République Algérienne Démocratique et Populaire
Ministère de L'enseignement Supérieur et de la Recherche Scientifique
Université Saad Dahleb Blida
Faculté des Sciences et Technologie
Département d'aéronautiques et des études spatiales

Mémoire de Fin de cycle

En vue de l'obtention du diplôme Master recherche en aéronautique
Option : télécommunications spatiales

Thème

Détection des drones à l'aide des réseaux de neurones artificiels.

Réalisé par
Melle DJABRI Nour El Houda & Mme HAMLAT Souad EPS.ZAIDI

Promotrice :

Dr Nawal CHEGGAGA

Soutenue publiquement devant la commission d'examen composée de :

Dr. Lila MOUFFOK	Professeur	Président
Dr. Sofiane TAHRAOUI	Professeur	Examinateur

Année Universitaire 2019/2020

Remerciements

*Je tiens tout d'abord à remercier sincèrement notre promotrice, Mme **Cheggaga Nawal**, pour nous avoir proposé ce projet passionnant. Je lui suis également très reconnaissante pour son engagement et sa disponibilité tout au long de ce travail. Son soutien et ses conseils avisés lors de nos discussions m'ont non seulement été d'une très grande aide, mais ont également été une importante source d'inspiration.*

*Mes sincères remerciements à toute ma famille, particulièrement ma mère **Fatiha**, mon père **Mohamed**, toutes mes sœurs **Asmaa**, **Kawther** et **Kounouz**, mon frère **Gholem** ainsi que mon cousin et mes deux neveux **Allaa eddine** et **Hodheifa**. Je vous remercie du fond du cœur pour votre compréhension, votre confiance et votre soutien.*

Très reconnaissante envers ceux qui m'ont appuyé et encouragé à effectuer ce mémoire, je tiens à remercier tous ceux qui ont contribué à ce travail de recherche, qu'ils trouvent ici l'expression de toute ma reconnaissance.

*Je tiens également à remercier une personne chère à mon cœur. Qui m'a toujours encouragé et aidé à mener à bien ce projet de fin d'études. Peu importe le nombre de fois que je dois te remercier, je ne pourrai jamais assez te remercier pour tout ce que tu as fait pour moi. Merci du fond du cœur, **Ammar**, je n'oublierai jamais ta présence à mes côtés, tes efforts et ta volonté de m'aider à tout prix, merci pour ta gentillesse et ta générosité.*

Nour El Houda

Remerciements

Je tiens à remercier vivement tout le corps enseignant et plus particulièrement notre chère promotrice Mme CHEGGAGA Nawal pour son encadrement particulier, sa patience et sa disponibilité, je lui exprime ma profonde gratitude pour son aide précieuse, ses judicieux conseils et ses suggestions avisées qui ont amélioré ma réflexion et aidé à mener à bien ce présent travail.

Je tiens à adresser mes plus vifs remerciements aux membres du jury, pour avoir accepté d'évaluer ce travail.

Je remercie mon cher khelifa pour son soutien, son amour et sa compréhension qui m'ont toujours donné du courage, je lui exprime mon grand amour et ma gratitude pour son encouragement et sa patience durant tout ce temps.

Mes vifs remerciements pour mes chers parents qui ont toujours cru en moi et mes capacités, qui voient aujourd'hui le fruit de plusieurs années de sacrifices et de prières, je leurs exprimes mes sentiments les plus profonds pour leur confiance et leur soutien.

Mes remerciements sincères vont également à mes chers frères qui ont été toujours là pour moi, à ma belle-famille et surtout ma belle-mère pour sa compréhension et son affection.

Je tiens à remercier mes amis et toute personne qui a contribué au bon déroulement de ce travail.

Je dédie ce modeste travail pour

Mon cher fils Yastene,

Mon époux et mes parents.

Souad

Résumé

L'intelligence artificielle a simplifié tant de tâches dans notre vie quotidienne, et a révolutionné plusieurs domaines, tels que la médecine, la robotique, l'aérospatial, etc. Dans notre présent travail. Nous avons utilisé une intelligence artificiel basée sur les réseaux de neurones artificiels pour la détection des drones, nous avons pour cela, opté pour le logiciel Python : Anaconda3, pour les bonnes performances qu'il montre dans le domaine de la reconnaissance et la classification des images. Nous avons essayé plusieurs architectures de réseau de neurone artificiel pour détecter et classer des images en deux classes : drone ou non drone. Nous avons d'abord testé les performances des modèles VGG à différents blocs, puis nous avons utilisé quelques méthodes de régularisation ainsi que l'apprentissage par transfert, sur trois différentes bases de données. Les résultats obtenus montrent que l'apprentissage par transfert où nous avons enregistré le modèle VGG-16 est le programme le plus performant pour la classification des images.

***Mots clés** : Intelligence Artificielle, drones, python Anaconda3, VGG-16,* l'apprentissage par transfert*, réseaux de neurones.*

Abstract

Artificial Intelligence has made our daily life easier, and has revolutionized several fields, such as medicine, robotics, aerospace, etc.

In our present work, we used an artificial intelligence based on artificial neural networks for the detection of drones, for this we opted for the Python software: Anaconda3, for the good performance it shows in the field of recognition and classification of images.

We have tried several artificial neural network architectures to detect and classify images into two classes: drone or non-drone. We first tested the performance of the VGG models in different blocks, then we used some regularization methods as well as transfer learning, on three different databases. The results obtained show that transfer learning based on VGG-16 is the most efficient model for classification of images.

***Key words:** Artificial Intelligence, drones, python Anaconda3, VGG-16, transfer learning, neural networks.*

ملخص

لقد بسط الذكاء الاصطناعي الكثير من المهام في حياتنا اليومية حيث أحدث ثورة في العديد من المجالات: مثل الطب والروبوتات والفضاء وما إلى ذلك. في عملنا الحالي استخدمنا ذكاءً اصطناعيًا قائمًا على الشبكات العصبية الاصطناعية لاكتشاف الطائرات بدون طيار، لذلك اخترنا برنامج بايثون: أناكوندا *3* وهذا لأدائه الجيد في مجال التعرف على الصور وتصنيفها. لقد جربنا العديد من الشبكات العصبية الاصطناعية لاكتشاف الصور وتصنيفها إلى فئتين: الطائرات بدون طيار والاجسام الطائرة الغير معروفة. اختبرنا أولاً أداء نماذج *VGG* على طبقات مختلفة ثم استخدمنا بعض طرق التنظيم وكذلك نموذج نقل التعلم على ثلاث قواعد بيانات مختلفة. تظهر النتائج المتحصل عليها أن نموذج نقل التعلم المبني على نموذج *VGG-16* ثلاث طبقات هو النموذج الأكثر كفاءة لتصنيف الصور.

الكلمات المفتاحية: الذكاء الاصطناعي، طائرات بدون طيار، بايثون: أناكوندا3، *VGG-16*، نقل التعلم، الشبكات العصبية.

Table de matières :

Liste des figures :

Les abréviations :

API : Interface de Programmation Applicative.

ADALINE : ADAptive LInear NEuron.

BEOS : BE Operating System.

CNN : Convolutionnal Neural Networks.

CNRI : Corporation for National Research Initiatives.

CGI : Conseillers en Gestion et Informatique.

CPU : Central Processing Unit.

CSV : Comma-Separated Values.

DGAC : Direction Générale de l'Aviation Civile.

2D : Deux Dimensions.

FC : Fully Connected layer.

FPS : Frames per second.

FTP : File Transfer Protocol.

GPU : Graphical Processing Unit.

GHZ : GigaHertZ.

GO : GigaOctet.

HALE : Haute Altitude Longue Endurance.

HOG : Histogramme des Gradients Orientés.

HTML : HyperText Markup Language.

HD : Haute Définition.

IA : Intelligence Artificielle.

IJCAI : International Joint Conferences on Artificial Intelligence.

IDE : Integrated Development Environment.

MALE : Moyenne Altitude Longue Endurance.

MAC OS : Macintosh Operating System.

MS-DOS : Microsoft Disk Operating System.

MB : Megabyte.

Ms : Millisecondes.

MAP : Mean Average Precision.

ONEIROS : Open-ended Neuro-Electronic Intelligent Robot Operating System.

OS : Operating System.

OVNI : Objet Volant Non Identified.

PSF : Python Software Foundation.

PMC : Perceptron Multi Couche.

PHP : Hypertext Preprocessor.

RDF : la Reconnaissance Des Formes.

RF : Radio Fréquences.

RNN : Reccurent Neural Networks.

RNA : Réseau de Neurone Artificiel.

RVB : Rouge, Vert et Bleu.

ReLU : Rectified Linear Unit.

RAM : Random Access Memory.

SVM : Support Vector Machine.

SSD : Single Shot Multibox Detector.

TCP : Très Courte Portée.

UAVs : Unmanned Aerial Vehicle.

UCAV: Unmanned Combat Aerial Vehicle.

VP : Vrai Positif.

VN : Vrai Négatif.

VGG : Visual Geometry Group.

XML : eXtended Markup Language.

Liste des tableaux :

Introduction générale

Parmi les inventions qui ont marqué ces dernières années l'apparition des drones, également appelés Unamed Aerial Vehicle en anglais. Ce sont de petits appareils volants pilotés à distance, ils étaient à leurs débuts destinés principalement pour les missions militaires telles que l'observation et la surveillance des zones de conflit, l'obtention d'informations, le positionnement électromagnétique, l'identification de cibles, etc. Cependant, ces quelques dernières années, le domaine des drones civils a connu une très grande évolution. Il peut même constituer un nouveau secteur industriel dynamique et créer des opportunités de croissance et d'emploi s'il répond à la règlementation indiquée par la DGAC (Direction générale de l'Aviation civile) concernant leur pilotage et leur utilisation.

Ces petits aéronefs présentent plusieurs avantages, tels que la prise des vues panoramiques dans le monde du cinéma, la surveillance des terres agricoles, en agriculture, la prévision météorologique, l'atteinte des zones jugées d'accès difficile et dangereux pour la collection des données, dans le domaine de la recherche scientifique ou encore, le transport des organes, en médecine, notamment aux États-Unis dans le Maryland, où a eu lieu le premier transport d'organe, pour une greffe, réduisant ainsi considérablement le temps de livraison et les coûts de transport. En revanche, et comme toute technologie, cet objet représente des inconvénients lorsqu'il est piloté de manière irresponsable, il peut directement ou indirectement porter atteinte à la sécurité des individus ou bien des états tel que le survol d'aéroport ou des sites sensibles et critiques comme la maison blanche, le bureau du Premier Ministre Japonais, la centrale nucléaire française, le Golden Gate, la prison à haute surveillance de la Colombie-Britannique, mais également la surveillance de la vie privée des gens comme les personnalités politiques et les célébrités. C'est la raison pour laquelle la détection des drones est devenue un véritable défi et il est donc nécessaire de développer des solutions puissantes, fiables et peu coûteuses pour les détecter. Plusieurs techniques existent déjà comme les capteurs acoustiques, l'utilisation des radars qui montrent des performances limitées en raison d'un très faible signal radar réfléchi des plus petits drones, l'interception de communication radiofréquence RF (Radio Fréquence) est également utilisée, mais de nouveaux drones sont programmés sans communication [1].

Dans cette présente étude, nous proposons une nouvelle méthode pour la détection des drones mini / micro de toutes formes à l'aide des réseaux de neurones artificiels et plus exactement le Deep Learning qui vient révolutionner le monde. Tout d'abord, une caméra est utilisée pour la collecte des données statistique et dynamique qui sont directement transférées sur un ordinateur équipé du logiciel Python : Anaconda3. Elles sont ensuite traitées à l'aide d'un algorithme de traitement d'image, puis, en utilisant un algorithme à base des réseaux de neurones convolutionnels, les images obtenues sont comparées à celles déjà existantes sur notre base de données. Une décision est alors prise pour valider l'apparition ou non du drone dans le périmètre du capteur visuel.

Ce travail fera l'objet de trois chapitres :

- Le premier chapitre est consacré à la théorie des réseaux de neurones artificiels RNA. Après avoir donné un aperçu sur les notions de base de l'intelligence artificielle et de l'historique

des RNA, nous parlerons du neurone biologique et de sa modélisation au neurone formel. Nous donnerons par la suite, les différents éléments de base de cet outil mathématique ainsi que quelques aspects qui nous seront utiles pour le reste de notre travail. Nous finirons par citer les différents travaux utilisant la technologie des réseaux de neurones artificiels.

- Le deuxième chapitre est dédié à la détection d'image à l'aide des réseaux de neurones convolutionnels. Nous présenterons tout d'abord quelques généralités sur les drones et les images ainsi que la procédure envisagée pour la détection des objets d'une image. Nous détaillerons ensuite le concept des CNN (Convolution Neural Network) et les VGG-16-. Nous conclurons enfin par des méthodes d'amélioration des performances de ces derniers.

- Dans le troisième et dernier chapitre, nous implémenterons notre programme sur le logiciel Python en utilisant plusieurs architectures basées sur la technologie des CNN. Chaque résultat obtenu sera suivi par des observations et d'une analyse.

Chapitre 1

Les Réseaux
De
Neurones Artificiels

Introduction :

Le cerveau humain est capable d'apprendre et de réaliser des raisonnements complexes, il est constitué d'un très grand nombre de neurones. C'est la raison pour laquelle l'humain a voulu développer des modèles de réseaux de neurones artificiels pour des objectifs bien précis tels que la modélisation et la compréhension du fonctionnement du cerveau et même pour réaliser des architectures et des algorithmes basés sur l'intelligence artificielle.

Le réseau de neurones artificiels (RNA) est l'une des technologies les plus récentes à utiliser la reconnaissance et la classification d'objets. Il utilise la modélisation mathématique conventionnelle afin de pouvoir modéliser et approximer des systèmes assez difficiles et complexes. Cependant, malgré tous les progrès importants réalisés dans la classification à l'aide de réseaux de neurones, divers problèmes liés aux applications de ces derniers n'ont pas été entièrement résolus [2].

De façon générale, ce qui nous intéresse est le fait qu'un réseau de neurones permet la détection et l'identification des drones grâce à une analyse en temps réel d'images prises sur site.

Dans ce chapitre, nous présenterons les réseaux de neurones. Tout d'abord, nous commencerons par une explication simple des différentes technologies d'apprentissage automatique. Nous décrirons ensuite l'historique des RNA et expliquerons pourquoi nous utiliserons ce réseau. Nous présenterons aussi les notions de base d'un réseau de neurones, les fonctions d'activation et les différentes architectures de ce réseau. Nous continuerons en expliquant les différents types d'apprentissage (supervisé et non supervisé), ainsi que ses types. Nous citerons également les différents domaines d'applications de l'intelligence artificielle, basée notamment sur des applications des réseaux de neurones ainsi que leurs usages. Enfin, nous terminerons en notant quelques avantages et inconvénients des RNA et nous citerons les procédures pour créer ce réseau.

1.1 Le maching learning :

L'apprentissage automatique est une technologie d'intelligence artificielle dont la base est que les machines peuvent apprendre par elles-mêmes, elle effectue une analyse prédictive basée sur des modèles statistiques.

Il existe de nombreux modes d'apprentissage, tels que l'apprentissage supervisé, l'apprentissage non supervisé, l'apprentissage par renforcement, etc.

Dans le cas d'un apprentissage supervisé, il est nécessaire de fournir des données à la machine et même de lui donner la réponse attendue pour l'aider à faire des prédictions correctes.

Ensuite, pour le cas d'un apprentissage non-supervisé, la machine ne doit pas être informée de ce qu'elle doit donner en résultat, elle regroupe simplement les données selon la similitude.

Et dans le cas de l'apprentissage par renforcement, la machine apprendra grâce aux récompenses. Cela signifie que l'amélioration de ses performances sera rentable, ce qui augmentera les chances que ce comportement se reproduise une autre fois [3].

Dans notre projet, nous utiliserons l'apprentissage supervisé, ce dernier sera expliqué et analysé pour but de créer un modèle statistique qui nous permette de classifier correctement des données ensuite il sera utilisé pour créer un modèle dynamique afin de détecter les drones.

L’intelligence artificielle fait partie intégrante de la digitalisation, qui a modifiée durablement notre société. Ce qui était il y a quelques années encore de l’ordre de la science-fiction est désormais réalité.

Le concept d'apprentissage automatique existe depuis les années 1950, mais maintenant les systèmes d'apprentissage automatique sont devenus plus complexes, la puissance de calcul a augmenté de façon exponentielle et la quantité de données disponibles sur Internet est astronomique. Ce sont quelques raisons qui ont rendu ces systèmes à la mode depuis ces dernières années [3].

Dans le contexte des systèmes d'apprentissage automatique, certains termes pertinents doivent toujours être compris afin de mieux comprendre les principes de l'apprentissage automatique.

- **Intelligence artificielle :**

La recherche sur l'intelligence artificielle (IA) vise à créer des machines qui pourraient agir comme des humains. En fait, les ordinateurs et les robots doivent analyser leur environnement et prendre la meilleure décision possible. C'est pourquoi les robots doivent se comporter intelligemment selon nos normes.

Mais cela pose un problème : quels critères devons-nous utiliser pour juger de notre intelligence ?

Aujourd'hui, l'intelligence artificielle ne peut pas simuler tout l'être humain (en particulier l'intelligence émotionnelle). Au lieu de cela, certains aspects sont isolés pour traiter des tâches précises et spécifiques. Ceci est souvent appelé intelligence artificielle faible (IA faible).

- **Réseau neuronal :**

En tant que branche de la recherche en intelligence artificielle, la neuro-informatique tente également de concevoir davantage d'ordinateurs basés sur des modèles cérébraux. Elle estime que le système nerveux est abstrait, c'est-à-dire se débarrasser de leurs caractéristiques biologiques et confiner à leur mode de fonctionnement.

Les réseaux de neurones artificiels sont principalement des méthodes mathématiques abstraites. Les réseaux neuronaux (fonctions mathématiques ou algorithmes) sont assemblés comme un cerveau humain et peuvent faire face à des tâches complexes. La force de la chaîne entre les neurones est différente et peut s'adapter au problème.

- **Big Data :**

Le terme « Big Data » ou « mégadonnés » écrit simplement un énorme ensemble de données dont le nombre dépasse le niveau d'analyse humaine.

Ces dernières années, la couverture croissante de Big Data dans les médias est due à leur source. En fait, dans de nombreux cas, le flux d'informations est créé à partir des données des utilisateurs (intérêts, profils, données personnelles) collectées par des entreprises telles que Google, Amazon ou Facebook afin de pouvoir s'adapter plus précisément aux besoins des clients.

Les systèmes informatiques traditionnels ne peuvent plus évaluer de manière satisfaisante cette quantité de données : les logiciels conventionnels ne peuvent trouver que ce que l'utilisateur veut. C'est pourquoi nous avons besoin de systèmes d'apprentissage automatique qui nous permettront de découvrir et de mettre en œuvre des relations jusque-là inconnues.

- **Data-Mining :**

Le Data mining est une analyse de Big Data. En effet, la collection elle-même n'a pas encore beaucoup de valeur. Les fonctionnalités pertinentes doivent être téléchargées et évaluées. Le Data mining diffère de l'apprentissage automatique en ce qui concerne principalement l'utilisation de modèles reconnus, tandis que cette dernière recherche de nouveaux modèles [4].

1.2 Historique :

Aujourd'hui, les réseaux de neurones sont utilisés dans de nombreux domaines à cause de leurs propriétés, en particulier leur capacité d'apprentissage.

• 1890 : W. James, célèbre psychologue américain introduit le concept de mémoire associative et propose ce qui deviendra une loi de fonctionnement pour l'apprentissage sur les réseaux de neurones connue plus tard sous le nom de loi de Hebb.

• 1943 : J. Mc Culloch et W. Pitts, ce sont les premiers à montrer que des réseaux de neurones formels simples peuvent réaliser des fonctions logiques, arithmétiques et symboliques complexes (tout au moins au niveau théorique).

• 1949 : D. Hebb, physiologiste américain explique le conditionnement chez l'animal par les propriétés des neurones eux-mêmes. Ainsi, un conditionnement de type pavlovien tel que, nourrir tous les jours à la même heure un chien, entraîne chez cet animal la sécrétion de salive à cette heure précise même en l'absence de nourriture. La loi de modification des propriétés des connexions entre neurones qu'il propose explique en partie ce type de résultats expérimentaux [5].

• 1958 : Rosenblatt qui conçoit le perceptron (premier réseau de neurones artificiel) qui est inspiré du système visuel (en termes d'architecture neurobiologique) et possède une couche de neurones d'entrée "perceptive" ainsi qu'une couche de sortie "décisionnelle". Ce réseau parvient à apprendre, à identifier des formes simples et à calculer certaines fonctions logiques [6].

• 1969 : M. Minsky et S. Papert publient un ouvrage qui met en exergue les limitations théoriques du perceptron. Limitations alors connues, notamment concernant l'impossibilité de traiter par ce modèle des problèmes non linéaires. Ils étendent implicitement ces limitations à tout modèle de réseaux de neurones artificiels. Leur objectif est atteint, il y a abandon financier des recherches dans le domaine (surtout aux États-Unis), les chercheurs se tournent principalement vers l'IA et les systèmes à bases de règles [5].

• Il faudra attendre le début des années 80 pour que l'intérêt dans ce domaine soit de nouveau présent [6].

• 1982 : Hopfield démontre l'intérêt d'utiliser les réseaux récurrents dits « feed-back » pour la modélisation des processus. Les réseaux récurrents constituent alors la deuxième grande classe de réseaux de neurones, avec les réseaux de type perceptron dits "feed-forward".

• Werbos conçoit son algorithme de rétro propagation qui ne sera pourtant popularisé qu'en 1986 par Rumelhart.

1.3 L'intérêt de l'utilisation des réseaux de neurones artificiels :

Comme nous le définirons dans les paragraphes suivants, les réseaux de neurones formels ont des propriétés approximatives et peuvent être exprimés comme suit :

Un réseau neuronal se compose d'un nombre limité de couches neuronales cachées, qui ont toutes la même fonction d'activation et un neurone de sortie linéaire, et peuvent se rapprocher de toute fonction bornée suffisamment régulière avec une précision arbitraire dans le champ fini de son espace variable.

De plus, par rapport à d'autres outils mathématiques couramment utilisés, les réseaux de neurones nécessitent moins de paramètres ajustables (poids de connexion). En fait, les réseaux de neurones ne sont pas utilisés pour effectuer des approximations de fonctions connues.

Le problème le plus courant est de trouver une relation entre un ensemble de résultats dans un processus donné et un ensemble de données d'entrée correspondant aux mesures effectuées. Nous supposons que cette relation existe malgré le fait que :

- Les mesures sont limitées en nombre ;
- Elles sont certainement affectées par le bruit ;
- Toutes les variables qui déterminent le résultat du processus n'ont pas besoin d'être mesurées.

En d'autres termes, l'ingénieur recherche un modèle du processus sur lequel il travaille à partir des mesures disponibles, on dit qu'il effectue une modélisation « boîte noire ». Dans la "syntaxe" des réseaux de neurones, les données à partir desquelles nous voulons construire un modèle sont appelées des exemples. Par conséquent, la raison pour laquelle les réseaux de neurones sont considérés comme un bon choix est que nous pouvons trouver une approximation à partir des mesures disponibles du modèle RNA.

Dans l'ensemble, le réseau de neurones utilise mieux les mesures disponibles par rapport aux méthodes d'approximation non linéaires traditionnelles.

Ce gain peut être important lorsque le processus modélisé dépend de plusieurs variables, par exemple dans le cas d'un processus de mise en forme avec plusieurs types de non-linéarités et plusieurs paramètres matériels et technologiques [7].

1.4 Notion de base sur les réseaux de neurones :

1.4.1 Le neurone biologique :

Les neurones sont des cellules du système nerveux. Ces cellules nerveuses sont responsables de la réception et de la transmission des influx nerveux. Le système nerveux compte plus de 1000 milliards de neurones interconnectés entre eux, les neurones ne sont pas tous identiques, leurs comportements les diffèrent en fonction de leurs positions dans le cerveau. Un neurone peut se décomposer en trois régions principales :

- Un corps cellulaire qui contient le noyau, la machine biochimique qui produit les enzymes et d'autres molécules qui contribuent à la survie des cellules nerveuses ;
- Des dendrites qui sont des parties multi-ramifiées qui reçoivent les influx nerveux, leur taille est de quelques dizaines de micromètres ;
- Un axone unique ; les axones sont les structures allongées qui transmettent les influx à partir du corps cellulaire. L'axone se ramifie à son extrémité où il se connecte aux dendrites des autres neurones, sa taille peut varier de quelques millimètres à plusieurs mètres ;

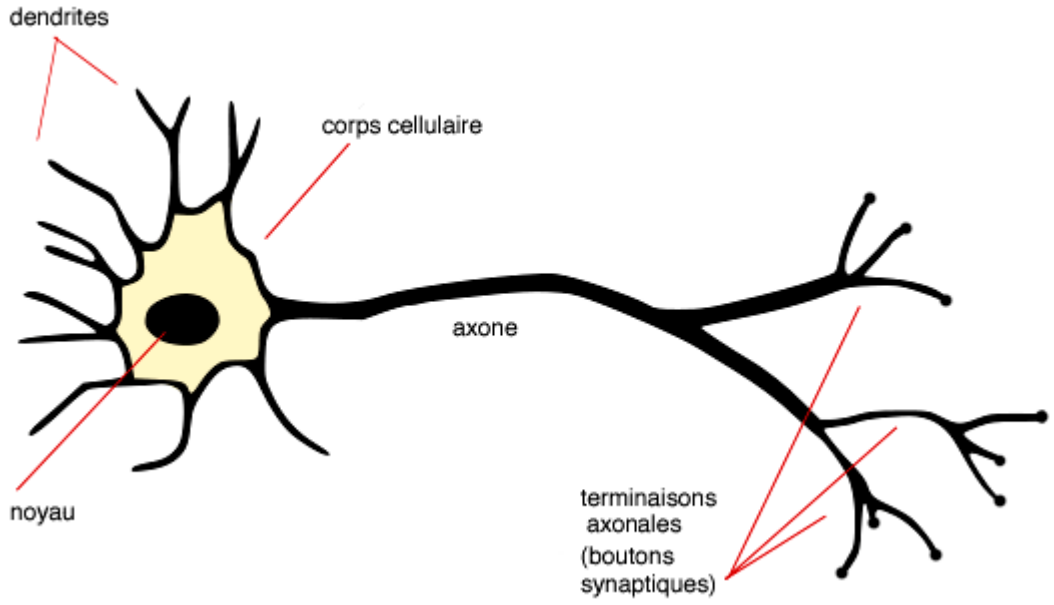


Figure 1.1 : un neurone biologique [8].

- Synapse : c'est l'endroit où se fait la connexion entre deux neurones, donc entre une dendrite et un axone. Anatomiquement, il s'agit d'une fente. L'une de ses berges est la terminaison d'un axone, l'autre est l'origine d'une dendrite. L'axone d'un neurone est donc lié à la dendrite du neurone suivant. Les synapses possèdent une sorte de mémoire qui leur permet d'ajuster leur fonctionnement, elles facilitent ou non le passage des influx nerveux, cette plasticité est à l'origine des mécanismes d'apprentissage [8].

1.4.2 Le neurone formel :

Un neurone formel ou bien artificiel est une modélisation mathématique du neurone biologique basée sur la structure des neurones à l'intérieur d'un cerveau humain pour modéliser et résoudre les problèmes complexes.

Alors un neurone est une fonction mathématique non linéaire et bornée, dont les variables de cette fonction sont appelées des « entrées » et la valeur de la fonction est appelée sa « sortie ».

On a l'habitude de représenter graphiquement un neurone comme dans la figure (1.2) ci-dessous [9] :

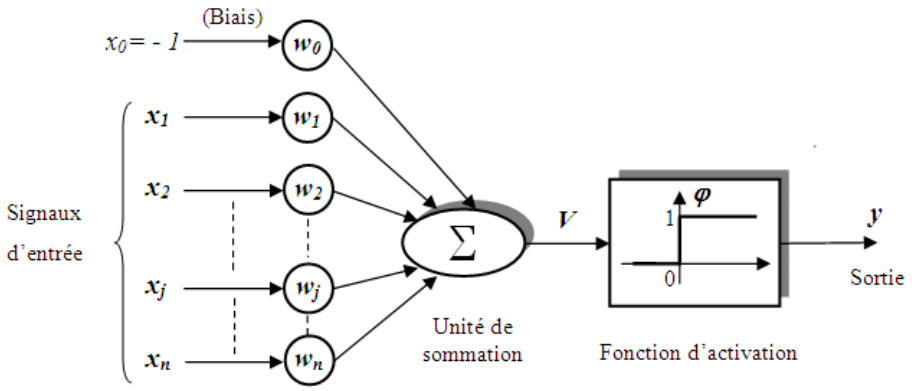


Figure 1.2 : modèle du neurone artificiel [10].

- x_1, x_2,…, x_j,…. x_n sont les entrées externes, elles proviennent soit des sorties d'autres neurones soit de stimuli sensoriels (capteur visuel, sonore,...) ;
- y est la sortie ;
- w_0, w_1……, w_n sont les poids synaptiques du neurone associés à chaque entrée, ils représentent l'efficacité synaptique du neurone biologique ($w_j<0$: synapse inhibitrice, $w_j>0$: synapse excitatrice) ; ces poids pondèrent les entrées et peuvent être modifiés par apprentissage ;
- Biais : généralement prend les valeurs +1 ou bien -1, il permet de varier le seuil de déclanchement du neurone et l'ajustement des poids et du biais lors de l'apprentissage ;
- Le noyau : intègre toutes les entrées et le biais et calcule la sortie en utilisant une fonction d'activation qui est souvent non linéaire [11].

1.4.3 La structure de réseau de neurone :

Avant de commencer l'explication sur la structure d'un réseau de neurones, certains termes doivent être clarifiés :

- Le poids de connexion est une valeur $w_{i,j}$ associée à l'attribut d'entrée j d'un neurone i. Il correspond au poids synaptique du neurone biologique ;
- La fonction de sommation d'un neurone est fonction de l'accumulation de ses différentes entrées, résultant en une somme pondérée de celles-ci. Cette valeur, servant de paramètre d'activation du neurone, est également appelée potentiel de membrane ;

- La fonction de transfert d'un neurone est une fonction permettant de calculer l'état (activé ou inactif) du neurone en fonction du potentiel membranaire calculé par la fonction de combinaison. Cette fonction est également appelée fonction d'activation.

Les neurones formels sont des unités fonctionnelles de base. Comme le montre la figure 1.3, la structure du neurone formel est inspirée de son homologue biologique.

Le neurone formel est fourni par un vecteur d'entrée $\vec{X}$ de taille d, $\vec{X}$= (x_1, x_2, ..., x_d), et son poids de connexion w_i (weigth) est associé à chaque attribut du vecteur. Les attributs d'entrée des neurones formels et leurs poids de connexion correspondent aux dendrites et aux synapses sur leurs homologues biologiques.

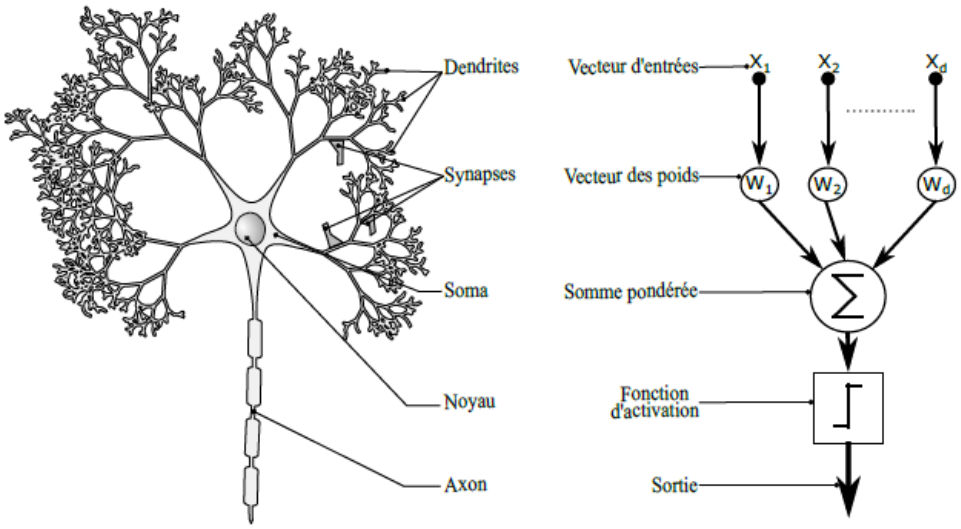


Figure 1.3 : mise en correspondance neurone biologique / neurone artificiel [12].

Par la suite, la somme calculée est envoyée à la fonction d'activation pour générer la valeur de la sortie neuronale. La fonction d'activation liée à la sortie traduit les noyaux et les axones dans la structure biologique des neurones.

Toutes les valeurs d'entrée sont extraites de l'accumulateur, ce qui permet à la fonction de sommation ou de combinaison de générer une somme pondérée des valeurs d'entrée. Cette fonction de sommation correspond évidemment au corps cellulaire (Soma) dans la structure des neurones biologiques [13].

1.4.4 Le comportement de réseau de neurone :

Lors du fonctionnement d'un réseau de neurones, le neurone formel produira une sortie variable basée sur des attributs d'entrée provenant de l'environnement externe ou des neurones voisins en

fonction de son emplacement dans le réseau. La variable de sortie est une valeur différentielle, qui peut distinguer l'état actif ou l'état inactif du neurone.

Lorsque les attributs d'entrée sont reçus, ils passent par une fonction de pondération avec les poids des connexions. Ensuite, toutes les valeurs pondérées seront transmises à la fonction de sommation. Ainsi, la somme pondérée est calculée. Selon la pondération, il existe deux types d'unités neuronales.

- **Unité produit scalaire :** où la pondération entre le vecteur d'entrée et le vecteur des poids de connexions est un produit scalaire. La somme pondérée est calculée par l'équation suivante :

$$Ps = \sum_{i=1}^{D}(w_i \cdot x_i) \quad (1)$$

- **Unité de distance :** où la pondération calculée représente la distance entre le vecteur des poids de connexion et le vecteur d'entrée. Cette distance peut être calculée sur différents niveaux (L1 : Manhattan, L2 : Euclidienne,). L'équation 2 présente la formule de calcul de la distance Euclidienne :

$$D_{l2} = \sum_{i=1}^{D}(w_i - x_i)^2 \quad (2)$$

La somme est ensuite envoyée à la fonction d'activation, souvent reconnue par la fonction de transfert, qui génère une valeur de sortie basée sur la valeur de seuil qui a été entrée en paramètre.

Une description détaillée de la fonction d'activation sera fournie dans les paragraphes suivants de ce chapitre.

En résumé, le neurone utilise l'équation 3 pour calculer sa valeur de sortie. La forme de cette équation reflète la non-linéarité du traitement des réseaux neuronaux. La valeur de sortie montre l'état des neurones qui seront envoyés aux neurones en aval ou directement à l'environnement externe [13].

$$S = f\left(\sum_{i=1}^{D}\|w_i \cdot x_i\|\right) \quad (3)$$

Notez que les équations décrivant le comportement des neurones artificiels n'introduisent pas la notion de temps. En fait, c'est le cas de la plupart des modèles de réseaux de neurones actuels, ce sont des modèles discrets et synchrones dont le comportement des composants ne varie pas dans le temps [5].

1.5 La fonction d'activation :

La fonction d'activation est inspirée du potentiel d'action qui est un phénomène électrique entre deux neurones biologiques, le neurone reçoit des signaux d'autres neurones à travers les dendrites. Le poids synaptique, est multiplié par le signal entrant. Les signaux des dendrites sont accumulés dans le corps cellulaire et si la force du signal résultant dépasse un certain seuil, le neurone transmet

le message à l'axone. Sinon, le signal est endommagé et ne se propage pas davantage. Le potentiel d'action est donc la variation de la force du signal indiquant si la communication doit se faire ou non [14].

La fonction de transfert ou la fonction d'activation dans un réseau de neurones artificiels est en général, une fonction non linéaire monotone croissante. Par ailleurs, les fonctions sont de qualités diverses : elles peuvent être déterministes, continues, discontinues ou aléatoires [15].

1.5.1 Les types de la fonction de transfert :

- **Fonction d'activation linéaire :**

C'est une fonction simple généralement de la forme f(x) = a.x où a ∈ R, cette fonction n'apporte pas un grand changement sur l'entrée, on aura une situation de proportionnalité :

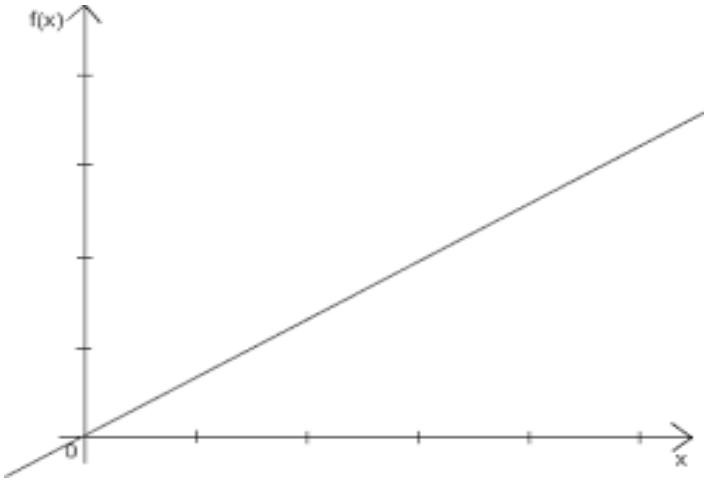


Figure 1.4 : fonction linéaire f(x) = ax [9].

- **Fonction d'activation non linéaire :**

Sont les fonctions les plus utilisées puisqu'elles permettent une séparation presque parfaite des données non linéairement séparables, elles sont indispensables pour résoudre les problèmes les plus complexes et aussi pour la raison que les fonctions linéaires ne peuvent être utilisées que pour les réseaux de neurones à une seule couche :

Exemple des fonctions non linéaire :

- **Fonction Sigmoïde :**

C'est une fonction qui permet la conversion des entrées pour les réduire entre 0 et 1, et d'exprimer les valeurs sous forme de probabilités. Lorsque la valeur est un très grand nombre positif la probabilité est de 1, et contrairement, lorsque la valeur est un très grand nombre négatif, la probabilité est alors de 0. Seules les petites valeurs influent réellement sur la sortie.

La fonction sigmoïde présente plusieurs défauts :

- Elle engendre des sorties toujours positives même pour les valeurs d'entrées négatives ;
- L'aplatissement assez important de la courbe engendre la saturation de neurones ;
- Elle n'est pas rapide puisqu'elle comprend la fonction exponentielle.

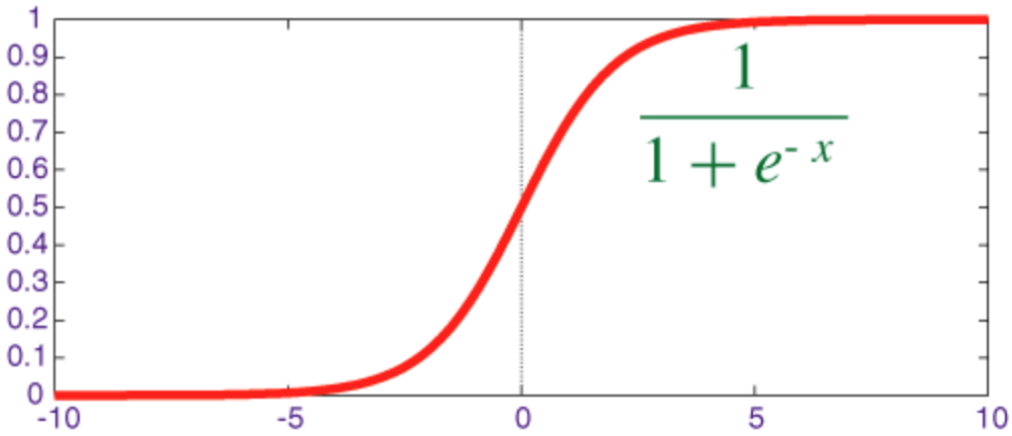


Figure 1.5 : la fonction sigmoïde [16].

- **Fonction tangente :**

La fonction tangente est comme la fonction sigmoïde mais la seule différence entre elles est que la fonction tangente est centrée sur zéro et qu'elle écrase toutes les valeurs comprises entre -1 et 1 (négatives et positives). Malgré cet avantage supplémentaire, la fonction tangente hyperbolique présente encore les mêmes inconvénients.

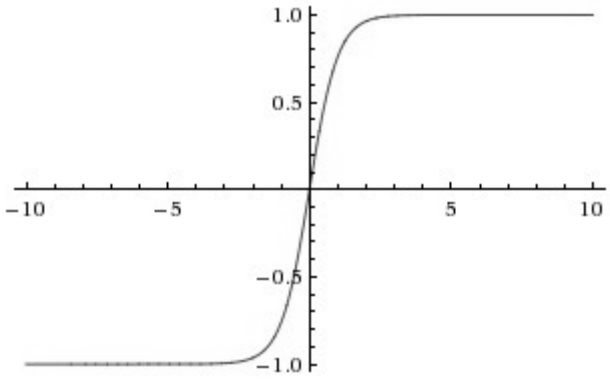


Figure 1.6 : la fonction tangente hyperbolique [9].

- **Fonction unité linéaire rectifiée :**

La fonction Relu ou bien unité linéaire rectifiée, est utilisée pour résoudre la saturation des deux fonctions sigmoïde et tanh, elle est représentée sous la forme de f(x) = max (0, x), si l'entrée est négative alors la sortie est 0, et si l'entrée est positive, la sortie est x, le réseau de neurones converge rapidement en utilisant cette fonction et ne sature pas.

Mais en dehors de cet avantage, le neurone reste inactif si l'entrée est négative, et aussi les poids ne sont pas mis à jour, donc le réseau ne pourra pas apprendre.

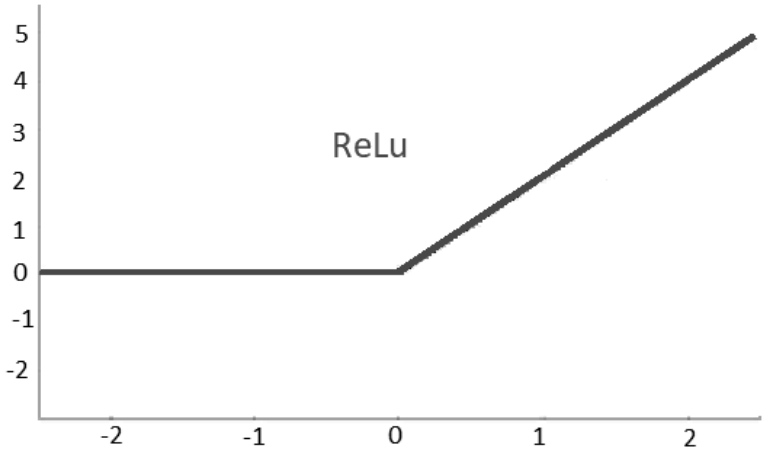


Figure 1.5 : La fonction Relu [9].

- **Fonction softmax :**

Softmax est une fonction d'activation spécialisée pour les réseaux de classification d'un problème à plusieurs classes. Elle réalise une exponentielle normalisée (c'est-à-dire que la somme des sorties doit être égale à 1). Cette contrainte supplémentaire permet de faire converger l'apprentissage plus rapidement qu'il ne le ferait autrement. En combinaison avec la fonction d'erreur d'entropie croisée (voir le chapitre 3), elle permet de modifier les réseaux perceptron multicouche pour l'estimation des probabilités de classes, Softmax est mis en œuvre via une couche de réseau de neurones juste avant la couche du résultat. La couche Softmax doit comporter le même nombre de nœuds que la couche du résultat. Son Intervalle de sortie est (-∞ ; +∞) [17].

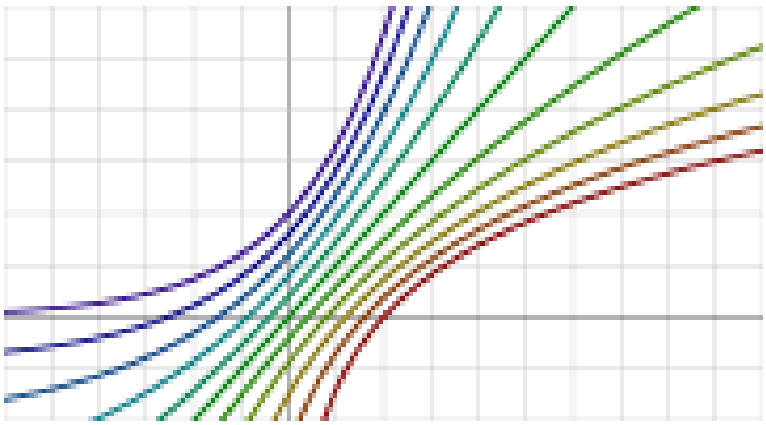


Figure 1.8 : la fonction softmax [18].

1.6 Le réseau de neurones :

Le réseau de neurones est une combinaison de fonctions élémentaires appelées neurones formels ordonnés dans des plans appelés couches.

Les neurones peuvent être reliés par des connexions pondérées avec les neurones des autres couches ou avec celles de la même couche. Chaque processeur élémentaire calcule une seule sortie en utilisant les variables d'entrées.

En générale, un neurone fait deux choses :

1. Il reçoit les entrées des autres neurones pour les faire combiner ensemble ;
2. Il effectue la transformation de données vers la sortie du neurone.

1.6.1 Architecture de réseaux de neurones :

Selon la topologie de connexion des neurones, on peut les classer en deux grandes catégories :

- Réseaux non bouclés (statique ou feed forward) ;
- Réseaux bouclés (dynamique, feed back ou récurrent).

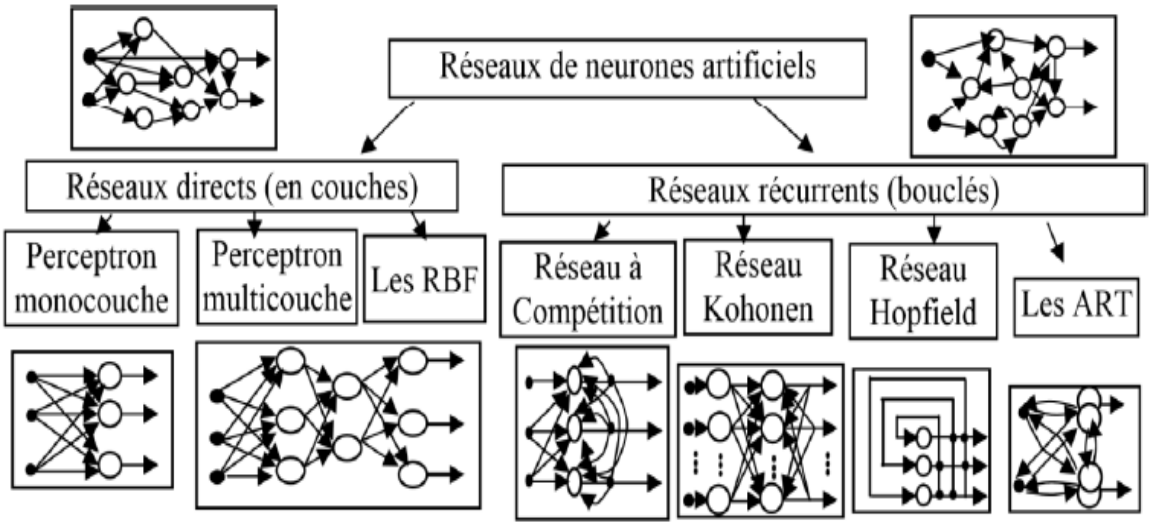


Figure 1.9 : les topologies des réseaux de neurones artificiels [19].

1.6.1.1 Réseaux statiques ou non bouclé :

Un réseau de neurones non bouclé (aussi dit statique ou bien réseau à propagation avant) est représenté comme un graphe dont les nœuds sont les neurones. L'information se propage à travers les couches de ce réseau, de l'entrée vers la sortie sans retour en arrière. Ce type de réseau est utilisé pour effectuer des tâches d'approximation de fonction non linéaire, de la classification ou de la modélisation de processus statiques non linéaires, les exemples les plus connus sont celui du perceptron simple et du perceptron multicouches [9].

- **Le perceptron simple :**

Le perceptron simple est un réseau monocouche et acyclique (il ne contient pas de boucles) qui ne comporte qu'une seule couche de neurones dont la fonction d'activation est de type « pas unitaire », ce qui implique un seul vecteur de poids. Toutes les unités de l'entrée sont connectées à celles de la sortie.

Ce réseau fonctionne comme un classeur linéaire, autrement dit, il peut seulement séparer ou classer les données linéairement séparables grâce à sa structure [2].

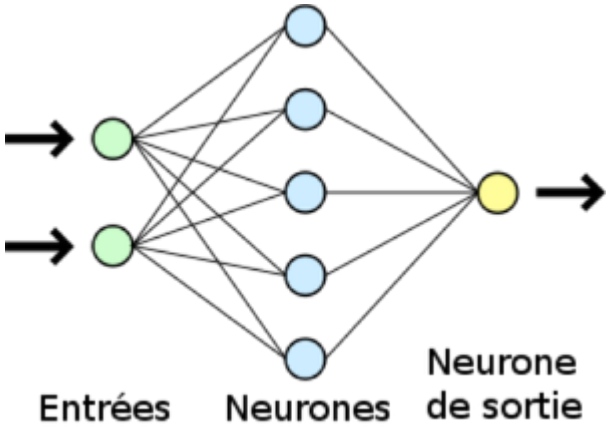


Figure 1.10 : schéma d'un perceptron simple [20].

- **Réseaux multicouches :**

C'est le réseau de neurones statique le plus utilisé. Les neurones sont arrangés par couche. Les neurones de la première couche reçoivent le vecteur d'entrée, ils calculent leurs sorties qui sont transmises aux neurones de la seconde couche qui calculent eux-mêmes leurs sorties et ainsi de suite de couche en couche jusqu'à celle de sortie. Selon leur disposition, on trouve deux types de couches :

- la couche de sortie ou bien la couche de décision où on récupère les résultats finaux de traitement, le nombre de neurones sur la couche de sortie dépend des résultats qu'on veut obtenir. Cette couche peut être représentée par un perceptron simple ou bien par un neurone avec une fonction d'activation linéaire appelé ADALINE (ADAptive LInear Neuron) [13].

- Les couches cachées (couches de prétraitement), sont des couches placées entre les entrées et la couche de sortie, elles sont de type perceptron simple et cela, pour que ces couches puissent préparer les données en utilisant des fonctions d'activation non-linéaires.

Chaque neurone dans la couche cachée est connecté à tous les neurones de la couche précédente et de la couche suivante, et il n'y a pas de connexions entre les cellules d'une même couche.

Sa topologie permet de résoudre des problèmes non linéairement séparables et il suit un apprentissage supervisé.

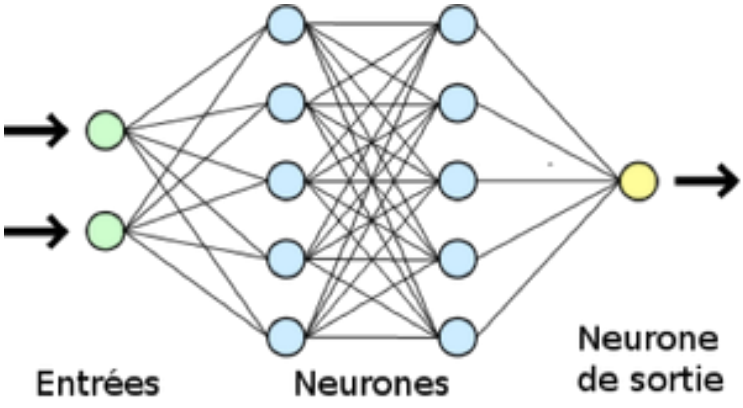


Figure 1.11 : schéma d'un PMC [20].

1.6.1.2 Réseaux récurrents :

Les réseaux récurrents appelés aussi réseaux bouclés ou RNN (Recurrent Neural Network) sont des réseaux avec une ou plusieurs liaisons vers l'arrière, régis par une ou plusieurs équations différentielles, résultent de la composition des fonctions réalisées par chacun des neurones et des retards associés à chacune des connexions. La fonction d'activation peut affecter les réseaux pendant une longue période, ce qui lui permet de converger vers des résultats plus précis.

Ces réseaux sont utilisés généralement pour le traitement de séquences (les signaux de taille variable) tel que la parole, le traitement de texte, le processus de commande ou le filtrage. Le comportement dynamique d'un réseau de neurones bouclé s'avère très complexe [5].

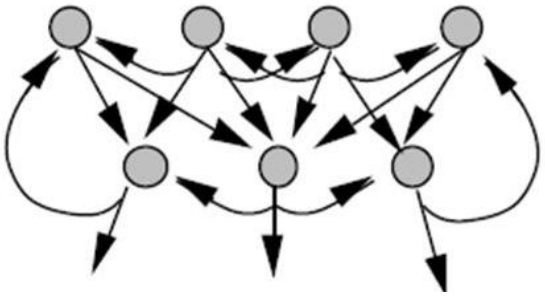


Figure 1.12 : schéma d'un réseau récurrent [21].

1.6.1.3 Réseau à connexion complète :

Il s'agit de la structure d'interconnexion la plus générale (figure 1.12). Chaque neurone est connecté à tous les neurones du réseau (et à lui-même) [5].

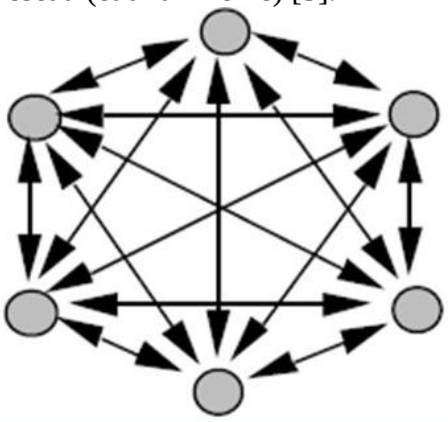


Figure 1.13 : schéma d'un réseau à connexion complète [21].

1.7 Apprentissage :

L'apprentissage est vraisemblablement la propriété la plus intéressante des réseaux neuronaux. Elle ne concerne cependant pas tous les modèles, mais les plus utilisés.

L'apprentissage est une phase du développement d'un réseau de neurones durant laquelle le comportement du réseau est modifié jusqu'à l'obtention du comportement désiré. D'une autre manière, l'apprentissage consiste à calculer les paramètres de sorte que les sorties du réseau soient aussi proches que possible des résultats recherchés.

Ces paramètres peuvent être :

- Le code de la classe à laquelle appartient la forme que l'on veut classer ;
- La valeur de la fonction que l'on veut approcher ou celle de la sortie du processus que l'on veut modéliser ;
- La sortie souhaitée du processus à commander.

Dans le cas des réseaux de neurones artificiels, on ajoute souvent à la description du modèle l'algorithme d'apprentissage dit algorithme d'optimisation, le modèle sans apprentissage présente en effet peu d'intérêt. Ces algorithmes consistent à minimiser l'écart entre la/les réponse(s) réelle(s) et la/les réponse(s) que l'on désire avoir en modifiant les paramètres par étapes successives appelées itérations. Lorsque l'algorithme d'apprentissage se déroule, la sortie s'adapte mieux.

Dans la majorité des algorithmes actuels, les variables modifiées pendant l'apprentissage sont les poids des connexions. L'apprentissage est la modification des poids du réseau dans le but d'accorder la réponse du réseau aux exemples et à l'expérience. On ne peut souvent pas décider à priori des valeurs des poids des connexions d'un réseau pour une application donnée.

À l'issu de l'apprentissage, les poids sont fixés : c'est alors la phase d'utilisation. Certains modèles de réseaux sont à apprentissage permanent. Dans ce cas il est vrai que l'apprentissage ne s'arrête jamais, cependant on peut toujours distinguer une phase d'apprentissage et une phase d'utilisation. Cette technique permet de conserver au réseau un comportement adapté malgré les fluctuations dans les données d'entrées.

Au niveau des algorithmes d'apprentissage, deux classes principales ont été définies selon lesquelles l'apprentissage est dit supervisé ou non supervisé. Cette distinction est basée sur la forme des exemples d'apprentissage. Dans le cas de l'apprentissage supervisé, les exemples sont des couples (Entrée, Sortie associée) alors que l'on ne dispose que des valeurs (Entrée) pour l'apprentissage non supervisé. Remarquons cependant que les modèles à apprentissage non supervisé nécessitent avant la phase d'utilisation, une étape de labélisation effectuée par l'opérateur, qui n'est qu'une part de supervision [5].

Alors l'apprentissage peut être exprimé par minimisation de la fonction perte quadratique (à la puissance 2) ou de celle d'une fonction d'entropie en classification :

$$\mathrm{Q}(w, w_0) = \sum_{i=1}^{n} Q_i = \sum_{i=1}^{n} [y_i - f(x; w, w_0)]^2$$

Où :

- $f(x; w, w_0)$: est la fonction de transfert ;
- w,w_0 : les paramètres à modifier qui sont successivement les poids synaptiques et le biais ;
- x : les variables explicatives, les entrées ;
- y_i : les variables estimées.

1.7.1 Apprentissage supervisé :

Pour les réseaux à apprentissage supervisé, on présente au réseau : à la fois les entrées et les sorties souhaitées pour ces entrées. Le réseau doit ajuster ses poids de façon à réduire l'écart entre la réponse désirée et la sortie du réseau. Cette procédure est répétée jusqu'à ce qu'un critère de performance soit satisfait de la façon suivante : dans le cas d'une discordance entre la réponse obtenue par le réseau est la réponse désirée, le poids synaptique des neurones actifs doit être affaibli. Par contre, là où il y a correspondance, le processus consiste à augmenter le poids des neurones actifs. Par conséquent, un modèle neuronal à règle d'apprentissage supervisé passe par une procédure de calibration des paramètres libres pendant la phase d'apprentissage. Ensuite, les circuits mémorisés seront généralisés sur les entrées aléatoires pendant la phase de décision.

Les algorithmes se basant sur la règle d'apprentissage supervisé sont très recommandés dans plusieurs domaines tels que les analyses prédictives, les analyses financières, la détection de fraudes, etc. Parmi les algorithmes d'apprentissage supervisés, les plus utilisés sont les algorithmes de rétro-propagation de gradient (ou bien d'erreur) et la règle Delta, ces derniers sont très appliqués avec le modèle de perceptron multicouches voir PMC.

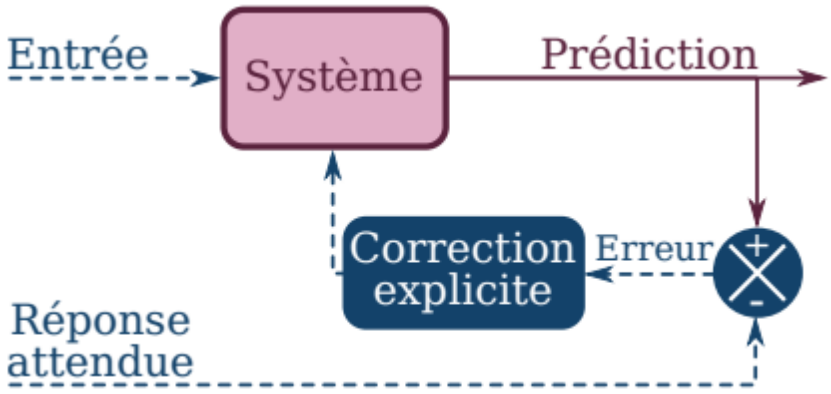


Figure 1.14 : bloc de l'apprentissage supervisé [22].

1.7.2 L'algorithme de rétro-propagation de gradient :

L'algorithme de rétro-propagation est un algorithme de gradient itératif conçu pour minimiser l'erreur quadratique (à une puissance de 2) entre les sorties obtenues du réseau multicouches et celles souhaitées. Cette minimisation est obtenue grâce à la configuration du contrepoids adéquat. L'erreur(e) est la différence entre la valeur désirée (d) pour le neurone de sortie et sa valeur calculée par propagation (x). Ce signal d'erreur permet de définir une fonction de coût :

$$C(W) = M[C_l(W)] = M[\sum_j e_{lj}{}^2(W)] \qquad \text{Avec } (e_{lj} = d_{lj} - x_{lj})$$

- $C(W)$ est une fonction coût ;
- j indique un numéro d'indice pour les neurones de sortie ;
- l indique un exemple d'apprentissage ;
- M est l'opérateur de moyennage, c'est une estimation de la moyenne temporelle dans le cas stochastique.

La fonction de transfert recommandée pour cet algorithme doit être continue, non-linéaire et différenciable.

Les étapes de l'algorithme :

1/ Initialisation des poids à des valeurs aléatoires de faible grandeur ;

2/ Sélection d'un exemple d'apprentissage $(E, d)_l$ dans la base d'apprentissage ;

3/ Présentation de la forme d'entrée (E) sur la couche d'entrée du réseau ;

4/ Calcul par propagation de la sortie obtenue (o) ;

5/ Si erreur en sortie alors pour tous les neurones i (depuis la sortie jusqu'à l'entrée) :

Si i est un neurone de sortie alors : $y_i = 2\, f'(a_i).(d_i - x_i)$;

Si i est un neurone caché (ou d'entrée) alors : $y_i = 2f'(a_i).\sum_k(w_{ki}, y_i)$;

(K : neurones compris entre la couche actuelle et la couche de sortie)

6/ Application de la procédure de gradient. μ est un gain fixé par l'utilisateur :

$$w_{ij}(t+1) = w_{ij}(t) + \mu.y_i.x_j \ ;$$

7/ Tant que l'erreur est trop importante, retour à l'étape 2 [21].

1.7.3 Apprentissage non supervisé :

Pour les réseaux à apprentissage non supervisé, aucune information sur la réponse désirée n'est fournie au réseau. On présente une entrée au réseau et on le laisse s'évoluer librement jusqu'à ce qu'il se stabilise. Ce comportement est connu sous le nom « auto organisation », Les algorithmes basés sur cette règle d'apprentissage sont alimentés, au cours de la phase d'apprentissage par une base d'apprentissage constituée d'un ensemble de stimuli avec une certaine redondance. Ces algorithmes d'apprentissage sont souvent appelés algorithmes « clustering », ce qui signifie regroupement et séparation.

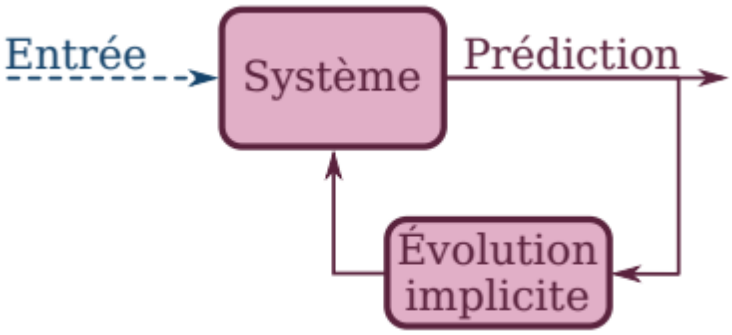


Figure 1.15 : bloc de l'apprentissage non supervisé [22].

1.8 Type des réseaux de neurones artificiels :

1.8.1 Réseaux Hopfield :

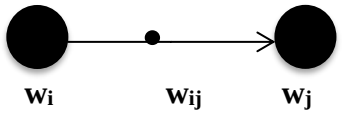
Il s'agit d'un réseau constitué de neurones à deux états (-1 et 1, ou 0 et 1), dont la loi d'apprentissage est la règle de Hebb (1949), qui veut qu'une synapse améliore son activité si et seulement si l'activité de ses deux neurones est corrélée (c'est-à-dire que le poids d'une connexion entre deux neurones augmente quand les deux neurones sont activés au même moment) [23].

- **La loi de Hebb :**

La loi de Hebb (1949) s'applique aux connexions entre neurones, il existe deux règles à suivre pour l'appliquer :

1. Renforcement synaptique : si deux neurones de chaque côté d'une connexion sont activés simultanément (de manière synchrone), alors la force de cette connexion est sélectivement renforcée.
2. Affaiblissement synaptique : si deux neurones de chaque côté d'une connexion sont activés de manière asynchrone, alors cette connexion est sélectivement affaiblie ou éliminée [24].

La loi de Hebb peut être modélisée par les équations suivantes :



$$w_{ij}(t+1) = w_{ij}(t) + \partial w_{ij}(t) \quad \text{où} \quad \partial w_{ij}(t) = x_i \cdot x_j ;$$

xi, xj : les valeurs d'activation des neurones i et j.

wij(t+1) : le nouveau poids exprimé par la loi de Hebb.

wij(t) : l'ancien poids.

∂wij(t) : la dérivée partielle du poids et qui exprime la coactivité entre le neurone i et j, elle est modélisée comme la valeur de produit des valeurs d'activation [5].

1.8.2 Réseaux Kohonen :

Ces réseaux sont inspirés des observations biologiques du fonctionnement des systèmes nerveux de perception des mammifères. Une loi de Hebb modifiée (tenant compte de l'oubli) est utilisée pour l'apprentissage. La connexion est renforcée dans le cas où les neurones reliés ont une activité simultanée et diminuée dans le cas contraire (alors qu'il ne se passait précédemment rien dans ce cas). Tous ces réseaux ont des applications dans la classification, le traitement d'image, l'aide à la décision et l'optimisation, ces réseaux sont plus complexes mais très proches de la réalité [13].

Ce réseau utilise la carte de Kohonen, qui se compose de deux couches, où la première couche constitue l'entrée du réseau et la deuxième couche constitue la sortie. La structure de cette carte va être présentée dans la figure ci-dessous. Les neurones de cette carte sont disposés dans une certaine topologie. Plusieurs topologies sont possibles, telles que des cartes carrées ou des cartes rectangulaires [23].

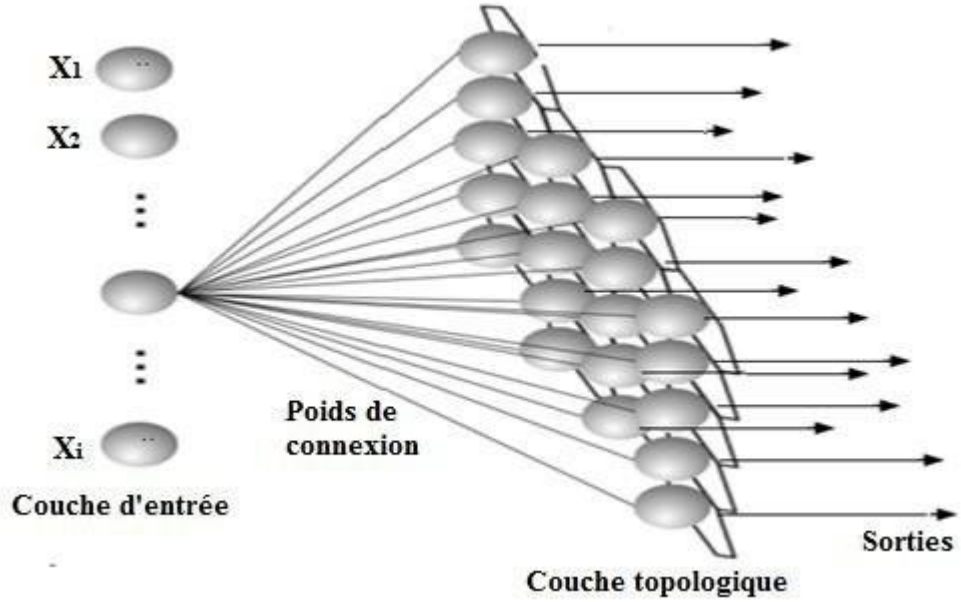


Figure 1.16 : la structure de la carte de kohonen [25].

1.9 Les domaines d'intelligence artificielle :

Les autres domaines et technologies d'intelligence artificielle : robotique, systèmes multi-agents, SVM, réseaux bayésiens, apprentissage machine par renforcement, programmation par contraintes, raisonnements à partir de cas, ontologies, logiques de description, algorithmes génétiques, etc.

On peut trouver d'autres domaines et technologies d'intelligence artificielle ajoutés à ceux déjà mentionnés précédemment et ce ne sont que quelques exemples pour illustrer la richesse et la variété d'intelligence artificielle. Il peut résoudre des problèmes non linéairement séparables et il suit un apprentissage supervisé avec la règle de correction de l'erreur.

Le tableau académique des domaines de l'intelligence artificielle (IJCAI) retient cinq domaines :

- traitement du langage naturel ;
- vision (ou traitement du signal) ;
- apprentissage automatique ;
- systèmes multi-agents ;
- robotique [26].

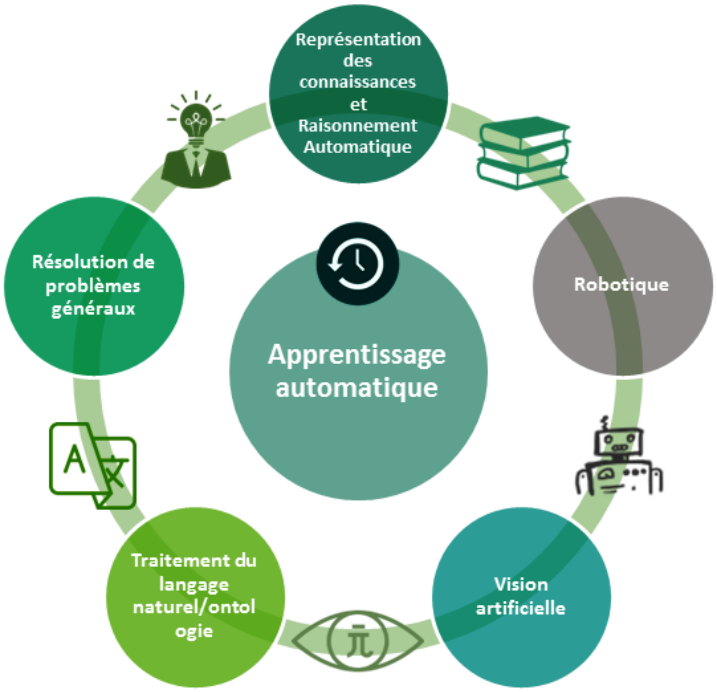


Figure 1.17 : les domaines de l'intelligence artificielle [27].

1.10 Les réseaux de neurones artificiels et leurs applications :

Lorsqu'il s'agit de traiter de grandes quantités de données, sans savoir où la solution s'orientera, les réseaux de neurones artificiels sont un outil puissant. Ils sont généralement utilisés dans le domaine de la reconnaissance de l'écriture manuscrite, de la reconnaissance d'images et de la voix, où un système informatique recherche certaines caractéristiques afin de mener à bien leur mission.

Un réseau de neurones artificiels peut également être utilisé pour effectuer tout type de prédiction ou de simulation. Cela s'applique, par exemple, aux prévisions météorologiques, aux diagnostics médicaux ou aux marchés boursiers.

Dans l'industrie, les réseaux de neurones artificiels sont parfois utilisés dans le cadre des technologies de surveillance d'activité, pour détecter tout écart par rapport aux valeurs déterminées et prendre automatiquement les contre-mesures nécessaires, ou pour fixer indépendamment des valeurs cibles en tenant compte de l'évaluation des données effectuées par les réseaux.

De nos jours, le développement de l'apprentissage non supervisé des réseaux de neurones artificiels permet d'étendre considérablement ses performances et ses domaines d'application. Parmi les applications les plus importantes des réseaux de neurones artificiels d'apprentissage automatique figurent les exemples bien connus d'Alexa, Siri (d'Appel) et l'assistant vocal de Google [28].

1.11 L'utilisation des réseaux de neurones :

Les réseaux de neurones ont la capacité extraordinaire de comprendre et d'extraire des règles et des tendances à partir de données complexes, bruyantes et imprécises. Ils peuvent être utilisés pour extraire des modèles et détecter des tendances basées sur des fonctions mathématiques complexes

qui sont difficiles (voire impossibles) à modéliser à l'aide de techniques d'analyse ou de paramétrage traditionnelles.

L'un des avantages des réseaux de neurones est la capacité de prédire avec précision des données qui ne sont pas des données d'entraînement, un processus appelé généralisation. Au vu de ces caractéristiques et de sa large gamme d'applications, les réseaux de neurones sont particulièrement adaptés à l'application de problèmes spécifiques dans les domaines de la recherche scientifique, commerciale et industrielle [19].

Voici de nombreux domaines dans lesquels les réseaux de neurones sont appliqués avec succès :

- Traitement du signal ;
- Prédiction ;
- Maîtrise des processus ;
- Robotique ;
- Classification ;
- Prétraitement des données ;
- Reconnaissance de formes ;
- Analyse d'image et synthèse vocale ;
- Diagnostiques et suivi médical ;
- Marché boursier et prévisions ;
- Demande de crédits ou de prêts immobiliers.

1.12 Les avantages et les inconvénients des réseaux de neurone :

Le réseau neuronal peut exécuter n'importe quelle fonction non linéaire jusqu'à ce qu'il atteigne un certain degré de fiabilité. Il peut également implémenter des fonctions dynamiques et avoir n'importe quel nombre d'entrées et de sorties. Nous énumérerons les avantages et les inconvénients des réseaux de neurones artificiels.

1.12.1 Avantages des réseaux de neurones :

• La capacité d'exprimer n'importe quelle fonction, qu'elle soit linéaire ou non, simple ou complexe ;

• Une faculté d'apprendre à partir d'exemples représentatifs par rétro propagation d'erreurs. L'apprentissage (ou la construction de modèles) se fait automatiquement ;

• Insensible aux données non fiables ou au bruit ;

• Facile à utiliser, le travail personnel exigeait beaucoup moins qu'une analyse statistique classique. Aucune compétence en mathématiques et calculs statistiques requis ;

• Moins de mauvais comportements en cas de petite quantité de données ;

• Pour un utilisateur novice, l'apprentissage est plus facile à comprendre que la complexité des statistiques multivariées.

1.12.2 Inconvénients des réseaux de neurones :

• Il n'y a pas de méthode systématique de définir la topologie optimale du réseau et le nombre de neurones à placer dans la couche cachée ;

• Le choix des valeurs initiales des poids du réseau et le réglage du pas d'apprentissage jouent un rôle important dans la vitesse de convergence ;

• Le problème du sur-apprentissage qui ne se généralisera pas.

• Les connaissances acquises par le réseau neuronal sont encodées par la valeur du poids synaptique, le réseau neuronal est donc une boîte noire et les utilisateurs ne peuvent pas comprendre ces connaissances [19].

1.13 Procédure de développement d'un réseau de neurones :

Dans les paragraphes suivants, nous expliquerons le développement classique d'un réseau de neurones, que nous décomposerons en sept étapes principales :

1.13.1 Collecte des données :

Le but de cette étape est de collecter des données pour développer et tester des réseaux de neurones. Pour les applications de données réelles, l'objectif est de collecter suffisamment de données pour former une base de données représentative qui peut fonctionner lors de l'utilisation du système neuronal.

Les fonctions réalisées par les calculs statistiques, les modèles qu'il fournit, ne sont valables que dans le domaine dans lequel ils ont été ajustés. En d'autres termes, lorsque les données fournies sont complètement différentes des données utilisées dans l'apprentissage, cela conduira à des résultats complètement imprévisibles.

1.13.2 Analyse des données :

En général, il est préférable d'effectuer une analyse des données pour identifier les caractéristiques discriminantes et détecter ou distinguer les données. Ces caractéristiques sont des entrées du réseau neuronal et doivent généralement être représentées de manière représentative. La détermination des caractéristiques affectera la taille du réseau (et donc le temps de simulation), les performances du système (capacité de séparation, taux de détection) et le temps de développement (temps d'apprentissage).

L'analyse statistique des données peut aider à éliminer les valeurs aberrantes et les données redondantes.

Pour la classification, l'expérimentateur doit déterminer :

- le nombre de classes appartenant aux données ;
- la classe qui appartient à chaque donnée.

1.13.3 Séparation des bases de données :

Pour construire une application basée sur des réseaux de neurones, deux bases de données sont nécessaires. Afin de développer une application à base de réseaux de neurones, il est nécessaire de disposer de deux bases de données : l'une pour l'apprentissage et l'autre pour tester le réseau généré et déterminer ses performances. Pour contrôler la phase d'apprentissage, il est souvent préférable de disposer d'une troisième base de données, appelée « base de validation croisée », qui permet d'arrêter l'apprentissage de manière appropriée pour obtenir de bons résultats de généralisation. Le critère d'arrêt de l'apprentissage est généralement calculé en fonction de la fonction de coût, qui caractérise la différence entre la valeur de sortie obtenue et la valeur de référence (la réponse attendue pour chaque exemple donné).

Il n'y a pas de règles pour déterminer quantitativement ce partage, qui est généralement un compromis basé sur la quantité de données disponibles et le temps d'apprentissage. Cependant, chaque base de données doit satisfaire aux contraintes de représentativité de chaque classe de données et doit généralement refléter la vraie distribution, c'est-à-dire la probabilité d'occurrence de diverses classes.

1.13.4 Choix d'un réseau de neurones :

Il existe de nombreux types de réseaux de neurones, chacun ayant ses propres avantages et inconvénients. Le choix du réseau peut dépendre de :

- La tâche à réaliser (classification, association, contrôle de processus, séparation aveugle de sources...) ;
- La nature de données.

Ce choix dépend également de notre maîtrise ou connaissance de certains réseaux, ou du temps disponible pour tester des architectures plus efficaces.

1.13.5 Mise en forme des données pour un réseau de neurones :

Habituellement, la base de données doit être prétraitée pour s'adapter à l'entrée et à la sortie du réseau neuronal. Le prétraitement actuel implique la réalisation d'une normalisation appropriée, qui prend en compte l'amplitude des valeurs acceptées par le réseau.

1.13.6 Apprentissage du réseau de neurones :

Tous les modèles de réseaux neuronaux nécessitent un apprentissage. Notons que plusieurs types d'apprentissage peuvent s'adapter au même type de réseau de neurones. Les critères de choix sont généralement la vitesse de convergence ou les performances de généralisation.

Certains algorithmes d'apprentissage sont chargés de déterminer les paramètres architecturaux du réseau neuronal. Si ces techniques ne sont pas utilisées, les meilleurs paramètres d'architecture peuvent être obtenus en comparant les valeurs de performance de différentes architectures de réseaux neuronaux.

1.13.7 Validation :

Après avoir formé le réseau de neurones, il est impératif de le tester dans une base de données différente de celle utilisée pour l'apprentissage ou la validation croisée. Ce test peut évaluer les

performances du système neuronal et détecter le type de données qui ont un problème à l'origine. Si les performances ne sont pas satisfaisantes, il sera nécessaire de modifier l'architecture du réseau ou de modifier la base d'apprentissage (caractéristiques distinctives ou représentativité des données de chaque classe) [29].

Conclusion :

Dans ce chapitre, nous avons essayé de fournir un bref aperçu sur les réseaux de neurones artificiels, les différents types d'architectures et des réseaux de neurones existants, ainsi qu'une explication de l'apprentissage supervisé et non supervisé. Nous avons également parlé de la procédure de développement d'un réseau de neurones, qui fait partie de notre travail et est essentiel pour une simulation réussite. Par la suite, nous avons introduit les domaines d'utilisation des réseaux de neurones. Mais nous avons dans un premier temps parlé en détails de notre modèle de calcul de base : définition, application, caractéristiques ainsi que les utilisations des RNAs.

Contrairement aux méthodes classiques qui ont montré leurs limites, les réseaux de neurones ont montré leur tendance à s'adapter à des problèmes complexes grâce à leur grande capacité de calcul et d'apprentissage. Le grand avantage de l'RNA est qu'il peut résoudre des problèmes sans écrire de règles complexes, tout en étant tolérant pour les erreurs. Ils sont utilisés dans divers domaines, tels que : la reconnaissance de formes et le traitement d'images, le traitement de données, etc. Ce sont cependant de véritables boites noires qui ne permettent pas d'interpréter les modèles construits. Si une erreur système se produit, il est presque impossible d'en déterminer la cause.

Afin d'obtenir de bons résultats, la mise en œuvre de réseaux de neurones nécessite quelques précautions :

- Les réseaux de neurones épargnent les approximations des fonctions non linéaires, il est donc nécessaire de s'assurer que le modèle n'est pas linéaire.

- Une séquence suffisamment riche d'apprentissage, de test et de validation avec des données représentatives suffisantes ; identifiant les paramètres ayant un impact significatif sur le problème à résoudre, et un algorithme d'apprentissage puissant.

Chapitre 2

Éléments A Détecter
Et
Techniques De Détection

Introduction :

La diffusion des drones (UAVs) mini/micro constitue une menace croissante pour la sécurité nationale et commerciale. Faciles à produire, bon-marché, faciles à piloter, difficiles à détecter, les drones commerciaux sont une des menaces se diffusant le plus rapidement pour les intérêts civils et militaires.

Nous sommes dans un monde où le drone prend de la hauteur. En l'espace de quelques années les ventes de ces petits bolides volants ont explosé, ils déferlent aux quatre coins de la planète, il y aurait plus d'un million de drones civils dans le monde.

Ils sont utilisés essentiellement pour les loisirs, mais les professionnels les utilisent également : drones agricoles, drones géomètres ou drones d'urgence, ces objets volants identifiés sont partout, ils nous envahissent. Alors, que faut-il penser de cette envolée avec ce nouveau ballet aérien ? Y aura-t-il vraiment plus d'accidents ? Et si ces engins étaient utilisés pour nuire, espionner ou attaquer, comment les détecter pour s'en protéger ? Les réponses se trouvent dans ce chapitre [30] [31].

Le développement de l'intelligence artificielle ces dernières années a beaucoup fait évoluer le domaine de la détection d'objet grâce aux modèles des réseaux de neurones profond et au deep learning. Les CNN sont les meilleurs voire les plus performants des algorithmes utilisés pour classer ou localiser un flux de données d'une image enregistrée à l'aide d'un capteur visuel.

Dans ce chapitre, nous procéderons à la présentation des différents types, tailles et formes des drones. Puis, nous parlerons de la détection et de la reconnaissance des formes de ceux-ci à l'aide des CNN. Nous ferons aussi une description des CNN, et les méthodes utilisées pour l'amélioration de ses performances. Enfin, nous terminerons ce chapitre par une conclusion qui résumera les points abordés.

2.1 Notion de base sur les drones :

2.2.1 La définition d'un drone :

Un drone est un aéronef qui peut voler et effectuer des missions sans humain à bord. Cette première caractéristique de base a prouvé leur raison de désigner les véhicules aériens sans pilote (UAV). Le mot "drone" est originaire de l'anglais et signifie « bourdon », ou « bourdonnement », est couramment utilisé en français en référence au bruit que certains d'entre eux font en volant.
La désignation de drone est très restrictive puisqu'elle ne couvre qu'un véhicule aérien. En fait, il ne s'agit que d'un élément du système conçu et déployé pour effectuer une ou plusieurs tâches. C'est pourquoi les experts disent "système de drone".

Dans toutes les autres conditions étant égales par ailleurs, le principe du drone peut être comparé au modèle aéronautique, de sorte que le petit modèle puisse être manipulé par la télécommande.
Cependant, les drones se divisent en deux catégories : ceux qui nécessitent effectivement une assistance du pilote au sol, par exemple lors du décollage et de l'atterrissage, et ceux qui sont complètement autonomes. Cette autonomie de guidage peut être étendue à la prise de décision opérationnelle afin de réagir à d'éventuels événements aléatoires au cours de la mission ; c'est la deuxième caractéristique fondamentale des drones [30].

Figure 1 : le « Kettering Bug », souvent considéré comme l'ancêtre des drones modernes [32].

2.1.2 La taille d'un drone :

La taille des drones aériens va de quelques centimètres pour les modèles miniatures à quelques mètres pour les drones dédiés (surveillance, renseignement, combat, transport, loisir). Pour les drones de longue endurance, le temps de vol varie de quelques minutes à 40 heures.

Figure 2.2 : les différentes tailles de drones [32] [33].

2.1.3 La forme des drones :

De quelques dizaines de grammes à une quinzaine de tonnes, de quelques centimètres à une quarantaine de mètres, la taille et le poids des drones sont fondamentalement variables ; les performances requises par la mission d'une part et la nature et l'importance de la charge utile d'autre part sont déterminantes.

La plupart des drones sont comparables aux avions, sauf que leur forme n'est pas dictée par celle d'un fuselage qui doit loger au moins un pilote (confortablement). Il existe de nombreuses configurations de drones. Chaque machine est très différente dont certaines sont innovantes [10].

Figure 2.3 : les différentes formes des drones [32] [34].

2.1.4 Les types de drone :

La classification la plus logique des UAVs adopte la taille ou les performances (principalement porté sur l'ampleur des opérations) comme critères principaux ; éventuellement le type de mission s'il s'agit d'une spécificité particulière (usage maritime par exemple). Nous distinguons [30] [32] :

2.1.4.1 Les drones miniatures :

Cette catégorie comprend tous les drones d'une envergure inférieure à 50 centimètres, qui ne peuvent descendre que de quelques centimètres. Il est lui-même divisé en deux familles :

Mini drone : sa taille est comprise entre 15 et 50 cm, ce type de drone est spécialement utilisé pour la collecte et la transmission d'images de jour comme de nuit ce qui est très prometteur. Le coût unitaire des mini drones varie de quelques centaines à quelques milliers de dollars.

Figure 2.4 : un mini drone [32].

Micro drone : sa taille ne dépasse pas 15 cm, il pèse environ 50 grammes, avec une vitesse de croisière d'environ 50 km/h, une autonomie d'une vingtaine de minutes et un rayon d'action d'une dizaine de kilomètres. Les tâches fournies par le domaine militaire sont la reconnaissance de route, l'évaluation des dégâts ou l'observation d'objectifs fixes. Lorsqu'ils sont utilisés dans un environnement urbain, ces véhicules peuvent voler au-dessus ou autour des pâtés de maisons et peuvent même pénétrer dans les bâtiments. Par conséquent, ils sont utilisés à des fins civiles, profitant ainsi aux pompiers, à la police ou à tout autre personnel fournissant des services de sécurité.

Figure 2.5 : un micro drone [35].

2.1.4.2 Les drones de court rayon d'action :

Aussi connus sous le nom de drones TCP, ou dans le langage militaire "drones de capitaine" sont conçus pour voir au-dessus des collines à quelques kilomètres. Leur envergure est de 0,5 à 2 mètres, ils ont généralement des ailes fixes et leur vitesse est faible (une dizaine de kilomètres par heure) et il est difficile d'éviter les obstacles.
Les évolutions technologiques requises dans cette catégorie concernent principalement les capteurs (évitement d'obstacles et caméra miniaturisée jour / nuit).

Figure 2.6 : un drone européen à long rayon d'action pour les militaires [36].

2.1.4.3 Les drones tactiques à moyen rayon d'action :

Puisqu'ils peuvent être dotés de vitesse basse (150 km/h) ou rapide (700 km/h), ces drones représentent la classe moyenne avec des performances variables. La masse au décollage est inférieure à une tonne, le rayon d'action est de 30 à 500 kilomètres, la hauteur peut atteindre 5000 mètres avec une bonne endurance de 2 à 8 heures. Nous ne testerons pas notre programme sur ce type de drone puisqu'il s'agit d'un grand drone.

Figure 2.7 : drone tactique à moyen rayon d'action [30].

2.1.4.4 Les drones maritimes tactiques :

La particularité des drones maritimes aéroportés vient de la double restriction d'utilisation, qui nécessite des adaptations techniques complexes : il nécessite une autonomie considérable (au moins 5 heures), et il doit être capable d'atterrir avec un vent fort sur une plateforme étroite, partiellement entouré d'obstacles et soumis à des mouvements de grande amplitude, se balançant et roulant dans une mer forte.

Figure 2.8 : drone tactique maritime [37].

2.1.4.5 Les drones à voilure fixe :

Pour ce type de drone, la capacité à résister à la gravité peut être assurée par la présence d'une ou plusieurs ailes rigides. C'est le profil spécifique de d'aile qui provoque la portance lorsque l'aéronef est exposé au vent relatif. Les drones standards en forme d'avion et une série d'équipements de forme plus originale entrent dans cette catégorie.

Lorsque les ailes ne se distinguent pas du corps de l'aéronef, elles sont généralement appelées « aile volante » (voir figure 2.9).

Les drones à voilure fixe ont l'avantage de pouvoir voler sur de longues distances et sont très adaptés aux tâches de cartographie. Cependant, la nécessité de maintenir une vitesse de déplacement minimale et un contrôle limité de leurs déplacements ne permettent pas de manœuvres précises autour des objets.

De plus, leurs décollages nécessitent une vitesse horizontale initiale. Il faut donc les lancers pour commencer à voler.

Figure 2.9 : drones dits « aile volante » [32].

2.1.4.6 Les drones à voilure tournante :

S'il s'agit d'un drone à voilure tournante, l'aéronef sera maintenu en l'air par un ou plusieurs rotors. Le corps principal de chaque hélice est parallèle au sol et exerce une force verticale sur l'air lors de la rotation. Cette catégorie comprend des appareils similaires aux hélicoptères, mais aussi des drones aux formes plus spécifiques.

Le principal avantage des drones à voilure tournante est qu'ils ont la capacité de rester en vol stationnaire, ce qui permet de prendre des photos dans des conditions de meilleure stabilité, et ainsi d'augmenter l'angle de vue des objets d'intérêt.

Les hélicoptères sont des équipements bien connus dans l'aviation traditionnelle. Pour ce type d'aéronef a voilure tournante, c'est un rotor unique appelé « rotor principal » qui maintient le drone en l'air. Ceci empêchera l'aéronef de tourner sur lui-même lorsque le rotor principal tourne, selon le principe de l'action et de la réaction, un rotor de queue ou rotor dit « anti-couple » doit être prévu (voir figure 2.10).

Figure 2.10 : l'hélicoptère est un sous-type de drone à voilure tournante [32].

Les multicoptères, également appelés « multicopters » en anglais, sont très populaires. Ils sont équipés de plusieurs bras avec les mêmes hélices fixées à l'extrémité. Le nombre d'hélices et leur configuration peuvent varier considérablement (voir figure 2.11). Cela montre que le sens de rotation de l'hélice est toujours alterné.

En fait, si toutes les hélices tournaient dans la même direction avec la même vitesse, l'appareil tournerait naturellement autour du centre de masse. La configuration et le nombre des hélices affectent la stabilité, l'autonomie et la sécurité du vol.

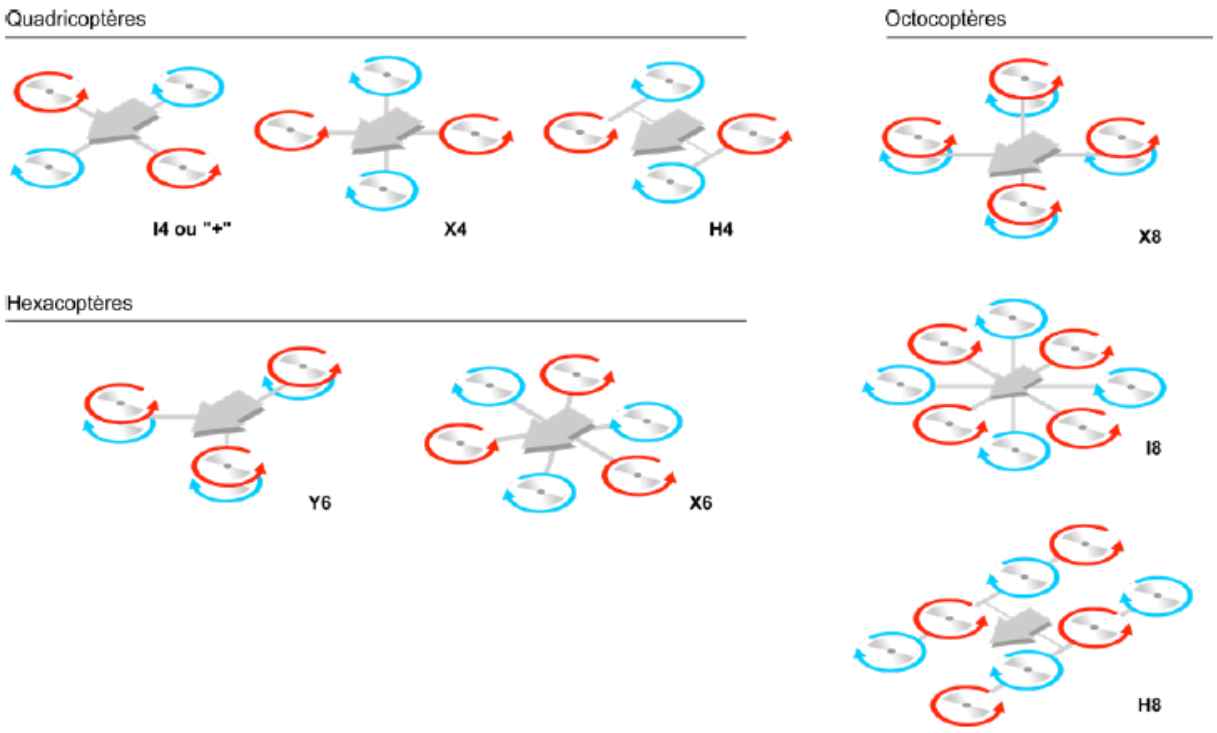


Figure 2.11 : les configurations fréquentes des hélices dites multicoptères [32].

Figure 2.12 : drone quadricoptère X4 [38].

Figure 2.13 : drone octocoptère I8 [39].

En raison de leur bon rapport qualité / prix, les multicoptères à quatre hélices ou quadricoptères (voir figure 2.12) connaissent un grand succès. Leur conception est très simple, mais si l'un des rotors est arrêté ou endommagé, ils ne pourront pas continuer à voler de manière stable, il est donc compréhensible qu'ils ne soient pas conseillés pour des missions impliquant des vols près ou au-dessus des gens.

Les hélices sont parfois divisées en deux parties sur chaque bras du drone, comme dans le modèle Y6 ou X8 représenté sur les figures 2.14 et 2.15, ce qui améliore la sécurité du vol. Si l'un des rotors coaxiaux tombe en panne, l'avion peut encore voler de manière stable.

Figure 2.14 : drone hexacoptère Y6 [40].

Figure 2.15 : drone octocoptère X8 [41].

En général, on peut dire que les hexacoptères (six moteurs) et les octocoptères (huit moteurs comme le montre la figure 2.13) sont utilisés par des professionnels.

2.1.4.7 Les drones de longue endurance :

Le temps de vol est compris entre 12 et 48 heures, ils appartiennent donc à la catégorie des « grands » drones, dont la taille est principalement déterminée par la charge utile et la grande quantité de carburant nécessaire à la mission. Selon la hauteur de vol de l'engin, la catégorie elle-même est divisée en deux parties : comme un avion, plus l'altitude de vol est élevée, plus la vitesse de vol est rapide et plus la distance parcourue est importante. Nous ne serons pas intéressés par ce genre de drone qui ressemble beaucoup à un avion et qui est très gros. Nous nous concentrerons uniquement sur les mini et micros drones.

Les drones MALE :

L'altitude de vol pour cette catégorie varie de 5 000 à 12 000 mètres, ce qui permet une distance allant jusqu'à 1 000 km. Les MALE opérationnels les plus connus sont le Hunter et le Predator américain.

Figure 2.16 : drone MALE [42].

Les drones HALE :

Dans cette catégorie, les dimensions de HALE atteignent la taille des avions de transport de classe Airbus A320, offrant une autonomie sur une distance de plusieurs milliers de km (10 000 km et plus), volant au-dessus du trafic aérien ordinaire (civil et militaire) jusqu'à 20 000 m d'altitude.

Les capacités de ces drones doivent être comparées aux capacités des avions sans pilote tels que les avions espions ou les avions de renseignement électronique Sigint, et aux capacités des satellites d'observation ou d'alerte.

Figure 2.17 : drone HALE [43].

2.1.4.8 Les drones de combat :

L'UCAV est conçu comme un véritable avion de combat sans pilote. Il s'agit bien sûr de drone offensif équipé de missiles ou de bombes guidées, dont la tâche est de réaliser des missions d'attaque au sol très précises, et même à long terme, ils peuvent également assurer la défense aérienne en tant que force de police aérienne.

Figure 2.18 : drones de combat UCAV [44] [45].

Comme nous l'avons déjà mentionné dans les paragraphes précédents, nous n'allons pas nous intéresser à ce type de drones qui sont très similaires aux avions et qui sont très volumineux. On se concentrera uniquement sur les mini et micros drones.

- ***Les autres types de drones :***

En plus des ailes volantes et des multicoptères, qui ont déjà conquis de nombreuses industries, il existe toute une série de drones moins connue. Les drones hybrides en font partie. Ils essaient de combiner le meilleur des deux modes, Autrement dit, combiner la vitesse de déplacement et

l'autonomie de l'aile volante avec la capacité de maintenir un vol stationnaire. Dans la figure ci-dessous, nous présentons un exemple de forme hybride de drone nommé VertiKUL.

Figure 2.19 : drone hybride appelé vertiKul [46].

Il existe encore des types de drones plus originaux. Parmi eux, on peut citer les drones en forme de ballon et même les « ornithoptères » qui volent grâce au battement d'ailes, comme un insecte, un oiseau ou une chauve-souris (voir figure 2.20).

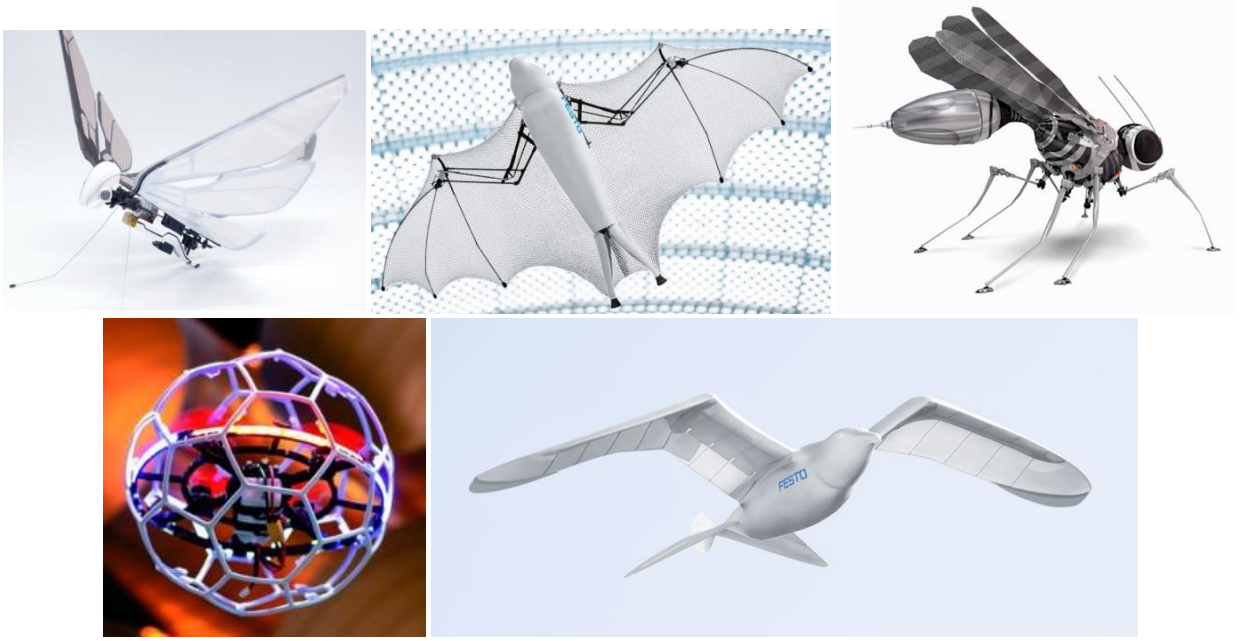


Figure 2.20 : drone de forme de ballon et drones ornithoptères [47] [48] [49] [33].

2.2 La détection d'un objet :

2.2.1 Définition :

La détection d'objets est un domaine de recherche très actif visant à classer et localiser des zones / régions d'une image ou d'un flux vidéo. Ce domaine est situé à l'intersection des deux autres : classification de l'image et localisation de l'objet. Pour avoir une bonne méthode de détection

d'objets, il est nécessaire de disposer d'un algorithme de détection de zone puissant et d'un bon algorithme de classification d'image.

Ci-dessous quelques méthodes utilisées pour la détection d'objets :

- **Détection d'objet basée sur les caractéristiques :** Il s'agit d'un ensemble de techniques et de méthodes conçues pour trouver des zones d'image avec des caractéristiques locales importantes, telles que des changements soudains et locaux d'intensité, des changements de texture, des points spécifiques, etc.

- **Détection d'objet Viola Jones :** C'est une méthode de détection d'objets dans des images numériques proposée par les chercheurs Paul Viola et Michael Jones en 2001. Ils ont proposé l'utilisation de fonctionnalités, c'est-à-dire la synthèse et la représentation d'informations calculées à partir de valeurs de pixels. Viola et Jones ont défini une fonction très simple, la pseudo-Haar, qui est calculée par la différence entre la somme des pixels de deux ou plusieurs régions rectangulaires adjacentes.

- **Classifications SVM avec caractéristiques HOG :** Une autre méthode traditionnelle et similaire consiste à utiliser HOG (Histogramme des Gradients Orientés) et SVM (Support Vector Machine) pour la classification. Bien qu'il soit meilleur que Viola-Jones, il nécessite tout de même une fenêtre coulissante de plusieurs échelles.

- **Détection d'objet d'apprentissage en profondeur :** Cette méthode utilise le réseau de neurones convolutifs pour faire apprendre à la machine les caractéristiques intrinsèques de l'objet afin qu'il puisse être reconnu.

  Nous utiliserons cette méthode de détection d'objets dans notre projet pour classer les images d'un drone.

2.2.2 Applications de détection d'objets :

- ✓ Contrôle de la qualité industrielle : la détection d'objets est également utilisée dans les processus industriels pour identifier les produits. La recherche d'objets spécifiques par inspection visuelle est une tâche de base qui implique de multiples processus industriels tels que le tri, la gestion des stocks, le traitement, la gestion de la qualité, l'emballage ;

- ✓ la reconnaissance faciale ;

- ✓ la sécurité : que ce soit pour l'identité faciale, d'appels, ou de scan,… ;

- ✓ voitures autonomes.

2.2.3 Principe de la reconnaissance des formes :

La reconnaissance des formes est la science de la définition d'algorithmes pour déterminer quelles formes ou objets observés sont similaires, ou en d'autres termes, les types d'objets connus qui leur sont associés.

Dans la littérature on distingue deux types de RDF :

- **RDF Structurelle** : représentation des formes à l'aide de grammaires ;
- **RDF Statistique** : représentation numérique des formes ; c'est ce que nous utiliserons dans notre étude car c'est le plus utilisé dans notre cas. Dans la reconnaissance des formes, chaque observation est représentée par un vecteur X de D paramètres réels, appelé vecteur forme, tel que :

$$X^i = [X_1 \dots \dots \dots X_D^i]^T$$

Ce vecteur sera représenté par un point dans cet espace qui est connu sous le nom espace de représentation (figure 2.21). Les paramètres de ce vecteur traduisent l'état du système étudié. Ils sont le résultat de l'analyse des signaux mesurés par les capteurs installés sur le système.

La reconnaissance des formes a pour objectif de décider à laquelle des classes w1 w2 w3 doit être associé une nouvelle forme ; autrement dit chercher dans quelle zone de l'espace se situe la nouvelle observation (figure 2.22). On peut toujours faire le parallèle avec l'objectif de la reconnaissance de drone qui définit à quel mode de fonctionnement correspond une nouvelle observation [9].

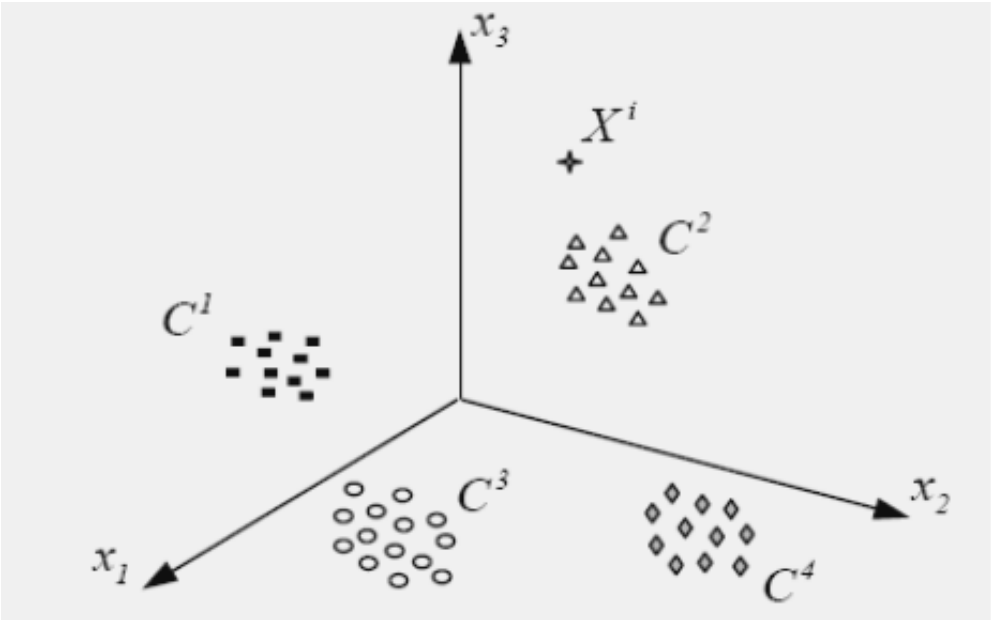


Figure 2.21 : caractéristique d'une observation par un vecteur forme représenté par un point dans R^D (D=3) [9].

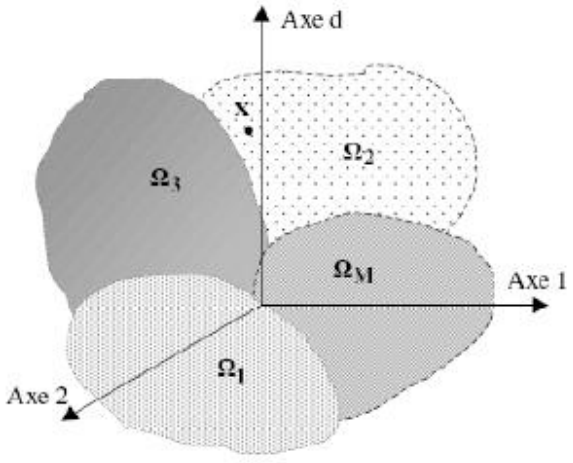


Figure 2.22 : objectif en reconnaissance des formes : associer une nouvelle observation X à l'une des classes [9].

2.3 Notion de base :

2.3.1 Définition d'une image :

Une image est une représentation plane d'une scène ou d'un objet généralement situé dans un espace tridimensionnel. Elle est produite par le contact de la lumière émise par les objets formant la scène avec un capteur (caméra, scanner, radiographie,). C'est en fait une représentation spatiale de la lumière.

L'image est un ensemble de points auquel est affectée une grandeur physique (luminance, couleur). Ces grandeurs peuvent être continues (image analogique) ou bien discrètes (images digitales). Mathématiquement, l'image représente une fonction continue IF, appelée fonction image de deux variables spatiales représentée par IF (x, y), qui a pour objectif de mesurer la nuance du niveau de gris de l'image aux coordonnées (x, y).

La fonction Image peut être représentée sous la forme suivante :

$IF : R^2 \rightarrow R$ avec R: ensemble des réelles.
$(x, y) \rightarrow IF(x, y)$ x et y : deux variables réelles [50].

2.3.2 Les différents types de format d'image :

- **Image couleur RVB :** l'œil humain utilise trois types de photocellules qu'on appelle cellules « cônes » pour analyser la couleur. Ils sont sensibles aux basses, moyennes ou hautes fréquences (rouge, vert, bleu). Par conséquent, pour représenter la couleur du pixel, nous devons donner trois nombres, qui correspondent aux doses des trois couleurs de base : rouge, vert et bleu. Ainsi, une image couleur peut être exprimée par trois matrices correspondant respectivement aux couleurs primaires.

- **Image d'intensités :** c'est une matrice où chaque élément est un nombre réel compris entre 0 (noir) et 1 (blanc). On parle aussi d'images en niveaux de gris, car des valeurs comprises entre 0 et 1 indiquent des niveaux de gris différents.

- **Image binaire :** c'est une matrice rectangulaire dont l'élément vaut 0 (noir) ou 1 (blanc) [50].

2.3.3 Caractéristiques de l'image :

L'image est un ensemble structuré d'information caractérisé par les paramètres suivants :

2.3.3.1 Pixel :

Le Pixel est une abréviation du mot "Picture element", une unité de surface utilisée pour définir la base d'une image numérique. Il implémente un point donné (x, y) sur le plan d'image.

L'information affichée par le pixel est le niveau de gris (ou couleur) obtenu à partir de la position correspondante dans l'image réelle. La différence entre une image monochrome et une image couleur réside dans la quantité d'informations contenues dans chaque pixel. Par exemple, dans une image couleur (RVB : rouge, vert et bleu), la valeur de pixel de chaque couleur est représentée par trois octets.

2.3.3.2 Dimension & Résolution :

La dimension est la taille de l'image. Elle prend la forme d'une matrice dont les éléments sont des valeurs numériques qui représentent des intensités lumineuses (pixels). Le nombre de lignes de cette matrice multipliée par le nombre de colonnes donne le nombre total de pixels dans une image.

Par contre, la résolution est la clarté ou la finesse des détails obtenus par un moniteur ou une imprimante lors de la génération d'une image. Sur un écran d'ordinateur, la résolution est exprimée en nombre de pixels par unité de mesure (pouces ou centimètres). Le terme résolution est également utilisé pour indiquer le nombre total de pixels horizontaux et verticaux sur le moniteur. Plus le nombre est élevé, meilleure est la résolution.

2.3.3.3 Voisinage :

Le plan de l'image est divisé en termes de formes rectangulaires ou hexagonales qui permettent ainsi d'exploiter la notion de voisinage (voir figure 2.23). Le voisinage d'un pixel est formé par l'ensemble des pixels se situant autour de lui. On définit aussi l'assiette (écran plat) comme étant l'ensemble des pixels caractérisant le voisinage pris en compte autour d'un pixel.

On distingue deux types de voisinage :

- **Voisinage à 4** : On ne prend en considération que les pixels qui ont un coté commun avec le pixel considéré.
- **Voisinage à 8** : On prend en compte tous les pixels qui ont au moins un point en liaison avec le pixel considéré.

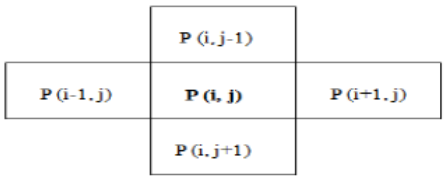


Figure 2.23 : voisinage à 4 [50].

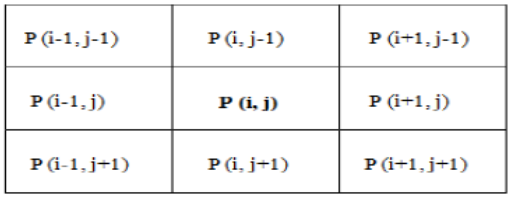


Figure 1.24 : voisinage à 8 [50].

2.3.3.4 Niveau de gris :

Le niveau de gris indique la luminosité d'un pixel, sa valeur minimale est 0, mais sa valeur maximale dépend de la profondeur de numérisation de l'image. Dans une image couleur avec une profondeur de 8 bits, le nombre de niveaux de gris est de 255. Chaque pixel peut adopter un niveau entre 0 et 255, contrairement à une image binaire où les pixels peuvent prendre seulement deux possibilités : soit 0, soit 255.

Pour le cas d'une image en couleurs non linéaires ou bien une image vue à partir d'un écran vidéo (avec une correction gamma). Chaque pixel peut être calculé par la formule suivante :

$$Gris = 0{,}299 * composante\ rouge + 0{,}587 * composante\ verte + 0{,}114 * composante\ bleue.$$

La formule tient compte de la sensibilité de l'œil à la couleur. Ainsi la présentation des niveaux de gris devient indépendante de la couleur et se limite uniquement à la luminosité des pixels individuels. La somme des trois coefficients vaut 1 et on remarque une forte inégalité entre ceux-ci, une lumière verte apparaît plus claire que les autres lumières rouges et bleues.

2.3.3.5 Contraste :

C'est l'opposition évidente entre les deux zones de l'image. L'image de contraste montre une bonne répartition dynamique des valeurs de gris dans toutes les plages de valeurs possibles, avec un blanc très clair et un noir profond. En revanche, les images à faible contraste ont une plage dynamique inférieure et la plupart des pixels ont des valeurs de gris très proches.

Le contraste global d'une image est défini par Michelson comme suivant :

Si L1 et L2 sont la luminosité de deux zones adjacentes A1 et A2 de l'image, le contraste est alors défini par le rapport suivant :

$$C = \frac{L1 - L2}{L1 + L2}$$

Ce contraste de Michelson est compris entre 0 et 1 avec une modulation analogique [51].

2.3.3.6 Luminance :

C'est le degré de luminosité des points dans une image, c'est-à-dire la clarté des couleurs. Elle est également définie comme étant le quotient de l'intensité lumineuse d'une surface par l'aire apparente de cette surface.

Une bonne luminance se caractérise par :

- Des images lumineuses (brillantes) ;

- Un bon contraste pour éviter les images dans lesquelles la gamme de contraste tend vers le blanc ou le noir ; ces images entraînent des pertes de détails dans les zones sombres ou lumineuses ;
- Pas de parasites.

2.3.3.7 Bruit :

Le bruit (parasite) dans l'image est défini lorsque l'intensité d'un pixel change très fortement par rapport à d'autres pixels adjacents, il provient de l'éclairage des dispositifs optiques et électroniques du capteur. C'est un parasite qui représente certains défauts (poussière, petits nuages, baisse momentanée de l'intensité électrique sur les capteurs…). Il se traduit par des taches de faible dimension et dont la distribution sur l'image est aléatoire.

2.3.3.8 Contour :

Les contours représentent la frontière entre les objets de l'image ou bien entre l'objet et l'arrière-plan, en d'autres termes, c'est la limite entre deux pixels adjacents ou le niveau de gris est significativement différent.

Dans les images numériques, les contours sont situés entre des pixels appartenant à des zones d'intensités moyennes différentes. Ce sont des profils de type "saut d'amplitude". Le profil peut également correspondre à des changements locaux d'intensité maximale ou minimale. C'est le contour du "toit".

2.3.3.9 La vidéo :

La vidéo est une image continue à une certaine vitesse. L'œil humain a la particularité de visualiser environ 20 images par seconde. Ainsi, en affichant plus de 20 images par seconde, il est possible de tromper les yeux et de leur faire croire que les images sont en mouvement. La fluidité de la vidéo est caractérisée par le nombre d'images par seconde (frame rate), exprimé en FPS (frames par seconde).

D'un autre côté, la vidéo au sens multimédia a généralement du son, c'est-à-dire des données audios [52].

2.4 La reconnaissance d'images :

Pour un ordinateur, la classification des images par leur contenu (par exemple, avions, voitures, drones ou autres objets) s'avère très compliqué. Pour cela, l'ordinateur procède à une analyse pour identifier les caractéristiques de chaque image.

En deep learning, les informations d'entrée (ici, les images) seront analysées couche par couche. Par exemple, la première couche d'un réseau de neurones artificiels peut vérifier la couleur des pixels d'une image. Ensuite, chaque pixel est traité par un neurone différent. Dans la couche suivante, les contours et les formes seront finalement reconnus, et des caractéristiques encore plus complexes seront reconnues bientôt dans la couche qui suit.

Un algorithme flexible est ainsi créé à partir des informations collectées et cartographiées. Les résultats de l'analyse d'une couche seront transférés à la couche suivante et l'algorithme sera modifié dans le processus. Par conséquent, l'ordinateur peut déterminer si l'image appartient à la catégorie « drone », « avion » ou « voiture » après de nombreuses opérations.

Au début du processus, des instructions sont entrées manuellement dans le système pour corriger les erreurs d'allocations, alors que l'algorithme subit des modifications. Ce n'est qu'après cette étape que le système peut automatiquement améliorer ses capacités de reconnaissance d'image.

En modifiant les connexions établies entre les neurones du réseau et en ajustant les poids des variables dans l'algorithme, nous pouvons nous assurer que les changements dans les modèles d'entrée (par exemple, des images de drones présentées de façon différentes) aboutissent de plus en plus au même modèle de sortie (détection de drone) sans erreur. Plus nous consacrons de temps à « l'apprentissage » du système d'imagerie, plus ce dernier s'améliore.

Pour les observateurs externes, il n'est pas toujours possible de comprendre les motifs reconnus par les ordinateurs d'apprentissage profond qui mènent à leurs décisions. C'est d'autant plus difficile que le système lui-même optimise constamment ses propres règles de décision.

2.4.1 Le deep learning :

Le deep learning ou apprentissage profond est un type d'intelligence artificielle dérivé du machine learning (apprentissage automatique) où la machine est capable d'apprendre par elle-même, contrairement à la programmation où elle se contente d'exécuter à la lettre des règles prédéterminées.

2.4.2 Le fonctionnement du deep learning :

Son fonctionnement implique des ordinateurs utilisant des réseaux de neurones artificiels qui imitent la structure du cerveau humain pour traiter de grandes quantités de données. Chaque fois que de nouvelles informations sont croisées, les connexions existantes entre les neurones changent et s'étendent. Sa fonction est de permettre au système d'apprendre des choses de manière autonome sans intervention manuelle, tout en améliorant ses performances ainsi que ses qualités de décision et de prévision.

Un réseau neuronal artificiel composé d'une couche neurale d'entrée, d'une ou plusieurs couches intermédiaires et d'une couche de sortie est utilisé pour traiter les informations autrement dit un perceptron multicouche (PMC). Les informations entrantes ou le vecteur d'entrée arrive à la couche d'entrée. Ensuite, il est transmis à la couche intermédiaire par des neurones artificiels, où il reçoit un « poids ». Enfin, un certain schéma est adopté lors de l'atteinte du niveau de sortie. Plus un réseau de neurones artificiels contient de couches, plus l'intelligence artificielle peut supporter la complexité.

L'un des algorithmes les plus performants du deep learning est les réseaux de neurones convolutionnels.

Le programme est conçu sur les réseaux de neurones convolutifs d'apprentissage en profondeur CNN et l'apprentissage par transfert, mais afin d'améliorer les performances de CNN, il est

nécessaire d'ajouter certaines méthodes, telles que la régularisation du décrochage et l'augmentation des données. Tout cela sera expliqué dans les paragraphes suivants.

2.4.3 Introduction au CNN :

Les réseaux convolutifs, de l'anglais convolutional neural networks (CNN), sont une forme particulière de réseaux neuronaux multicouches dont la conception est inspirée de l'architecture des cortex visuels des organismes vivants des mammifères.

Ces réseaux de neurones peuvent classer les informations des plus simples aux plus complexes. Ils sont composés de plusieurs couches de neurones et de plusieurs fonctions mathématiques avec des paramètres ajustables, qui peuvent prétraiter une petite quantité d'informations. Les CNN sont caractérisés par leur première couche convolutive (généralement une à trois couches). Comme son nom l'indique, la couche convolutive est basée sur le principe mathématique de la convolution et tente de reconnaître la présence d'un motif (par exemple, dans un signal ou une image) [53] [14].

Pour les images, le premier calque convolutif peut détecter les contours des objets (tels que les cercles), le second calque convolutif peut transformer les contours en objets (tels que les roues), et les calques suivants (pas nécessairement des convolutions) peuvent utiliser ces contours pour distinguer les informations des voitures et motos. L'étape d'apprentissage des objets connus permet de trouver les meilleurs paramètres en affichant la machine, par exemple, des milliers d'images de chiens, de voitures ou de sports. L'un des défis est de trouver un moyen d'ajuster ces paramètres aussi rapidement et efficacement que possible. Les réseaux de neurones convolutifs ont de nombreuses applications dans la reconnaissance d'images, de vidéos ou de traitement du langage naturel [53].

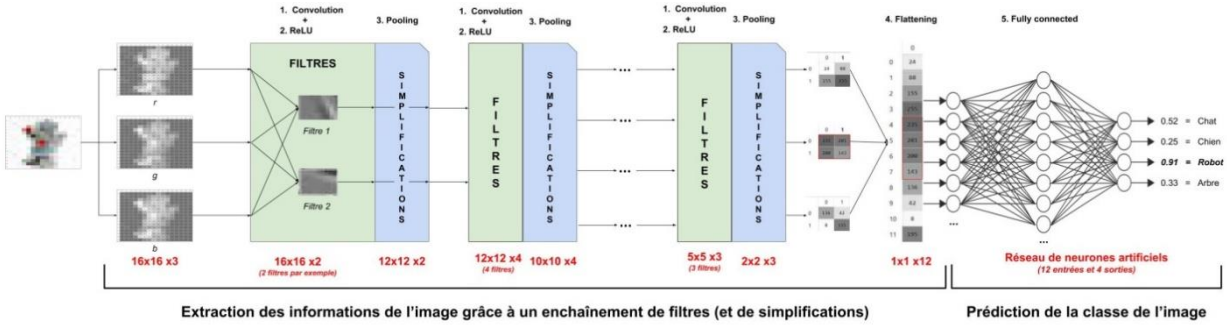


Figure 2.25 : fonctionnement d'un réseau de neurones [53].

2.4.4 Les couches d'un réseau de neurones convolutifs :

Pour exécuter un réseau neuronal convolutif, nous avons besoin de quatre types de couches : couche convolutive, couche de pooling, couche de correction ReLU et couche entièrement connectée (fully connected).

2.4.4.1 La couche de convolution :

La convolution est le cœur d'un réseau de neurones convolutifs. À l'origine, une convolution est une formule mathématique (on parle de produit de convolution) qui est souvent utilisée pour la modification des images car il permet de faire ressortir les caractéristiques suivantes :

- La mise en évidence des traits verticaux, horizontaux, diagonaux… ;
- Le flou d'image (blur), lissage des textures ou à l'inverse "sharpen" ;
- L'inversion de couleurs ;
- etc.

Ces caractéristiques ne peuvent être obtenues que lorsqu'un bon filtre est utilisé. En fait, la convolution prend juste une image et un filtre (qui est une autre image) comme entrée, effectue des calculs et renvoie une nouvelle image (généralement plus petite).

➢ **Filtre :**

Pour utiliser le filtre d'un réseau de neurones convolutifs, il faut prédéfinir ces paramètres qui sont le kernel, le stride, et le padding, leurs valeurs sont générées aléatoirement lors de l'initialisation. Au fur et à mesure que le réseau apprend, les valeurs des filtres sont mises à jour pour améliorer les résultats du CNN. Par conséquent, la valeur du filtre fait partie des variables (poids, biais...) qui changent au cours du processus d'apprentissage du réseau.

Il faut noter que plus la convolution se produit tardivement dans le CNN, plus les filtres deviennent complexes, plus la détection des formes est difficile. En effet, après quelques applications de filtres, l'image ne ressemble plus à l'image d'origine (seules les informations "principales" laissées par le filtre précédent sont conservées). Là où la première convolution montrera des lignes verticales, la dernière convolution pourra trouver la structure en nid d'abeille (en raison de la conversion d'image) : le CNN accumulera des informations détaillées pour trouver l'idée globale [14].

- **Les types de convolution :** il existe plusieurs types de convolutions, même si en général on utilise celle de base, il peut s'avérer utile de connaître les outils à notre disposition.

- **Convolution classique :** c'est celle dont nous avons parlé ci-dessus et qui est la plus utilisée. Elle contient trois paramètres : la taille de la convolution (appelée **kernel** et souvent en 3×3), le **stride** qui représente le décalage du kernel entre chaque calcul, et le **padding** qui est la manière dont on peut "dépasser" de l'image pour appliquer la convolution.

- **Dilated convolution**, identique à la convolution précédente sauf que le kernel est éclaté (on prend, par exemple, un pixel sur deux pour calculer la convolution avec un paramètre supplémentaire qui est la **dilation rate**, il représente le nombre de pixels à ignorer.

- **Transposed convolution**, qui construit la sortie comme si on inversait une convolution sur l'image [53].

Ces images montrent comment les différentes convolutions fonctionnent :

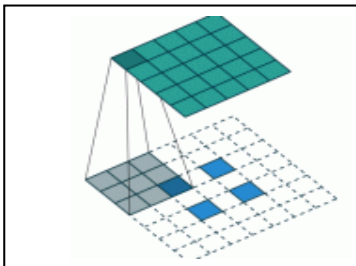


Figure 2.26 : dilated convolution [53].

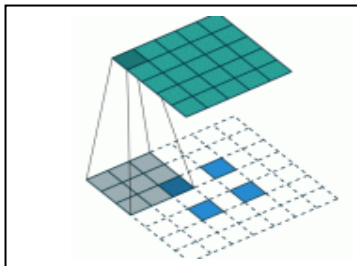


Figure 2.27 : convolution classique [53].

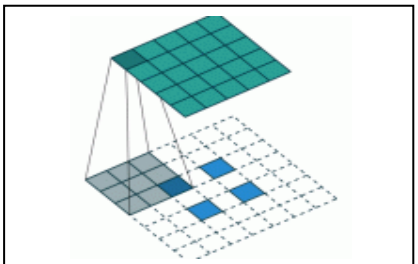


Figure 2.28 : transposed convolution [53].

- **Separable convolution** :

C'est une convolution qui peut être décomposée en deux ou plusieurs convolutions simples, ce type de convolution n'est employé que rarement. Elle fonctionne comme suit :

Au début on applique une convolution classique sur chaque Channel pour avoir 3 images issues de l'image originale décomposée en RGB. Ensuite ces 3 images fusionnent (pixel à pixel) pour former le résultat final de la convolution. Le nombre de calculs est donc élevé (rappelons qu'on a souvent 64, 128, 256 ou 512 channels). Cette technique (cas de convolution séparable utilisée en deep learning) a pour particularité de pouvoir être calculée directement sur tous les channels en même temps. Par conséquent, on calculera la convolution d'un seul pixel sur tous ses canaux, ce qui nous donnera une seule valeur. On fera de même avec les autres pixels, un à un (toujours transversalement c'est à dire en profondeur), et ça aboutira sur une image directe (pas besoin d'en ajouter beaucoup à la fin). Cette décomposition a pour but de gagner une partie des calculs.

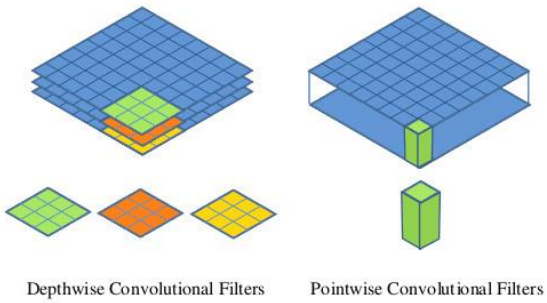


Figure 2.29 : les deux techniques de la séparable convolution [53].

2.4.4.2 La couche de rectification linéaire :

La fonction d'activation ReLU ou la fonction de "rectification" très utilisée en Deep Learning. Cette fonction est souvent appliquée à la sortie d'une couche de convolution dans un réseau de neurones convolutionnels pour de multiples raisons :

- Les valeurs obtenues à la sortie d'une couche de convolution sont linéaires par rapport à celles d'entrées à cause des opérations réalisées dans cette couche (addition /multiplications)

pour cela l'utilisation de la fonction ReLU est recommandée, elle peut briser une partie de la linéarité en supprimant une partie des valeurs (toutes celles négatives) donc elle a comme avantage d'accélérer les calculs ;

- Dans l'image, la linéarité n'est que peu, voire pas significative (par exemple, le changement entre les valeurs de pixels peut être important dans une zone, l'image a des coins...).

En ne modifiant pas les données positives, la ReLU n'affecte pas les traits qui sont mis en évidence par la convolution, au contraire, elle les met davantage en évidence en creusant l'écart (valeurs négatives) "entre" deux caractéristiques et il est possible de faire cela puisque la fonction ReLU remplace toutes les valeurs négatives par la valeur zéro.

Le ReLU est un meilleur choix pour l'utiliser dans notre programme, il existe d'autre ReLU tel que le Leaky ReLU: il detruit la linéarité des données dans la moindre mesure.

2.4.4.3 La couche de pooling :

Le pooling est une opération qui simplifie une image, il s'agit de remplacer un carré de pixels par une seule valeur (généralement 2×2 ou 3×3). De cette manière, la taille de l'image est réduite et simplifiée (lissée).

Pour appliquer le pooling, on commence par sélectionner un carré de pixels de taille 2×2 (pour un pooling de 2×2). Ensuite, on calcule la valeur qui remplacera ce carré. Puis, on décale ce carré vers la droite d'une case par exemple si le stride (pas) vaut 1. Généralement *le* stride vaut 1 ou 2.

On doit choisir parfaitement le stride pour que la sélection d'image s'adapte partout sans erreur de calcul, même avec des pixels manquants.

Une fois arrivé au bout à droite, on recommence tout à gauche en refaisant les mêmes étapes mais cette fois-ci en décalant une fois vers le bas (d'un pas égal au stride).

Il existe plusieurs types de pooling :

- Le "***max pooling***", il prend la valeur maximale de la sélection. C'est le type de pooling le plus utilisé pour son avantage de calcul (immédiat), et permet de simplifier efficacement l'image.
- Le "***mean pooling***" (ou average pooling), qui prend la moyenne des pixels de la sélection : on calcule la somme de toutes les valeurs puis on divise par le nombre de valeurs. On obtient donc une valeur intermédiaire pour représenter ce lot de pixels.
- Le "***sum pooling***", c'est la moyenne sans avoir divisé par le nombre de valeurs (on ne calcule que leur somme) [53] [14].

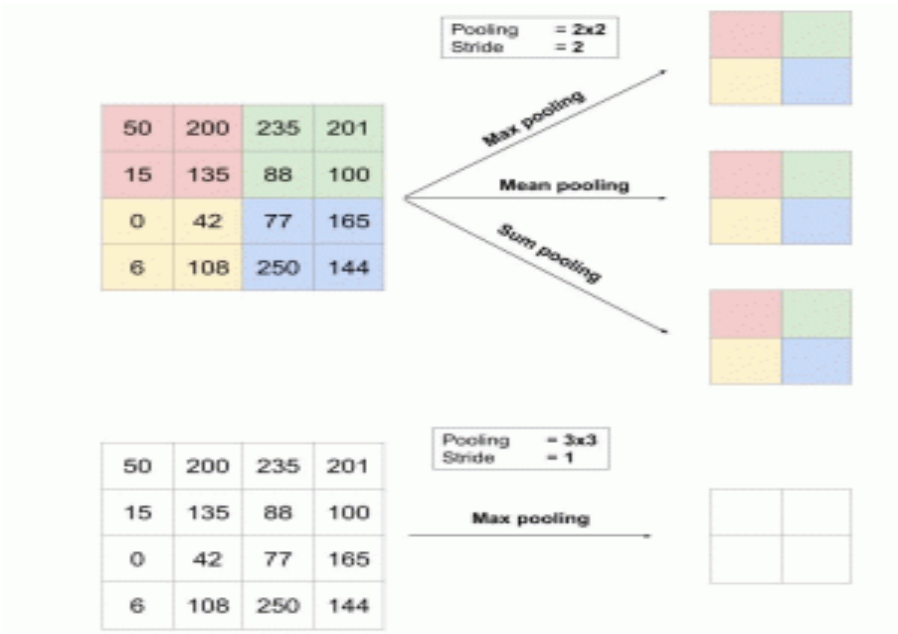


Figure 2.30 : calcul du pooling sur une image 4×4. Un pooling de 2×2 signifie que l'on sélectionne les pixels en carrés de 2×2. Le stride indique de combien de cases décaler le carré à chaque fois [53].

2.4.4.4 La couche de fully connected :

En anglais fully connected layer « FC » s'applique aux entrées précédemment aplaties, où chaque entrée est connectée à tous les neurones. Les couches fully-connected apparaissent généralement à la fin de l'architecture CNN et peuvent être utilisées pour optimiser les objectifs, tels que les notes des cours.

- **Le flattening :**

La Dernière étape de la partie "extraction des informations" est l'aplatissement ou bien le flatting en anglais qui consiste simplement à mettre bout à bout toutes les images (matrices) que nous avons pour en faire un (long) vecteur. Récupérer les pixels ligne par ligne (en fait, ce ne sont plus des images ou des pixels, mais des tableaux de nombres, donc les pixels sont ces nombres) et les ajouter au vecteur final [51].

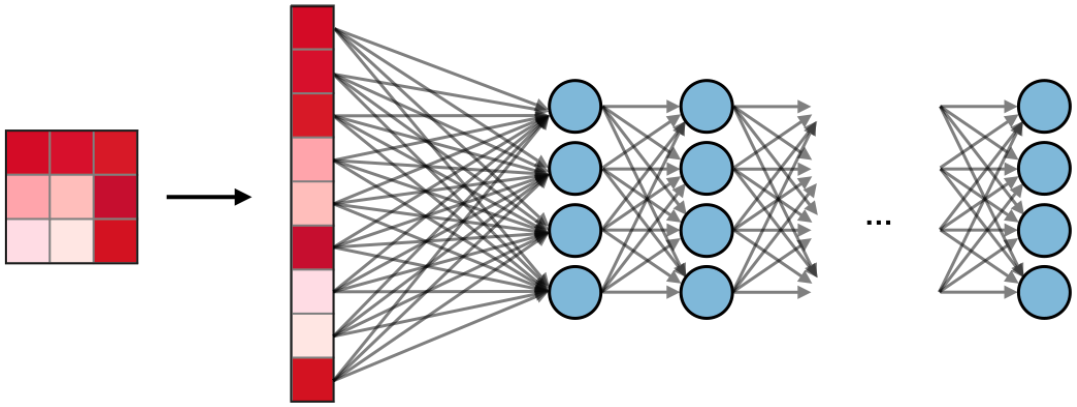


Figure 2.31 : la mise à plat des pixels après la couche de pooling [14].

2.4.5 Modèle VGG -16 :

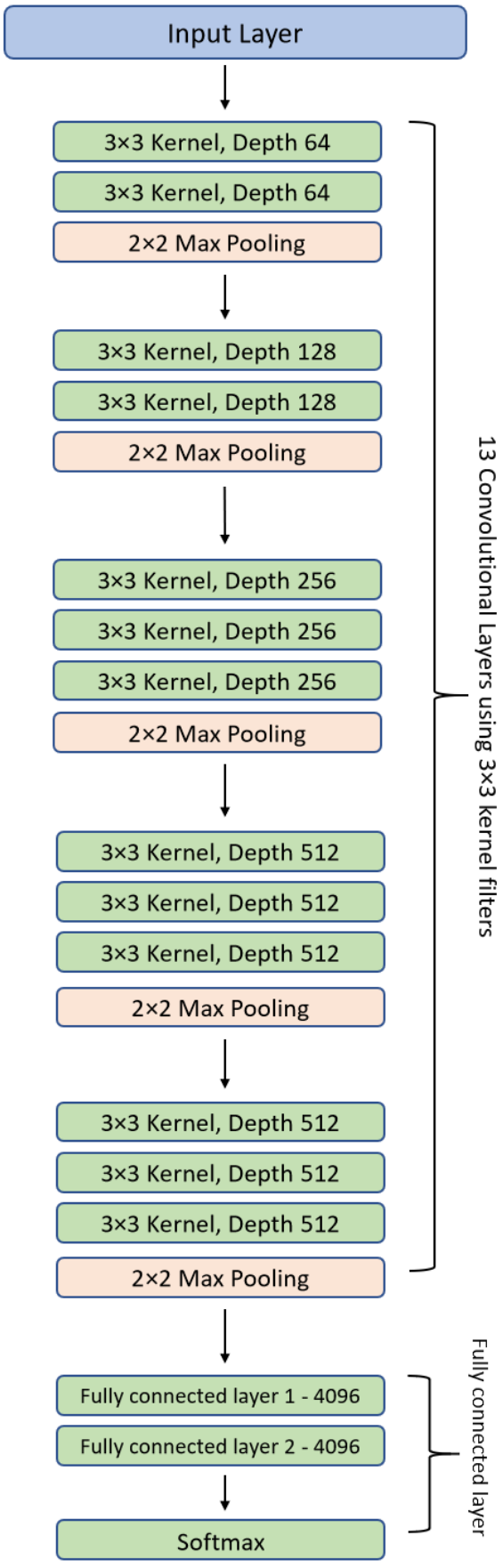


Figure 2.32 : l'architecture du modèle VGG -16 [54].

Comme le montre la figure ci-dessus, l'architecture du modèle VGG-16 se compose de 13 couches de convolution, 2 couches entièrement connectées et 1 classificateur SoftMax.

Karen Simonyan et Andrew Zisserman ont créé un réseau à 16 couches composé de couches convolutives et entièrement connectées. Dans un souci de simplicité, ils n'utilisent que des couches convolutives 3 × 3 empilées les unes sur les autres. En 2014, ils ont présenté l'architecture VGG-16- dans leur article Réseau convolutionnel très profond pour la reconnaissance d'images à grande échelle « Very Deep Convolutional Network for Large Scale Image Recognition ».

Nous expliquerons plus en détail la structure du réseau VGG-16 ci-dessous :

- La première et la deuxième couche de convolution sont constituées de 64 filtres de caractéristiques de base « feature kernel filters » de taille 3 x 3. Lorsque l'image d'entrée (image RVB avec une profondeur de 3) est transmise aux première et seconde couches convolutives, les dimensions passent à 224x224x64. Ensuite, la sortie générée est transmise à la couche de pooling maximale « max pooling layer » par un pas (stride) de 2.

- Le troisième et la quatrième couche de convolution sont constitués de 124 filtres de caractéristiques de base « feature kernel filters » de taille 3 x 3. Ces deux couches sont suivies par la couche de pooling maximale « max pooling layer » avec un pas (stride) de 2 et le résultat est réduit à 56x56x128.

- Les cinquième, sixième et septième couches sont des couches convolutives avec une taille de filtre de 3x3. Tous les trois utilisent 256 cartes de caractéristiques « feature maps ». Ces couches sont suivies par la couche de pooling maximale « max pooling layer » avec un pas (stride) de 2.

- Les couches huit à treize sont des ensembles de couches convolutives constituées de 512 kernel filtres de taille 3x3. Ceux-ci sont suivis par la couche de pool maximale « max pooling layer » en 1 pas (stride).

- Les couches quatorze et quinze sont des couches cachées entièrement connectées de 4096 unités, suivies d'une couche de sortie softmax de 1 000 unités (seizième couche). [54].

2.4.6 Les méthodes pour améliorer les performances des CNN :

- **La régularisaion :**

Cette technique clé d'apprentissage automatique (le machine learning) vise à limiter le «sur-apprentissage» et à contrôler les erreurs de variance pour améliorer les performances.

Lors de l'apprentissage de modèles, la régularisation peut imposer des contraintes pour favoriser des modèles simples plutôt que des modèles complexes. En d'autres termes, il réduit les erreurs de type de variance et améliore la généralité de la solution. Il existe de nombreuses formes de régularisation, en fonction de l'objectif souhaité et des hypothèses sur le problème.

Par conséquent, la régularisation euclidienne de la régression aux petits carrés favorisera les coefficients faibles, tandis que la régularisation au lasso (utilisée lorsque le nombre de variable d'entrée est grand) favorisera la « sparsité » de la représentation, en forçant l'algorithme à ne considérer qu'une petite partie des données et en ignorant le reste.

Dans le cas des réseaux de neurones, les méthodes de régularisation les plus courantes sont :

- **Le dropout** : les poids (paramètres du réseau neuronal) sont qualité de l'algorithme d'apprentissage (l'autre étant le biais). Remplacés par zéro au hasard pendant l'entraînement.

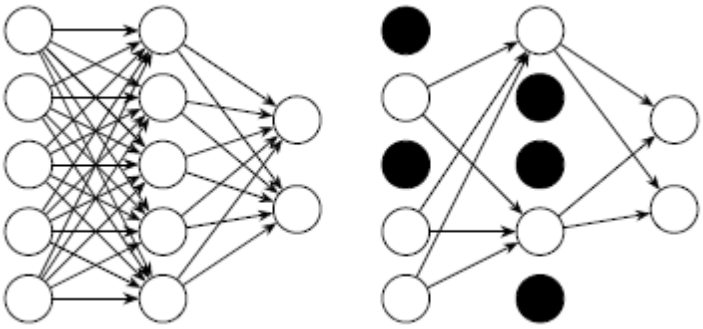


Figure 2.33 : un exemple de dropout standard [55].

Le réseau de gauche est entièrement connecté mais celui de droite avait des neurones abandonnés avec une probabilité de 0,5. La suppression n'est pas appliquée au calque de sortie.

- **l'Early Stopping** : l'apprentissage s'arrête plus tôt pour supporter des modèles simples ou la régularisation décrite ci-dessus.

Figure 2.34 : Early Stopping utilisé pour loss et l'accuracy [56] [57].

En outre, la régularisation garantit généralement certaines propriétés théoriques des algorithmes, assurant également leurs bonnes performances, comme la stabilité ou une meilleure borne de généralisation dans le cas d'une méthode à noyau.

Nous allons maintenant expliquer quelques termes nécessaires à comprendre avant de terminer le reste des méthodes.

L'overfitting : Ce phénomène de « sur-apprentissage » réduit les performances et la capacité de généralisation des algorithmes d'apprentissage automatique. La technique la plus courante pour limiter cette situation est la régularisation, qui peut réduire les erreurs de type variance.

Un algorithme d'apprentissage, tel que supervisé, recherche le modèle qui exprime le mieux la relation entre les données. L'overfitting se produit lorsqu'un algorithme sur-apprend (overfit), c'est-à-dire lorsqu'il apprend à partir de données, mais aussi à partir de patterns (schémas, structures) non liés au problème, comme le bruit. Ainsi, le sur-apprentissage est caractérisé par des erreurs de type variance très élevées. Plus précisément, lorsqu'on utilise des modèles très complexes sur des problèmes très simples mais bruyants, on observe généralement ce phénomène.

La variance : C'est l'une des deux erreurs utilisées pour définir l'algorithme d'apprentissage tente d'approcher d'une manière ou d'une autre la relation exacte entre les variables d'entrées et de sorties du problème, c'est-à-dire le « modèle réel ». Il peut essayer d'utiliser le modèle le plus complet et le plus complexe pour s'assurer qu'il comprend toutes les subtilités du problème. Mais malheureusement, les données sont souvent bruyantes, autrement dit, elles sont altérées par la présence d'un signal aléatoire de faible intensité (appelé bruit).

Il est à noter que la variance représente la "distance" entre le modèle réel de la famille de modèles considérée et la solution atteinte par l'algorithme.

Le biais : comme mentionné ci-dessus, le biais est l'une des deux erreurs utilisées pour déterminer la qualité de l'algorithme d'apprentissage.

L'algorithme utilise un modèle plus simple que le problème que nous essayons d'apprendre, et ne peut donc pas résoudre toute la complexité. Cette erreur dans les hypothèses du modèle est appelée « biais ».

À mesure que le modèle s'approche de la complexité du problème, le biais deviendra plus petit. A l'inverse, si le modèle est trop simple, l'écart sera très élevé. La nature de l'erreur dépend du type de problème considéré. Dans tous les cas, l'erreur totale ne sera jamais nulle en raison du bruit. Cependant, il peut être très faible.

Le biais est parfois défini comme la « distance » entre le meilleur modèle qu'un algorithme peut apprendre et le modèle réel. En apprentissage automatique, nous recherchons généralement un équilibre entre le biais et la variance afin que ces deux erreurs soient approximativement égales [58].

2.4.7 Techniques d'augmentation des données d'image :

La démonstration initiale a montré que les performances des données étaient dues à de simples transformations (telles que le retournement horizontal, l'augmentation de l'espace colorimétrique et le recadrage aléatoire). Ces transformations codent pour bon nombre des invariants énumérés ci-dessus, ce qui pose un défi aux tâches de reconnaissance d'image.

Ci-dessous, nous expliquons l'augmentation des données basées sur des manipulations d'images par des transformations géométriques et de nombreuses autres fonctions de traitement d'image. La classe des augmentations qui sera discutée pourrait être caractérisée par la facilité de leur mise en œuvre. Nous décrirons également diverses méthodes d'augmentations géométriques dans le cadre de leur « sécurité » d'utilisations. Cela fait référence à la possibilité de conserver l'étiquette après le traitement. En revanche, une transformation de sauvegarde sans étiquette peut améliorer la capacité du modèle à générer une réponse en indiquant qu'il n'est pas sûr de ses prédictions. Cependant, la création d'étiquettes précises pour chaque augmentation de données non

sécurisée est un processus qui demande beaucoup de calculs. Sans la fonction de traitement d'image, l'étiquette ne peut pas être modifiée sous une certaine amplitude de distorsion.

Sans la fonction de traitement d'image, l'étiquette ne peut pas être modifiée avec une certaine distorsion. Ceci est illustré par la conception de la croissance prenant en compte les données et le défi de l'élaboration de politiques générales d'élargissement. C'est un point important pour les augmentations géométriques énumérées ci-dessous.

- **Retournement :**

L'inversion de l'axe horizontal est beaucoup plus courante que l'inversion de l'axe vertical. Cette augmentation est l'une des plus faciles à mettre en œuvre. Pour les ensembles de données qui incluent la reconnaissance de texte, il ne s'agit pas d'une transformation préservant les étiquettes.

- **Espace colorimétrique :**

Les données d'image numérique sont généralement codées sous la forme d'un tenseur de dimension (hauteur × largeur × canaux de couleur). L'augmentation de l'espace des canaux de couleur est une autre stratégie très simple et pratique à mettre en œuvre. Il suffit simplement d'isoler un seul canal de couleur tel que R, V ou B.

En isolant la matrice et en ajoutant 2 matrices nulles d'autres canaux de couleur pour terminer cette opération, l'image peut être rapidement convertie en une représentation dans le canal de couleur. De plus, la valeur RVB peut être facilement contrôlée par de simples opérations matricielles pour augmenter ou diminuer la luminosité de l'image.

Des augmentations de couleur plus avancées résultent de l'obtention d'un histogramme de couleur qui décrit l'image. La modification de la valeur d'intensité dans ces histogrammes entraîne des changements d'éclairage similaires à ceux utilisés dans les applications de retouche photo.

- **Recadrage :**

Cette technique peut être utilisée comme étape de traitement pratique pour les données d'image avec des dimensions de hauteur et de largeur mixtes, en recadrant le bloc de couleur central de chaque image. En outre, le recadrage aléatoire peut également être utilisé pour fournir un effet très proche de la traduction.

La différence entre le recadrage aléatoire et la traduction est que le premier réduira la taille de l'entrée, par exemple (256,256) → (224,224), tandis que le second conservera les dimensions spatiales de l'image. En fonction du seuil de réduction sélectionné pour le recadrage, il se peut qu'il ne s'agisse pas d'une transformation pour protéger l'étiquette.

- **Rotation :**

L'augmentation de la rotation se fait en faisant tourner l'image dans le sens horaire ou antihoraire sur l'axe de 1° à 359°. La sécurité des augmentations de rotation est fortement déterminée par le paramètre de degré de rotation. Lorsque le degré de rotation augmente, l'étiquette de données n'est plus stockée après la transformation.

Traduction :

Le déplacement d'image vers la gauche, la droite, le haut ou le bas peut être une transformation très utile pour éviter un écart de position dans les données. Lorsque l'image originale est traduite dans une direction, des valeurs vides (telles que 0s ou 255s) peuvent être utilisées pour remplir l'espace restant, ou un bruit aléatoire ou gaussien peut être utilisé pour remplir l'espace restant. Ce remplissage préserve la taille spatiale de l'image améliorée.

Injection de bruit :

L'injection de bruit consiste à injecter une matrice de valeurs aléatoires généralement dérivée d'une distribution gaussienne. L'injection de bruit a été testée par Moreno-Barea et Al. L'ajout de bruit à l'image peut aider les CNN à obtenir des fonctions plus puissantes.

Transformations de l'espace colorimétrique :

Les données d'image sont codées dans 3 matrices empilées, dont chacune a une taille de (hauteur × largeur). Ces matrices représentent les valeurs de pixel d'une seule valeur de couleur RVB. Le biais d'éclairage est l'un des problèmes de reconnaissance d'image les plus courants. Par conséquent, l'efficacité de la conversion d'espace colorimétrique (également appelée transformation photométrique) est conceptuellement très intuitive. Une solution rapide pour les images trop lumineuses ou trop sombres consiste à parcourir l'image en boucle et à réduire ou augmenter la valeur des pixels d'une quantité fixe. Une autre opération rapide de l'espace colorimétrique consiste à combiner des matrices de couleurs RVB individuelles.

Il existe également une autre transformation qui vise à limiter les valeurs de pixel à une certaine valeur minimale ou maximale. La représentation inhérente des couleurs dans les images numériques les rend adaptées à de nombreuses stratégies d'amélioration.

La transformation d'espace colorimétrique peut également être obtenue à partir d'applications d'édition d'images. Les valeurs de pixel de l'image dans chaque canal de couleur RVB sont assemblées pour former un histogramme de couleur. Celui-là peut être manipulé pour appliquer des filtres qui modifient les caractéristiques de l'espace colorimétrique de l'image.

Au fur et à mesure que l'espace colorimétrique augmente, la liberté dans la créativité s’accroît. La modification de la distribution des couleurs de l'image peut résoudre les problèmes d'éclairage rencontrés lors de test des données, Comme indiqué dans la figure ci-dessous.

Figure 2.35 : exemple de couleur augmentation [59].

L'inconvénient de la transformation de l'espace colorimétrique est l'augmentation de la mémoire, des coûts de transformation et du temps de formation. De plus, les transformations de couleur peuvent ignorer des informations de couleur importantes et ne sont donc pas toujours une transformation préservant les étiquettes [59].

2.4.8 Le Transfer Learning :

Le "**transfer learning**" consiste à utiliser un CNN qui a été formé pour une tâche spécifique afin de le rendre dédié à d'autres problèmes.

Prenons un exemple simple pour illustrer le transfert learning : une personne qui veut apprendre à jouer de la guitare, si elle sait déjà jouer du piano, elle pourra apprendre plus facilement. En d'autres termes, elle utilisera ses connaissances musicales pour apprendre à jouer de nouveaux instruments.

Dans le contexte de l'analyse de données, la plupart des algorithmes d'apprentissage automatique sont basés sur l'hypothèse que l'ensemble des données utilisées pour l'apprentissage et l'ensemble des données de test appartiennent au même espace de descripteurs et ont la même distribution de probabilité. Cependant, dans de nombreuses applications, ce n'est pas le cas, et le recyclage des modèles à forte intensité de données et de calcul, en particulier dans l'apprentissage en profondeur, peut être optimisé avec l'apprentissage par transfert : Les connaissances obtenues à partir du jeu de données de formation, ci-après dénommé « source », sont « transférées » afin de pouvoir traiter correctement le nouvel ensemble de données, appelé « cible ».

En pratique, il existe un grand nombre de méthodes d'apprentissage par transfert, telles que :

- **L'instance based transfer** (basée sur la transmission) : elle effectue la transmission en pondérant et / ou en sélectionnant certaines données de l'ensemble « source » pour les introduire dans l'ensemble du jeu « cible » ;
- **Feature based transfer** (transfert basé sur les fonctionnalités) : il modifie les espaces descripteurs « source » et / ou « cible » pour les rendre similaires ;

- **Le model based transfer** (transfert basé sur un modèle) ou **parameter based transfer** (transfert basé sur des paramètres) : il modifie le modèle lui-même (qu'il s'agisse de réseau neuronal, SVM, arbre de décision...) pour transférer les connaissances. Il est à noter qu'en deep learning, les modèles pré-entraînés dont les concepts sont de plus en plus courants dans la littérature appartiennent à cette famille [58].

Conclusion :

Nous avons tout d'abord expliqué quelques connaissances fondamentales sur les drones, qui sont à la base de notre projet. Nous avons ensuite établi les notions de base des caractéristiques d'une image, ainsi qu'une présentation du concept de notre travail où nous avons établi un aperçu général des techniques et des méthodes utilisées dans le domaine de la détection d'images. Nous avons également donné une explication détaillée du concept des réseaux de neurones convolutifs, ainsi que leur fonctionnement et leurs domaines d'application. Enfin, nous avons expliqué les VGG16 et les méthodes pour améliorer les performances des CNN ainsi que le transfer learning.

Il faut savoir que les types de drones que nous voulons détecter sont les micros et mini drones uniquement quelle que soit leur forme. Nous éliminerons donc les drones volumineux de notre travail.

Chapitre 3

Méthodes De Détection Et Résultats

Introduction :

Après avoir décrits de façon théorique les réseaux de neurones et le problème de la reconnaissance des drones dans les chapitres précédents, ce chapitre sera consacré aux résultats expérimentaux que donne l'implémentation du programme. Premièrement, une description des bibliothèques utilisés dans notre travail sera exposée. Deuxièmement, nous prendrons d'abord un petit ensemble de données afin d'analyser les performances des architectures avant de prendre l'ensemble complet (plusieurs expériences et tests seront menés pour entraîner et tester le réseau afin d'atteindre les performances attendues). Nous enregistrerons troisièmement notre modèle sur le Transfer Learning car celui-ci montre une performance très grande. Nous terminerons enfin ce chapitre par une conclusion résumant et interprétant les résultats obtenus.

3.1 La plateforme TensorFlow :

En novembre 2015, TensorFlow est passé Open Source par Google, c'est une plateforme de développement informatique numérique. Depuis son lancement, TensorFlow n'a cessé de croître et est rapidement devenu l'un des Frameworks les plus largement utilisés pour l'apprentissage profond (Deep Learning) et donc les réseaux de neurones.

Son nom est principalement inspiré du fait que les opérations actuelles sur les réseaux de neurones sont principalement réalisées à l'aide de tables de données multidimensionnelles nommées Tenseurs (Tensor). Un tenseur 2D équivaut à une matrice. Actuellement, Google propose des produits majeurs basés sur TensorFlow, tels que Gmail, Google Photos et la reconnaissance vocale.

Le principal avantage de TensorFlow est qu'il fournit un ensemble de flux de travail pour créer et former des modèles à l'aide de Python, JavaScript ou Swift, et les déployer facilement dans le cloud, sur site, dans un navigateur ou sur n'importe quel appareil, quelle que soit la langue utilisée [7].

3.2 Le logiciel et les librairies utilisées dans l'implémentation :

3.2.1 La plateforme Keras :

Keras est un API de réseau neuronal de haut niveau (Deep Learning) écrit en Python et peut être exécuté sur TensorFlow ou Theano. Son développement se concentre sur l'expérimentation rapide, et il peut passer d'une idée à un résultat dans les plus brefs délais. C'est la clé d'une bonne recherche.

Cette interface fait partie des travaux de recherche du projet ONEIROS, et son principal auteur et mainteneur est l'ingénieur François Chollet. L'équipe TensorFlow de Google a décidé en 2017 de prendre en charge Keras dans la bibliothèque principale de TensorFlow. Le fondateur a expliqué que Keras a été conçu comme une interface et non comme un environnement d'apprentissage de bout en bout. Il fournit un ensemble d'abstractions de niveau supérieur et plus intuitif. Cela facilite la configuration du réseau neuronal indépendant d'une bibliothèque informatique back-end appelé aussi arrière-plan qui spécifie les étapes de sortie du logiciel qui produit les résultats [8].

3.2.2 Configuration utilisé dans l'implémentation :

La configuration matérielle utilisée dans notre implémentation est :

- Un MacOs Catalina i7 CPU 2.8 GHZ ;
- Carte graphique : Intel HD Graphics 6301536 MB ;

- AM de taille 16 GO ;
- Version : 10.15.5 ;
- Disque dur SSD.

Et les logiciels et les bibliothèques installés sont :

- Python 3.7 ;
- Tenserflow 2.3 ;
- Keras 2.3.1.

Avant de commencer à exécuter et à expliquer la procédure, il est nécessaire de comprendre la terminologie des résultats, telle que la précision et la perte :

- **Précision (Accuracy) :**

Proportion des prédictions correctes d'un modèle de classification. Dans la classification à classes multiples, la précision est définie comme suit [60] :

$$\text{Précision} = \frac{\text{Prédictions correctes}}{\text{Nombre total d'exemples}}$$

Dans la classification binaire, la précision est définie ainsi :

$$\text{Précision} = \frac{\text{Vrais positifs + Vrais négatifs}}{\text{Nombre total d'exemples}}$$

Notre projet utilise cette classification étant donné que nous n'avons que deux classes uniquement (0 ou 1).

Vrai positif (VP) : un exemple du modèle prédisant correctement la classe positive. Quand notre modèle déduit qu'une image particulière est un drone quand il s'agit bien d'un drone.

Vrai négatif (VN) : un exemple du modèle prédisant correctement la classe négative. Quand notre modèle déduit qu'une image particulière est un non drone quand il s'agit bien d'un non drone.

- **Cross entropie binaire :**

Il s'agit d'une fonction de perte pour les tâches de classification binaire, et seules deux options (oui ou non, A ou B, 0 ou 1, gauche ou droite) sont utilisées pour répondre aux questions. Il est également possible de répondre à plusieurs questions indépendantes en même temps, par exemple lors d'une classification multi-étiquettes ou d'une segmentation d'images binaires.
Formellement parlant, cette perte est égale à la moyenne de la perte d'entropie croisée catégorique pour de nombreuses tâches à deux catégories.

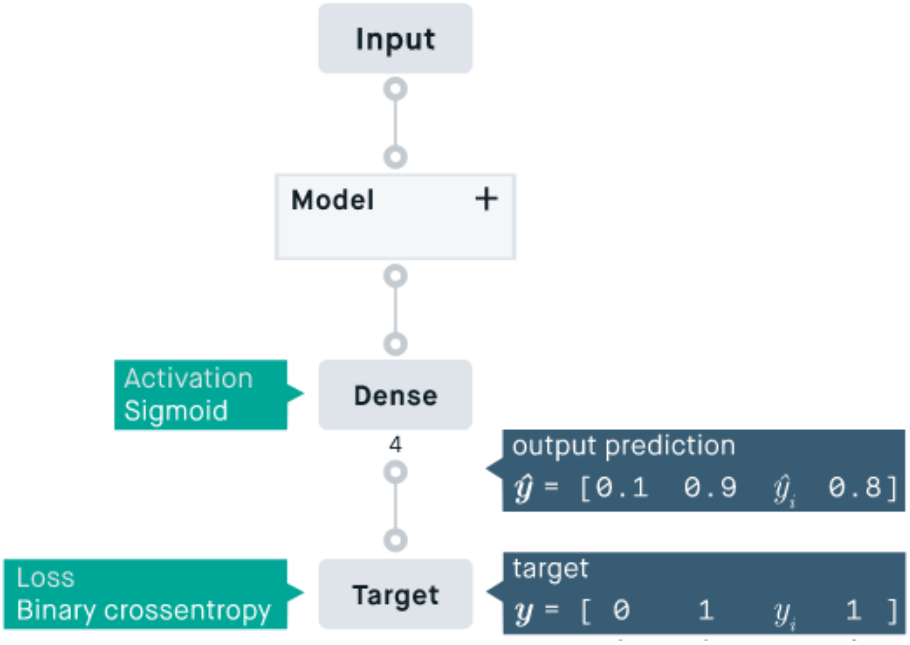


Figure 3.1 : illustration des étapes de classification de l'entropie croisée binaire de perte [61].

Mathématiques croisées binaires :

La fonction de la perte d'entropie croisée calcule la perte d'un exemple en calculant la moyenne suivante :

$$\text{Loss} = -\frac{1}{N}\sum_{i=1}^{N}[y_i \log(\hat{y}_i) + (1-y_i)\log(1-\hat{y}_i)]$$

Où $\hat{y}_i$ est l'i-ème valeur scalaire dans la sortie du modèle, y_i est la valeur cible correspondante et N est le nombre de valeurs scalaires dans la sortie du modèle.

Cela équivaut au résultat moyen de la fonction de la perte d'entropie croisée catégorique appliquée à de nombreux problèmes de classification indépendants, chaque problème n'ayant que deux classes possibles avec des probabilités cibles y_i et $(1-\hat{y}_i)$ [61].

3.4 Préparation de l'ensemble des données drones et non-drones :

L'étape de collecte d'images est effectuée par nous-mêmes via Google Images ou d'autres sites Web (tels que Pinterest). Ensuite, nous divisons la base de données en trois catégories nommées : tr, ts (pour la phase d'apprentissage) et ts1 (pour la phase de test). Puis, nous avons créé deux fichiers sur chaque catégorie, un pour les drones et un pour les non-drones.

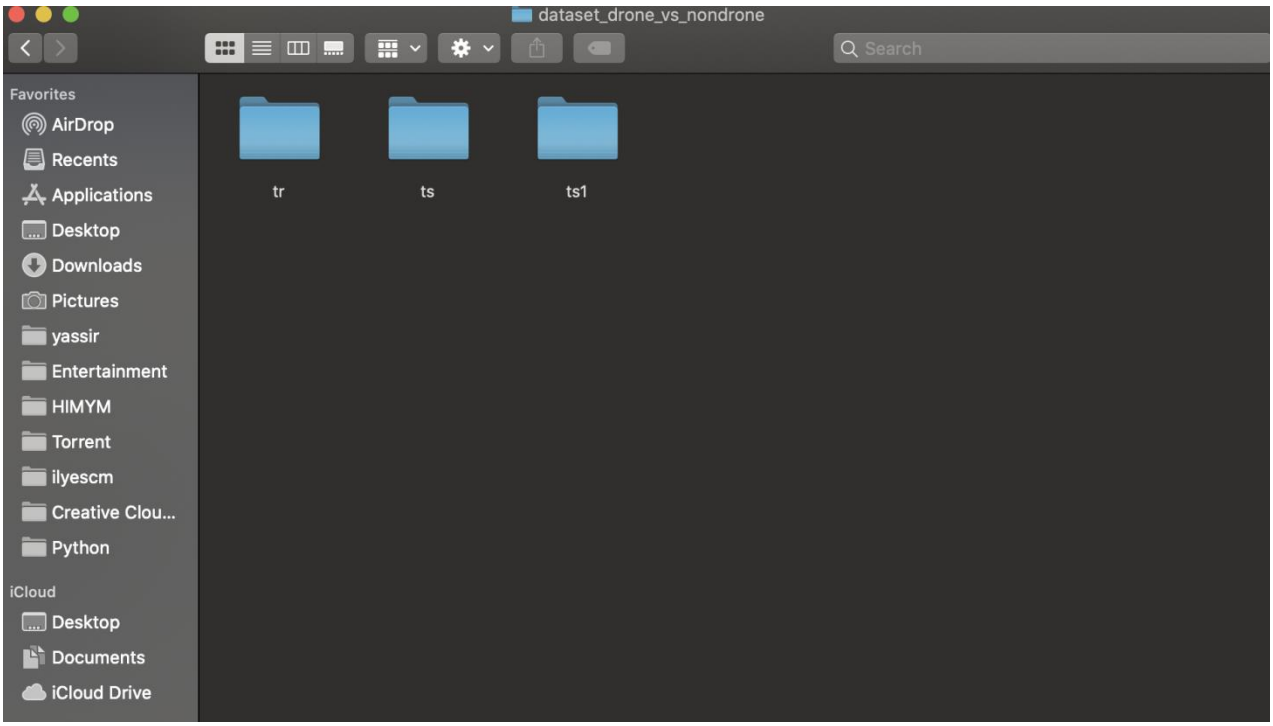


Figure 3.2 : les trois catégories de notre base de données.

Figure 3.3 : les deux classes trouvées sur chaque catégorie.

Nous avons rassemblé toutes les formes de drones (mini et micro) que nous avons pu trouver sur des sites Web ou sur Google Images et même dans des articles en prenant une capture d'écran. Nous avons également modifié les images en supprimant toutes les écritures et les visages, et même les objets ressemblant à des drones, afin que notre programme puisse facilement distinguer les caractéristiques des drones.

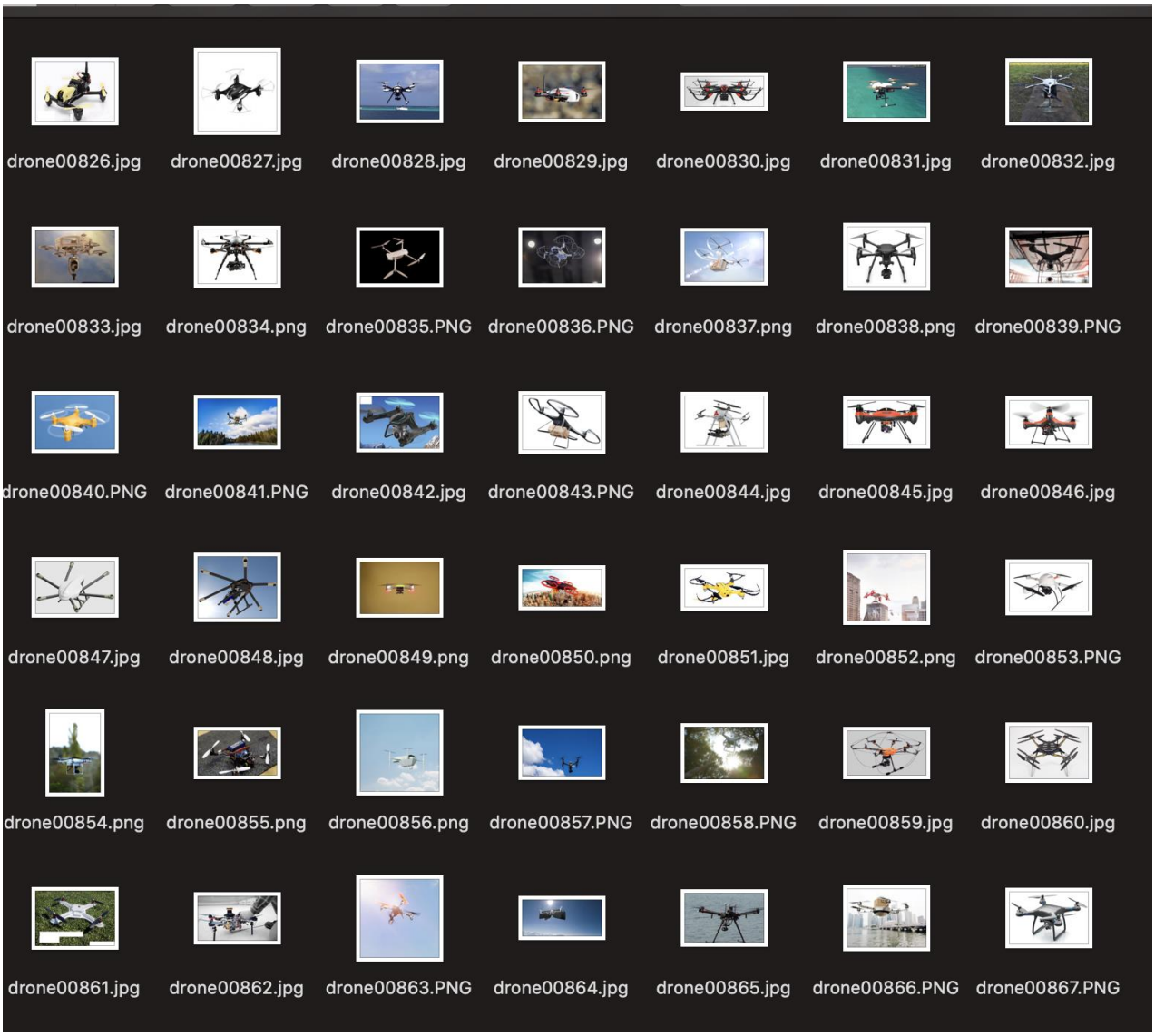


Figure 3.4 : les images trouvées sur la classe « drone ».

La sélection d'images non-drones consiste à choisir des objets volants qui ressemblent à des drones et qui sont volants. Nous avons collecté des images, notamment des oiseaux rapaces, des OVNIs, des sacs en plastique volants, ainsi que des ballons, des cerfs-volants et certaines images incluent des parachutes éloignés.

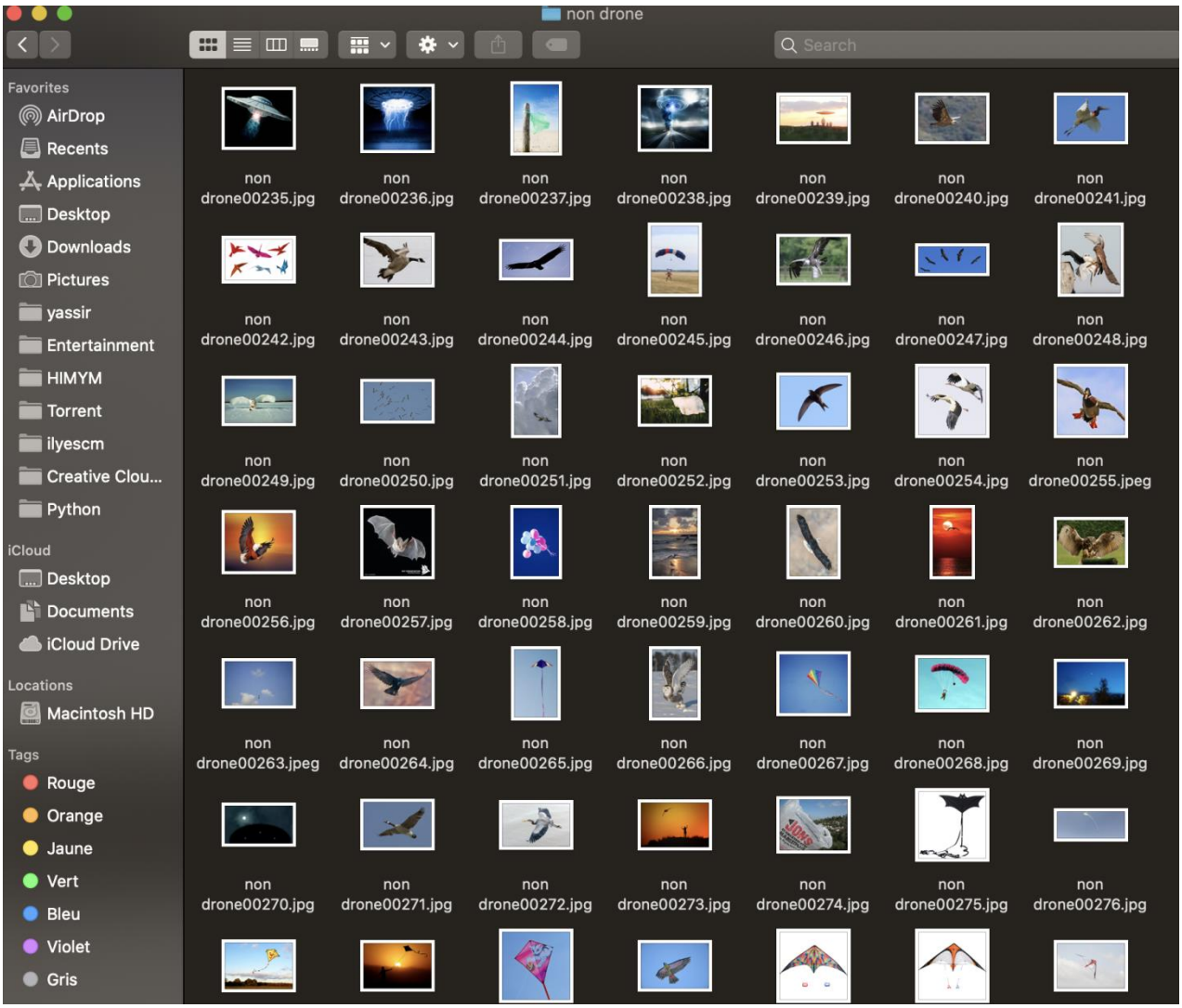


Figure 3.5 : les images trouvées sur la classe « non drone ».

Afin d'éviter les images en double, nous avons utilisé le logiciel Duplicate Media Finder qui détecte les images en double. Après avoir supprimé les images en double, nous devions donc remplacer ces images par d'autres images afin d'atteindre le nombre d'images souhaités.

3.5 Développer un modèle CNN de base :

Au cours de nos expérimentations, nous avons créé plusieurs modèles avec différentes architectures, les figures montrées dans les sections suivantes seront basées sur une petite base de données. Par conséquent, il est tout à fait normal qu'ils ne soient pas très performants. Après avoir essayé toutes les architectures, nous enregistrerons notre modèle sur l'architecture qui montre les meilleures performances et le moins de pertes, où nous allons essayer plus de bases de données.

Dans cette section, nous allons développer un modèle de réseau neuronal convolutif de base pour l'ensemble des données drones et non drones. Ci-dessous un tableau qui montre combien d'images nous avons utilisé dans l'apprentissage ainsi que le teste.

Tableau 0.1 : les données utilisées lors de l'apprentissage (entraînement et validation) et le test.

Les classes	Train-it (entraînement)	Test-it (la validation)	Test (le test)
Drone	1002 images	286 images	143 images
Non drone	235 images	67 images	34 images
La somme	1237 images	353 images	177 images

Un modèle de base établira la performance du modèle minimale à laquelle tous nos autres modèles pourront être comparés, ainsi qu'un modèle d'architecture que nous pourrons utiliser comme base d'étude et d'amélioration.

Les principes architecturaux généraux des modèles VGG constituent un bon point de départ car la structure modulaire de cette architecture est facile à comprendre et à mettre en œuvre.

L'architecture consiste à empiler des couches convolutives avec de petits filtres 3 × 3 suivis d'une couche de Max Pooling. Ensemble, ces couches forment un bloc, et ces blocs peuvent être répétés dans le nombre de filtres où chaque bloc est augmenté avec la profondeur du réseau tel que 32, 64, 128, 256 pour les quatre premiers blocs du modèle. Le remplissage est utilisé sur les couches convolutives pour garantir que les formes de hauteur et de largeur des cartes d'entités en sortie correspondent aux entrées.

Nous pouvons explorer cette architecture sur le problème drones et non drones ainsi que pour comparer les modèles avec cette architecture à un, deux et trois blocs.

Chaque couche utilisera la fonction d'activation ReLU et l'initialisation du poids He, qui sont généralement les meilleures pratiques. Par exemple, une architecture de style VGG à trois blocs où chaque bloc a une seule couche de convolution.

Nous pouvons créer une fonction nommée *define_model()* qui définira un modèle et le retournera prêt à être ajusté sur l'ensemble des données. Cette fonction peut ensuite être personnalisée pour définir différents modèles de base, par exemple des versions du modèle avec un, deux ou trois blocs de style VGG.

Le modèle sera ajusté avec une descente de gradient stochastique et nous commencerons avec un taux d'apprentissage conservateur de 0,001 et un élan de 0,9. Le problème est une tâche de classification binaire, nécessitant la prédiction d'une valeur de 0 ou de 1.

Une couche de sortie avec un nœud et une activation sigmoïde sera utilisée et le modèle sera optimisé à l'aide de la fonction de perte d'entropie croisée binaire qui calculera la perte entre les étiquettes vraies et les étiquettes prévues.

Nous devons ensuite préparer les données. Cela implique d'abord de définir une instance de *ImageDataGenerator* qui mettra à l'échelle les valeurs en pixels dans la plage de 0 à 1.

Puis, les littérateurs doivent être préparés pour les ensembles de données de train (entraînement), et de test.

Nous pouvons utiliser la fonction *flow_from_directory()* sur le générateur de données et créer un littérateur pour chacun des répertoires *train/* et *test/*. Il faut préciser que le problème est un problème

de classification binaire via l'argument « *class_mode* », et charger les images de taille 200 × 200 pixels via l'argument « *target_size* ». Nous fixerons la taille du lot à 64.

Nous pouvons ensuite ajuster le modèle à l'aide de l'itérateur de train (*train-it*) et utiliser l'itérateur de test (*test_it*) comme ensemble de données de validation pendant l'entraînement.

Le nombre d'étapes pour l'entraînement (train) et les l'itérateurs de test doit être spécifié. C'est le nombre de lots qui comprendront une époque. Cela peut être spécifié via la longueur de chaque littérateur, et sera le nombre total d'images dans les répertoires d'entraînement et de test divisé par la taille du lot (64).

Le modèle sera adapté pour 20 époques, un petit nombre pour vérifier si le modèle peut apprendre le problème.

Une fois ajusté, le modèle final peut être évalué directement sur l'ensemble des données de test et sur la précision de la classification rapportée.

Enfin, nous pouvons créer un tracé de l'historique collecté lors de l'entraînement stocké dans le répertoire « *history* » renvoyé par l'appel à *fit_generator()*.

L'historique contient la précision (accuracy) et la perte (loss) du modèle sur l'ensemble des données de test et de l'entraînement à la fin de chaque époque. Les tracés linéaires de ces mesures sur les époques d'entraînement fournissent des courbes d'apprentissage que nous pouvons utiliser pour avoir une idée de si le modèle est sur-ajusté, sous-ajusté ou s'il est bien ajusté.

La fonction *summary_diagnostics()* prend le répertoire historique et crée un seul chiffre avec un graphique linéaire de la perte et un autre pour la précision. La figure est ensuite enregistrée dans un fichier avec un nom de fichier basé sur le nom du script.

Maintenant que nous avons un *test_harness*, examinons l'évaluation de trois modèles de base simples.

3.5.1 Modèle VGG à un bloc :

Le modèle VGG à un bloc a une seule couche convolutive avec 32 filtres suivis d'une couche de Pooling Max.

L'exécution d'un exemple imprime d'abord la taille de l'entraînement et des jeux des données de test, confirmant que le jeu de données a été chargé correctement.

Le modèle est ensuite ajusté et évalué.

Résultats :

Sur l'affichage, nous pouvons lire que l'apprentissage comporte 1237 images classé en deux classes (drone / non drone). 353 images lors de la validation et le test est réalisé sur 177 images toujours classé en deux classes.

```
Found 1237 images belonging to 2 classes.
Found 353 images belonging to 2 classes.
Found 177 images belonging to 2 classes.

> 81.714
```

Dans ce cas, nous pouvons voir que le modèle a atteint une précision d'environ 81.7% sur l'ensemble des données de test.

Une courbe est également créée montrant un graphique linéaire pour la perte et un autre pour la précision du modèle sur les ensembles de données d'entrainement (bleu) et de test (orange).

En examinant ce graphique, nous pouvons voir que le modèle a sur-ajusté l'ensemble des données d'entraînement.

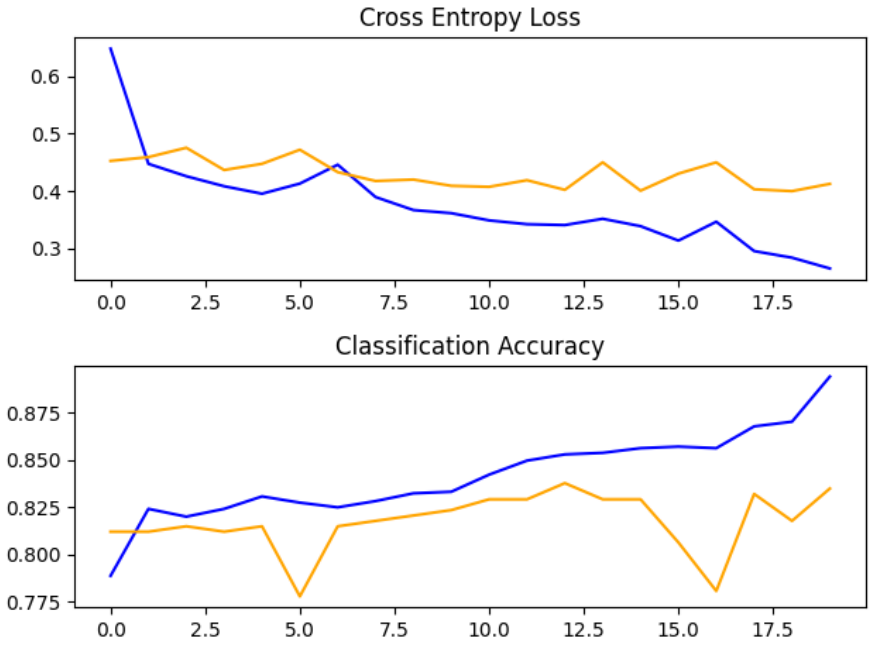


Figure 3.6 : tracés linéaires des courbes d'apprentissage de la perte et de la précision pour le modèle de base avec un bloc VGG sur l'ensemble des données drone et non drone.

3.5.2 Modèle à deux blocs VGG :

Le modèle VGG à deux blocs étend le modèle à un bloc et ajoute un deuxième bloc avec 64 filtres. Le modèle est ajusté et évalué et les performances sur l'ensemble des données de test sont signalées.

Résultats :

```
Found 1237 images belonging to 2 classes.
Found 353 images belonging to 2 classes.
Found 177 images belonging to 2 classes.

> 80.571
```

Dans ce cas, nous pouvons voir que le modèle a obtenu une légère diminution des performances d'environ 81.7% avec un bloc à environ 80.5% de précision avec deux blocs

En examinant le tracé des courbes d'apprentissage, nous pouvons voir qu'une fois de plus, le modèle semble avoir sur-ajusté l'ensemble des données d'entraînement, peut-être plus tôt, dans ce cas.

Ceci est probablement le résultat de la capacité accrue du modèle, et nous pouvons nous attendre à ce que cette tendance à un sur-ajustement plus précoce se poursuive avec le modèle suivant.

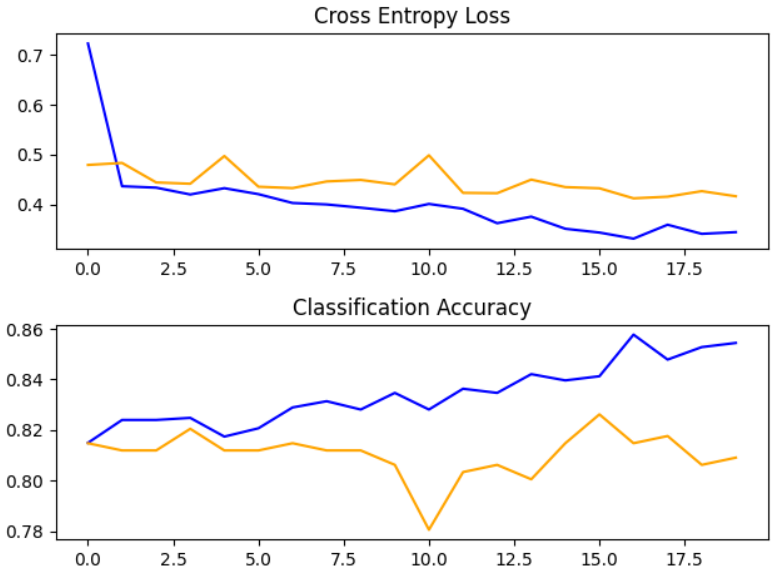


Figure 3.7 : tracés linéaires des courbes d'apprentissage de la perte et de la précision pour le modèle de base avec deux blocs VGG sur l'ensemble de données drone et non drone.

3.5.3 Modèle VGG à trois blocs :

Le modèle VGG à trois blocs étend le modèle à deux blocs et nous ajoutons seulement un troisième bloc avec 128 filtres.

Résultats :

```
                    Found 1237 images belonging to 2 classes.
                    Found 353 images belonging to 2 classes.
                    Found 177 images belonging to 2 classes.

> 80.000
```

Dans ce cas, nous pouvons voir que nous avons obtenu une diminution supplémentaire des performances d'environ 80.5% avec deux blocs à environ 80% de précision avec trois blocs.

En examinant le tracé des courbes d'apprentissage, nous pouvons voir une tendance similaire au sur-ajustement mais qui commence vers l'époque 6.

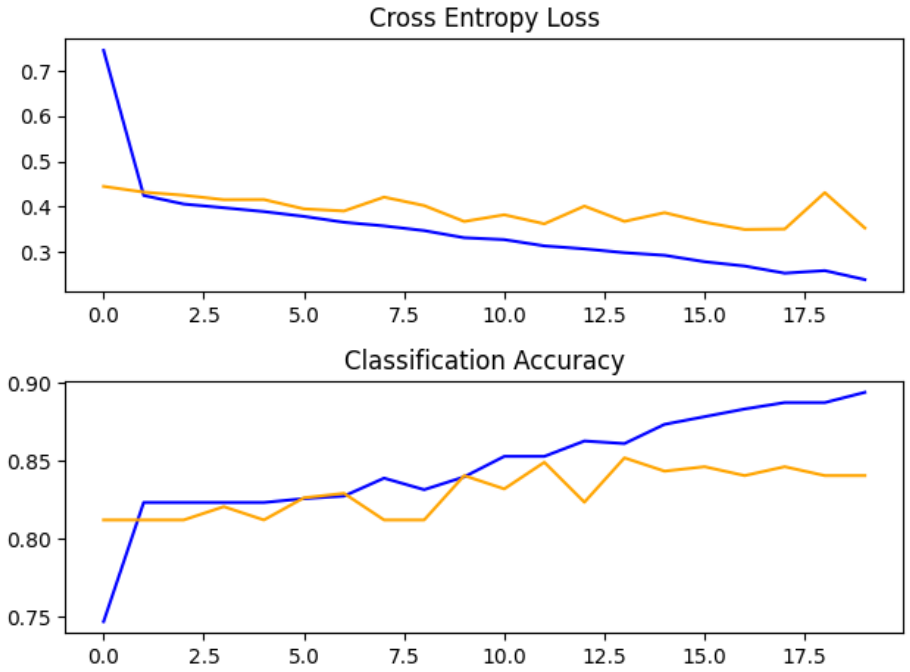


Figure 3.8 : tracés linéaires des courbes d'apprentissage de la perte et de la précision pour le modèle de base avec trois blocs VGG sur l'ensemble des données drone et non drone.

Discussion :

Nous avons exploré trois modèles différents avec une architecture basée sur VGG.

Les résultats peuvent être résumés ci-dessous en supposant une certaine variance dans ces résultats compte tenu de la nature stochastique de l'algorithme.

Tableau 0.2 : les résultats obtenus dans les trois architectures de VGG.

Le modèle	Précision
VGG 1	81.714%
VGG 2	80.571%
VGG 3	80.000%

Nous constatons une diminution des performances avec l'augmentation de la capacité, mais aussi un cas similaire de sur-ajustement survenant de plus en plus tôt dans la course.

Les résultats suggèrent que le modèle bénéficiera probablement des techniques de régularisation. Cela peut inclure des techniques telles que l'abandon, la décroissance du poids et l'augmentation des données. Ceci peut également améliorer les performances en encourageant le

modèle à apprendre des fonctionnalités qui sont davantage invariantes à la position en développant l'ensemble des données d'entraînement.

3.6 Développer des améliorations du modèle :

Dans cette section, nous commencerons par le modèle de base avec trois blocs VGG et explorerons quelques améliorations simples du modèle.

Après avoir examiné les courbes d'apprentissage du modèle pendant l'entraînement, le modèle a montré de forts signes de sur-ajustement. Nous pouvons explorer deux approches pour tenter de remédier à ce sur-ajustement : la régularisation du décrochage et l'augmentation des données.

On s'attend à ce que ces deux approches ralentissent le taux d'amélioration pendant l'entraînement et contrecarrent le sur-ajustement de l'ensemble des données d'entraînement. En tant que tel, nous augmenterons le nombre d'époques d'entraînement de 20 à 50 pour donner au modèle plus d'espaces pour le raffinement.

3.6.1 Régularisation des abandons :

En règle générale, une petite quantité de perte peut être appliquée après chaque bloc VGG, avec plus de perte appliquée aux couches entièrement connectées près de la couche de sortie du modèle.

Dans ce cas, un abandon de 20% est appliqué après chaque bloc VGG, avec un taux d'abandon plus élevé de 50% appliqué après la couche entièrement connectée dans la partie classificateur du modèle.

L'exécution de l'exemple ajuste d'abord le modèle, puis signale les performances du modèle sur l'ensemble de données de test d'attente.

Dans ce cas, nous pouvons constater une diminution des performances du modèle, passant d'environ 80% de précision pour le modèle de base à environ 77% avec l'ajout du décrochage.

Résultats :

```
                    Found 1237 images belonging to 2 classes.
                    Found 353 images belonging to 2 classes.
                    Found 177 images belonging to 2 classes.

> 77.143
```

En examinant les courbes d'apprentissage, nous pouvons voir que l'abandon a eu un effet sur le taux d'amélioration du modèle à la fois sur l'entraînement et sur les ensembles de test.

Le sur-ajustement a été réduit ou retardé, bien que les performances puissent commencer à ralentir vers la fin de la course.

Les résultats suggèrent que des périodes de formation plus poussées peuvent entraîner une amélioration supplémentaire du modèle. Il peut également être intéressant d'explorer un taux d'abandon légèrement plus élevé après les blocs VGG en plus de l'augmentation des périodes d'entraînement.

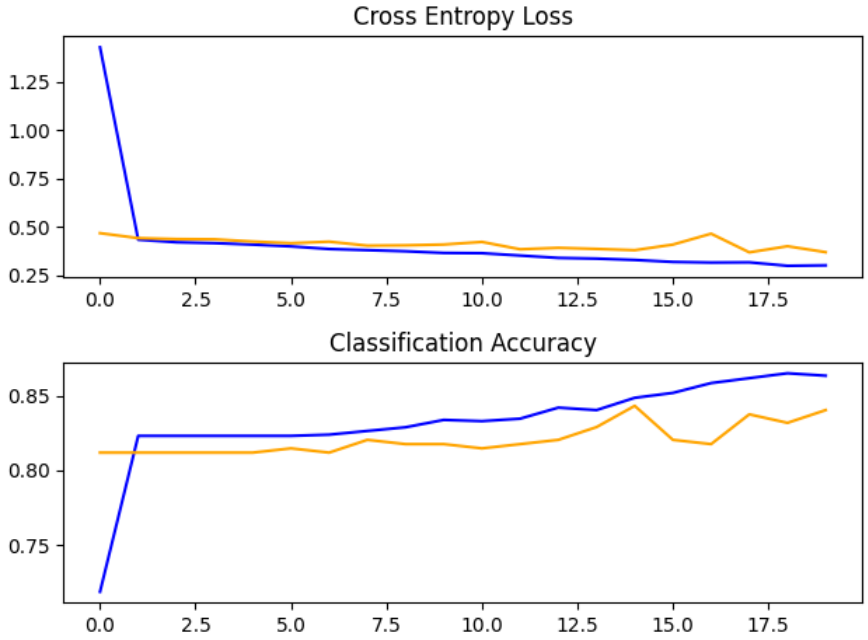


Figure 3.9 : graphiques linéaires des courbes d'apprentissage de la perte et de la précision pour le modèle de référence avec abandon sur l'ensemble des données drone et non drone.

3.6.2 Augmentation des données d'image :

Ces augmentations peuvent être spécifiées en tant qu'arguments de l'ImageDataGenerator utilisé pour l'ensemble des données d'entraînement. Les augmentations ne doivent pas être utilisées pour l'ensemble des données de test car nous souhaitons évaluer les performances du modèle sur les photographies non modifiées.

Cela nécessite que nous ayons une instance ImageDataGenerator distincte pour l'entraînement et l'ensemble des données de test, puis des itérateurs pour l'entraînement et les ensembles de test créés à partir des générateurs de données respectifs.

Dans ce cas, les photos de l'ensemble des données d'entraînement seront augmentées de petits décalages horizontaux et verticaux aléatoires (10%) et de retournements horizontaux aléatoires qui créent une image miroir d'une photo. Les photos des étapes d'entraînement et de test auront leurs valeurs de pixels mises à l'échelle de la même manière.

L'exécution de l'exemple ajuste d'abord le modèle, puis signale les performances du modèle sur l'ensemble des données de test d'attente.

Dans ce cas, nous pouvons constater une diminution des performances d'environ 2%, passant d'environ 80% pour le modèle de base à environ 78% pour le modèle de base avec une simple augmentation des données.

Résultats :

```
Found 1237 images belonging to 2 classes.
Found 353 images belonging to 2 classes.
Found 177 images belonging to 2 classes.

>78.286
```

En examinant les courbes d'apprentissage, nous pouvons voir le modèle semble être capable d'apprendre davantage, la perte sur l'entraînement et l'ensemble des données de test diminuant. La répétition de l'expérience avec 100 époques ou plus aboutira très probablement à un modèle plus performant.

Il peut être intéressant d'explorer d'autres augmentations susceptibles d'encourager davantage l'apprentissage de caractéristiques indifférentes à leur position dans l'entrée, comme les rotations mineures et les zooms.

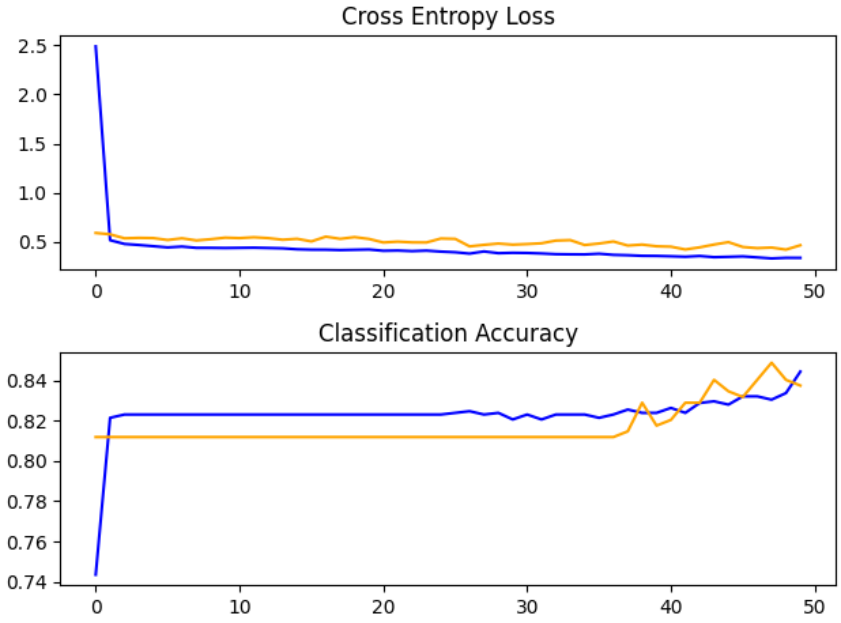


Figure 3.10 : graphiques linéaires des courbes d'apprentissage de la perte et de la précision pour le modèle de base avec augmentation des données sur le jeu de données drone et non drone.

Discussion :

Nous avons exploré deux améliorations différentes du modèle de base.

Les résultats peuvent être résumés ci-dessous, même si nous devons encore supposer une certaine variance dans ces résultats compte tenu de la nature stochastique de l'algorithme :

- **VGG3** + abandon (dropout) : **77.143%**
- Augmentation des données + **VGG3** : **78.286%**

Comme suspecté, l'ajout de techniques de régularisation ralentit la progression des algorithmes d'apprentissage et réduit le sur-ajustement, ce qui se traduit par une diminution des performances sur l'ensemble des données exclues. Il est probable que la combinaison des deux approches avec une augmentation supplémentaire du nombre d'époques de formation se traduira par de nouvelles améliorations.

Ce n'est que le début des types d'améliorations qui peuvent être explorées sur cet ensemble de données. En plus des modifications apportées aux méthodes de régularisation décrites, d'autres méthodes de régularisation pourraient être explorées, telles que la décroissance du poids et l'arrêt précoce.

Cela peut valoir la peine d'explorer les modifications apportées à l'algorithme d'apprentissage, telles que les modifications du taux d'apprentissage, l'utilisation d'un calendrier d'apprentissage ou un taux d'apprentissage adaptatif tel qu'Adam.

D'autres architectures de modèles peuvent également valoir la peine d'être explorées. On s'attend à ce que le modèle de base choisi offre plus de capacité que ce qui pourrait être requis pour ce problème et un modèle plus petit peut être plus rapide à entraîner et à son tour pourrait entraîner de meilleures performances.

3.6.3 L’apprentissage par transfert :

L'apprentissage par transfert consiste à utiliser toute une partie d'un modèle formé sur une tâche connexe.

Un modèle utile pour l'apprentissage par transfert est l'un des modèles VGG, comme le VGG-16 avec 16 couches qui, au moment de son développement, a obtenu les meilleurs résultats sur le défi de classification de photos ImageNet.

Le modèle est composé de deux parties principales, la partie extracteur de caractéristiques du modèle qui est composée de blocs VGG et la partie classificateur du modèle qui est composée de couches entièrement connectées et de la couche de sortie.

Nous pouvons utiliser la partie d'extraction de caractéristiques du modèle et ajouter une nouvelle partie classificateur du modèle qui est adaptée à l'ensemble de données drones et non drones. Plus précisément, nous pouvons maintenir les poids de toutes les couches convolutives fixés pendant l'entraînement, et former uniquement de nouvelles couches entièrement connectées qui apprendront à interpréter les caractéristiques extraites du modèle et à effectuer une classification binaire.

Ceci peut être réalisé en chargeant le modèle VGG-16-, en supprimant les couches entièrement connectées de l'extrémité de sortie du modèle, puis en ajoutant les nouvelles couches entièrement connectées pour interpréter la sortie du modèle et effectuer une classification. La partie classifiante du modèle peut être supprimée automatiquement en définissant l'argument « *include_top* » sur « *False* », ce qui nécessite également que la forme de l'entrée soit également spécifiée pour le modèle, dans ce cas (224, 224, 3). Cela signifie que le modèle chargé se termine à la dernière couche

de Pooling Max, après quoi nous pouvons ajouter manuellement une couche d'aplatissement (Flatten layer) et les nouvelles couches de classification (New Classifier Layers).

Dans ce cas, peu de formation sera nécessaire, car seule la nouvelle couche entièrement connectée et de sortie a des poids pouvant être entraînés. En tant que tel, nous fixerons le nombre d'époques d'entraînement à 10.

Le modèle VGG-16- a été formé sur un jeu de données de défi ImageNet spécifique. En tant que tel, il est configuré pour s'attendre à ce que les images d'entrée aient la forme 224x224 pixels. Nous utiliserons cela comme taille cible lors du chargement de photos à partir de l'ensemble de données drones et non drones.

Le modèle s'attend également à ce que les images soient centrées. Autrement dit, pour avoir les valeurs de pixel moyens de chaque canal RVB telles que calculées sur l'ensemble des données d'apprentissage ImageNet soustraites de l'entrée. Keras fournit une fonction pour effectuer cette préparation pour des photos individuelles via la fonction *preprocess_input()*.

Néanmoins, nous pouvons obtenir le même effet avec ImageDataGenerator en définissant l'argument « *featurewise_center* » sur « *True* » et en spécifiant manuellement les valeurs moyennes de pixels à utiliser lors du centrage comme valeurs moyennes de l'ensemble de données d'entraînement ImageNet : [123.68, 116.779, 103.939].

L'exécution de l'exemple ajuste d'abord le modèle, puis signale les performances du modèle sur l'ensemble des données de test d'attente.

Dans ce cas, nous pouvons voir que le modèle a obtenu des résultats très impressionnants avec une précision de classification d'environ 97% sur l'ensemble des données de test d'exclusion.

Résultats :

```
                    Found 1237 images belonging to 2 classes.
                    Found 353 images belonging to 2 classes.
                    Found 177 images belonging to 2 classes.

> 97.6
```

En examinant les courbes d'apprentissage, nous pouvons voir que le modèle s'adapte rapidement à l'ensemble des données. Il ne montre pas de sur-ajustement important, bien que les résultats suggèrent qu'une capacité supplémentaire du classificateur et/ ou l'utilisation de la régularisation pourraient être utiles.

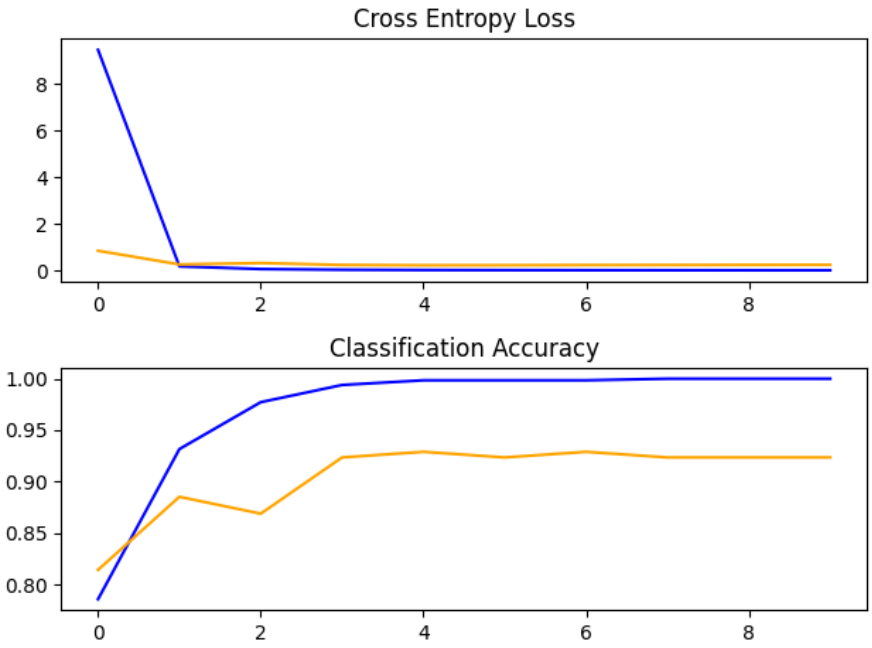


Figure 3.11 : graphiques linéaires des courbes d'apprentissage de perte et de précision pour le modèle d'apprentissage par transfert VGG16 sur l'ensemble des données drones et non drones.

3.6.4 Finalisation du modèle et prédictions :

Le processus d'amélioration du modèle peut se poursuivre aussi longtemps que nous avons des idées et le temps et les ressources pour les tester.

À un moment donné, une configuration finale du modèle doit être choisie et adoptée. Dans ce cas, nous garderons les choses simples et utiliserons l'approche d'apprentissage par transfert VGG-16- comme modèle final.

Tout d'abord, nous finaliserons notre modèle en ajustant un modèle sur l'ensemble des données d'entraînement et en enregistrant le modèle dans un fichier pour une utilisation ultérieure. Nous allons ensuite charger le modèle sauvegardé et l'utiliser pour faire une classification sur une seule image.

3.6.5 Préparation de l'ensemble des données finales :

Un modèle final est généralement adapté à toutes les données disponibles, telles que la combinaison de tous les ensembles de données d'entraînement et de test.

La première étape consiste à préparer l'ensemble des données d'entraînement de sorte qu'il peut être chargé par la *classe ImageDataGenerator* via la fonction *flow_from_directory()*. Plus précisément, nous devons créer un nouveau répertoire avec toutes les images d'entraînement organisées en sous-répertoires *drones/* et *non drones/* sans aucune séparation en répertoires *train/* ou *test/*.

Cela peut être réalisé en mettant à jour le script. Dans ce cas, nous allons créer un nouveau dossier *finalize_drones_vs_nondrones/* avec des sous-dossiers *drone /* et *non drone /* pour l'ensemble de données d'entraînement et de test.

3.6.6 L'enregistrement du modèle final :

Nous sommes maintenant prêts à ajuster un modèle final sur l'ensemble de données d'entraînement et de test.

Le *flow_from_directory()* doit être mis à jour pour charger toutes les images du nouveau répertoire *finalize_drones_vs_nondrones/*.

Une fois ajusté, nous pouvons enregistrer le modèle final dans un fichier h5 en appelant la fonction *save()* sur le modèle et passer le nom de fichier choisi.

Après avoir exécuté cet exemple, nous avons maintenant un gros fichier de 84,6 mégaoctets avec le nom « *final_model.h5* » dans notre répertoire de travail actuel.

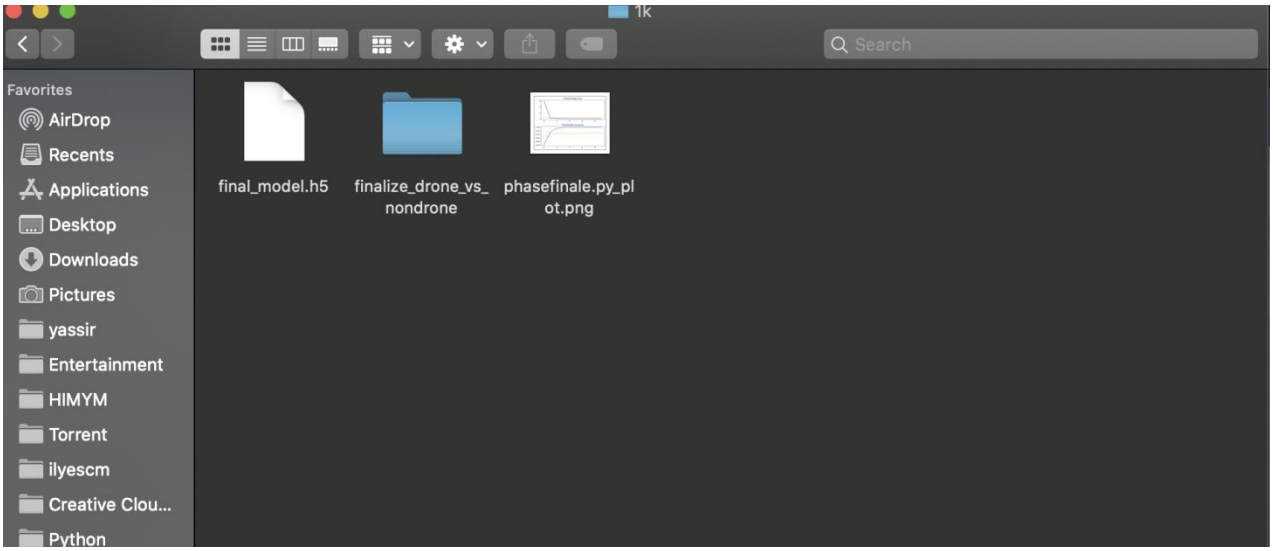


Figure 3.12 : une capture d'écran sur le fichier créé h5.

- **Les classifications :**

Nous pouvons utiliser notre modèle enregistré pour faire une classification sur de nouvelles images.

Le modèle suppose que les nouvelles images sont en couleur et qu'elles ont été segmentées de manière à ce qu'une image contienne au moins un drone ou un non drone.

Ci-dessous, une image qui n'a pas d'étiquette, mais nous pouvons clairement dire qu'il s'agit d'un drone. Nous avons enregistré cette image dans notre répertoire de travail actuel avec le nom de fichier « *a.jpg* ».

Figure 3.13 : drone (a.jpg).

Nous prétendrons qu'il s'agit d'une image entièrement nouvelle et invisible, préparée de la manière requise, et verrons comment nous pourrions utiliser notre modèle enregistré pour prédire entièrement ce que l'image représente. Pour cet exemple, nous nommerons la classe « *0* » pour « *Drone* ».

Remarque : les sous-répertoires d'images, un pour chaque classe, sont chargés par la fonction *flow_from_directory()* dans l'ordre alphabétique et attribués à un entier pour chaque classe. Le sous-répertoire « *drone* » vient avant « *non drone* », donc les étiquettes de classe reçoivent les entiers :

- ***drone = 0 ;***
- ***non drone = 1.***

Cela peut être changé via l'argument « *classes* » en appelant *flow_from_directory()* lors de l'apprentissage du modèle.

Tout d'abord, nous pouvons charger l'image et changer sa taille de 224 × 224 pixels. Elle peut ensuite être redimensionnée pour avoir un seul échantillon dans un ensemble de données. Les valeurs de pixel doivent également être centrées pour correspondre à la façon dont les données ont été préparées lors de l'apprentissage du modèle. La fonction *load_image()* implémente cela et retournera l'image chargée prête pour la classification.

Ensuite, nous pouvons charger le modèle comme dans la section précédente et appeler la fonction *predict()* pour prédire (classifier) le contenu de l'image sous la forme d'un nombre compris entre « *0* » et « *1* » pour « *drone* » et « *non drone* » respectivement.

L'exécution de l'exemple charge et prépare d'abord l'image, puis charge le modèle et prédit correctement que l'image chargée représente un « *drone* » ou une classe « *0* ».

> 0

> drone

Comme nous pouvons le voir, la dernière architecture (Transfer Learning) montre de très bons résultats, donc pour celle-ci seule, nous exécuterons plus de données.

Tableau 0.3 : les données utilisées lors de l'apprentissage (l'entraînement et la validation) et le test.

Les classes	Train-it (l'entraînement)	Test-it (la validation)	Test (test)
Drone	3327 images	1003 images	449 images
Non drone	1149 images	351 images	162 images
La somme	4476 images	1354 images	611 images

Les résultats :

Sur l'affichage, nous pouvons lire que l'apprentissage comporte 4476 images classé en deux classes (drone / non drone). 1354 images lors de la validation et le test est réalisé sur 611 images toujours classé en deux classes.

```
Found 4476 images belonging to 2 classes.
Found 1354 images belonging to 2 classes.
Found 611 images belonging to 2 classes.

>94.205
```

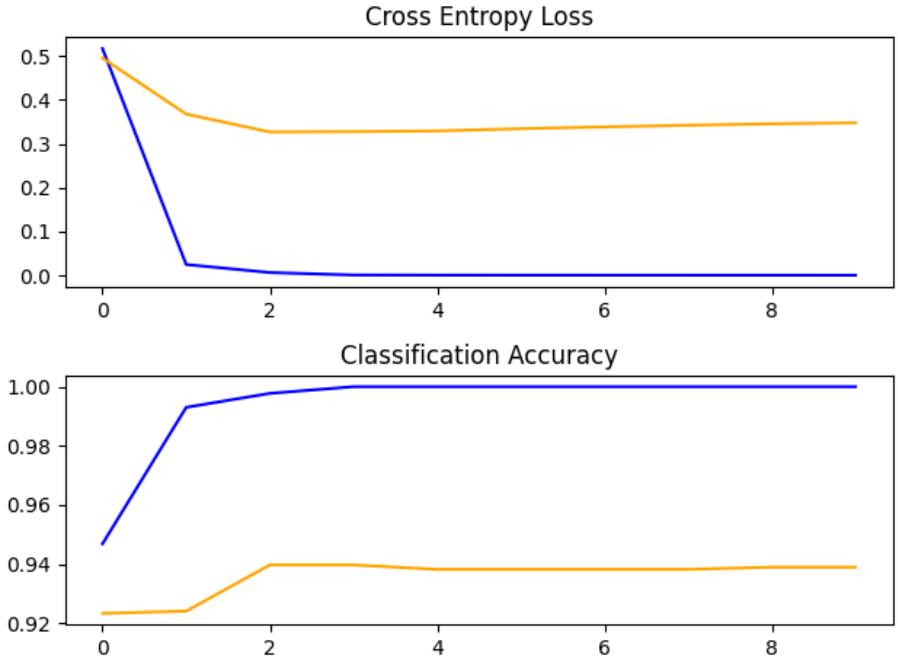


Figure 3.14 : graphiques linéaires des courbes d'apprentissage de perte et de précision pour le modèle d'apprentissage par transfert VGG16 sur l'ensemble des données drones et non drones.

Discussion :

Après l'époque deux, nous voyons clairement que la perte et la précision demeure constante et que nous avons une très grande perte.

Tableau 0.4 : les données utilisées lors de l'apprentissage (entraînement et validation) et le test.

Les classes	Train-it (l'entraînement)	Test-it (la validation)	Test (test)
Drone	7000 images	2000 images	1000 images
Non drone	2,338 images	666 images	333 images

Les résultats :

Sur l'affichage, nous pouvons lire que l'apprentissage comporte 9338 images classé en deux classes (drone / non drone). 2666 images lors de la validation et le test est réalisé sur 1333 images toujours classé en deux classes.

```
Found 9338 images belonging to 2 classes.
Found 2666 images belonging to 2 classes.
Found 1333 images belonging to 2 classes.

>94.570
```

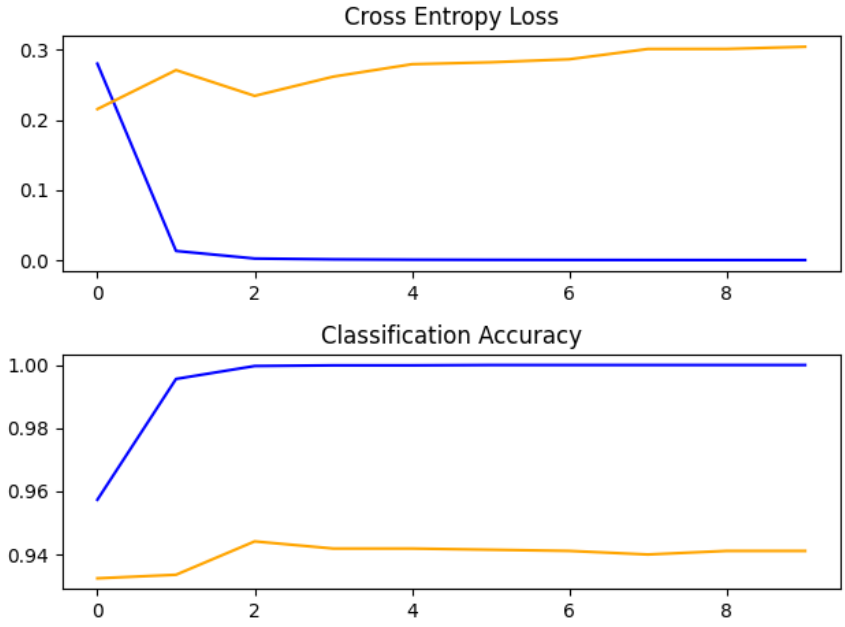


Figure 3.15 : graphiques linéaires des courbes d'apprentissage de perte et de précision pour le modèle d'apprentissage par transfert VGG16 sur l'ensemble de données drones et non drones.

Discussion :

Nous voyons clairement que la perte de test (orange) qui est censée diminuer, semble légèrement augmenter. Les performances ne semblent pas beaucoup s'améliorer (elles restent inchangées après la deuxième période).

Tableau 0.5 : les résultats obtenus de transfert learning à base de VGG16 avec différente quantité d'images.

1er essaie	2ème essaie	3ème essaie
97.6%	94.205%	94.570%

Discussion :

Nous constatons une diminution des performances, cela est dû au centrage manuel que nous avons fixé sur notre architecture. En effet, l'élément important sur chaque image (drone / non drone) n'est pas toujours positionné au centre de l'image. Pour régler ce problème, nous devons soit, utiliser uniquement les données où les images sont correctement centrées, soit modifier toutes les images de manière à ce que l'élément important de chaque image soit au centre de l'image. On peut aussi changer les chiffres du centrage manuel.

3.7 La détection :

Tout d'abord, nous avons installé la version de tensorflow « 1.13.1 », et quelques packages nécessaires pour la détection d'objet. Nous avons par la suite, créé un fichier nommé « drone » contenant trois autres fichiers nommés : images, data, training.

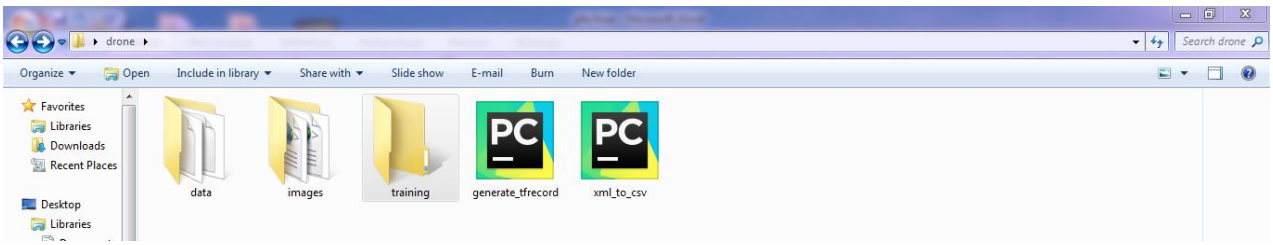


Figure 3.16 : une capture d'écran du fichier drone.

Nous avons aussi créé à l'intérieur du fichier images, deux autres fichiers : un fichier nommé "train" (les images d'apprentissage) et un autre fichier nommé "test" (les images de test).

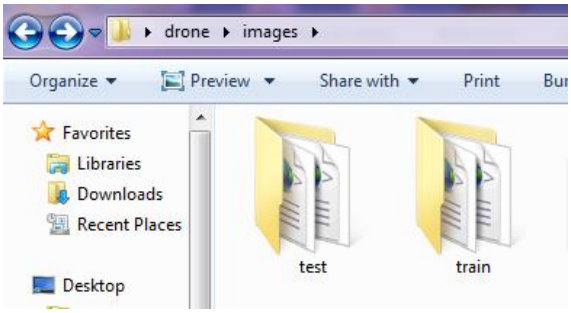


Figure 3.17 : une capture d'écran dans le fichier images.

Ensuite, il faut mettre les étiquettes pour chaque image et pour cela nous avons installé LabelImg qui nous permet d'encadrer les drones trouvés dans les images.

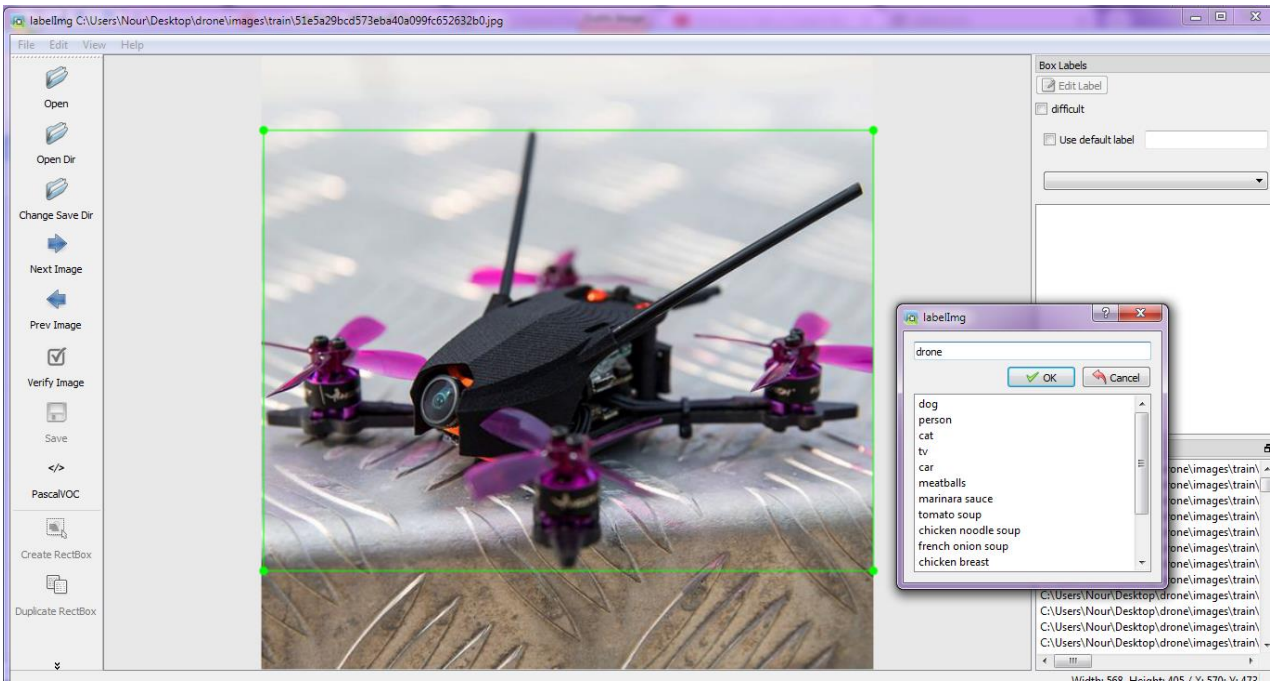


Figure 3.18 : l'encadrement des images à l'aide du labelImg.

Un fichier XML sera ensuite créé pour chaque image qui a les caractéristiques du cadre (dimensions, position dans l'image). Nous avons encadré 480 images en total et nous les avons divisées comme suit : 90% pour train et 10% pour test.

Donc 432 images dans le fichier train (la phase d'apprentissage) et 48 images dans le fichier test.

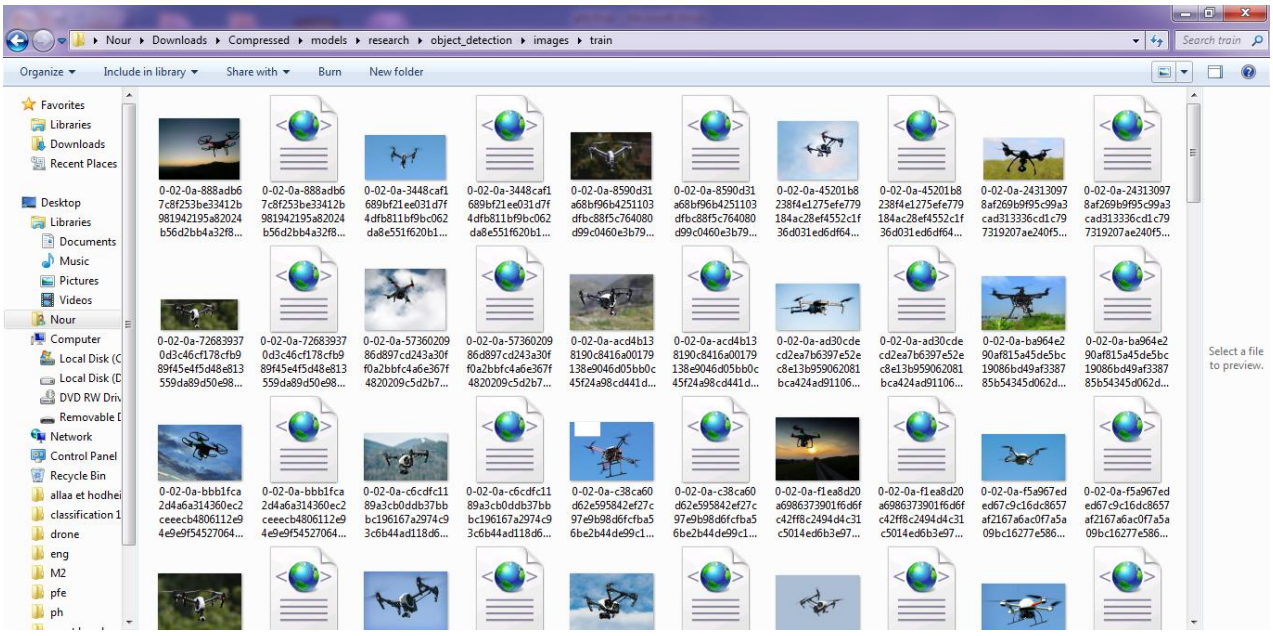
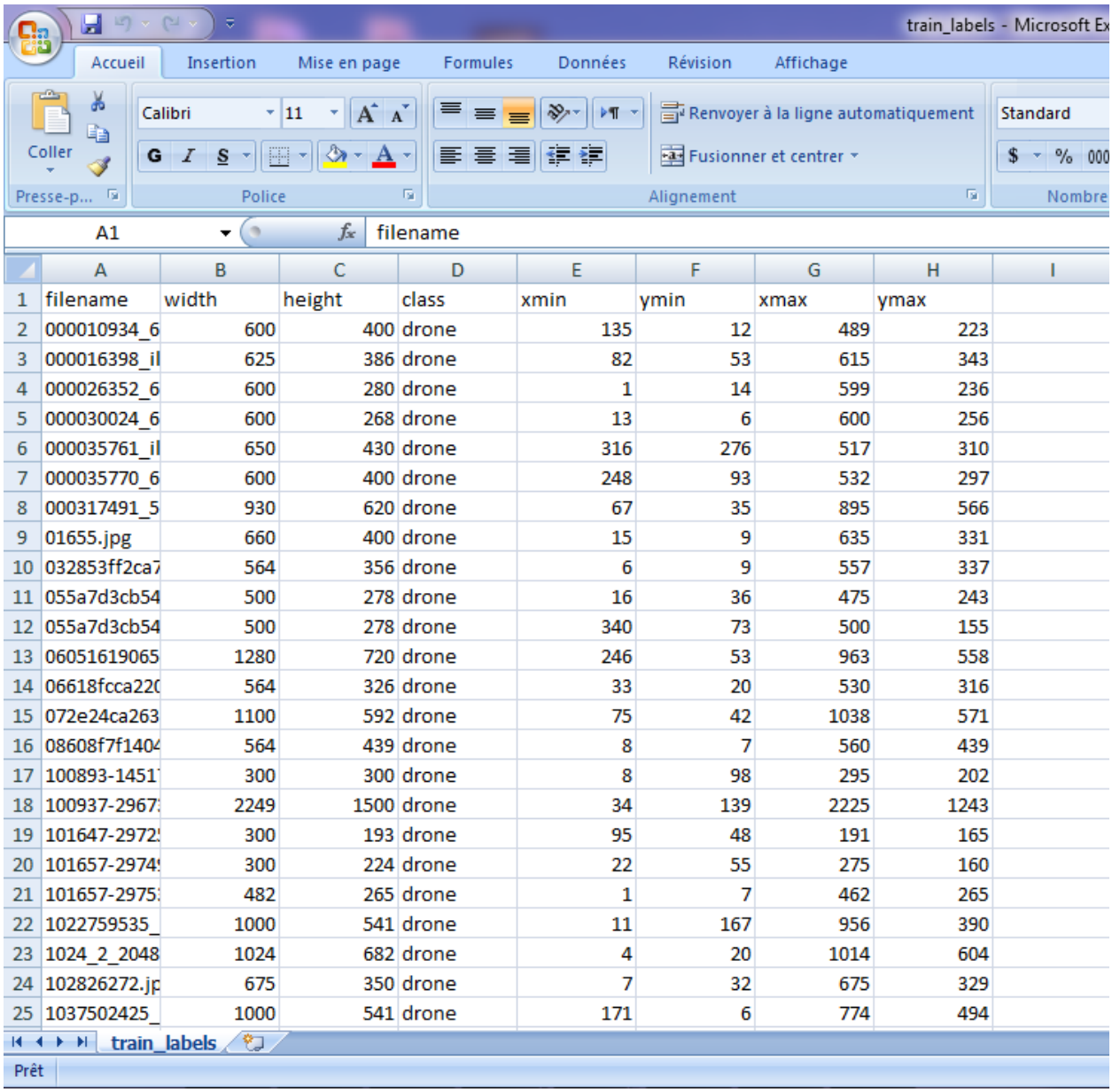


Figure 3.19 : une capture d'écran des images de train.

Ensuite, nous avons exécuté sur notre base (tensorflow) un programme qui converti les fichiers XML en CSV.



train_labels - Microsoft Ex

Accueil | Insertion | Mise en page | Formules | Données | Révision | Affichage

Coller | Calibri | 11 | Renvoyer à la ligne automatiquement | Standard | Fusionner et centrer | Presse-p... | Police | Alignement | Nombre

A1 | filename

	A	B	C	D	E	F	G	H	I
1	filename	width	height	class	xmin	ymin	xmax	ymax	
2	000010934_6	600	400	drone	135	12	489	223	
3	000016398_il	625	386	drone	82	53	615	343	
4	000026352_6	600	280	drone	1	14	599	236	
5	000030024_6	600	268	drone	13	6	600	256	
6	000035761_il	650	430	drone	316	276	517	310	
7	000035770_6	600	400	drone	248	93	532	297	
8	000317491_5	930	620	drone	67	35	895	566	
9	01655.jpg	660	400	drone	15	9	635	331	
10	032853ff2ca7	564	356	drone	6	9	557	337	
11	055a7d3cb54	500	278	drone	16	36	475	243	
12	055a7d3cb54	500	278	drone	340	73	500	155	
13	06051619065	1280	720	drone	246	53	963	558	
14	06618fcca22(	564	326	drone	33	20	530	316	
15	072e24ca263	1100	592	drone	75	42	1038	571	
16	08608f7f140-	564	439	drone	8	7	560	439	
17	100893-1451	300	300	drone	8	98	295	202	
18	100937-2967	2249	1500	drone	34	139	2225	1243	
19	101647-2972	300	193	drone	95	48	191	165	
20	101657-2974	300	224	drone	22	55	275	160	
21	101657-2975	482	265	drone	1	7	462	265	
22	1022759535_	1000	541	drone	11	167	956	390	
23	1024_2_2048	1024	682	drone	4	20	1014	604	
24	102826272.jp	675	350	drone	7	32	675	329	
25	1037502425_	1000	541	drone	171	6	774	494	

train_labels

Prêt

Figure 3.20 : une capture d'écran du fichier Microsoft Excel-train_labels.csv.

Enfin, une exécution est faite sur le fichier train et sur le fichier test qui nous donne deux fichiers : train.record et test.record dans le fichier data.

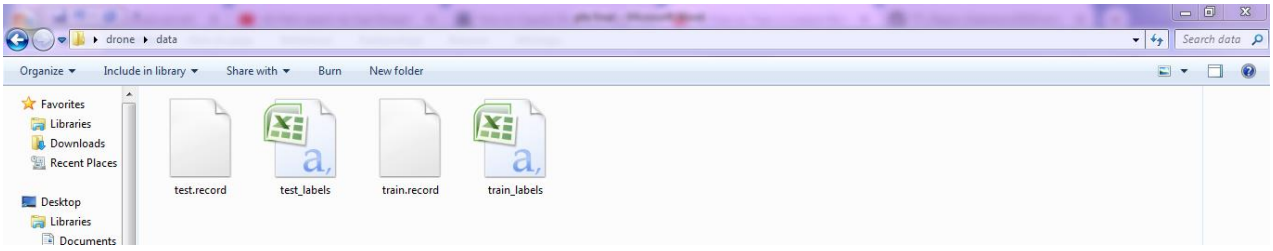


Figure 3.21 : une capture d'écran qui montre l'état final des fichiers créés.

Avant de commencer la formation, nous avons téléchargé le modèle ssd_mobilenet_v1_coco trouvé sur « COCO-trained models » (un ensemble de modèles formés par COCO). Celui-ci détecte les objets à une vitesse de 30ms et une précision (mAP) de 21. Il a été choisi pour sa rapidité d'entraînement.

Afin de réduire le temps d'apprentissage, nous avons exécuté notre modèle sur Google Colab. Il s'agit d'un service cloud basé sur Jupyter Notebook fourni par Google (gratuitement), conçu pour la formation et la recherche en apprentissage automatique [62].

Après 11 279 itérations, le loss (perte) est comprise entre 0,5 et 1, mais nous nous sommes arrêtés ici pour voir les résultats car le loss stagnait de toute manière entre 0,5 et 1.

Figure 3.22 : l'exécution sur Google Colab.

Enfin, nous avons essayé des images statiques et dynamiques que nous n'avons pas utilisées lors de la formation au test de robustesse. Les résultats seront affichés en bas.

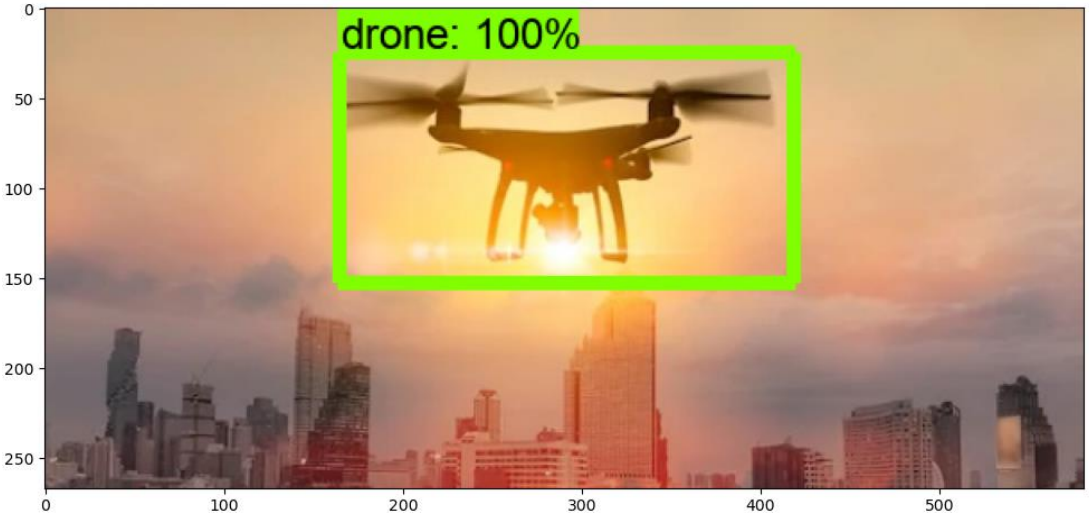


Figure 3.23 : détection d'un drone à 100%.

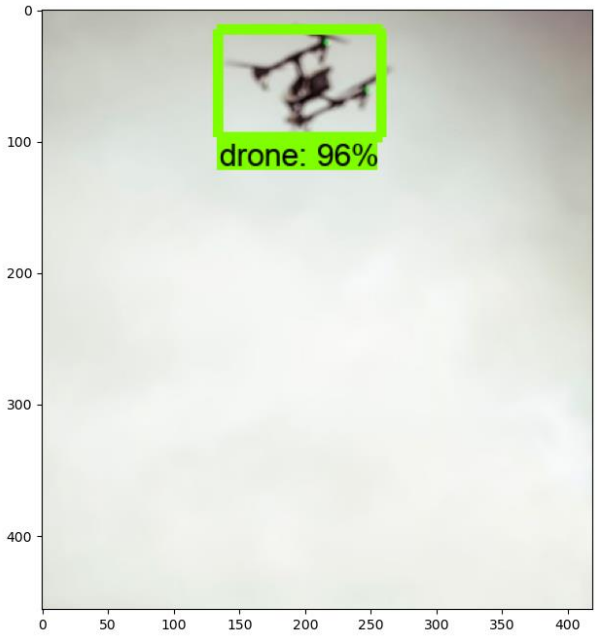


Figure 3.24 : détection d'un drone à 96%.

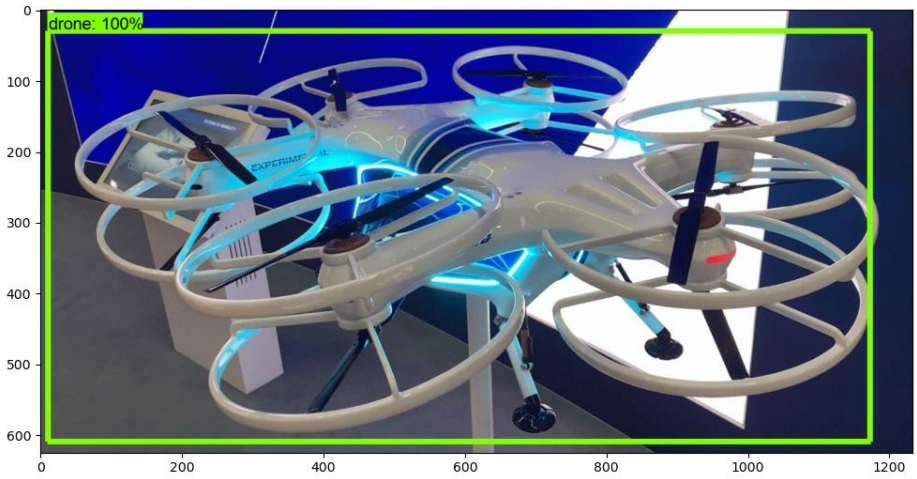


Figure 3.25 : détection d'un drone à 100%.

Figure 3.26 : détection d'un drone à 100%.

Figure 3.27 : pas de détection pour un sac plastique volant.

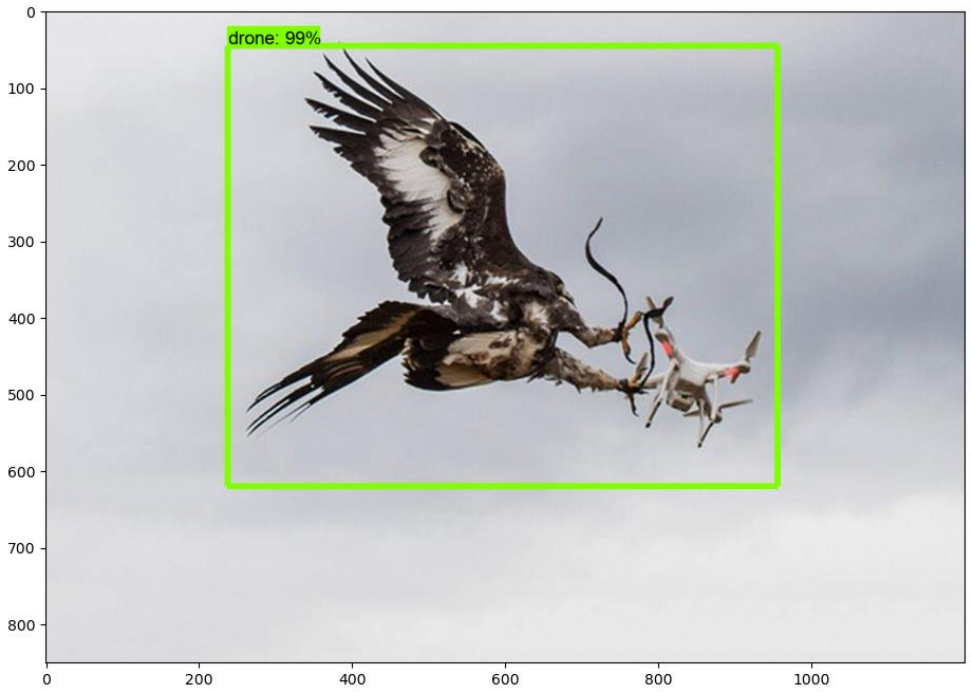


Figure 3.28 : détection d'un drone avec un oiseau à 99%.

Conclusion :

L'objectif de ce chapitre est de présenter les résultats de simulation obtenus à l'aide du logiciel Python sur les bases de données (drones / non drones) et de les commenter. Les travaux de ce mémoire consistent à faire la classification sur les images de drones, en utilisant les réseaux de neurones convolutifs, ceci en se basant sur deux critères de performance, l'erreur (Loss) et la précision (Accuracy). Lors de la conception, nous avons essayé toutes les architectures expliquées ci-dessus, et le programme a montré de très bonnes performances en termes de précision et de perte

lorsque nous avons utilisé la méthode d'apprentissage par transfert avec le modèle à trois blocs VGG16. Nous avons ensuite essayé trois bases de données plus volumineuses par rapport à la première mais avons remarqué une baisse de performances due au centrage manuel que nous avons fixé sur l'architecture.

La comparaison des résultats trouvés a montré que le nombre d'époque, la taille de la base de données et la profondeur des réseaux neuronaux, sont des facteurs importants pour obtenir les meilleurs résultats. Cependant, pour le Transfer Learning VGG16, nous avons constaté que plus les données sont volumineuses, moins nous avons de performances à cause de notre fixation manuelle de centrage.

Enfin, nous avons créé un modèle de détection de drones à l'aide d'images qui détectera ensuite les images statiques et dynamiques.

Conclusion générale

Le présent mémoire ne porte que sur la détection de micro et mini drones quelle que soit leur forme vu qu'elles sont les plus vulnérables. Ces dernières années, ces véhicules aériens sans pilote se sont rapidement développés dans les domaines civil et militaire et ont entraîné des risques considérables. Afin de les détecter, nous avons essayé d'implémenter dans notre programme plusieurs architectures de réseaux neuronaux convolutifs qui conduisent à la reconnaissance d'objets volants dans l'image.

Le but de ce projet de fin d'étude est de mettre en œuvre un programme puissant, facile et peu coûteux pour reconnaître et classer les drones de tous les autres objets volants.

Dans le premier chapitre, nous nous sommes familiarisés avec les réseaux de neurones et leurs fonctionnements, ainsi que leurs applications et utilisations.

Le passage en revue de plusieurs méthodes nous a guidés dans le choix de l'architecture la plus efficace lorsque nous avons essayé, dans le processus de mise en œuvre, les architectures telles que les VGG ou d'autres méthodes qui peuvent améliorer ses performances et le transfer learning.

Un programme réalisé avec le logiciel Anaconda3 sur la base de données (drone/ non drone) a été mis en place et les résultats ont été présentés et discutés. Au cours de la partie programmation, nous avons essayé toutes les architectures expliquées en détail dans le deuxième chapitre et nous avons trouvé que le programme fonctionnait très bien en termes d'erreur (loss) et de précision (accuracy) avec l'apprentissage par Transfer Learning sur le modèle à trois blocs VGG-16. Ensuite, nous avons essayé d'utiliser trois bases de données différentes pour analyser les performances des résultats mais du fait du centrage manuel fixé sur cette architecture, les performances se sont dégradées. Nous avons également constaté que le nombre d'itérations, la taille de la base de données et le nombre de blocs de réseau neuronal convolutif sont des facteurs importants pour obtenir les meilleurs résultats basés sur la perte et la précision. Nous avons également réalisé un programme pour la détection de drone avec l'image statique et dynamique.

Enfin, pour terminer, il faut savoir que nous avons rencontré quelques problèmes lors de la phase de mise en œuvre, l'utilisation d'un CPU nous a fait perdre trop de temps à cause du temps d'exécution trop long qui peut atteindre jusqu'à quatre heures lors de l'apprentissage. La solution est d'utiliser un ordinateur avec un GPU compatible (NVIDIA) qui peut considérablement réduire le temps d'exécution de l'apprentissage des réseaux de neurones convolutifs profonds vu le nombre de couches cachées et les bases de données très volumineuses. Nous suggérons en perspective d'ajouter des techniques de régularisation qui peuvent être une bonne solution pour augmenter les performances.

Bibliographie :

[1] : HUDGSON, Alice. *Drone livreur d'organes : le futire du transport dans le domaine de la greffe-Intel* [**en ligne**]. (01/05/2019) Disponible sur : < https://www.intel.fr/content/www/fr/fr/it-managers/healthcare-organ-drone-delivery.html > (consulté le 22/08/2020).

[2] : BAA, Mohammed et KHLEDJ, Omar. *Reconnaissance de caractères manuscrits ou de formes par réseau de neurones*. Mémoire de fin d'étude, Télécommunication : Université Djilali Bounaama Khemis Miliana, 2017, 84p.

[3] : BARREIRO LINDO, Flavio. *Interprétation d'images basée sur la technologie des réseaux de neurones*. Mémoire de bachelor, Informatique : Haute École de Gestion de Genève (HEG-GE), 2018, 86p.

[4] : IONOS. *Qu'est-ce que l'apprentissage automatique ?* **[en ligne]**. (Le 31/07/2019) Disponible sur : <https://www.ionos.fr/digitalguide/web-marketing/analyse-web/quest-ce-que-lapprentissage-automatique/ > (consulté le 20/07/2020).

[5] : CLAUDE, Touzet. *Les Reseaux De Neurones Artificiels, Introduction Au Connexionnisme* : Cours, Exercices Et Travaux Pratiques, 1992, 130p.

[6] : H. ABDI. *Les Réseaux de neurones*, édition presses universitaires de Grenoble, 1994.

[7] : CHAMEKH, Abdessalem. *Optimisation des procédés de mise en forme par les réseaux de neurones artificiels*. Thèse de doctorat en Sciences de l'ingénieur : Université d'Angers, 2008, 144p.

[8] : Le cerveau humain.(23/02/2012) [schéma] **In :** *TPE sur les robots et les Hommes*. Disponible sur : < https://tpe-robots-hommes-2012.e-monsite.com/medias/images/schema-neurone1.gif > (consulté le 18/04/2020).

[9] : MERZOUKA, Nouressadat. *Etudes Des Performances Des Réseaux De Neurones Dynamiques A Représenter Des Systèmes Réels: Une Approche Dans L'espace D'état*. Mémoire de Magister, contrôle : Universite De Setif 1, 2009, 134p.

[10] : M.Asencio., P.Gros., J.Patry, Les drones tactiques à voilure tournante dans les engagements contemporains. *Fondation pour la recherche stratégique 27 rue Damesme– 75013 PARIS*, 2010, n°8, 160p, (ISSN : 1966-5156).

[11] : DJERIRI, Youcef. Les réseaux de neurones artificiels. *Neural Network Applied To Power Quality Enhancement*, 2017, 39p.

[12] : CHEKROUN, Soufyane. Modèle du neurone formel de Mac Culloch et Pitts (avec biais). (2014) **In :** *ResearchGate*. Disponible sur : < https://www.researchgate.net/figure/Modele-du-neurone-formel-de-Mac-Culloch-et-Pitts-avec-biais-Il-sagit-de-realiser_fig1_328996656 > (consulté le 21/08/2020).

[13] : ABADI, Mehdi. *Réalisation d'un réseau de neurones "SOM" sur une architecture matérielle adaptable et extensible à base de réseaux sur puce "NoC"*. Thèse de doctorat, génie électrique : Université de Lorraine ; Université du Centre (Sousse, Tunisie), 2018, 197p.

[14] : FLORENT, Simon. *Deep Learning, les fonctions d'activation* [**en ligne**]. (Le 04/10/2018) Disponible sur : < https://www.supinfo.com/articles/single/7923-deep-learning-fonctions-activation > (consulté le 15/02/2020).

[15] : HOUSSOU, Mohammed. *Optimisation De La Structure Des Réseaux De Neurones Par Algorithmes Génétiques*. Mémoire de Magister, Electrique : Université De Boumerdès, 2005, 115p.

[16] : Rip Tutorial. *Machine-Learning Fonctions d'activation* [**en ligne**]. Disponible sur : < https://i.stack.imgur.com/YUC5S.png > (consulté le 10/08/2020).

[17] : Machine learning. *Réseaux de neurones à plusieurs classes* **[en ligne].** (Le 10/02/2020) Disponible sur : < https://developers.google.com/machine-learning/crash-course/multi-class-neural-networks/softmax?hl=fr > (consulté le 23/08/2020).

[18] : Deeply learning (le 26/09/2018). [Photo] **In** : *la fonction d'activation Par Bastien Maurice*. Disponible sur : < https://deeplylearning.fr/cours-theoriques-deep-learning/fonction-dactivation/ > (consulté le 23/08/2020).

[19] : HABIBOULLAH, Beha Dine et LADJEDEL, Billal. *Utilisation des réseaux de neurones artificiels pour la prédiction de la vitesse de vent*. Mémoire de fin d'étude, Electronique : Université Mohamed Boudiaf M'SILA, 2017, 78p.

[20] : L'Intelligence Artificielle- TPE. *Comparaison entre réseau de neurones biologique et artificiel* **[en ligne]**. (31/01/2020) Disponible sur : < https://iatpe2015.wordpress.com/ > (consulté le 22/08/2020).

[21] : NOUICER, Imane EPS. MADJOUB. *Optimisation des réseaux de neurones par l'AIS pour le traitement des données optiques*. Mémoire de Magister, Informatique : Université des sciences et de la technologie d'Oran Mohamed Boudiaf, 2014, 92p.

[22] : Ben Cheikh Othman & Biteur Youcef. *Étude et réalisation d'un système de poursuite du point de puissance maximale en utilisant les réseaux de neurones artificiels - Application au système photovoltaïque* -. Mémoire de fin d'étude, Génie électrique : Universite Kasdi Merbah Ouargla, 2017, 74p.

[23] : CHIKH, Mourad. *Modelisation Hydrologique Par L'approche Connexionniste : Cas Du Bassin De L'oued Sebdou (Tafna- Nord-Ouest Algérien)*. Mémoire de Magister, Hydrogéologie : Université Abou Bekr Belkaid Tlemcen, 2011, 99p.

[24] : Le connexionnisme. *1. Principes-LEAD* **[en ligne]**. Disponible sur : < http://leadserv.u-bourgogne.fr/IMG/pdf/2_Le_connexionnisme.pdf > (consulté le 22/08/2020).

[25] : EL HMAIDI, Abdellah. Carte auto organisatrice de Kohonen. Chaque sphère symbolise un neurone de la couche d'entrée ou de la couche de sortie (carte de Kohonen). (Janvier 2016) [schéma] **In :** *Etude de la variation saisonnière des paramètres physico-chimiques des sédiments superficiels de la retenue du barrage Sidi Chahed (Meknès, Maroc) : approche par carte auto-organisatrice SOM*. Disponible sur : < https://www.researchgate.net/figure/Carte-auto-organisatrice-de-Kohonen-Chaque-sphere-symbolise-un-neurone-de-la-couche_fig2_304559519 > (consulté le 18/04/2020).

[26] : SENAT. *Pour une intelligence artificielle maîtrisée, utile et démystifiée - Rapport* [**en ligne**]. (Le 09/09/2020) Disponible sur : < https://www.senat.fr/rap/r16-464-1/r16-464-15.html > (consulté le 15/02/2020).

[27] : Iscomwiz Banque 3.0. Les sous-domaines de l'Intelligence Artificielle. [schéma] **In :** *L'Intelligence Artificielle en bref*. Disponible sur : < http://iscomwiz-banque3point0.e-monsite.com/pages/l-intelligence-artificielle-en-bref.html > (consulté le 15/08/2020).

[28] : IONOS. *Qu'est-ce qu'un réseau neuronal artificiel ?* **[en ligne]**. (Le 10/03/2020) Disponible sur :<https://www.ionos.fr/digitalguide/web-marketing/search-engine-marketing/quest-ce-quun-reseau-neuronal-artificiel/ > (consulté le 20/07/2020).

[29] : BENAMAR, Houmadi. *Étude Exploratoire D'outils Pour Le Data Mining*. Mémoire de fin d'étude : Université Du Québec À Trois-Rivières. Canada, 2007, 116p.

[30] : Conférence : *Mieux connaître les drones* - Onera, 9 septembre 2005.

[31] : DOMINIQUE, David Et JEAN, Panhaleux. *Les Drones Civils, Enjeux Et Perspectives*. N° 008816-01. Ministère De L'écologie, Du Développement Durable Et De L'énergie, 2015, 107p.

[32] : DUBOIS, Samuel., YVES, Vanhellemont., DE BOUW, Michael. *Les drones au service de la construction : technologies, enjeux et perspectives*. Centre Scientifique Et Technique De La Construction, 2019, 69p.

[33] : FRED. Micro Drone 3.0 (18/06/2015). [Photo] **In :** *Helicomicro*. Disponible sur : < https://www.helicomicro.com/2015/06/18/micro-drone-3-0/ > (consulté le 15/08/2020).

[34] : Hoppy King.com. *H-King Ridge Ryder Slope Wing EPO 913mm (PNF)* [Photo] **In :** *HobbyKing.com*. Disponible sur : < https://hobbyking.com/en_us/hobbykingtm-ridge-ryder-slope-wing-epo-913mm-pnf.html?___store=en_us > (consulté le 15/08/2020).

[35] : DUNCAN, Trevor. Rotor Drone Pro (27/11/2017). [Photo] In : *Rezo Camera Micro Drone RTF [VIDEO]*. Disponible sur : < https://www.rotordronepro.com/rezo-camera-micro-drone-rtf-video/#outer-popup > (consulté le 15/08/2020).

[36] : Futura science (26/09/2008). [Photo] In : un drone europeen à long rayon d'action pour les militaires Disponible sur : <https://www.futura-sciences.com/sciences/actualites/aeronautique-drone-europeen-long-rayon-action-militaires-16796/ > (consulté le 15/08/2020).

[37] : DUGAST, Stéphane. Plus de 120 heures de vols et près de 200 appontages de jour comme de nuit, le bilan depuis ses débuts à la fin du printemps 2012 du système de drone S-100 Camcopter et désigné dans la Marine par l'acronyme SERVAL pour Système Embarqué de Reconnaissance Vecteur Aérien Léger (24/07/2013). [Photo] **In :** *Ministère des Armées*. Disponible sur : < https://www.defense.gouv.fr/english/marine/a-la-une/drone-systeme-embarque-de-reconnaissance-vecteur-aerien-leger-serval > (consulté le 15/08/2020).

[38] : *OBJEKO*. Drone quadricoptère : meilleur drone et comparatif pour bien le choisir (27/10/2016). [Photo] **In :** *Objeko.com*. Disponible sur : < https://www.objeko.com/guide-achat-drone-quadricoptere-12165/ > (consulté le 15/08/2020).

[39] : GUTIERREZ, Peter. L'IoT sur le radar de l'opérateur de drone (30/05/2016). [Photo] **In :** *LotHub*. Disponible sur : < https://www.iothub.com.au/news/iot-on-drone-operators-radar-420134 > (consulté le 15/08/2020).

[40] : Direct Industry. *Drone hexarotor DS6-Y6* [Photo] **In :** *Direct Industry* Disponible sur : < https://www.directindustry.fr/prod/dronesys/product-181028-1781509.html > (consulté le 15/08/2020).

[41] : Amazon Business. *Xfold Spy-8urtf Supports Spy X8 Octocopter* [Photo] **In :** *Amazon.fr*. Disponible sur : < https://www.amazon.fr/Xfold-Spy-8urtf-Supports-Spy-X8-Octocopter/dp/B01HGIOSEC > (consulté le 15/08/2020).

[42] : GROIZELEAU, Vincent. *Un drone Harfang* (21/07/2011). [Photo] **In :** *Mer et Marine*. Disponible sur : < https://www.meretmarine.com/fr/content/dassault-en-negociation-pour-le-remplacement-des-drones-harfang > (consulté le 15/08/2020).

[43] : LAGNEAU, Laurent. *Selon le Pentagone, le drone perdu par l'US Navy volait à 34 km des côtes iraniennes quand il a été abattu* (20/06/2019). [Photo] **In :** *zone militaire opex360.com* Disponible sur : < http://www.opex360.com/2019/06/20/selon-le-pentagone-le-drone-perdu-par-lus-navy-volait-a-34-km-des-cotes-iraniennes-quand-il-a-ete-abattu/ > (consulté le 15/08/2020).

[44] : BEN BOUDAOUD Lynda, ALBANE Saadia. *Contrôle d'un groupe d'avions sans pilote (UAVs)*. Mémoire de fin d'étude, Informatique : Université A/Mira de Béjaia, 2013, 68p.

[45] : CHAPLEAU, *FCAS photo Dassault Aviation* (28/12/2015). [Photo] **In :** RP Defense. Disponible sur : < http://rpdefense.over-blog.com/2015/12/lancement-d-une-etude-technico-operationnelle-relative-au-futur-drone-aerien-de-combat.html > (consulté le 15/08/2020).

[46] : EVAN, Ackerman. Photo: KU Leuven (18/08/2014). [Photo] **In** : *Ieee Spectrum*. Disponible sur : < https://spectrum.ieee.org/automaton/robotics/drones/vertikul-uav-explores-practicalities-of-delivery-drones > (consulté le 15/08/2020).

[47] : Luxorion. Exemples de nano drones…virtuels ! En effet, on les retrouve sur internet et notamment sur Youtube mais il s'agit de canulars. Si une technologie de pointe est capable de créer de robots, leur utilité reste à démontrer si ce n'est à faire le buzz ! [Photo] **In :** *Luxorion*. Disponible sur : < http://www.astrosurf.com/luxorion/Physique/bug-flying-robot-hoax1.jpg > (consulté le 15/08/2020).

[48] : RAHMIL, David-Julien. *Oiseau, méduse, serpent... Quand les robots s'inspirent de la nature* (25/09/2018). [Photo] **In :** *L'ADN*. Disponible sur : < https://www.ladn.eu/tech-a-suivre/biotech/robots-nature-inspiration-animaux/ > (consulté le 15/08/2020).

[49] : Mallys. MetaFly, le drone qui vole comme un papillon (25/03/2019). [Photo] **In :** Mallys.Fr. Disponible sur : < https://mallys.fr/2019/03/25/metafly-le-drone-qui-vole-comme-un-papillon/ > (consulté le 15/08/2020).

[50] : MOKRI, Mohammed Zakaria. *Classification des images avec les réseaux de neurones convolutionnels*. Mémoire de fin d'études, Informatique : Université Abou Bakr Belkaid, 2017, 62p.

[51] : Gérard Laurent, Ecrans plats et vidéoprojecteurs - 2e éd : Principes, fonctionnement et maintenance.

[52] : l'encyclopédie informatique Comment Ça Marche. *Introduction à la vidéo numérique* [en ligne]. Disponible sur : < www.commentcamarche.net > (consulté le 22/08/2020).

[53] : Pensee Artificielle. *Focus : Le Réseau de Neurones Convolutifs* **[en ligne]**. (Le 11/01/2019) Disponible sur : < https://penseeartificielle.fr/focus-reseau-neurones-convolutifs/ > (consulté le 06/08/2020).

[54] : SRIKANTH, Tammina. Transfer learning using VGG-16 with Deep Convolutional Neural Network for Classifying Images. *International Journal of Scientific and Research Publications*, 2019, vol 9, n°10, p. 143-149, (ISSN 2250-3153).

[55] : Shahrokh, Valaee., Labach, Alex., Salehinejad, Hojjat. An example of standard dropout. The left network is fully connected, and the right has had neurons dropped with probability 0.5. Dropout is not applied to the output layer (2019). [Photo] **In :** Survey of Dropout Methods for Deep Neural Networks. Disponible sur : < https://www.researchgate.net/publication/332778873_Survey_of_Dropout_Methods_for_Deep_Neural_Networks > (consulté le 15/04/2020).

[56] : RAHUL, Jain. *Why "early-stopping" works as Regularization?* (08/02/2020). [Schéma] **In :** *Medium.* Disponible sur : < https://medium.com/@rahuljain13101999/why-early-stopping-works-as-regularization-b9f0a6c2772 > (consulté le 15/08/2020).

[57] : Deeplearning4j. Early Stopping. [schéma] **In :** *Deeplearning4j*. Disponible sur : < https://deeplearning4j.konduit.ai/tuning-and-training/early-stopping > (consulté le 15/08/2020).

[58] : Data Analytics Post. *Lexique* **[en ligne]**. Disponible sur : < https://dataanalyticspost.com/Lexique/ > (consulté le 22/07/2020).

[59] : Connor Shorten & Taghi M. Khoshgoftaar. A survey on Image Data Augmentation for Deep Learning. *Journal du Big Data*, 2019, vol 6, n°60, 48p.

[60] : Google developers. *Glossaire du machine learning* [**en ligne**]. (05/03/2020) Disponible sur : < https://developers.google.com/machine-learning/glossary?hl=fr#l > (consulté le 22/08/2020).

[61] : Peltarion. *Crossentropie binaire* [**en ligne**]. Disponible sur : < https://peltarion.com/knowledge-center/documentation/modeling-view/build-an-ai-model/loss-functions/binary-crossentropy > (consulté le 22/08/2020).

[62] : HENRI, Michel. *Google Colab : Le Guide Ultime* [**en ligne**]. (Le 4/11/2019) Disponible sur : < https://ledatascientist.com/google-colab-le-guide-ultime/ > (consulté le 22/08/2020).