Il est donc possible d'utiliser un réseau de neurones pour réaliser une mémoire ; on parle alors de mémoire neuronale. [13]

4.3 Apprentissage supervisé et non-supervisé

L'apprentissage supervisé et l'apprentissage non-supervisé sont faciles à distinguer :

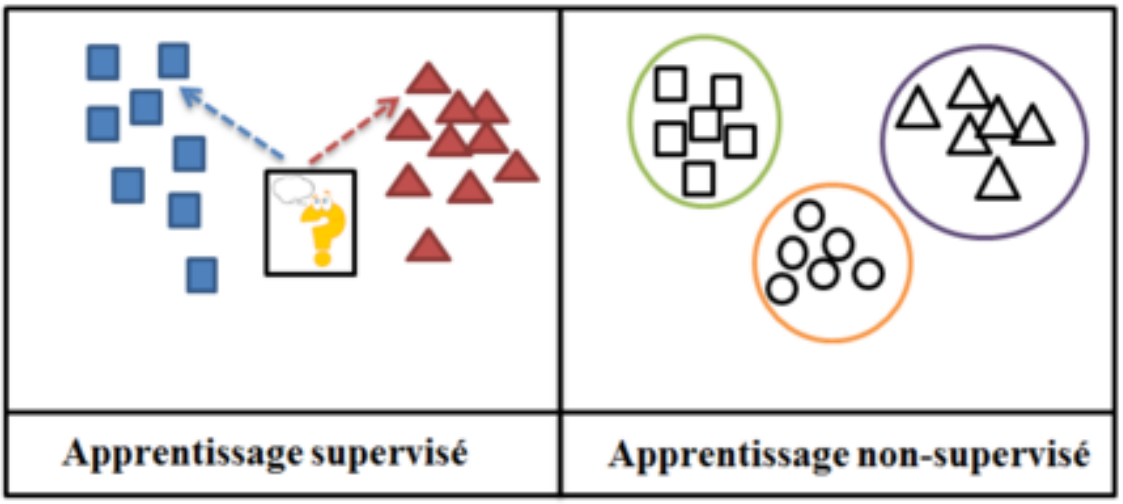

Figure 3.6 : Différences entre apprentissage supervisé et non-supervisé.

- Un apprentissage est dit supervisé lorsque le réseau est forcé à converger vers un état final précis, en même temps qu'un motif lui est présenté ; par exemple on dispose d'éléments déjà classés comme des articles en rubrique : cuisine, sport, culture …etc. On veut classer un nouvel élément,

donc le réseau ou programme doit lui attribuer un nom parmi les classes données.

- À l'inverse, lors d'un apprentissage non-supervisé, le réseau est laissé libre de converger vers n'importe quel état final lorsqu'un motif lui est présenté. On prend comme exemple qu'on a des éléments non classés, comme des images de fleurs ; si deux fleurs ont la même forme, elles sont en rapport avec une même plante correspondante.

4.4 Surapprentissage

Il arrive souvent que les exemples de la base d'apprentissage comportent des valeurs approximatives ou bruitées. Si on oblige le réseau à répondre de façon quasi parfaite relativement à ces exemples, on peut obtenir un réseau qui est biaisé par des valeurs erronées.

Par exemple, imaginons qu'on présente au réseau des couples (x_i , $f(x_i)$) situés sur une droite d'équation y=ax+b, mais bruités de sorte que les points ne soient pas exactement sur la droite. S'il y a un bon apprentissage, le réseau répond ax+b pour toute valeur de x présentée. S'il y a surapprentissage, le réseau répond un peu plus que ax+b ou un peu moins, car chaque couple (x_i , $f(x_i)$) positionné en dehors de la droite va influencer la décision : il aura appris le bruit en plus, ce qui n'est pas souhaitable.

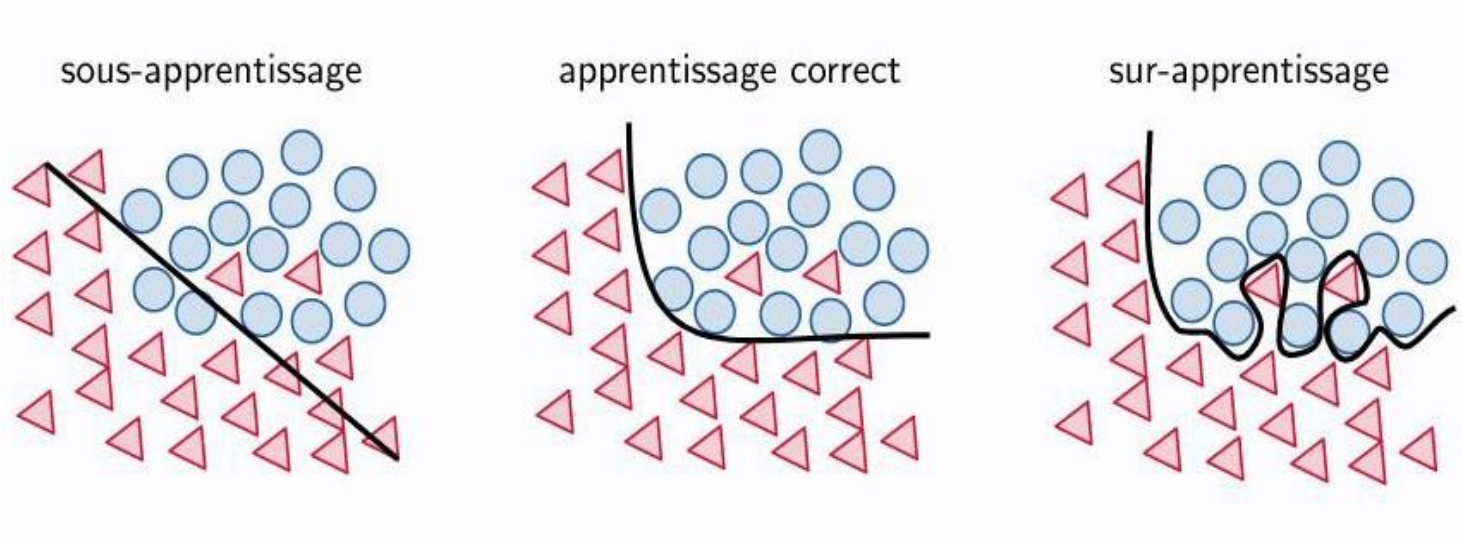

Figure 3.7 : Exemples de sous-apprentissage apprentissage correct et surapprentissage.

Pour éviter le surapprentissage, il existe une méthode simple : il suffit de partager la base d'exemples en 2 sous-ensembles. Le premier sert à l'apprentissage et le second sert à l'évaluation de l'apprentissage. Tant que l'erreur obtenue sur le deuxième ensemble diminue, on peut continuer l'apprentissage, sinon on arrête.[13]

4.5 Rétropropagation

La rétropropagation consiste à rétropropager l'erreur commise par un neurone à ses synapses et aux neurones qui y sont reliés. Pour les réseaux de neurones, on utilise habituellement la rétropropagation du gradient de l'erreur, qui consiste à corriger les erreurs selon l'importance des éléments qui ont justement participé à la réalisation de ces erreurs : les poids synaptiques qui contribuent à engendrer une erreur importante se verront modifiés de manière plus significative que les poids qui ont engendré une erreur marginale.

4.6 Elagage

L'élagage (pruning, en anglais) est une méthode qui permet d'éviter le surapprentissage tout en limitant la complexité du modèle. Elle consiste à supprimer des connexions (ou synapses), des entrées ou des neurones du réseau une fois l'apprentissage terminé. En pratique, les éléments qui ont la plus petite influence sur l'erreur de sortie du réseau sont supprimés. Deux exemples d'algorithmes d'élagage sont :

- Optimal brain damage (OBD) de Yann LeCun et al.
- Optimal brain surgeon (OBS) de B. Hassibi et D. G. Stork.

5. TensorFlow

5.1 Définition

Tensor Flow est un outil open source d'apprentissage automatique développé par Google le 9 Novembre 2015 et publié sous Licence Apache.

Il est fondé sur l'infrastructure DistBelief, initiée par Google en 2011, et est doté d'une interface pour Python, Julia et R.

TensorFlow est l'un des outils les plus utilisés en IA dans le domaine de l'apprentissage machine. TensorFlow fournit des API stables en Python et C. Des API sans rétro-compatibilité garantie en C++, Go, Java, JavaScript et Swift. Des packages faits par des tiers sont disponibles en C#, Haskell, Julia, R, Scala, Rust, Ocaml et Crystal. [14]

5.2 Historique de TensorFlow

5.2.1 DisBelief

À partir de 2011, Google Brain a développé un outil propriétaire d'apprentissage automatique fondé sur l'apprentissage profond. Son utilisation a augmenté rapidement à travers les différentes filiales d'Alphabet autant dans le milieu commercial que dans la recherche. Google a assigné de nombreux ingénieurs informaticiens, dont Jeffrey Dean, pour simplifier et réordonner le code de DisBelief en une bibliothèque logicielle plus rapide et plus solide qui est devenue Tensor fLow. En 2009, l'équipe, dirigée par Geoffrey Hinton, avait implémenté la rétropropagation du gradient généralisée et d'autres améliorations qui ont permis la création de réseaux neuronaux ayant une précision considérablement meilleure. Par exemple, une réduction de 25 % d'erreur dans la reconnaissance automatique de la parole a été obtenue.

5.2.2 TensorFlow

Tensor Flow est la deuxième génération du système de Google Brain. La version 1.0.0 est sortie le 11 février 2017 Alors que l'implémentation de référence tourne sur un seul appareil, Tensor Flow peut être lancé sur plusieurs CPU et GPU (avec

des extensions optionnelles telles que CUDA ou SYCL (en) pour GPGPU). Tensor Flow est disponible en version 64-bits pour Linux, macOS, Windows et pour les plateformes mobiles sur Android et iOS.

Son architecture flexible permet le développement sur plusieurs variétés de plateformes (CPU, GPU, TPU), allant du PC de bureau à des clusters de serveurs et des mobiles aux dispositifs de bords.

En juin 2016, Jeff Dean a mentionné que 1 500 dépôts github mentionnaient Tensor Flow, dont seulement cinq étaient de Google. [14]

5.2.3 TensorFlow lite

En mai 2017, Google a annoncé qu'une couche logicielle spécifique serait créée pour le développement sur Android, Tensor Flow Lite, à partir d'Android Oreo.

5.3 Utilisations

5.3.1 Rank Brain

Le 26 octobre 2015, Google a officiellement sorti RankBrain (en), adossé à Tensor Flow. [14]

5.3.2 Le Pentagone

Le 6 mars 2018, le site américain Gizmodo a révélé l'existence d'un partenariat entre l'entreprise Google et le Pentagone, destiné à aider ce dernier à analyser des images de drones par l'usage de Tensor Flow, sans pouvoir donner plus d'indications sur l'implication de l'entreprise. Google a déclaré : « La technologie labellise des images qui seront analysées par des humains et ne sert qu'à un usage non offensif ». [14]

5.3.3 DeepDream

DeepDream se fonde en totalité sur l'architecture de Tensor Flow pour la base algorithmique du logiciel.

6. Conclusion

Ce chapitre englobe l'importance de la machine Learning et l'intelligence artificielle. L'ensemble de toutes ces technologies est utilisé au sein du développement scientifique et plus précisément dans les domaines de l'intelligence artificielle et le machine Learning. On ne peut pas s'en passer de ces outils car la création de notre classificateur d'images se base sur ces technologies. Dans le chapitre suivant nous allons utiliser TensorFlow 2 pour la création d'un modèle neuronal multicouche et exploiter les fondements du deep Learning supervisé dans la perspective d'automatiser l'analyse des images satellitaires.

CHAPITRE 4

Analyse d’images satellitaires par Tensor Flow

1. Introduction

En guise de créer notre classificateur intelligent pour les zones vertes sur des images satellitaires, nous avons choisi Colaboratory comme environnement de développement. Cet environnement incorpore le notebook Jupyter et nous permet de superviser le développement peu importe où on est via un accès cloud et internet. Dans ce chapitre on présentera les outils utilisés ainsi que la solution développée dans la perspective d'une supervision automatisée des espaces verts.

2. Jupyter

2.1 Définition

Jupyter est une application web utilisée pour programmer dans plus de 40 langages de programmation, dont Python, Julia, Ruby, R, ou encore Scala. C'est un projet communautaire dont l'objectif est de développer des logiciels libres, des formats ouverts et des services pour l'informatique interactive. Jupyter est une évolution du projet IPython. Jupyter permet de réaliser des calepins ou notebooks, c'est-à-dire des programmes contenant à la fois du texte en markdown et du code en Julia, Python, R... Ces calepins sont utilisés en science des données pour explorer et analyser des données. [15]

2.2 Historique

En 2014, Fernando Pérez annonce un projet dérivé d'IPython nommé Projet Jupyter. IPython continue d'exister comme interprète de commande Python et noyau pour Jupyter. Tandis que le notebook et d'autres composants d'IPython qui ne sont pas liées à un langage sont passés sous le nom de Jupyter. Le nom du projet Jupyter fait référence :

- aux trois principaux langages de programmation pris en charge par Jupyter, à savoir Julia, Python et R,
- aux carnets de notes de Galilée sur la découverte des lunes de Jupiter.

Le projet Jupyter développe et maintient les logiciels interactifs Jupyter Notebook, JupyterHub et JupyterLab. [15]

2.3 Logiciels

Jupyter utilise plusieurs logiciels qu'on cite :

- Jupyter Notebook
- JupyterHub
- JupyterLab
- Jupyter Book

2.4 Les environnements Jupyter

2.4.1 Online

- Binder
- Colaboratory
- Azure Notebooks

2.4.2 Windows

- Jupyter Portable
- Anaconda (Python distribution)

2.4.3 Linux

- Jupyter Lab
- Anaconda (Python distribution). [15]

Pour la création de notre classificateur on a utilisé Colaboratory pour faciliter l'accès immédiat au programme n'importe où nous somme.

3. Colaboratory

3.1 Définition

Colaboratory, ou plutôt appelé "Colab" est un service cloud partiellement gratuit de Google Research. Il permet à n'importe quelle personne d'écrire et d'exécuter un code Python de son choix tout en restant dans le navigateur. C'est un environnement particulièrement adapté au machine learning ou l'apprentissage automatique, et à l'analyse de données et le DATA SCIENCE, Colab est un service hébergé de notebooks Jupyter qui ne nécessite aucune configuration et permet d'accéder gratuitement à des ressources informatiques, dont des GPU. [16]

3.2 Avantages et limites

Les avantages de Colab sont très nombreuses dont on site :

- Aucune configuration requise.
- Accès gratuit aux GPU.
- Partage facile.

Cependant, Les ressources de Colab ne sont pas illimitées, et l'accès n'est pas garanti. De plus, les limites d'utilisation sont susceptibles de fluctuer. Ces contraintes sont nécessaires pour maintenir un accès gratuit aux ressources de Colab. [17]

3.3 Colab et l’apprentissage machine

Colab permet d’importer des données d’images, d'entraîner un classificateur de ces images importées et d’évaluer le modèle de ces derniers, tout cela avec quelques lignes de code. Les notebooks Colab exécutent ce code sur les serveurs Cloud de Google. Quelle que soit la puissance de votre ordinateur. Vous n’avez besoin que d’un navigateur. [16]

3.4 Applications et utilités

Colab est très largement utilisé par la communauté de la machine Learning, par exemple dans les applications suivantes :

- Premiers pas avec TensorFlow.
- Développement et entraînement de réseaux de neurones.
- Expérimentation avec les TPU.
- Dissémination de la recherche en IA.
- Création de tutoriels.

4. Programme apprentissage par Tensor Flow 2

Avant d’écrire le programme, on avait besoin d’une base de données pour l’apprentissage.

4.1 Base de données pour des images satellites

On a utilisé une base de données de plusieurs images satellitaires. La base de données comporte 10 classes et a été enregistrée dans Google Drive pour réaliser notre modèle réseau de neurones en exploitant TensorFlow 2.

Cette base de données est constituée de 27026 images qui sont classifiées et catégorisées dans le tableau suivant :

Catégorie	Catégorie en français	Taille de l'image
Annual crop	Récolte annuel	64 * 64 pixels
Forest	Forêt	
Herbaceous vegetation	Végétation herbacée	
Highway	AutoRoute	
Industrial	Zone industriel	
Pasture	Pâturage	
Permanent crop	Récolte permanentes	
Residential	Zone résidentiel	
River	Rivier	
Sea lake	Lac marin	

Tableau 4.1 : les noms des classes de la base de données.

4.2 Descriptions des étapes de développement du modèle

4.2.1 Chargement de la base de données sur Colab

Pour accéder aux images satellitaires de la base de données, on a besoin de charger les autorisations nécessaires, le chemin de la racine est alors : /content/gdrive/My Drive/2750

```
[ ]  from google.colab import drive

     drive.mount('/content/gdrive', force_remount=False)
     DataSet_path = 'gdrive/My Drive/2750'
```

Figure 4.1 : Chargement de la base de données.

Après exécution, le programme demande de récupérer un code d'accès à partir d'un URL donné :

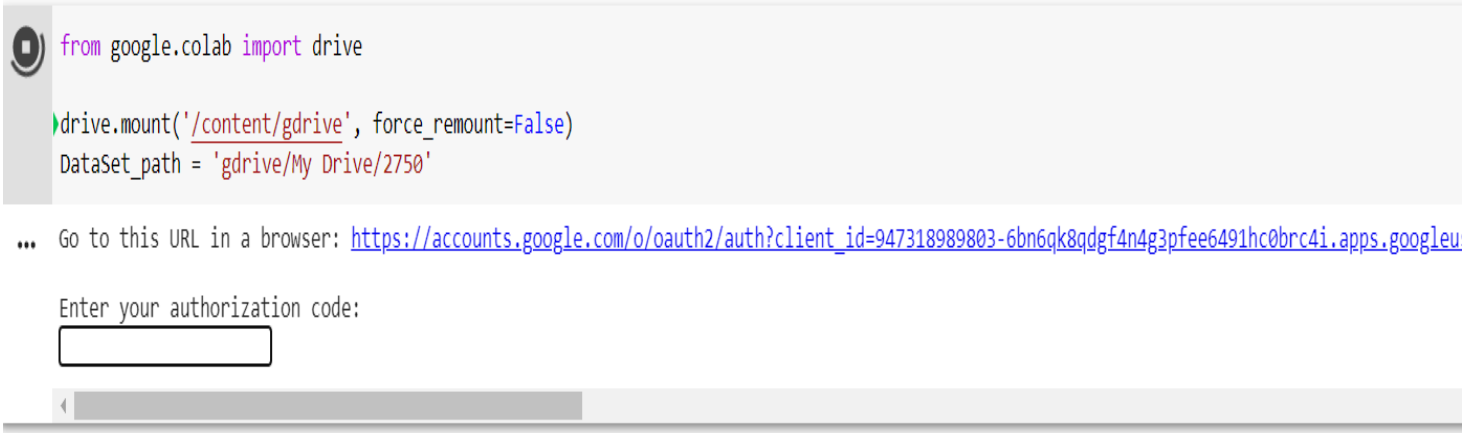

Figure 4.2 : Demande du code d'accès pour les autorisations drive.

Une fois avoir eu le code d'accès, l'exécution commence, et le chemin de la base de données est chargé dans la variable ***DataSet_Path***.

```
[ ]  from google.colab import drive

     drive.mount('/content/gdrive', force_remount=False)
     DataSet_path = 'gdrive/My Drive/2750'
```

```
     Mounted at /content/gdrive
```

Figure 4.3 : Résultat de l'exécution pour l'accès Drive.

4.2.2 Importation des bibliothèques

Pour créer un classificateur et notre solution intelligente qui nécessitait au début un apprentissage, il y a des bibliothèques qu'on doit importer pour assurer le fonctionnement du programme, après l'exécution les bibliothèques seront importées.

```
import matplotlib.pyplot as plt
import numpy as np
import os
import PIL
import tensorflow as tf

from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.models import Sequential
```

Figure 4.4 : Importation des bibliothèques.

L'exécution ne montre aucun résultat, les bibliothèques sont implémentées.

4.2.3 Affichage de la base de données

Cette section nous permet d'afficher la liste des classes et images avec son chemin.

```
from IPython.display import Image, display
import glob
i=1
for filename in glob.glob(DataSet_path+'/**/*.jpg'):
  print(str(i) +" : "+ filename )
  #display(Image(filename))
  i=i+1
```

Figure 4.5 : Affichage de la base de données.

Après exécution, le programme affiche une très longue liste qui inclut l'ensemble des images de la base de données, la figure suivante présente un échantillon de cette dernière :

```
24022 : gdrive/My Drive/2750/Forest/Forest_1604.jpg
24023 : gdrive/My Drive/2750/Forest/Forest_352.jpg
24024 : gdrive/My Drive/2750/Forest/Forest_1514.jpg
24025 : gdrive/My Drive/2750/Forest/Forest_824.jpg
24026 : gdrive/My Drive/2750/Forest/Forest_889.jpg
24027 : gdrive/My Drive/2750/HerbaceousVegetation/HerbaceousVegetation_1738.jpg
24028 : gdrive/My Drive/2750/HerbaceousVegetation/HerbaceousVegetation_107.jpg
24029 : gdrive/My Drive/2750/HerbaceousVegetation/HerbaceousVegetation_1482.jpg
24030 : gdrive/My Drive/2750/HerbaceousVegetation/HerbaceousVegetation_784.jpg
24031 : gdrive/My Drive/2750/HerbaceousVegetation/HerbaceousVegetation_210.jpg
24032 : gdrive/My Drive/2750/HerbaceousVegetation/HerbaceousVegetation_399.jpg
24033 : gdrive/My Drive/2750/HerbaceousVegetation/HerbaceousVegetation_1357.jpg
24034 : gdrive/My Drive/2750/HerbaceousVegetation/HerbaceousVegetation_2934.jpg
24035 : gdrive/My Drive/2750/HerbaceousVegetation/HerbaceousVegetation_755.jpg
24036 : gdrive/My Drive/2750/HerbaceousVegetation/HerbaceousVegetation_830.jpg
24037 : gdrive/My Drive/2750/HerbaceousVegetation/HerbaceousVegetation_913.jpg
24038 : gdrive/My Drive/2750/HerbaceousVegetation/HerbaceousVegetation_2948.jpg
24039 : gdrive/My Drive/2750/HerbaceousVegetation/HerbaceousVegetation_2410.jpg
24040 : gdrive/My Drive/2750/HerbaceousVegetation/HerbaceousVegetation_100.jpg
24041 : gdrive/My Drive/2750/HerbaceousVegetation/HerbaceousVegetation_1116.jpg
```

Figure 4.6 : Résultat de l'affichage de données.

4.2.4 Apprentissage

Cette étape nous permet de diviser notre base de données en deux parties, une partie pour l'apprentissage qui va prendre 60% de la base originale et les 40% qui restent dédiés à la validation.

```
[ ]  #Base de donnée pour apprentissage
     train_ds = tf.keras.preprocessing.image_dataset_from_directory(
       DataSet_path,
       validation_split=0.6,
       subset="training",
       seed=123,
       image_size=(64, 64),
       batch_size=32)

     class_names = train_ds.class_names
     print(class_names)


     #base de données pour validation
     val_ds = tf.keras.preprocessing.image_dataset_from_directory(
       DataSet_path,
       validation_split=0.4,
       subset="validation",
       seed=123,
       image_size=(64, 64),
       batch_size=32)

     class_names2 = val_ds.class_names
     print(class_names)
```

Figure 4.7 : Apprentissage

Après exécution :

```
Found 27026 files belonging to 10 classes.
Using 10811 files for training.
['AnnualCrop', 'Forest', 'HerbaceousVegetation', 'Highway', 'Industrial', 'Pasture', 'PermanentCrop', 'Residential', 'River', 'SeaLake']
Found 27026 files belonging to 10 classes.
Using 10810 files for validation.
['AnnualCrop', 'Forest', 'HerbaceousVegetation', 'Highway', 'Industrial', 'Pasture', 'PermanentCrop', 'Residential', 'River', 'SeaLake']
```

Figure 4.8 : Résultat de l'apprentissage.

4.2.5 visualisation des données

Pour visualiser les 25 premières images de l'ensemble de données d'entraînement tout en précisant leur classes on utilise les commandes suivantes :

```
plt.figure(figsize=(10,10))
for images, labels in train_ds.take(1):
 for i in range(25):
    plt.subplot(5,5,i+1)
    plt.xticks([])
    plt.yticks([])
    plt.grid(False)
    plt.imshow(images[i].numpy().astype("uint8"))
    plt.title(class_names[labels[i]])
plt.show()
```

Figure 4.9 : Visualiser quelques données.

Après exécution :

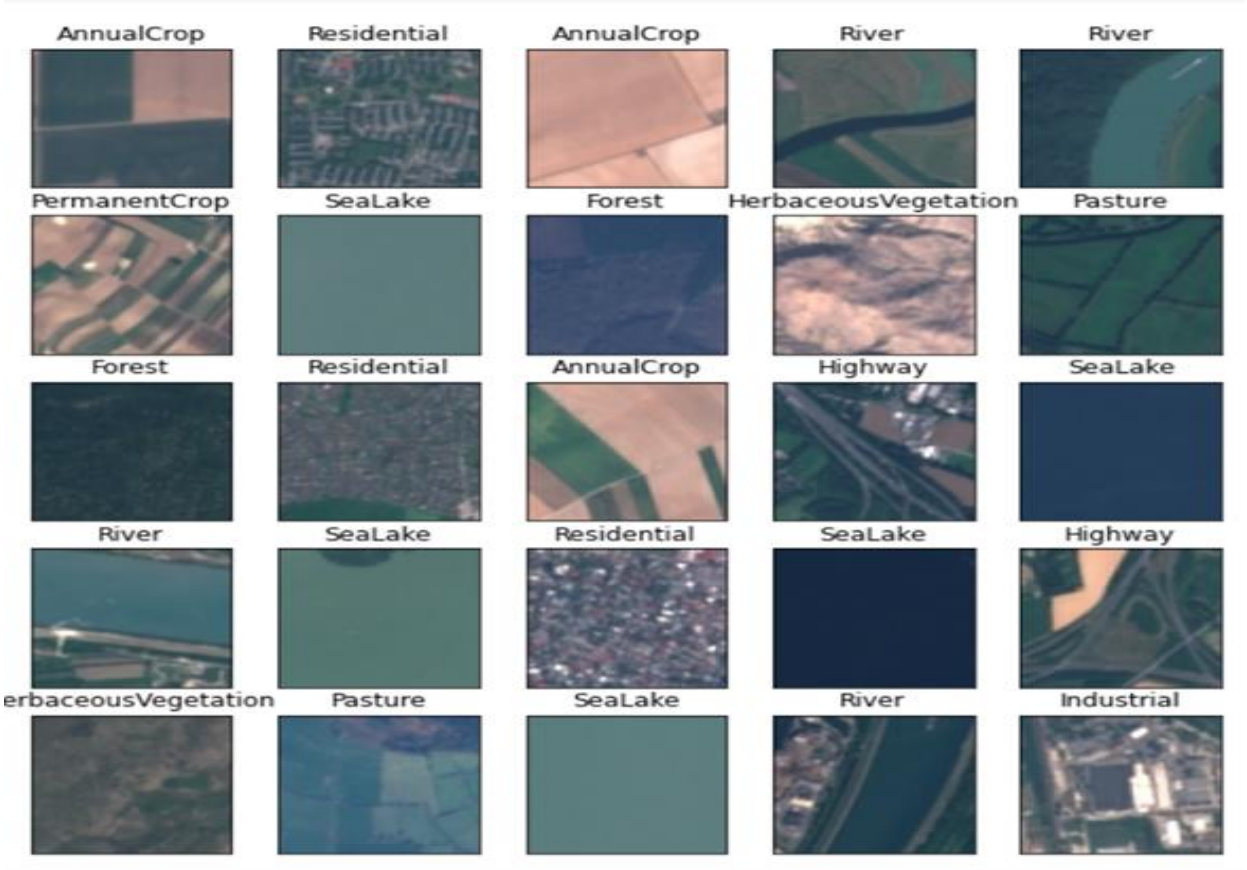

Figure 4.10 : Résultat de visualisation.

4.2.6 Configuration de l'ensemble de données pour les performances

Pour éviter le blocage des entrées / sorties on utilise les commandes suivantes qui consiste en l'utilisation de la mémoire cache.

```
AUTOTUNE = tf.data.AUTOTUNE

train_ds = train_ds.cache().shuffle(1000).prefetch(buffer_size=AUTOTUNE)
val_ds = val_ds.cache().prefetch(buffer_size=AUTOTUNE)
```

Figure 4.11 : Configuration de l'ensemble de données pour les performances

L'exécution de ce script n'est pas visible.

4.2.7 Standardisation des données

Les images RVB sont codées dans une plage de [0, 255], ceci n'est pas adapté pour un réseau neuronal ce qui nous exige une normalisation des données vers une plage de [0, 1] en utilisant un calque de redimensionnement.

```
normalization_layer = layers.experimental.preprocessing.Rescaling(1./255)

normalized_ds = train_ds.map(lambda x, y: (normalization_layer(x), y))
image_batch, labels_batch = next(iter(normalized_ds))
first_image = image_batch[0]
# Notice the pixels values are now in `[0,1]`.
print(np.min(first_image), np.max(first_image))
```

Figure 4.12 : Standardisation des données

L'Exécution de cette étape affiche la valeur maximale et minimale des pixels de la première image qui sont effectivement comprises entre [0, 1].

4.2.8 Création du modèle

Pour créer notre modèle on a décidé d'utiliser 128 unités et une fonction d'activation, dans notre cas c'était la fonction RELU.

Rectified Linear Unit (ReLU) c'est la fonction d'activation la plus utilisée et la plus simple. Cette fonction nous permet d'effectuer un filtrage sur nos données qui laisse passer les valeurs positives (x > 0) dans les couches choisies du réseau de neurones.

```
model = tf.keras.models.Sequential([
  tf.keras.layers.Flatten(input_shape=(64, 64, 3)),
  tf.keras.layers.Dense(128, activation='relu'),
  tf.keras.layers.Dropout(0.2),
  tf.keras.layers.Dense(10)
])
```

Figure 4.13 : Création du modèle.

Les commandes suivantes compilent le modèle créé :

```
model.compile(optimizer='adam',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])
```

Figure 4.14 : Compilation du modèle.

Cette partie donne le sommaire du modèle créé :

```
model.summary()
```

Figure 4.15 : Afficher le sommaire du modèle.

Après exécution, un tableau s'affiche qui résume les informations du modèle créé :

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
flatten_1 (Flatten)          (None, 12288)             0
_________________________________________________________________
dense_2 (Dense)              (None, 128)               1572992
_________________________________________________________________
dropout_1 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_3 (Dense)              (None, 10)                1290
=================================================================
Total params: 1,574,282
Trainable params: 1,574,282
Non-trainable params: 0
```

Figure 4.16 : Résumé du modèle.

Les lignes suivantes aident à faire l'entraînement (apprentissage) du modèle répété 10 fois epochs=10.

```
#phase d'apprentissage
epochs=10
history = model.fit(
  train_ds,
  validation_data=val_ds,
  epochs=epochs
)
```

Figure 4.17 : Phase d'apprentissage

Après exécution, le résultat de l'entraînement s'affiche.

```
Epoch 1/10
338/338 [==============================] - 243s 720ms/step - loss: 44.6328 - accuracy: 0.1115 - val_loss: 2.2989 - val_accuracy: 0.1087
Epoch 2/10
338/338 [==============================] - 5s 14ms/step - loss: 2.2969 - accuracy: 0.1132 - val_loss: 2.2962 - val_accuracy: 0.1087
Epoch 3/10
338/338 [==============================] - 5s 14ms/step - loss: 2.2949 - accuracy: 0.1144 - val_loss: 2.2956 - val_accuracy: 0.1087
Epoch 4/10
338/338 [==============================] - 5s 14ms/step - loss: 2.2944 - accuracy: 0.1139 - val_loss: 2.2954 - val_accuracy: 0.1087
Epoch 5/10
338/338 [==============================] - 5s 14ms/step - loss: 2.2942 - accuracy: 0.1144 - val_loss: 2.2954 - val_accuracy: 0.1087
Epoch 6/10
338/338 [==============================] - 5s 14ms/step - loss: 2.2942 - accuracy: 0.1144 - val_loss: 2.2953 - val_accuracy: 0.1087
Epoch 7/10
338/338 [==============================] - 5s 14ms/step - loss: 2.2942 - accuracy: 0.1144 - val_loss: 2.2954 - val_accuracy: 0.1087
Epoch 8/10
338/338 [==============================] - 5s 14ms/step - loss: 2.2942 - accuracy: 0.1144 - val_loss: 2.2954 - val_accuracy: 0.1087
Epoch 9/10
338/338 [==============================] - 5s 14ms/step - loss: 2.2942 - accuracy: 0.1130 - val_loss: 2.2954 - val_accuracy: 0.1087
Epoch 10/10
338/338 [==============================] - 5s 14ms/step - loss: 2.2942 - accuracy: 0.1144 - val_loss: 2.2954 - val_accuracy: 0.1087
```

Figure 4.18 : Résultat de l'entraînement.

- Nombre d'images

Il existe une commande qui permet de connaître le nombre d'images contenues dans notre base de données :

```
image_count = len(list(glob.glob(DataSet_path+'/**/*.jpg')))
print(image_count)
```

Figure 4.19 : Connaître le nombre d'image

Après exécution le programme affiche qu'on a 27 026 images.

```
image_count = len(list(glob.glob(DataSet_path+'/**/*.jpg')))
print(image_count)

27026
```

Figure 4.20 : Nombre d'images dans la base de données.

4.2.9 Visualisation des résultats de l'entraînement

Pour visualiser la précision et les pertes des données de notre modèle on se sert des lignes suivantes :

```
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']

loss = history.history['loss']
val_loss = history.history['val_loss']

epochs_range = range(epochs)

plt.figure(figsize=(8, 8))
plt.subplot(1, 2, 1)
plt.plot(epochs_range, acc, label='Training Accuracy')
plt.plot(epochs_range, val_acc, label='Validation Accuracy')
plt.legend(loc='lower right')
plt.title('Training and Validation Accuracy')

plt.subplot(1, 2, 2)
plt.plot(epochs_range, loss, label='Training Loss')
plt.plot(epochs_range, val_loss, label='Validation Loss')
plt.legend(loc='upper right')
plt.title('Training and Validation Loss')
plt.show()
```

Figure 4.21 : Visualisation des résultats de l'entraînement

L'exécution affiche deux figures l'une présente la précision de l'apprentissage et de la validation du modèle, et l'autre présente les pertes :

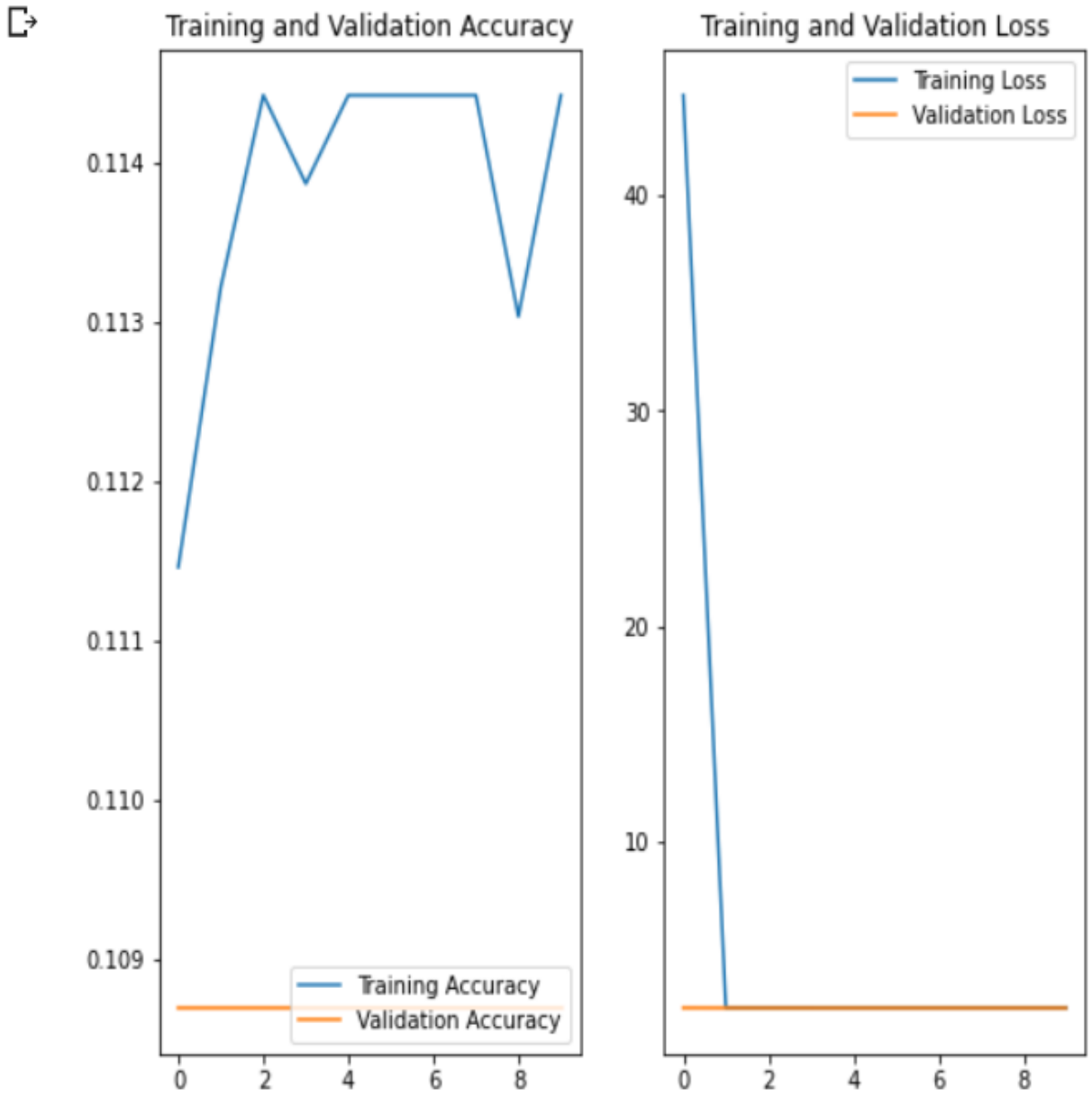

Figure 4.22 : Résultat des performances de l'entraînement.

La précision de l'ensemble d'entraînement augmente lentement jusqu'à presque 0.2 (20%) pendant les 2 premières époques. Cependant, la précision sur l'ensemble de validation reste stable à environ 0.104 tout au long de l'entrainement. Ce qui concerne la perte du jeu de l'entrainement, Il diminue dès la première époque et s'aligne ; tandis la perte du jeu de validation ne change pas durant le processus de l'entrainement.

4.2.10 Surapprentissage

Il existe plusieurs façons de lutter contre le sur-ajustement, on va utiliser l'augmentation de données et dropout.

❖ Augmentation de données

L'augmentation des données se fait en générant des données d'entraînement supplémentaires à partir des exemples existants en les augmentant à l'aide de transformations aléatoires qui produisent des images d'apparence crédible. Cela permet d'exposer le modèle à davantage d'aspects des données.

```
[ ]  data_augmentation = keras.Sequential(
       [
         layers.experimental.preprocessing.RandomFlip("horizontal",
                                                      input_shape=(64,
                                                                   64,
                                                                   3)),
         layers.experimental.preprocessing.RandomRotation(0.1),
         layers.experimental.preprocessing.RandomZoom(0.1),
       ]
     )
```

Figure 4.23 : Méthode de l'augmentation des données.

Cette commande n'affiche aucun résultat.

Pour visualiser le résultat de la commande précédente :

```
[ ]  plt.figure(figsize=(10, 10))
     for images, _ in train_ds.take(1):
       for i in range(9):
         augmented_images = data_augmentation(images)
         ax = plt.subplot(3, 3, i + 1)
         plt.imshow(augmented_images[0].numpy().astype("uint8"))
         plt.axis("off")
```

Figure 4.24 : Visualisation du résultat de l'augmentation de données.

Après exécution :

Figure 4.25 : Résultat de la visualisation

❖ Dropout

1. Définition de Dropout

Le terme Dropout veut dire l'abandon, il désigne le décrochage des unités (cachées et visibles) dans le réseau neuronal. En termes simples, le décrochage se réfère à ignorer les unités (c.-à-d. les neurones) pendant la phase d'entraînement de certains ensembles de neurones qui est choisie au hasard. Par « ignorer », on veut dire que ces unités ne sont pas prises en compte lors d'une passe en avant ou en arrière.

2. L'intérêt d'utiliser Dropout

Une couche entièrement connectée occupe la plupart des paramètres et, par conséquent, les neurones développent une co-dépendance les uns avec les autres pendant l'entraînement, ce qui limite la puissance individuelle de chaque neurone conduisant à un suréquipement des données d'entraînement.

On crée alors un autre modèle comme suit :

```
num_classes=10
model = Sequential([
  data_augmentation,
  layers.experimental.preprocessing.Rescaling(1./255),
  layers.Conv2D(16, 3, padding='same', activation='relu'),
  layers.MaxPooling2D(),
  layers.Conv2D(32, 3, padding='same', activation='relu'),
  layers.MaxPooling2D(),
  layers.Conv2D(64, 3, padding='same', activation='relu'),
  layers.MaxPooling2D(),
  layers.Dropout(0.2),
  layers.Flatten(),
  layers.Dense(128, activation='relu'),
  layers.Dense(num_classes)
])
```

```
[ ] model.compile(optimizer='adam',
                  loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
                  metrics=['accuracy'])

    model.summary()
```

Figure 4.26 : Méthode de dropout.

Après exécution, le modèle sera créé et l'entraînement fait.

On visualise alors les résultats de cet entraînement.

```
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']

loss = history.history['loss']
val_loss = history.history['val_loss']

epochs_range = range(epochs)

plt.figure(figsize=(8, 8))
plt.subplot(1, 2, 1)
plt.plot(epochs_range, acc, label='Training Accuracy')
plt.plot(epochs_range, val_acc, label='Validation Accuracy')
plt.legend(loc='lower right')
plt.title('Training and Validation Accuracy')

plt.subplot(1, 2, 2)
plt.plot(epochs_range, loss, label='Training Loss')
plt.plot(epochs_range, val_loss, label='Validation Loss')
plt.legend(loc='upper right')
plt.title('Training and Validation Loss')
plt.show()
```

Figure 4.27 : Résultats de l'entraînement du nouveau modèle.

Après exécution :

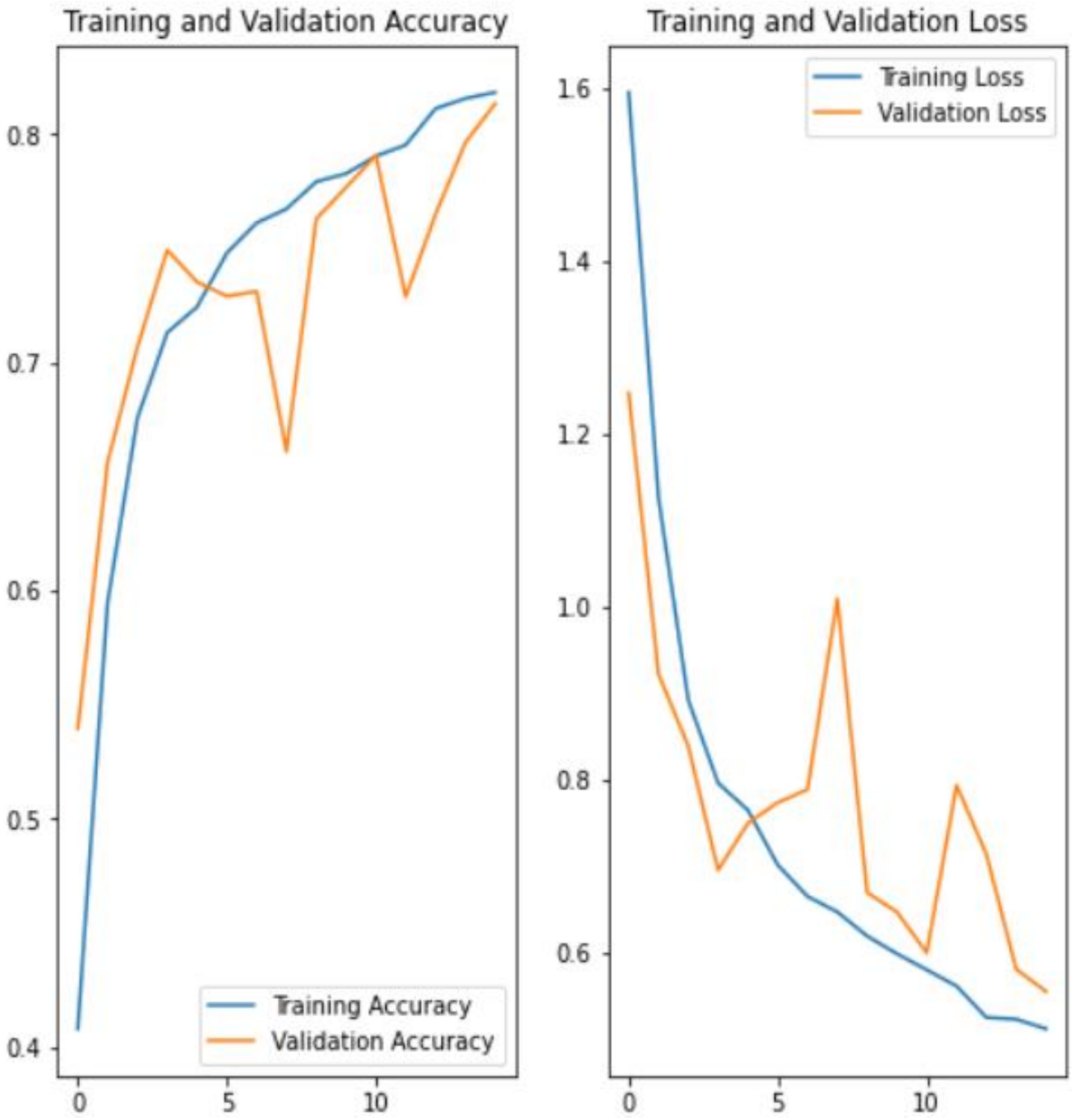

Figure 4.28 : Résultat de performance du nouveau modèle.

4.2.11 Prédire sur de nouvelles données

```
imsat_url = "imsat_url = ""
imsat_path = tf.keras.utils.get_file('image_satellitaire', origin=imsat_url)

img = keras.preprocessing.image.load_img(
    imsat_path, target_size=(64, 64)
)
img_array = keras.preprocessing.image.img_to_array(img)
img_array = tf.expand_dims(img_array, 0) # Create a batch

predictions = model.predict(img_array)
score = tf.nn.softmax(predictions[0])

print(
    "This image most likely belongs to {} with a {:.2f} percent confidence."
    .format(class_names[np.argmax(score)], 100 * np.max(score))
)"
imsat_path = tf.keras.utils.get_file('image_satellitaire', origin=imsat_url)

img = keras.preprocessing.image.load_img(
    sunflower_path, target_size=(64, 64)
)
img_array = keras.preprocessing.image.img_to_array(img)
img_array = tf.expand_dims(img_array, 0) # Create a batch

predictions = model.predict(img_array)
score = tf.nn.softmax(predictions[0])

print(
    "This image most likely belongs to {} with a {:.2f} percent confidence."
    .format(class_names[np.argmax(score)], 100 * np.max(score))
)
```

Figure 4.29 : Prédiction sur de nouvelles données.

5. CONCLUSION

Ce chapitre a présenté, en premier lieu, un aperçu assez détaillé sur les outils de programmation utilisés, Jupyter et Colab, tout en donnant leurs définitions, historiques et utilités. Par la suite, on a décrit chaque étape de notre programme, qu'on a exécutée afin de les comprendre et avoir les résultats souhaités.

CONCLUSION GENERALE

Ce mémoire avait pour ambition de réaliser une surveillance au fil du temps d'un écosystème vert choisi. Pour aboutir à cet objectif une série d'étapes a été effectuée :

Tout d'abord on a introduit la notion d'imagerie satellitaire qui a agi comme source de photos pour notre projet tout en expliquant l'intérêt de l'utilisation du SAR dans l'imagerie satellitaire et ses applications dans la télédétection.

En second lieu, on a assuré à définir globalement l'intelligence artificielle et ses aspects vue son importance tout en utilisant des outils de développement tels que : TensorFlow, Colab, Jupyter.

Après la récolte d'informations, on a commencé l'implémentation de notre classificateur d'image sous Colab qui à son tour a donné la classe de l'image entrée d'une zone choisie pour faire sa comparaison avec d'autres images de la même zone au fil du temps.

Pour conclure notre travail, on peut dire qu'on a atteint l'objectif souhaité, qui consiste à créer un modèle neuronal via les outils de l'intelligence artificielle qui peut être exploité pour la surveillance d'écosystèmes verts (dégradation ou réhabilitation). Ce projet peut être développé avec un complément de base de données pour être plus précis et pour surveiller davantage de paramètres et plus de détails, aussi il possède d'innombrables perspectives et dimensions.

BIBLIOGRAPHIE

[1] https://fr.wikipedia.org/wiki/Imagerie_satellite (consulté le 17 Avril 2021)

[2] https://www.applisat.fr/generalites-satellites/imagerie-satellitaire-radar-exemples-applications (consulté le 17 Avril 2021)

[3] https://www.universalis.fr/encyclopedie/teledetection/ (consulté le 30 Avril 2021)

[4] https://fr.wikipedia.org/wiki/Radar_%C3%A0_synth%C3%A8se_d%27ouverture (consulté le 1 Avril 2021)

[5] Meyer, Franz. «Radar spatial à synthèse d'ouverture : principes, accès aux données et techniques de traitement de base», dans le manuel de SAR : Méthodologies complètes pour la surveillance forestière et estimation de la biomasse. Lieu d'édition : Éditeur. (Avril 2019)

[6] https://sentinels.copernicus.eu/web/sentinel/user-guides/sentinel-1-sar/acquisition-modes/stripmap (consulté le 9 Juin 2021)

[7] https://www.radartutorial.eu/20.airborne/ab08.en.html (consulté le 9 Juin 2021)

[8] https://tel.archives-ouvertes.fr/tel-01082960/document (consulté le 4 Mai 2021)

[9] https://fr.wikipedia.org/wiki/Intelligence_artificielle (consulté le 19 Mai 2021)

[10] : https://inventiv-it.fr/intelligence-artificielle-forte-de-quoi-sagit-il/ (consulté le 21 Mai 2021)

[11] https://inventiv-it.fr/intelligence-artificielle-faible-de-quoi-sagit-il/ (consulté le 21 Mai2021)

[12] https://fr.wikipedia.org/wiki/Apprentissage_automatique (consulté le 22 Mai2021)

[13]https://fr.wikipedia.org/wiki/R%C3%A9seau_de_neurones_artificiels#Mode_supervis%C3%A9_ou_non (consulté le 22 Mai 2021)

[14] https://fr.wikipedia.org/wiki/TensorFlow (consulté le 25 Mai 2021)

[15]https://fr.wikipedia.org/wiki/Jupyter (consulté le 25 Mai 2021)

[16] https://research.google.com/colaboratory/faq.html?hl=fr (consulté le 25 Mai 2021)

[17]https://colab.research.google.com/notebooks/intro.ipynb?utm_source=scs - (consulté le 25 Mai 2021)