الجمهورية الجزائرية الديمقراطية الشعبية
République Algérienne démocratique et populaire

وزارة التعليم العالي و البحث العلمي
Ministère de l'enseignement supérieur et de la recherche scientifique

جامعة سعد دحلب البليدة
Université SAAD DAHLAB de BLIDA

كلية التكنولوجيا
Faculté de Technologie

قسم الإلكترونيك
Département d'Électronique

Mémoire de Master

Filière Télécommunication
Spécialité Réseaux & Télécommunication

Présenté par

Bentouati Sara

&

Bentiba Fella

Application de Deep Learning dans la classification d'image.

Proposé par : Amirouche Nesrine

Année Universitaire 2020-2021

Remerciements

Nous tenons à remercier en premier lieu, Dieu tout puissant pour la volonté, la santé et le courage qu'il nous a donné pour suivre nos études.

Nous souhaitons exprimer également nos remerciements à notre encadreur Mme. AMIROUCHE pour son aide si précieuse, sa patience et surtout ses orientations et l'intérêt qu'elle nous a accordé tout au long de l'élaboration de ce travail.

Sans oublier les membres du jury qui ont accepté l'évaluation de ce travail.

Nos remerciements sincères vont également à toutes personnes contribuant de près ou de loin à l'élaboration de ce travail.

Finalement, Nous remercions grandement nos enseignants qui ont enrichi nos connaissances et de nous avoir guidé durant cette formation professionnelle.

Dédicaces

A celle qui attend mon retour chaque jour

A celle qui m'a comblées d'affection, d'amour et de tendresse, et qui a veillé à côté de mon berceau pour consoler mes cris de douleurs, et qui n'a jamais cessé de le faire.

Ma mère

A celui qui fait le plus brave des hommes, m'ouvrant ses bras dans les sombres moments et m'aidant à aller de l'avant vers le meilleur, et qui m'a tant soutenu moralement et matériellement.

Mon père

A mes chères sœurs : Sara, Nabila, Nesrine, Hadjer

A mon frère : Amine

A mon fiancé : Toufik qui m'a aidé dans les moments difficiles.

A mon binôme : BENTOUTI Sara qui m'a supporté durant toute la durée de la réalisation de ce projet et a su excuser mes retards.

Fella.

Dédicaces

Je dédie ce mémoire à

Mes parents :

Ma mère, qui a œuvré pour ma réussite, de par son amour, son soutien, tous les sacrifices consentis et ses précieux conseils, pour toute son assistance et sa présence dans ma vie, reçois à travers ce travail aussi modeste soit-il, l'expression de mes sentiments et de mon éternelle gratitude.

Mon père, qui peut être fier et trouver ici le résultat de longues années de sacrifices et de privations pour m'aider à avancer dans la vie. Puisse Dieu faire en sorte que ce travail porte son fruit ; Merci pour les valeurs nobles, l'éducation et le soutient permanent venu de toi.

A toute ma famille BENTOUATI et ARGOUB.

A mon binôme Fella BENTIBA pour les excellents et les durs moments passés ensemble, ainsi que sa famille.

Sara.

ملخص:

تهدف هذه الأطروحة إلى تقديم أحدث ما توصلت إليه التكنولوجيا في التعلم العميق في مجال الاتصالات السلكية واللاسلكية. وهو يتألف من الكشف عن مساهمة التعلم العميق في الاتصالات، ثم تقديم وعرض مفهوم الذكاء الاصطناعي والتعلم الآلي والتعلم العميق للتوسع في الشبكات العصبية والخوارزميات المختلفة. الخطوة التالية هي إنشاء نموذج تصنيف للصور يعتمد على شبكات CNN باستخدام مكتبات مثل TensorFlow وKeras .

كلمات المفاتيح: الاتصالات السلكية واللاسلكية، الذكاء الاصطناعي، التعلم الآلي، التعلم العميق، الشبكات العصبية, الخوارزميات, شبكات CNN ,TensorFlow, Keras

Résumé :

Ce mémoire vise à fournir un état de l'art du Deep Learning dans les télécommunications. Il s'agit d'exposer la contribution du Deep Learning dans les télécommunications, puis d'introduire et de présenter le concept d'intelligence artificielle, de Machine Learning et de Deep Learning en l'étendant aux réseaux de neurones et à différents algorithmes. L'étape suivante consiste à créer un modèle de classification d'images basé sur les réseaux CNN en utilisant des bibliothèques comme TensorFlow et Keras.

Mots clé : Télécommunication, Intelligence Artificielle, Machine Learning, Deep Learning, Réseaux de neurones, Algorithme, CNN networks, TensorFlow, Keras.

Abstract :

This thesis aims to provide a state-of-the-art of Deep Learning in telecommunications. It consists of exposing the contribution of Deep Learning in telecommunication, and then introducing and presenting the concept of Artificial Intelligence, Machine Learning and Deep Learning expanding to neural networks and different Algorithms. The next step is to create an image classification model based on CNN networks using libraries like TensorFlow and Keras.

Keywords : Telecommunication, Artificial Intelligence, Machine Learning, Deep Learning, Neural Networks, Algorithm, CNN network, TensorFlow, Keras.

Listes des acronymes et abréviations

AI : Artificial Intelligence (Intelligence artificielle)

ANN : Artificial Neural Network (Réseau neuronal artificiel)

Cifar : Canadian Institute For Advanced Research

CNN : Convolutional Neural Network (Réseau Neuronal Convolutionnel)

CPU : Central Processing Unit

GPU : Graphic Processing Unit

DL : Deep Learning (Apprentissage profond)

GAN : Generative Adversarial Neural Network (Réseau neuronal adversarial génératif)

IoT : Internet of Things (Internet des Objets)

LTSM : Long Short Term Memory (Mémoire à long terme)

LTE : Long Term Evolution (Evolution à long terme)

MIMO : Multiple Input, Multiple Output (Entrées multiples, Sorties multiples)

ML : Machine Learning (Apprentissage Automatique)

MLP : Multi-Layer Perceptron (Perceptron multicouche)

MNIST : Modified/Mixed National Institute of Standards and Technology (Institut national modifié de normes et de technologies)

NLP : Natural Language Processing (Traitement de langue naturelle)

NOMA : Non Orthogonal Multiple Access (Accès multiple non orthogonal)

RASH : Random Access Channel (Canal d'accés aléatoire)

Relu : Rectified Linear Unit

RNN : Recurrent Neural Network (Réseau neuronal récurent)

SCMA : Satellite-Code Division Multiple Access (Système d'accès multiple à code clair)

Table des matières

Introduction générale 1

Chapitre 1 Les problèmes liés aux nouvelles techniques en télécommunication, et l'apport du Deep Learning 3

1.1 Introduction 4

1.2 Principaux problèmes que le Deep Learning permet de résoudre 5

1.2.1. Représentation du schéma de codage/décodage 5

1.2.2 Détection d'anomalies 6

1.2.3 Prévision du trafic 7

1.3 Conclusion 8

Chapitre 2 Historique et définition du Deep Learning 9

2.1 Introduction 10

2.1.1 Intelligence artificielle 10

2.1.2 Machine Learning 11

2.2 Deep Learning 11

2.2.1 Historique et évolution du Deep Learning 11

Les premiers réseaux de neurones 11

La création du « Perceptron » 11

La première version du modèle de rétropropagation continue 12

Le tout premier perceptron multicouche 12

La publication du livre "Perceptrons" qui révèle certaines imperfections que présente Le perceptron de Rosenblatt 12

Le premier hiver de l'intelligence artificielle 12

L'invention du Neocognitron 12

La création du réseau Hopfield, et l'adoption pratique de la rétropropagation dans les réseaux neuronaux 13

La création de la machine de Boltzmann 13

La création de NetTalk et la réussite de la rétropropagation dans le réseau neuronal

L'entraînement d'un réseau neuronal convolutif à reconnaitre les chiffres, et la publication de la première version du Théorème d'Approximation Universelle 13

La publication d'un article sur un type d'architecture de réseau neuronal révolutionnaire 14

L'utilisation des GPU pour la formation des réseaux neuronaux profonds 14
Le lancement de la base de données « ImageNet » 14
Le modèle CNN « AlexNet » remporte le concours de classification d'images d'Imagenet 14
La création du GAN : le réseau neuronal adversarial génératif 15
Un champion humain du jeu complexe du Go perd contre Le modèle d'apprentissage par renforcement profond 15
L'obtention du prix Turing par Yoshua Bengio, Geoffrey Hinton et Yann LeCun 15
2.2.2 Définition 15
2.2.3 Les réseaux des neurones 16
Le neurone 17
Principe de base 17
Les fonctions d'activation 18
2.2.4 Topologie des réseaux de neurones 20
Propagation vers l'avant de l'information (Feed-forward) 20
Réseaux récurrents (Feed-back) 22
2.3 L'apprentissage en Deep Learning 24
2.3.1 Apprentissage supervisé 24
2.3.2 Apprentissage non supervisé 24
2.3.3 Apprentissage par renforcement 24
2.4 Avantages et Inconvénients des réseaux de neurones 23
2.4.1 Les avantages 25
2.4.2 Les inconvénients 25
2.5 Conclusion 25
Chapitre 3 Exploitation du Deep Learning 26
3.1 Introduction 26
3.2 Les problèmes typiques 27
3.2.1 Types de tâches 27
3.3 Les algorithmes de Deep Learning 29
3.3.1 Les Auto encodeurs 29
3.3.2 Réseaux de neurones convolutifs (CNN) 30
Les couches de réseaux de neurones convolutifs 31

3.4 La classification d'image en Télécommunication 36
3.4.1 Introduction sur le traitement d'image 36
3.4.2 La vision artificielle et le traitement d'image 36
3.5 Conclusion 37
Chapitre 4 Implémentation et résultat 39
4.1 Introduction 39
4.2 Présentation des outils 39
4.2.1 Matériel 39
4.2.2 Logiciels 40
4.3 Base de données 43
4.4 Implémentation 46
4.4.1 L'architecture de notre réseau 46
Le modèle CNN sur CIFAR-10 46
Le modèle CNN sur MNIST : 48
4.5 Résultats obtenus et discussion 50
4.5.1 Résultats obtenus pour le modèle CIFAR-10 50
4.5.2 Résultats obtenus pour le modèle MNIST 52
4.6 Tableau de comparaison 53
4.7 Conclusion 54
Conclusion générale 55
Annexe 56
Bibliographie 64

Liste des figures

Figure 2.1. Intelligence Artificielle, Machine Learning et Deep Learning. 10
Figure 2.2. Un neurone artificiel. 17
Figure 2.3. Modélisation d'un neurone artificiel. 18
Figure 2.4. Représentation graphique de la fonction Sigmoïde 19
Figure 2.5. Représentation graphique de la fonction ReLu 19
Figure 2.6. Exemple d'un réseau de neurones artificiel. 20
Figure 2.7. Perceptron mono-couche. 21
Figure 2.8. Perceptron Multi-couches. 22
Figure 2.9. Réseaux récurrents. 22
Figure 2.10. Un exemple d'un réseau récurrent qui se déroule 23
Figure 3.1. L'architecture d'un auto encodeur 30
Figure 3.2. Les réseaux de neurones convolutifs 31
Figure 3.3. L'opération de convolution 32
Figure 3.4. Principe de la fonction ReLU 33
Figure 3.5. Exemple de principe du Pooling 33
Figure 3.6. Principe de la couche entièrement connectée 34
Figure 3.7. Un résumé des types d'architectures de réseaux de neurones 35
Figure 4.1. Logo Python 40
Figure 4.2. Logo TensorFlow 41
Figure 4.3. Interface Anaconda 42
Figure 4.4. Images de CIFAR-10 classes 44
Figure 4.5. Images de MNIST classes 45
Figure 4.6. L'architecture du modèle CNN sur CIFAR-10 46
Figure 4.7. Configuration de modèle CNN sur CIFAR-10 47
Figure 4.8. Illustration du rôle de la couche Flatten 48
Figure 4.9. L'architecture du modèle CNN sur MNIST 48
Figure 4.10. Configuration de modèle CNN sur MNIST 49
Figure 4.11. Précision et l'erreur de modèle CIFAR-10 (15 époques) 50
Figure 4.12. Précision et l'erreur de modèle CIFAR-10 (30 époques) 51
Figure 4.13. Précision et l'erreur de modèle CNN sur MNIST (15 époques) 52
Figure 4.14. Précision et l'erreur de modèle CNN sur MNIST (30 époques) 53

Liste des tableaux

Tableau 4.1. Base de données utilisées 45
Tableau 4.2. Comparaison des résultats 53

Introduction générale

Avec la popularisation de l'intelligence artificielle et son émergence graduelle en tant que technologie de base à l'origine de développements importants dans un large spectre de domaines, l'utilisation de Machine Learning et de Deep Learning a connu une progression énorme. Selon de nombreuses recherches et études, l'intelligence artificielle (IA), le Machine Learning et le Deep Learning devraient figurer parmi les carrières les mieux rémunérées et les plus profitables dans les prochaines années.

La vision par ordinateur est le domaine académique qui vise à obtenir une compréhension de haut niveau des informations de bas niveau fournies par les pixels bruts des images numériques.

Les robots, les moteurs de recherche, et bien d'autres ont des applications qui incluent la classification d'objets plus spécifiquement, la classification d'image. On peut dire que cette dernière est un sous-domaine de la vision par ordinateur, qui repose sur le Machine Learning et le Deep Learning. Au cours de la dernière décennie, le domaine de cet apprentissage a été dominé par les réseaux de neurones profonds, qui tirent parti des améliorations de la puissance de calcul et de la disponibilité des données. Un sous-type d'un réseau neuronal appelé réseau neuronal convolutif (CNN) est bien adapté aux tâches liées à l'image.

Le réseau neuronal est entraîné à rechercher différentes caractéristiques, telles que les arêtes, les angles et les différences de couleur sur l'image et pour les combiner en formes plus complexes.

Pour notre mémoire, nous passerons en revue des notions théoriques en commençant par la contribution importante du Deep Learning dans le domaine de télécommunication, le concept de base de l'IA, le Machine Learning et le Deep Learning et leur évolution, s'étendant aux réseaux de neurones et enfin aux réseaux convolutifs.

Après avoir présenté les notions générales et les problèmes résolus, nous prendrons une problématique qui est la classification d'image, nous montrerons comment créer un modèle avec des architectures différentes en appliquant les modèles CIFAR 10 et MNIST comme

modèles sur la base d’images. On évaluera au fur et à mesure les résultats de nos tests. On fera appel à plusieurs bibliothèques pour faciliter l'implémentation et accélérer le processus d'apprentissage.

Chapitre 1 Les problèmes liés aux nouvelles techniques en télécommunication

1.1 Introduction

Internet, les appareils mobiles et autres technologies sans fil se développent de jour en jour, pénétrant dans tous les domaines de notre vie quotidienne, du travail au divertissement. Selon Cisco, le trafic Internet mondial atteindra environ 30 Go par habitant, dont plus de 63 % proviendront des appareils sans fil et mobiles [1] , de sorte que le trafic des smartphones dépassera le trafic des PC la même année.

Ce phénomène est principalement dû à l'évolution de l'Internet à haut débit et, plus particulièrement, à la popularisation et à l'utilisation généralisée des smartphones et des plans de données accessibles associés. Bien que la 4G et son évolution à long terme (LTE) soient considérées comme une technologie mature, la technologie et l'architecture radio sont continuellement améliorées, comme dans le cadre de la norme LTE Advanced, une amélioration majeure de la LTE. Toutefois, à long terme, la prochaine génération de télécommunications (5G) est envisagée et bénéficie d'un élan considérable de la part de l'industrie et des chercheurs. En outre, avec le déploiement des applications de l'Internet des objets (IoT), des villes intelligentes, etc., une nouvelle pléthore de services 5G est apparue avec des exigences de conception très divergentes et technologiquement difficiles. Les futurs systèmes 5G évoluent pour prendre en charge l'explosion des volumes de trafic mobile, la gestion agile des ressources du réseau pour maximiser l'expérience de l'utilisateur et l'extraction d'analyses fines en temps réel. L'accomplissement de ces tâches est difficile, car les environnements mobiles sont de plus en plus complexes, hétérogènes et évolutifs [2].

La diversité et la complexité croissantes des architectures de réseaux mobiles ont rendu difficiles la surveillance et la gestion de la multitude d'éléments de réseaux. C'est pourquoi l'intégration d'une intelligence machine polyvalente dans les futurs réseaux mobiles suscite un intérêt inégalé de la part des chercheurs [3] . Les chercheurs en réseaux commencent également à reconnaître la puissance et l'importance de Deep Learning, et explorent son potentiel pour résoudre les problèmes spécifiques au domaine des réseaux mobiles [4].

Les problèmes de réseaux mobiles et d'apprentissage profond ont fait l'objet de recherches indépendantes. Ce n'est que récemment que des croisements entre les deux domaines sont apparus [5].

Le Machine Learning et le Deep Learning sont dominants dans les domaines de la médecine, de la reconnaissance d'images, des voitures [6], etc. La curiosité et les débats préliminaires sur la viabilité de la 5G par l'incorporation d'algorithmes de Deep Learning se sont révélés d'une grande importance pour l'amélioration des infrastructures sociales, le renforcement de nos pratiques ainsi que l'expansion de la numérisation. Même si le Deep Learning et la technologie sans fil 5G sont souvent considérés comme des domaines d'étude distincts, ils ont un effet puissant lorsqu'ils sont combinés.

1.2 Principaux problèmes résolus par le Deep Learning

1.2.1 Représentation du schéma de codage/décodage

La génération de l'information à la source et la reconstruction de cette information au récepteur constituent respectivement les processus de codage et de décodage. Toutefois, en raison de la nature instable des canaux, certaines perturbations et certains bruits dans le signal peuvent entraîner la corruption des données [7]. En considérant les réseaux 5G, où de nouvelles technologies, telles que MIMO, accès multiple non orthogonal (NOMA), mmWave seront déployées, les schémas de codage/décodage doivent être adaptés pour fonctionner correctement. Ces schémas doivent caractériser plusieurs phénomènes qui peuvent avoir un impact sur la transmission des données, tels que la diffraction du signal, l'évanouissement, la perte de chemin et la diffusion.

Dans [8], les auteurs ont proposé une solution basée sur le Deep Learning pour paramétrer le mappage bit-à-symbole et la détection multi-utilisateurs. Comme nous utilisons une modulation non orthogonale, la détection multi-utilisateurs devient un problème lourd. Dans [9], les auteurs ont proposé un modèle de Deep Learning pour apprendre le processus de codage/décodage du système MIMO-NOMA afin de minimiser l'erreur quadratique moyenne totale des signaux des utilisateurs. Dans [10], les auteurs ont proposé un modèle de Deep Learning à utiliser dans un système d'accès multiple à code clair (SCMA), qui est une technique NOMA prometteuse basée sur le code, dans le but de minimiser le taux d'erreur sur les bits. Dans [11], les auteurs ont considéré les systèmes MU-SIMO (multiuser single-input multiple-output).

Un modèle simple de Deep Learning a été envisagé pour la conception conjointe de formes d'onde multi-utilisateurs du côté de l'émetteur et la détection de signaux non cohérents du

côté du récepteur. L'objectif principal était de réduire la différence entre les signaux émis et reçus.

1.2.2 Détection d'anomalies

Les futurs réseaux 5G conduiront avec différents types de dispositifs sur des réseaux sans fil hétérogènes avec des débits de données plus élevés, une latence plus faible et une consommation d'énergie plus faible. Des mécanismes de gestion autonomes seront nécessaires pour réduire le contrôle et la surveillance de ces réseaux complexes [12].

Les systèmes de détection des anomalies sont importants pour identifier les flux réseau malveillants qui peuvent avoir un impact sur les utilisateurs et les performances du réseau. Cependant, le développement de ces systèmes reste un défi considérable en raison du grand volume de données généré dans les systèmes 5G [13].

Dans [13], les auteurs traitent des systèmes de défense de la cybersécurité dans les réseaux 5G, en proposant l'utilisation de modèles d'apprentissage profond capables d'extraire les caractéristiques des flux réseau et l'identification rapide des cybermenaces.

Dans [14], les auteurs ont proposé une solution basée sur le Deep Learning pour détecter les anomalies dans le trafic réseau, en considérant deux types de comportement comme des anomalies de réseau : les cellules dormantes et le trafic en flèche. Les cellules dormantes peuvent être dues à des défaillances du matériel d'antenne ou à des défaillances du canal d'accès aléatoire (RACH) en raison d'une mauvaise configuration du RACH, tandis que l'augmentation du trafic peut résulter de la congestion du réseau, où le trafic augmente mais avec un débit relativement faible pour satisfaire la demande des utilisateurs. Rappelons que le RACH est le canal responsable de l'attribution des ressources radio aux utilisateurs. Ainsi, lorsque le RACH ne fonctionne pas correctement, nous avons effectivement une cellule endormie où aucune activité de transmission n'a lieu.

1.2.3 Prévision du trafic

On s'attend à ce que le trafic Internet soit multiplié par dix d'ici 2027. Cela agit comme un point d'ancrage crucial pour créer la nouvelle génération d'architecture de réseau cellulaire [15]. La prédiction du trafic pour le jour, l'heure ou même la minute à venir peut être utilisée pour optimiser les ressources système disponibles, par exemple en réduisant la consommation d'énergie, en appliquant un ordonnancement opportuniste ou en prévenant les problèmes dans l'infrastructure [15].

Les travaux présentés dans [16] [17] ont proposé une solution basée sur le Deep Learning pour prédire le trafic pour les mécanismes de découpage du réseau. Notons que la 5G repose sur l'utilisation du découpage du réseau afin d'accueillir différents services et locataires tout en les isolant virtuellement. Dans [16], un mécanisme proactif de découpage du réseau a été proposé et un modèle de Deep Learning a été utilisé pour prédire le trafic avec une grande précision. Dans [17], un mécanisme nommé DeepCog a été proposé dans un but similaire. DeepCog peut prévoir la capacité nécessaire pour allouer les futures demandes de trafic dans les tranches de réseau tout en minimisant les violations de demande de service et le surprovisionnement des ressources.

Dans [18], les auteurs ont proposé différents modèles de Deep Learning pour la prédiction du trafic Internet mobile. Les auteurs ont utilisé les différents modèles pour considérer les aspects spatiaux et temporels du trafic. Le trafic maximal, moyen et minimal a été prédit pour les modèles proposés.

Nous reconnaissons plusieurs autres avantages à l'utilisation de l'apprentissage profond pour résoudre les problèmes d'ingénierie des réseaux

- L'un des principaux avantages de Deep Learning est qu'il permet d'extraire automatiquement des caractéristiques de haut niveau à partir de données présentant une structure complexe et des corrélations internes. Le processus d'apprentissage n'a pas besoin d'être conçu par un humain, ce qui simplifie considérablement l'élaboration préalable des caractéristiques [19]. L'importance de cet aspect est amplifiée dans le contexte des réseaux mobiles, car les données mobiles sont généralement générées par des sources hétérogènes, sont souvent bruyantes et présentent des modèles

spatiaux et temporels non triviaux [20], dont l'étiquetage nécessiterait un effort humain exceptionnel.

- Le Deep Learning est capable de traiter de grandes quantités de données. Les réseaux mobiles génèrent des volumes élevés de différents types de données à un rythme rapide.

1.3 Conclusion

Le Deep Learning joue un rôle de plus en plus important dans le domaine des réseaux mobiles et sans fil. Dans cette thèse, nous résumons les principaux problèmes auxquels la communication sans fil est confrontée, nous nous concentrons sur l'apprentissage profond et ses applications, et nous étudions l'intersection entre ces deux domaines différents.

Un point de préoccupation majeur dans l'intégration de la communication sans fil et de Deep Learning reste celui de la performance. Les solutions adoptées doivent non seulement fournir la solution attendue, mais elles doivent le faire au bon moment.

En général, l'utilisation de Deep Learning dans les télécommunications a déjà produit de nombreuses contributions importantes et on s'attend à ce qu'elles évoluent encore plus dans un avenir proche malgré les nombreuses limites.

Chapitre 2 Définition et Historique du Deep Learning

2.1 Introduction

Avant de comprendre ce qu'est le Deep Learning, nous devons introduire deux concepts principaux : Le premier est le concept d'intelligence artificielle. Le second est le Machine Learning (Apprentissage Automatique).

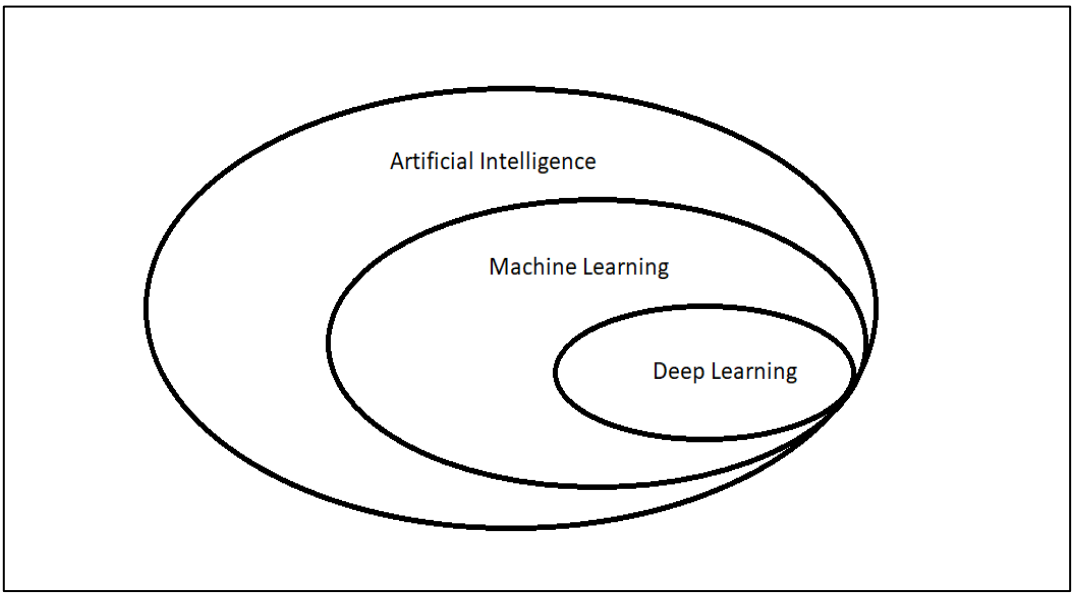


Figure 2.1. Intelligence Artificielle, Machine Learning et Deep Learning.

2.1.1 Intelligence artificielle

L'Intelligence Artificielle (IA) est un vaste domaine, où nous essayons d'imiter le comportement humain dans le but de rendre les machines si puissantes pour accomplir de nombreux types de tâches. Plus précisément, c'est la simulation des processus d'intelligence humaine par des machines, comme les ordinateurs, les systèmes embarqués, industriels, biomédicaux, financiers et bien d'autres encore. Ces processus comprennent l'apprentissage (l'acquisition d'informations et de règles d'utilisation de ces informations), le raisonnement (l'utilisation des règles pour parvenir à des conclusions approximatives ou définitives) et l'autocorrection. Les applications particulières de l'IA comprennent la reconnaissance vocale, la vision artificielle, le contrôle, la prédiction et divers autres domaines.

Ainsi, l'apprentissage automatique est un sous-domaine de l'IA et l'apprentissage profond est un sous-domaine de l'apprentissage automatique [21].

2.1.2 Machine Learning

Le Machine Learning (L'apprentissage Automatique) est un domaine de recherche en informatique qui traite des méthodes d'identification et de mise en œuvre de systèmes et algorithmes par lesquels un ordinateur peut apprendre, ce champ est généralement lié à l'intelligence artificielle.

A l'origine, ce domaine était consacré au développement de l'intelligence artificielle, mais en raison des limites de la théorie de la technologie qui existaient, il est devenu plus logique de concentrer ces algorithmes sur des taches spécifiques. La plupart des algorithmes (ML) tels qu'ils existent aujourd'hui se concentrent sur optimisation des fonctions.

Par conséquent, l'utilisation des algorithmes Machine Learning devient fréquemment un processus répétitif d'essais et d'erreurs, dans lequel le choix de l'algorithme parmi les problèmes donne des résultats de performance différents. Cela est acceptable dans certains contextes, mais dans le cas de la modélisation du langage et de la vision par ordinateur, cela devient problématique [22].

2.2 Deep Learning

2.2.1 Historique et évolution du Deep Learning

Les premiers réseaux de neurones

En 1943, les premiers réseaux de neurones ont été inventés par deux mathématiciens et neuroscientifiques du nom de Warren McCulloch et Walter Pitts.

Dans leur article "A Logical Calculus of the Ideas Immanent in Nervous Activity" Ils montrent le modèle mathématique d'un neurone biologique. Ce neurone de McCulloch Pitts a des capacités très limitées et ne possède aucun mécanisme d'apprentissage. Pourtant, il jettera les bases des réseaux neuronaux artificiels et de l'apprentissage profond [23].

La création du « Perceptron »

En 1957, et dans son article "The Perceptron : A Perceiving and Recognizing Automaton ", Le psychologue américain du nom de Franck Rosenblatt montre le nouveau modèle du neurone de McCulloch-Pitts, le "Perceptron", qui avait de véritables capacités d'apprentissage pour effectuer une classification binaire par lui-même. Cela inspire la révolution dans la recherche

sur les réseaux neuronaux peu profonds pour les années à venir, jusqu'au premier hiver de l'IA [24].

La première version du modèle de rétropropagation continue

Henry J. Kelley dans son article "Gradient Theory of Optimal Flight Paths" montre la toute première version du modèle de rétropropagation continue en 1960. Son modèle s'inscrit dans le contexte de la théorie du contrôle, mais il jette les bases d'un raffinement ultérieur du modèle qui sera utilisé dans les ANN (Artificial Neural Network) dans les années à venir [25].

Le tout premier perceptron multicouche

En 1965, Alexey Grigoryevich Ivakhnenko ainsi que Valentin Grigor'evich Lapa, créent une représentation hiérarchique d'un réseau de neurones qui utilise une fonction d'activation polynomiale et qui est formé à l'aide de la méthode de groupe de traitement des données (Group Method of Data Handling) (GMDH).

Il est maintenant considéré comme le tout premier perceptron multicouche, et Ivakhnenko est souvent considéré comme le père de l'apprentissage profond [26].

La publication du livre "Perceptrons" qui révèle certaines imperfections que présente Le perceptron de Rosenblatt

En 1969, Marvin Minsky et Seymour Papert publient le livre "Perceptrons" dans lequel ils montrent que Le perceptron de Rosenblatt ne peut pas résoudre des fonctions compliquées comme XOR. Pour de telles fonctions, les perceptrons doivent être placés dans plusieurs couches cachées, ce qui risque de compromettre l'algorithme d'apprentissage du perceptron. Ce revers déclenche l'hiver de la recherche sur les réseaux neuronaux [27].

Le premier hiver de l'intelligence artificielle

De 1974 à 1980, on connut le premier hiver de l'intelligence artificielle, période durant laquelle il n'y eu quasiment plus d'investisseurs pour financer les recherches en I.A. L'intelligence artificielle était sur le point de mourir [28].

L'invention du Neocognitron

En 1980, Kunihiko Fukushima invente le Neocognitron, la première architecture de réseau de neurones convolutifs capable de reconnaître des modèles visuels tels que des caractères écrits à la main [29].

La création du réseau Hopfield, et l'adoption pratique de la rétropropagation dans les réseaux neuronaux

En 1982, John Hopfield crée le réseau Hopfield, qui n'est autre qu'un réseau neuronal récurrent (Recurrent Neural Network ou RNN). Il sert comme un système de mémoire adressable par le contenu, et sera déterminant pour les autres modèles RNN de l'ère moderne de l'apprentissage profond [30].

D'autre part, Paul Werbos, sur la base de sa thèse de doctorat de 1974, propose l'utilisation de la rétropropagation pour la propagation des erreurs pendant la formation des réseaux neuronaux. Les résultats de sa thèse de doctorat conduiront à l'adoption pratique de la rétropropagation par la communauté des réseaux neuronaux dans le futur [31].

La création de la machine de Boltzmann

En 1985, David H. Ackley, Geoffrey Hinton et Terrence Sejnowski créent une machine de

Boltzmann qui est un réseau de neurones récurrent stochastique. Ce réseau de neurones ne possède qu'une couche d'entrée et une couche cachée, mais pas de couche de sortie [32].

La création de NetTalk et la réussite de la rétropropagation dans le réseau neuronal

En 1986, Terry Sejnowski crée NeTtalk, un réseau de neurones qui apprend à prononcer un texte anglais écrit, en lui montrant un texte en entrée et en comparant les transcriptions phonétiques [33].

D'autre part, Geoffrey Hinton, Rumelhart et Williams, dans leur article "Learning Representations by back-propagating errors" montrent la mise en œuvre réussie de la rétropropagation dans le réseau neuronal. Cela a ouvert les portes pour former facilement des réseaux de neurones profonds complexes, ce qui était le principal obstacle dans les premiers temps de la recherche dans ce domaine [34].

L'entraînement d'un réseau neuronal convolutif à reconnaitre les chiffres, et la publication de la première version du Théorème d'Approximation Universelle

En 1989, Yann LeCun utilise la rétropropagation pour entraîner un réseau de neurones convolutifs (Convolutional Neural Network ou CNN) à reconnaître des chiffres manuscrits. Il

s'agit d'une étape majeure, car elle jette les bases de la vision par ordinateur moderne utilisant l'apprentissage profond [35].

Par ailleurs, George Cybenko publie la première version du Théorème d'Approximation Universelle dans son article "Approximation by superpositions of a sigmoidal function". Il prouve que le réseau de neurones direct avec une seule couche cachée contenant un nombre fini de neurones peut approximer toute fonction continue. Il ajoute de la crédibilité au Deep Learning [36].

La publication d'un article sur un type d'architecture de réseau neuronal révolutionnaire

En 1997, Sepp Hochreiter et Jürgen Schmidhuber publient un article important sur la "mémoire à long terme" (Long Short Term Memory) (LSTM). Il s'agit d'un type d'architecture de réseau de neurones récurrents qui va révolutionner l'apprentissage profond dans les décennies à venir [37].

L'utilisation des GPU pour la formation des réseaux neuronaux profonds

En 2008, Le groupe d'Andrew NG à Stanford commence à promouvoir l'utilisation des GPU pour la formation des réseaux de neurones profonds afin de multiplier le temps de formation [38].

Cela pourrait apporter des avantages pratiques dans le domaine de l'apprentissage profond pour la formation efficace sur d'énormes volumes de données.

Le lancement de la base de données « ImageNet »

En 2009, Fei-Fei Li, professeur à Stanford, lance ImageNet qui est une base de données de 14 millions d'images libellées. Elle servira de référence aux chercheurs en apprentissage profond qui participeront chaque année aux concours ImageNet (ILSVRC) [39].

Le modèle CNN « AlexNet » remporte le concours de classification d'images d'Imagenet

AlexNet, un modèle CNN (un réseau de neurones convolutifs) implémenté par un GPU et conçu par Alex Krizhevsky, remporte le concours de classification d'images d'ImageNet en 2012 avec une précision de 84 %. C'est un énorme bond en avant par rapport à la précision de

75% que les modèles précédents étaient atteints par les modèles précédents. Cette victoire déclenche un nouveau boom de l'apprentissage profond au niveau mondial [40].

La création du GAN : le réseau neuronal adversarial génératif

Le "Generative Adversarial Neural Network" (Réseau neuronal adversarial génératif), également connu sous le nom de GAN, a été créé par Ian Goodfellow en 2014 [41].

Les GAN ouvrent de toutes nouvelles portes à l'apprentissage profond dans les domaines de la mode, de l'art et de la science, grâce à leur capacité à synthétiser des données réelles.

Un champion humain du jeu complexe du Go perd contre Le modèle d'apprentissage par renforcement profond

Le modèle d'apprentissage par renforcement profond de Deepmind bat un champion humain dans le jeu complexe du Go en 2016. Le jeu est beaucoup plus complexe que les échecs, cet exploit capte donc l'imagination de tous et porte la promesse de l'apprentissage profond à un tout autre niveau [42].

L'obtention du prix Turing par Yoshua Bengio, Geoffrey Hinton et Yann LeCun

Yoshua Bengio, Geoffrey Hinton et Yann LeCun remportent le prix Turing 2018 pour leur immense contribution aux avancements dans le domaine de l'apprentissage profond et de l'intelligence artificielle. C'est un moment marquant pour ceux qui ont travaillé sans relâche sur les réseaux neuronaux alors que toute la communauté de l'apprentissage automatique s'en était éloignée dans les années 1970 [43].

2.2.2 Définition

Le Deep Learning (Apprentissage Profond) est un domaine de recherche sur le Machine Learning (apprentissage Automatique) Basé sur un type spécifique de mécanisme d'apprentissage.

Il se caractérise par l'effort pour créer des modèles d'apprentissage à plusieurs niveaux, dans lequel les niveaux les plus profonds prennent en considération les résultats des niveaux précédents, les Transformant et en faisant toujours plus abstraction. Cette présentation des niveaux d'apprentissage est inspirée par la façon dont le cerveau traite l'information et apprend en répondant aux stimuli externes [22].

Le Deep Learning est basé sur l'idée des réseaux de neurones artificiels, ces derniers sont des modèles bien plus complexes que tous les autres modèles de Machine Learning car ils représentent des fonctions mathématiques avec des millions de coefficients (les paramètres).

De nos jours, Deep Learning est au centre de l'attention puisque sa réalisation est beaucoup plus importante que tout autre algorithme de Machine Learning dans des tâches aussi complexes, et avec une telle puissance, il est possible d'entraîner la machine sur des tâches plus avancées.

2.2.3 Les réseaux des neurones

Les réseaux de neurones artificiels sont un nouveau type de technologie informatique : Ce sont de multiples processeurs parallèles, hautement intégrés par un réseau de connexions qui créent un modèle distribué computationnel.

Leur architecture était donc très innovante dans les années 50, lorsqu'elles sont nées en parfaite analogie avec la structure du cerveau : de nombreux neurones biologiques étroitement connectés par des synapses à travers lesquels les calculs se propagent en parallèle dans le cortex cérébral. Cela permet d'accéder de nouvelles capacités : trouver la solution à des problèmes complexes en temps réel, résistance aux fautes et erreurs, auto-apprentissage, etc.

Pratiquement, tous les algorithmes de DL sont des réseaux neuronaux [44]. Ce sont des modèles de traitement de l'information qui simulent le fonctionnement du système nerveux biologique. C'est similaire à la façon dont le cerveau traite l'information au niveau du fonctionnement. Tous les réseaux neuronaux sont constitués de neurones inter connectés qui sont organisés en couches.

Le neurone

Ce qui forme les réseaux de neurones, ce sont les neurones artificiels inspirés du vrai neurone qui existe dans notre cerveau. La figure suivante montre une représentation d'un neurone artificiel :

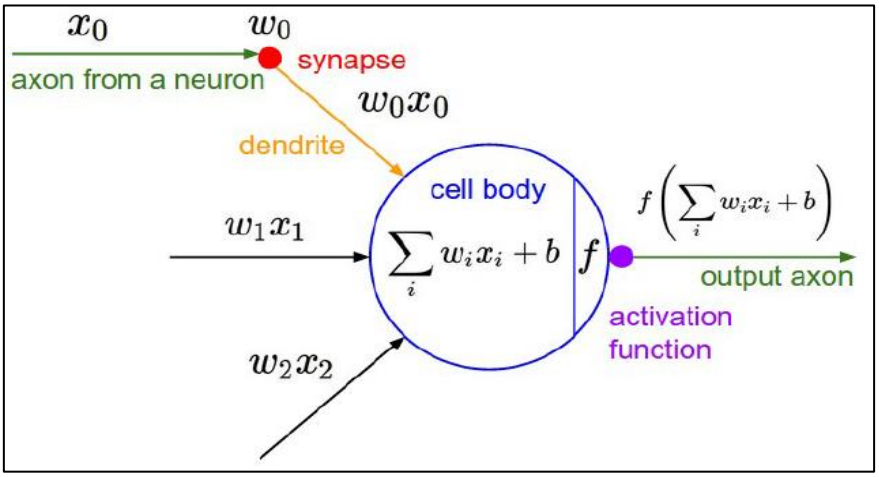


Figure 2.2. Un neurone artificiel.

Principe de base

L'unité de base du calcul dans un réseau de neurones artificiel est le neurone qui a été défini dans [45] en 1959. Un neurone artificiel reçoit des entrées de certains autres neurones ou d'une source externe ayant des valeurs numériques *x*1, *x*2, ...*xn* auxquels il est connecté par des synapses et calcule une sortie *y*. Chaque entrée *xi* a un poids associé *wi*, qui est attribué en fonction de son importance relative par rapport aux autres entrées. La valeur d'entrée *x* du neurone correspond à la somme pondérée de ses entrées en ajoutant une autre entrée ayant un poids *b* appelé biais. Ce biais peut être vu comme une entrée *x*0 supplémentaire dont la valeur est toujours de 1 ensuite, le neurone applique une fonction *f* sur cette somme, comme illustré à la figure 3 :

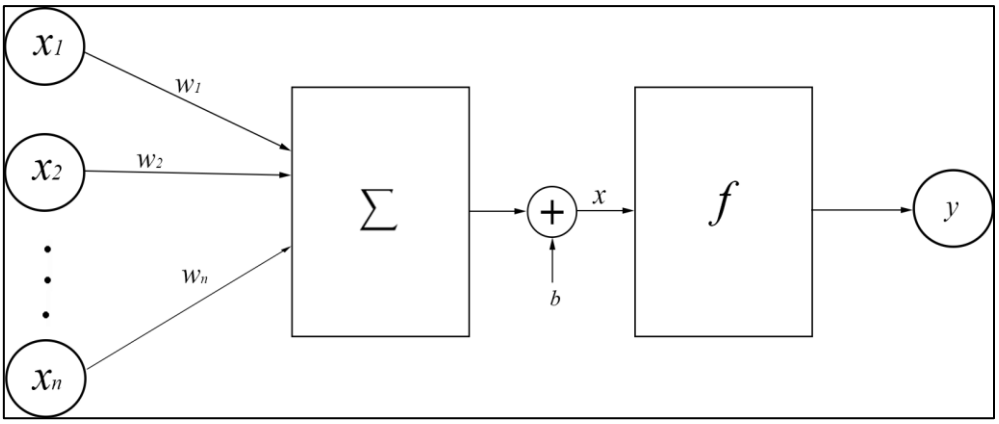


Figure 2.3. Modélisation d'un neurone artificiel.

La sortie du neurone, appelée activation de sortie, est calculée selon la formule suivante :

$$y = f \sum_{i=1}^{n} w_i x_i + b$$

Où *f* est la fonction d'activation, qui sera détaillée ultérieurement.

Les fonctions d'activation

Après que le neurone a effectué le produit entre ses entrées et ses poids, il applique également une non-linéarité sur ce résultat. Cette fonction non linéaire s'appelle la fonction d'activation.

La fonction d'activation est une partie importante du réseau neuronal. Ce que cette fonction a décidé est si le neurone est activé ou non. Il calcule la somme pondérée des entrées et ajoute le biais. Il s'agit d'une transformation non linéaire de la valeur d'entrée.

Après la transformation, cette sortie est envoyée à la couche suivante. La non-linéarité est si importante dans les réseaux de neurones, sans la fonction d'activation, un réseau de neurones est devenu simplement un modèle linéaire. Il existe de nombreux types de ces fonctions, parmi lesquelles nous trouvons [46] [47] :

- **La fonction sigmoïde :** Cette fonction est l'une des plus couramment utilisées. Il est borné entre 0 et 1, et il peut être interprété stochastiquement comme la probabilité

que le neurone s'active, et il est généralement appelé la fonction logistique ou le sigmoïde logistique.

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

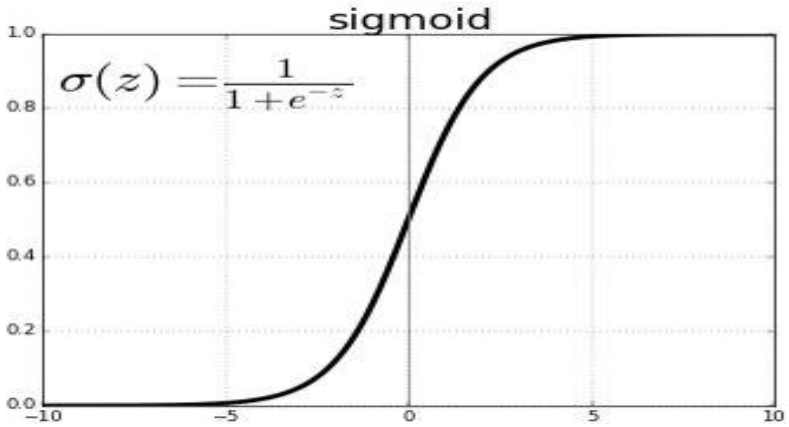


Figure 2.4. Représentation graphique de la fonction Sigmoïde *[65].*

- **La fonction ReLU :** La fonction ReLU est probablement la plus proche de sa correspondante biologique [47]. Cette fonction est récemment devenue le choix de nombreuses tâches (notamment en computer vision) [46]. Comme dans la formule ci-dessus, cette fonction renvoie 0 si l'entrée z est inférieure à 0 et retourne z lui-même s'il est plus grand que 0.

$$R(z) = \max(0, z)$$

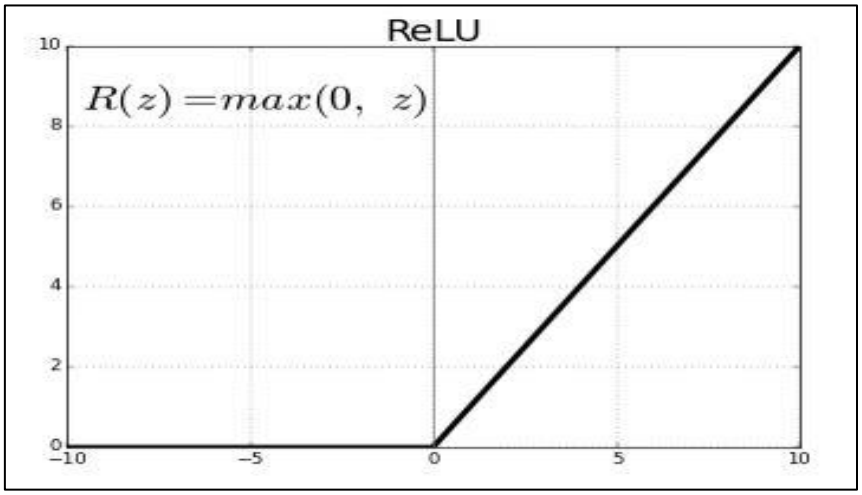


Figure 2.5. Représentation graphique de la fonction ReLu *[65].*

- **La fonction Softmax :** La fonction Softmax est une forme plus élucidée de la fonction sigmoïde. Il est utilisé pour la classification multi-classes. La fonction calcule les probabilités d'une classe particulière dans une fonction. Ainsi, la plage de valeurs de sortie de la fonction est comprise entre 0 et 1. La principale différence entre la fonction sigmoïde et la fonction Softmax est que la fonction sigmoïde peut être utilisée pour la classification binaire tandis que la fonction Softmax peut également être utilisée pour la classification multi-classes.

$$Softmax(x_i) = \frac{e^{x_i}}{\sum_{j=0}^{k} e^{x_k}}$$

En comparaison avec sigmoïde, la fonction ReLU accélère le temps d'apprentissage ce qui entraîne une efficacité calculatoire [48].

2.2.4 Topologie des réseaux de neurones

Nous distinguons deux types de topologies de réseaux de neurones :

Propagation vers l'avant de l'information (Feed-forward)

Le réseau de neurones à propagation avant -ou feed-forward neural network- était le premier et le plus simple réseau de neurones artificiel mis au point. Il contient plusieurs neurones disposés en couches. Les neurones des couches adjacentes ont des connexions entre eux. Toutes ces connexions ont des poids qui leur sont associés.

Un exemple de réseau de neurones à propagation avant est présenté sur la figure :

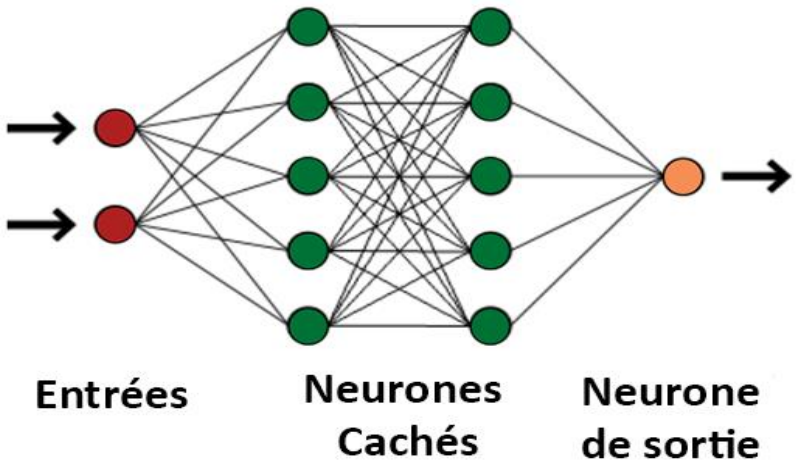


Figure 2.6. Exemple d'un réseau de neurones artificiel.

Dans un réseau à propagation avant, les informations ne se déplacent que dans un seul sens, en partant des neurones d'entrée, passant par les neurones cachées (s'ils existent) et en arrivant aux neurones de sortie.

Pour introduire les réseaux de neurones, nous allons commencer par présenter deux réseaux à propagation avant :

- **Réseaux à une couche (Feed-forward) :** Appelé aussi perceptron mono-couche a été introduit en 1958 par Franck Rosenblatt, Il s'agit d'un réseau de neurones à propagation avant le plus simple qui contient une couche cachée

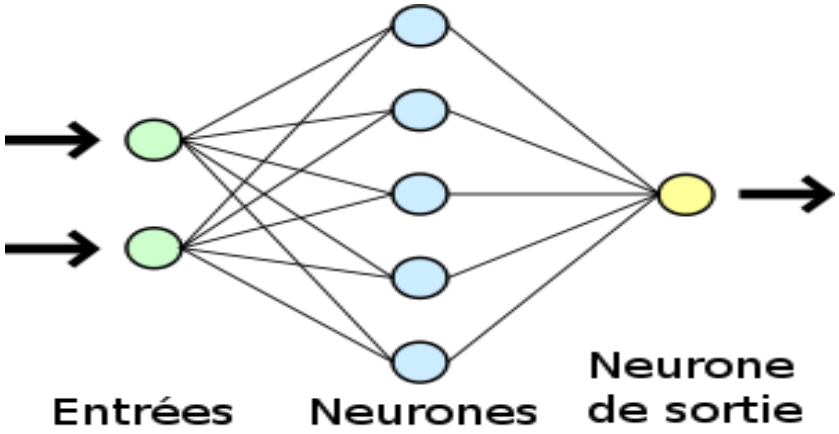


Figure 2.7. Perceptron mono-couche.

- **Réseaux du perceptron multicouche (multilayer perceptron MLP) :** Les perceptrons multicouches (MLP) également appelés les réseaux profonds (Feed-forward) effectuent la propagation vers l'avant de l'information, sont des réseaux de neurones acycliques structuré en couches.

  Cette classe de réseaux feed-forward diffère de la précédente car elle a une ou plusieurs couches de neurones cachés (couches cachées) entre les couches d'entrée et de sortie. Chaque couche a des connexions entrantes de la couche précédente et sortantes dans la suivante. Ce type d'architecture fournit au réseau une perspective globale car il augmente les interactions entre les neurones [49] [50] [51].

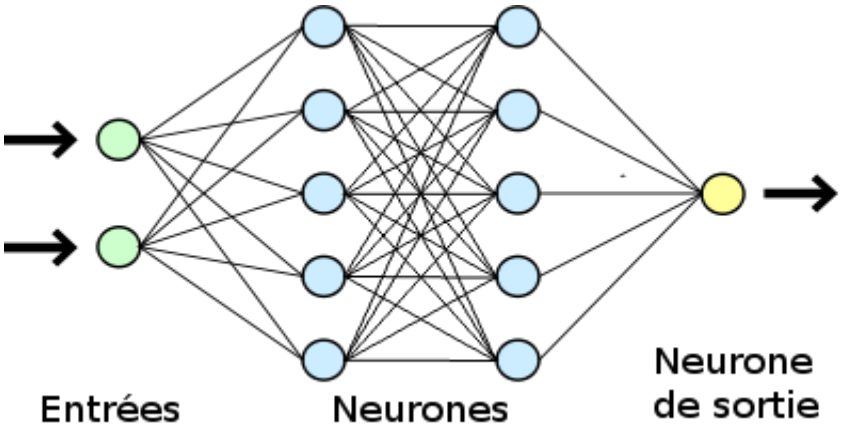


Figure 2.8. Perceptron Multi-couches.

Réseaux récurrents (Feed-back)

Un réseau de neurones récurrent (RNN, récurrent neural network) est un type de réseau de neurones artificiels principalement utilisé pour résoudre les problèmes de la reconnaissance vocale et le traitement automatique du langage naturel.

Un réseau récurrent diffère des précédents par le fait qu'il est cyclique, La présence de cycles à un impact profond sur les capacités d'apprentissage du réseau et sur ses performances, notamment rendre le système dynamique [49] [50]**.**

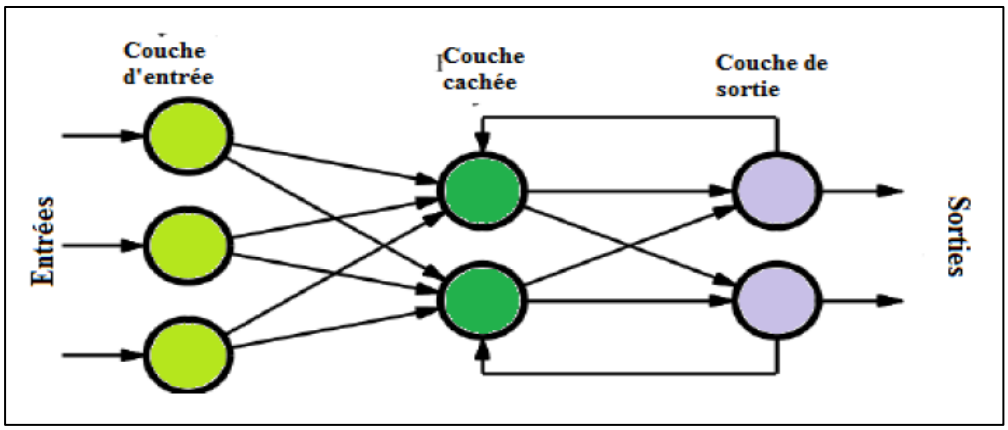


Figure 2.9. Réseaux récurrents.

Les couches de réseaux neuronaux récurrentes sont des entités primitives qui permettent aux réseaux neuronaux d'apprendre à partir de séquences d'entrées [52]. Si la séquence que nous traitons est une phrase de 3 termes par exemple, le réseau va être déroulé en un réseau neuronal à 3 couches, une couche pour chaque mot, la figure suivante [52] représente cette idée :

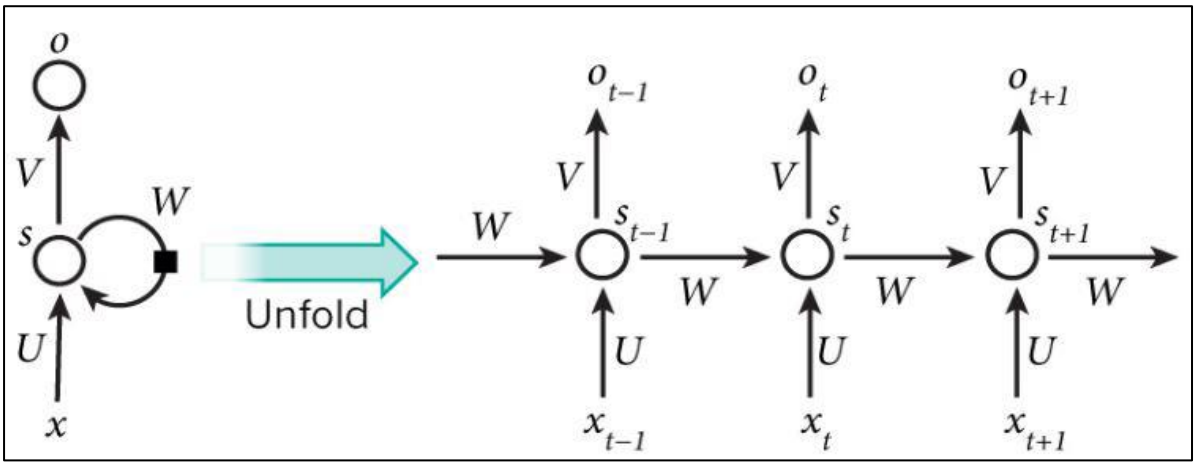


Figure 2.10. Un exemple d'un réseau récurrent qui se déroule *[52].*

Les formules qui dirigent les calculs dans RNN sont les suivantes :

- $\boldsymbol{x_t}$ est l'entrée au moment **t**.
- $\boldsymbol{U}$, $\boldsymbol{V}$, $\boldsymbol{W}$ sont les paramètres que le réseau va apprendre des donnés de l'apprentissage.
- $\boldsymbol{S_t}$ est l'état caché à l'instant **t**. C'est la « mémoire » du réseau, $\boldsymbol{S_t}$est calculé en fonction de l'état caché précédent et de l'entrée à l'étape courante avec :

$$\boldsymbol{S_t = f\ (U_{x_t} + W_{s_{t-1}})}$$

- Ou **f** est la fonction d'activation (une fonction nonlineair telle que : ReLU).
- $\boldsymbol{O_t}$ Est la sortie au moment **t**. Par exemple, si nous souhaitons prédire le mot suivant dans une phrase, ce serait un vecteur de probabilités à travers notre vocabulaire [53], nous utiliserons alors la fonction d'activation Softmax avec :

$$\boldsymbol{O_t = \text{Softmax}\ (V_{S_t})}$$

2.3 L’apprentissage en Deep Learning

L'apprentissage est le processus par lequel les paramètres libres d'un réseau de neurones s'adaptent, en utilisant un processus de stimulation à leur environnement. Le type d'apprentissage est déterminé par la façon dont ces adaptations ont lieu.

Un algorithme d'apprentissage est un ensemble de règles bien définies qui résolvent un certain souci d'apprentissage [54] [50]**.** Trois grandes classes d’apprentissage existent :

2.3.1 Apprentissage supervisé

L'apprentissage supervisé est la tâche d'apprentissage automatique la plus simple et la plus connue. Il est basé sur un certain nombre d'exemples pré classifiés, dans lesquels est connu à priori la catégorie à laquelle appartient chacune des entrées utilisées comme exemples.

L’algorithme le plus connu pour l’apprentissage supervisé est le gradient, qui est par définition, pour un neurone, l’erreur relative à ce neurone. Il peut en quelque sorte être vu comme la contribution du neurone à l’erreur globale.

A chaque back propagation, on calcule le gradient de chaque neurone, en commençant par ceux de la couche de sortie (les gradients des couches inférieurs se calculent à partir des gradients des couches supérieurs).

2.3.2 Apprentissage non supervisé

C’est de l’apprentissage par exploration où l’algorithme d’apprentissage ajuste les poids des liens entre neurones pour maximiser la qualité de classification des entrées.

2.3.3 Apprentissage par renforcement

Les sorties idéales ne sont pas connues directement. Aussi, les poids sont ajustés de façons aléatoires et la modification est conservé si l’impact est positif ou non.

2.4 Avantages et Inconvénients des réseaux de neurones

2.4.1 Les avantages

Les réseaux de neurones représentent des avantages, tels que :

- Les réseaux de neurones sont souples et génériques. Ils peuvent résoudre différents types de problèmes dont le résultat peut être : une classification, analyse de données, etc.

- Ils traitent des problèmes non structurés sur lesquels aucune information n'est disponible à l'avance.

- Les réseaux neuronaux se comportent bien parce que même dans des domaines très complexes.

2.4.2 Les inconvénients

Les réseaux de neurones ont aussi des inconvénients, tels que :

- La lenteur d'apprentissage [55].

- La difficulté de choisir des valeurs initiales des poids de connexion ainsi que l'adaptation du pas d'apprentissage [56] .

- En cas d'erreur dans les résultats de sorties, l'utilisateur n'a aucune information sur le fonctionnement interne [55].

2.5 Conclusion

Dans ce chapitre, Nous avons donné une introduction à l'intelligence artificielle ainsi qu'au Machine Learning. On a cité les différentes connaissances de base sur le Deep Learning en donnant un aperçu des différentes applications du Deep Learning, des réseaux neuronaux et nous avons expliqué le principe de fonctionnement de ces réseaux.

On a fini ce chapitre par un bref historique de l'évolution du Deep Learning et des réseaux neuronaux.

Chapitre 3 Exploitation du Deep Learning

3.1 Introduction

Le Deep Learning ou « apprentissage profond » est une famille d'algorithmes de machine Learning (apprentissage automatique) pour entraîner des réseaux de neurones composés de plusieurs couches internes et potentiellement un grand nombre.

Le développement du Deep Learning fut motivé en partie par l'échec des algorithmes traditionnels à résoudre quelques problèmes majeurs de l'intelligence artificiel telle que l'analyse audio [57, 58] (reconnaissance vocale, synthèse vocale…etc.), le traitement du langage naturel [59], le traitement et classification des images, la reconnaissance d'objet, la vision assistée par ordinateur [60, 61, 62] …etc.

Les progrès de l'apprentissage profond ont été possibles notamment grâce à l'augmentation de la puissance des ordinateurs et au développement de grandes bases de données (big data) [63].

Dans ce chapitre, nous présentons les problèmes susceptibles d'être résolus par le Deep Learning, ainsi que les algorithmes qui permettent de les résoudre.

3.2 Les problèmes typiques

Le Deep Learning peut résoudre de nombreux types de tâches, telles que la classification, la régression et la traduction. Dans la classification, l'algorithme spécifie à quelle catégorie appartiennent certaines entrées (la reconnaissance d'objet est un exemple de classification, où l'entrée est une image et la sortie est un code numérique qui reconnaît l'objet dans l'image). Régression, qui prédit la valeur d'entrée donnée (prévision du montant de la réclamation de l'assuré). La traduction est la tâche de convertir une série de symboles écrits dans une langue dans une autre langue. Les réseaux de neurones profonds ne sont pas un simple empilement de couches de neurones. En effet, pour résoudre des problèmes de plus en plus complexes.

Il est donc nécessaire de présenter des approches différentes suivant les problèmes que l'on cherche à résoudre. Il existe différents types de réseaux de neurones profonds qui visent à résoudre différents problèmes.

3.2.1 Types de tâches

Il existe de nombreux types de tâches résolus par Deep Learning, parmi eux :

- **Traitement automatique de langue naturelle (NLP) :** Le but étant d'extraire le sens des mots et voir des phrases pour faire de l'analyse de sentiments. L'algorithme va comprendre ce qui est dit dans un avis Google par exemple, ou va communiquer avec

des personnes via des Chatbots (un programme informatique capable de dialoguer avec un ou plusieurs humains par échange vocal ou textuel).

- **La reconnaissance faciale :** Un algorithme de Deep Learning va apprendre à détecter les yeux, le nez, la bouche sur une photo. Il va s'agir en premier lieu de donner un certain nombre d'images à l'algorithme, puis à force d'entraînement, l'algorithme va être en mesure de détecter un visage sur une image.

- **Colorisation automatique :** La colorisation de l'image pose le problème de l'ajout de couleurs aux photographies noir et blanc. Le Deep Learning peut être utilisé pour utiliser les objets et leur contexte dans la photographie pour colorer l'image, un peu comme un opérateur humain pourrait aborder le problème. Cette capacité tire parti des réseaux de neurones de convolution de grande qualité et cooptés pour le problème de la colorisation de l'image.

- **Recherche vocale et assistants à commande vocale :** Avec les grands géants de la technologie ont déjà fait d'énormes investissements dans ce domaine, des assistants à commande vocale peuvent être trouvés sur presque tous les smartphones. Les plus populaires sont Le Siri d'Apple, Microsoft Cortana, et Alexa d'Amazon.

- **Analyse des sentiments du texte :** Beaucoup d'applications ont des systèmes de révision basés sur des commentaires intégrés à leurs applications. La recherche sur le traitement du langage naturel et les réseaux de neurones récurrents ont parcouru un long trajet et il est maintenant tout à fait réalisable de déployer ces modèles sur le texte de votre application pour extraire des informations de niveau supérieur. Cela peut être très utile pour évaluer la polarité émotionnelle pour extraire des sujets significatifs à l'aide de modèles de reconnaissance d'entités nommées.

- **Analyses d'image :** Les cas d'usages les plus fréquents en analyse d'image sont la classification d'images (trouver la catégorie de l'objet principal de l'image), la détection d'objets (trouver où sont situés tous les objets de l'image, ainsi que leur catégorie) et la segmentation d'image (déterminer pour chaque pixel de l'image à quelle catégorie il appartient). Ces trois cas d'usage présentent une complexité croissante. Cependant, avec suffisamment de données, les réseaux de neurones

actuels permettent d'obtenir des performances similaires à celles des humains, voire supérieures pour ces problématiques.

3.3 Les algorithmes de Deep Learning

Avant d'en savoir plus sur les algorithmes, nous allons d'abord définir ce qu'est un algorithme.

- Un algorithme est un procédé qui permet de résoudre un problème sans avoir besoin d'inventer une solution à chaque fois [64].
- Un algorithme d'apprentissage est un ensemble de règles bien définies qui résolvent certains problèmes d'apprentissage sont difficiles à résoudre [50] [54].

Maintenant, nous allons entrer dans le domaine de Deep Learning. Cette série d'algorithmes permet de faire des progrès significatifs dans le domaine de la classification des images et du traitement du langage par exemple.

Le modèle d'apprentissage profond est construit sur le même modèle que le perceptron multicouche Décrit avant. Cependant, il convient de noter qu'il existe beaucoup plus de couches intermédiaires différentes. Chaque couche intermédiaire sera subdivisée en sous-sections, traitant de sous-problèmes plus simples et fournissant les résultats à la couche suivante, et ainsi de suite [65].

Il existe différents algorithmes de Deep Learning. Nous pouvons ainsi citer :

3.3.1 Les Auto encodeurs

Un Auto-encodeur (AE) est un réseau de neurone artificiel dans lequel l'entrée et la sortie sont identiques, il utilisé à des fins telles que la découverte de produits pharmaceutiques, le traitement d'images. Geoffrey Hinton a conçu les auto-encodeurs dans les années 1980 pour résoudre les problèmes d'apprentissage non supervisé (Unsupervised Learning). En effet, l'encodeur est constitué par un ensemble de couches de neurones, qui traitent les données afin d'obtenir une nouvelle représentation des données tandis que les couches de neurones du décodeur analysent les données encodées pour essayer de reconstruire les données d'origines. L'image de la figure ci-dessous montre l'architecture d'un auto-encodeur :

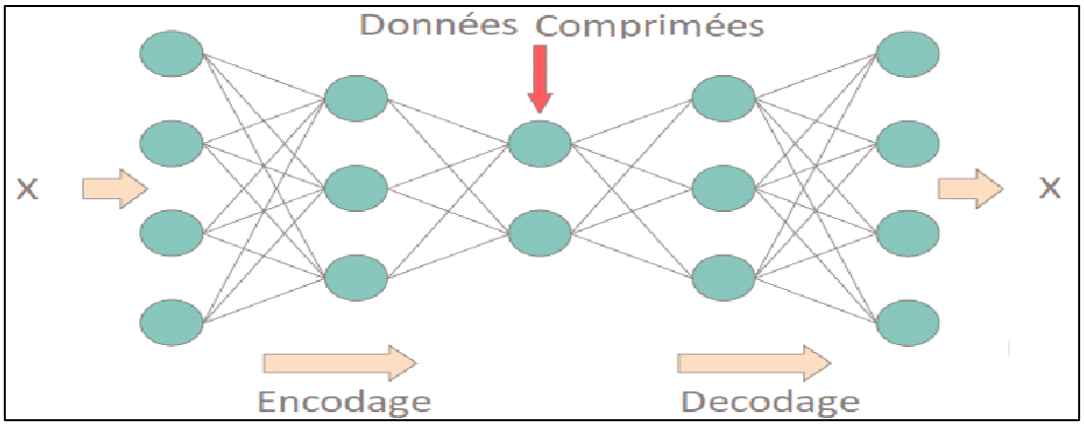


Figure 3.1. L'architecture d'un auto encodeur.

La couche la plus importante est la couche encodée, qui permet d'avoir une nouvelle représentation des données. Le nombre de neurones dans les couches cachées doit être inférieur à celui des couches d'entrées pour permettre aux couches cachées à apprendre plus de modèles de données et à ignorer les « bruits ». Si le nombre de neurones dans les couches cachées est supérieur à celui des couches d'entrée, le réseau neuronal aura trop de capacité pour apprendre des données. Dans un cas extrême, il pourrait simplement copier l'entrée dans les valeurs de sortie, y compris les bruits, sans extraire aucune information essentielle.

3.3.2 Réseaux de neurones convolutifs (CNN)

Les réseaux convolutifs ont été introduits pour la première fois par Fukushima [66], ils sont différents des autres formes de réseaux neuronaux artificiels, Le CNN est un type de réseau de neurones artificiels acycliques il a largement utilisé pour le traitement et classification d'images et détection d'objet et les systèmes de recommandation...etc.

Les réseaux de neurones convolutifs sont à ce jour les modèles les plus performants pour classer des images**.** Désignés par l'acronyme CNN, de l'anglais « Convolutional Neural Network », ils comportent deux parties bien distinctes. En entrée, une image est fournie sous la forme d'une matrice de pixels. Elle a deux dimensions pour une image aux niveaux de gris.

La couleur est représentée par une troisième dimension, de profondeur 3 pour représenter les couleurs fondamentales [Rouge, Vert, Bleu].

Première partie d'un CNN : Est la partie convolutive à proprement parler. Elle fonctionne comme un extracteur de caractéristiques des images. Une image est passée à travers d'une succession de filtres, ou noyaux de convolution, créant de nouvelles images appelées cartes

de convolutions. Certains filtres intermédiaires réduisent la résolution de l'image par une opération de maximum local. En fin, les cartes de convolutions sont mises à plat et concaténées en un vecteur de caractéristiques, appelé code CNN.

Deuxième partie est la partie classification : Ce code CNN obtenu en sortie de la partie convolutive est ensuite branché en entrée d'une deuxième partie, constituée de couches entièrement connectées appelées perceptron multicouche. Le rôle de cette partie est de combiner les caractéristiques du code CNN pour classer l'image.

La sortie est une dernière couche comportant un neurone par catégorie. Les valeurs numériques obtenues sont généralement normalisées entre 0 et 1, de somme 1, pour produire une distribution de probabilité sur les catégories [67].

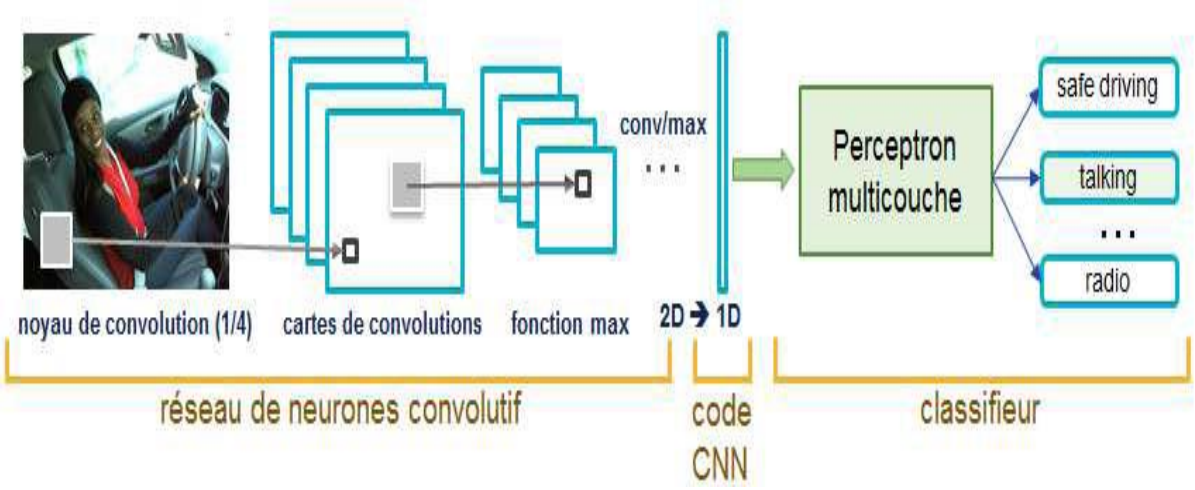


Figure 3.2. Les réseaux de neurones convolutifs.

Les réseaux de neurones convolutionels sont basés sur le perceptron multicouche (MLP), et inspirés du comportement du cortex visuel des vertébrés. Bien qu'efficaces pour le traitement d'images, les MLP ont beaucoup de mal à gérer des images de grande taille, ce qui est dû à la croissance exponentielle du nombre de connexions avec la taille de l'image.

Par exemple, si on prend une image de taille 32x32x3 (32 de large, 32 de haut, 3 canaux de couleur), un seul neurone entièrement connecté dans la première couche cachée du MLP aurait 3072 entrées (32*32*3). Une image 200x200 conduirait ainsi à traiter 120 000 entrées par neurone ce qui, multiplié par le nombre de neurones, devient énorme.

Les couches de réseaux de neurones convolutifs

Il y a plusieurs couches différentes pour construire ce type de réseau sont :

- La couche de convolution (CONV) qui traite les données d'un champ récepteur.

- La couche de mise en commun (Pooling) qui permet de compresser l'information en réduisant la taille de l'image intermédiaire (souvent par sous-échantillonnage).
- La couche de correction (ReLU) souvent appelée par abus 'ReLU' en référence à la fonction d'activation (Unité de rectification linéaire).
- La couche entièrement connectée (FC) qui est une couche de type perceptron.
- La couche de perte (LOSS).

- **Couche de convolution (CONV)**

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

Tout d'abord, une partie de l'image est connectée à la couche CONV pour effectuer une opération de convolution et calculer le produit scalaire entre le champ récepteur (c'est une région locale de l'image d'entrée ayant la même taille que celle du filtre). Le résultat de l'opération est un entier unique du volume de sortie. Ensuite, nous faisons glisser le filtre sur le champ récepteur suivant de la même image d'entrée par une foulée et refaisons la même opération. Cette opération est répétée par le même processus encore et encore jusqu'à ce que toute l'image soit parcourue.

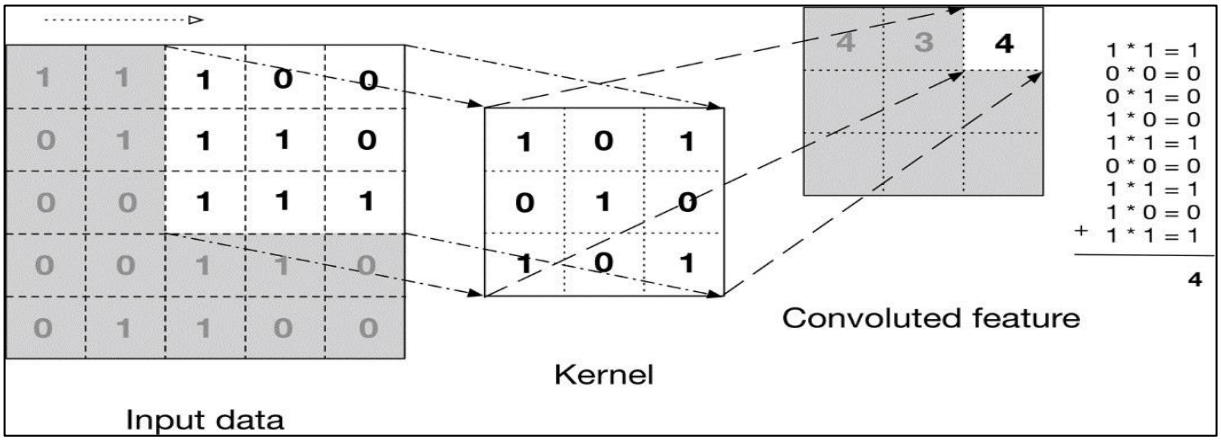


Figure 3.3. L'opération de convolution [68]**.**

La couche CONV contient également l'activation ReLU voir figure 3.4 pour que toutes les valeurs négatives soient mises à zéro.

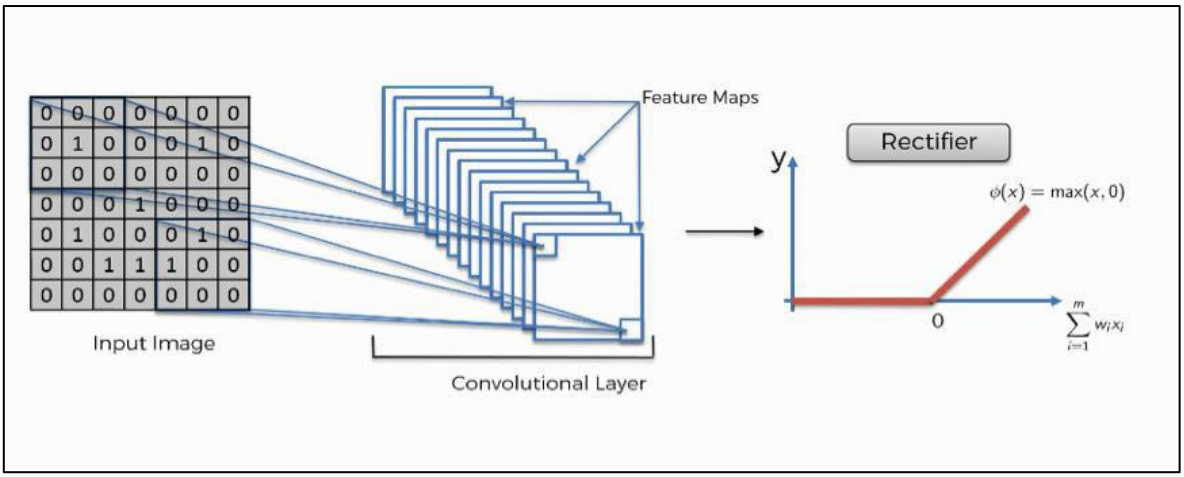


Figure 3.4. Principe de la fonction ReLU [68].

- **Couche de mise en commun (Pooling)**

La couche de Pooling est utilisée pour réduire le volume spatial de l'image d'entrée après la convolution. Elle est utilisée entre deux couches de convolution. Si nous appliquons FC (Fully Connected) après la couche CONV sans appliquer le pooling ou le pooling maximum, le calcul sera coûteux. Ainsi, la mise en commun maximale est le seul moyen de réduire le volume spatial de l'image d'entrée en codant l'information, la figure 3.5 montre le max pooling sur une fenêtre 2 × 2 [69] [70] [71].

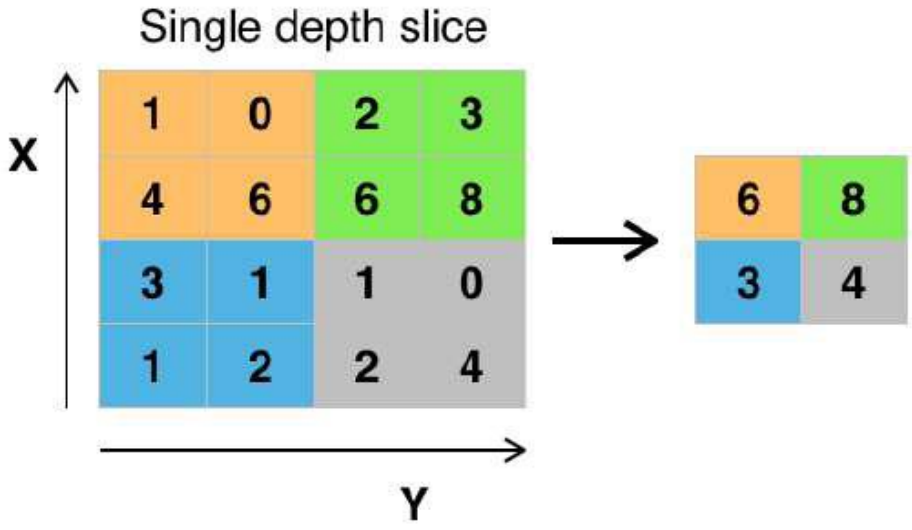


Figure 3.5. Exemple de principe du Pooling [72].

- **Couches de correction (ReLU)**

Il est possible d'améliorer l'efficacité du traitement en intercalant entre les couches de traitement une couche qui va opérer une fonction mathématique (fonction d'activation) sur les signaux de sortie. La fonction ReLU (abréviation de Unités Rectifié linéaires) :

$$F(x)=\max(0, x)$$

Cette fonction force les neurones à retourner des valeurs positives [69] [70].

- **Couche entièrement connectée (FC)**

Après plusieurs couches de convolution et de max-pooling, le raisonnement de haut niveau dans le réseau neuronal se fait via des couches entièrement connectées. Les neurones dans une couche entièrement connectée ont des connexions vers toutes les sorties de la couche précédente. Leurs fonctions d'activations peuvent donc être calculées avec une multiplication matricielle suivie d'un décalage de polarisation.

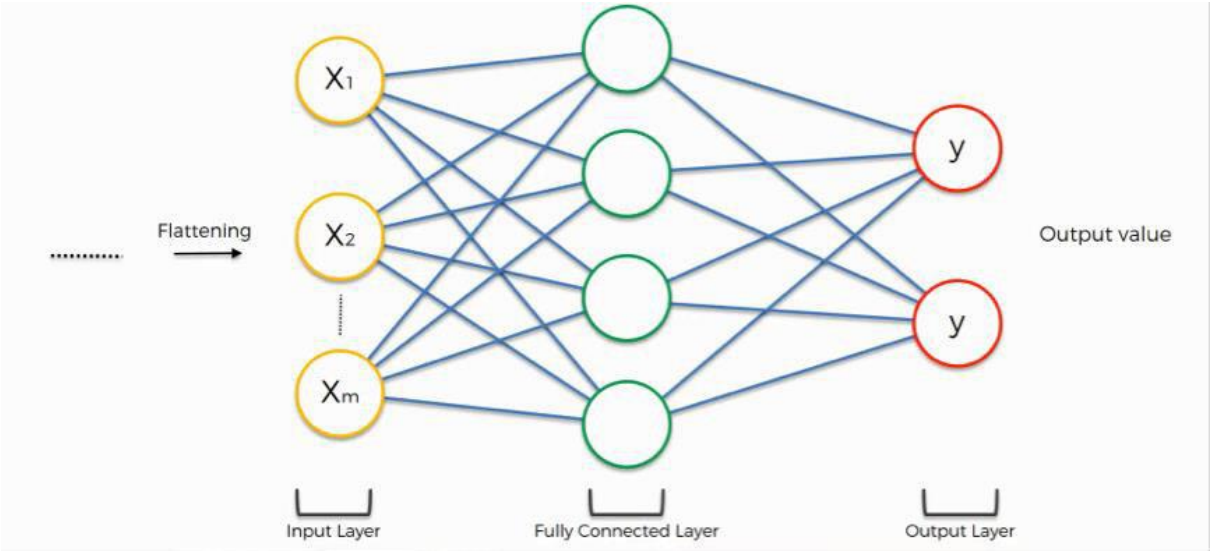


Figure 3.6. Principe de la couche entièrement connectée (FC).

- **Couche de perte (LOSS)**

La couche de perte spécifie comment l'entrainement du réseau pénalise l'écart entre le signal prévu et réel.

Elle est normalement la dernière couche dans le réseau. Diverses fonctions de perte adaptées à différentes tâches peuvent y être utilisées. La fonction « Softmax » permet de calculer la distribution de probabilités sur les classes de sortie.

Les réseaux neuronaux mentionnés, ne sont pas tous les réseaux qui existent, une excellente ressource qui résume peut-être toute les architectures sont en [73]. La figure suivante montre une représentation de ce travail.

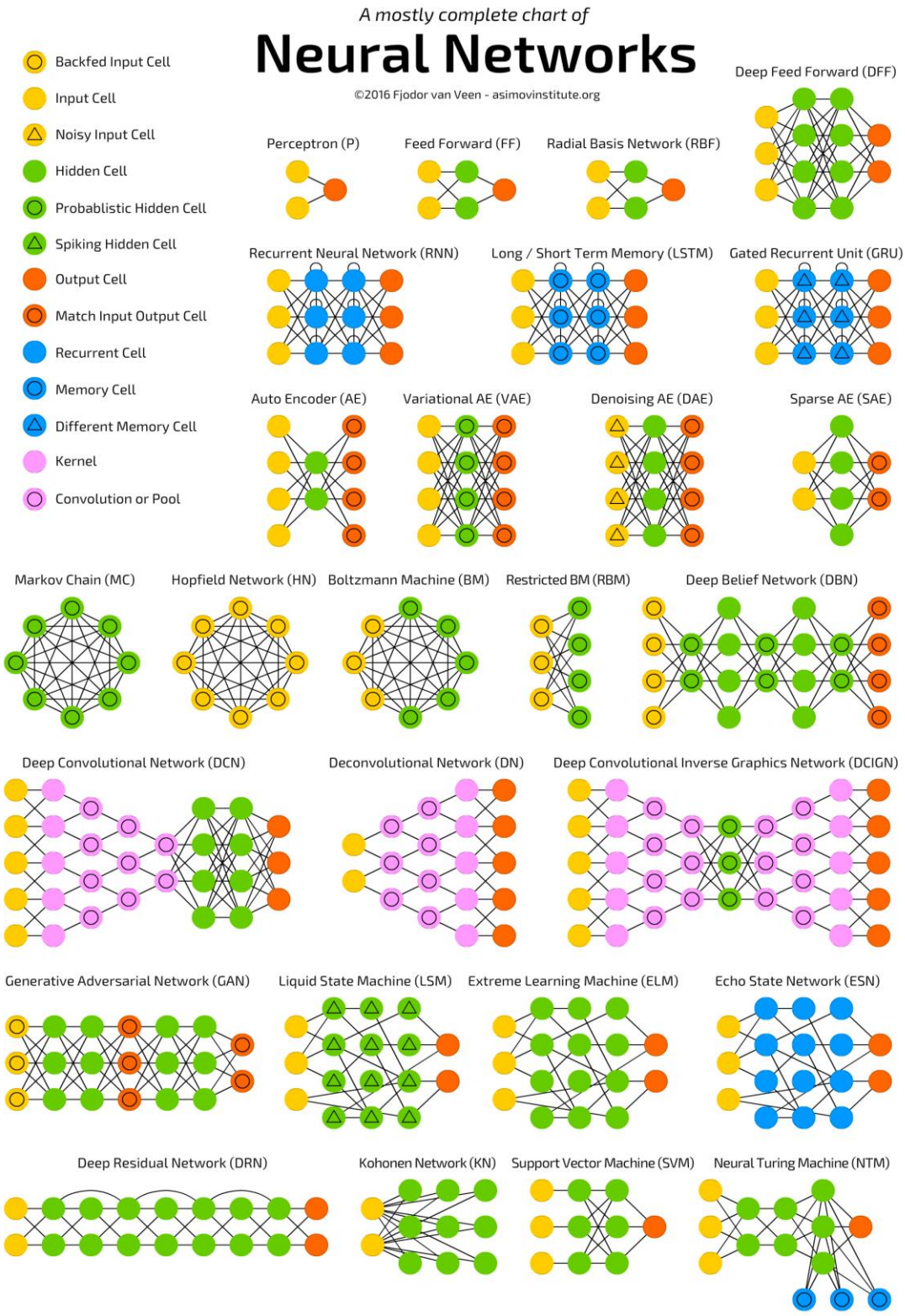


Figure 3.7. Un résumé des types d'architectures de réseaux de neurones.

3.4 La classification d'image en Télécommunication

3.4.1 Introduction sur le traitement d'image

Le traitement d'image est une partie du traitement de signal en télécommunication, consacrée aux images et aux vidéos.

Le traitement d'image est l'ensemble des opérations réalisées sur l'image, qui permettent de la modifier afin d'améliorer sa qualité, de faciliter son interprétation et d'en extraire des informations.

Les opérations de traitement peuvent être réparties en deux niveaux :

- **Le traitement de bas niveau** qui est construit autour des méthodes d'analyse d'image dans le but d'extraire des traits des images et de les analyser sans les interpréter (contours et texture, par exemple). Il s'agit de données de nature numérique.

- **Le traitement de haut niveau** intègre la totalité des méthodes permettant de traduire les caractéristiques issues du bas niveau (prise de décision, classification, IA). Il s'agit d'entités de nature symbolique liées à une représentation de la réalité extraite de l'image [74].

En résumé, la classification d'images est une branche de traitement d'image de haut niveau, et le traitement d'images est une branche de traitement du signal qui est un domaine étudié en télécommunication.

3.4.2 La vision artificielle et le traitement d'image

Le domaine de la vision artificielle (vision par ordinateur) et du traitement d'image est cette discipline qui consiste à convertir une image en données objets ou, plus explicitement, à identifier les objets contenus dans l'image par l'extraction et l'analyse de caractéristiques abstraites (Features) à partir des pixels, suivant un processus de reconnaissance de forme similaire à celui opéré par l'humain. Ce domaine en terme générique englobe plusieurs aspects de l'imagerie tel que : la compression d'images, segmentation, fusion d'images et classification d'images (qui est le centre de notre intérêt) [75].

3.5 Conclusion

Dans ce chapitre, on a commencé par déterminer les problèmes typiques du Deep Learning, ensuite, les algorithmes de l'apprentissage profond afin de résoudre quelques problèmes, toute en donnant en détail la méthode choisie dans notre travail de recherche qui est le CNN. Ces réseaux sont capables d'extraire des caractéristiques d'images présentées en entrée et de classifier ces caractéristiques.

A la fin, On a vu la corrélation entre la classification d'images et la télécommunication, et comment l'intelligence artificielle joue un rôle dans tout ça.

Le prochain chapitre traite les détails de la conception, ainsi que la méthode et les outils utilisés pour la réalisation de notre application.

Chapitre 4 Implémentation et résultat

4.1 Introduction

Après avoir présenté, les notions générales et les problèmes résolus par le Deep Learning. Dans ce chapitre, nous allons prendre une problématique particulière qui est la classification d'images, nous présentons les outils utilisés dans notre application, par la suite on va montrer comment créer notre modèle avec des architectures différentes, on va appliquer ces modèles sur la base d'images CIFAR 10 et MNIST. Pour cela, on va travailler avec le langage de programmation python et des bibliothèques comme Tensorflow et Keras pour l'apprentissage et la classification, et pour améliorer les performances des modèles, on va utiliser quelques techniques simples et efficaces comme Dropout.

4.2 Présentation des outils

4.2.1 Matériel

Les informations générales de matériel utilisé dans notre implémentation sont :

- Un PC portable HP
- Windows 10 Entreprise
- Processeur Intel ® Celeron® CPU N3060 @ 1.60GHz 1.60 GHz
- RAM 4Go
- Système d'exploitation 64bits, processeur x64

Les expériences ont été réalisées sur une machine virtuelle dans l'environnement Google Colab.

Colaboratory, souvent raccourci en Colab, est un produit de Google Research. Colab permet à n'importe qui d'écrire et d'exécuter le code Python de son choix par le biais du navigateur.

C'est un environnement particulièrement adapté au Machine Learning, à l'analyse de données et à l'éducation. En termes plus techniques, Colab est un service hébergé qui ne nécessite aucune configuration et permet d'accéder gratuitement à des ressources informatiques, dont les GPU.

4.2.2 Logiciels

Python : C'est un langage de programmation, un des langages de niveau supérieur, caractérisé par une écriture et une lecture simple, facile à apprendre, utilisant un style de programmation ouvert, et évolutif. Python est un langage polyvalent utilisé dans de nombreux domaines, tels que la création de programmes autonomes utilisant des interfaces graphiques connues et l'exécution de programmes Web, et son utilisation comme langage de script pour contrôler les performances des programmes les plus populaires. En général. Python peut être utilisé pour programmer des programmes simples pour les débutants et pour réaliser de grands projets en même temps que tout autre langage de programmation.il est souvent conseillé aux débutants en programmation d'apprendre cette langue car c'est l'une des langues de logiciel d'apprentissage les plus rapides [76].

Figure 4.1. Logo Python.

Keras : Keras [77] est un framework open source de Deep Learning pour le Python, capable de s'exécuter sur TensorFlow. Il est écrit par Francis Chollet, membre de l'équipe Google. Keras est utilisé dans un grand nombre de startups, de laboratoires de recherche [78] (dont le CERN, Microsoft Research et la NASA) et de grandes entreprises telles que Netflix, Yelp, Square, Uber, Google, etc. Il a été développé dans le cadre de l'effort de recherche du projet ONEIROS (Open-ended Neuro Electronic Intelligent Robot Operating System). En 2017, l'équipe TensorFlow de Google a pris la décision de fournir un support pour Keras et de l'intégrer dans la bibliothèque principale de TensorFlow. Il présente un ensemble d'abstractions de plus haut niveau et plus intuitives qui facilitent la configuration des réseaux neuronaux [77]

TensorFlow : est un framework de programmation pour le calcul numérique créé par Google et devenu un framework Open Source en novembre 2015, son architecture flexible permet un déploiement facile du calcul sur diverses plates-formes (CPU, GPU, TPU). Depuis, TensorFlow a continué à gagner une grand importance et popularité et rapidement devenu l'un des frameworks le plus utilisés pour le Deep Learning. Son nom s'inspire notamment du fait que les opérations actuelles des réseaux de neurones s'effectuent principalement via une table multidimensionnelle de données, appelée Tenseurs (Tensor) qui est l'équivalent d'une matrice [79].

Figure 4.2. Logo TensorFlow.

Anaconda : est une distribution libre et open source des langages de programmation Python et R appliqué au développement d'applications dédiées à la science des données et à Deep learning (traitement de données à grande échelle, analyse prédictive, calcul scientifique), qui vise à simplifier la gestion des paquets et de déploiement. Les versions de paquetages sont gérées par le système de gestion de paquets conda. La distribution Anaconda est utilisée par plus de 6 millions d'utilisateurs et comprend plus de 250 paquets populaires en science des données adaptés pour Windows, Linux et MacOS [80]

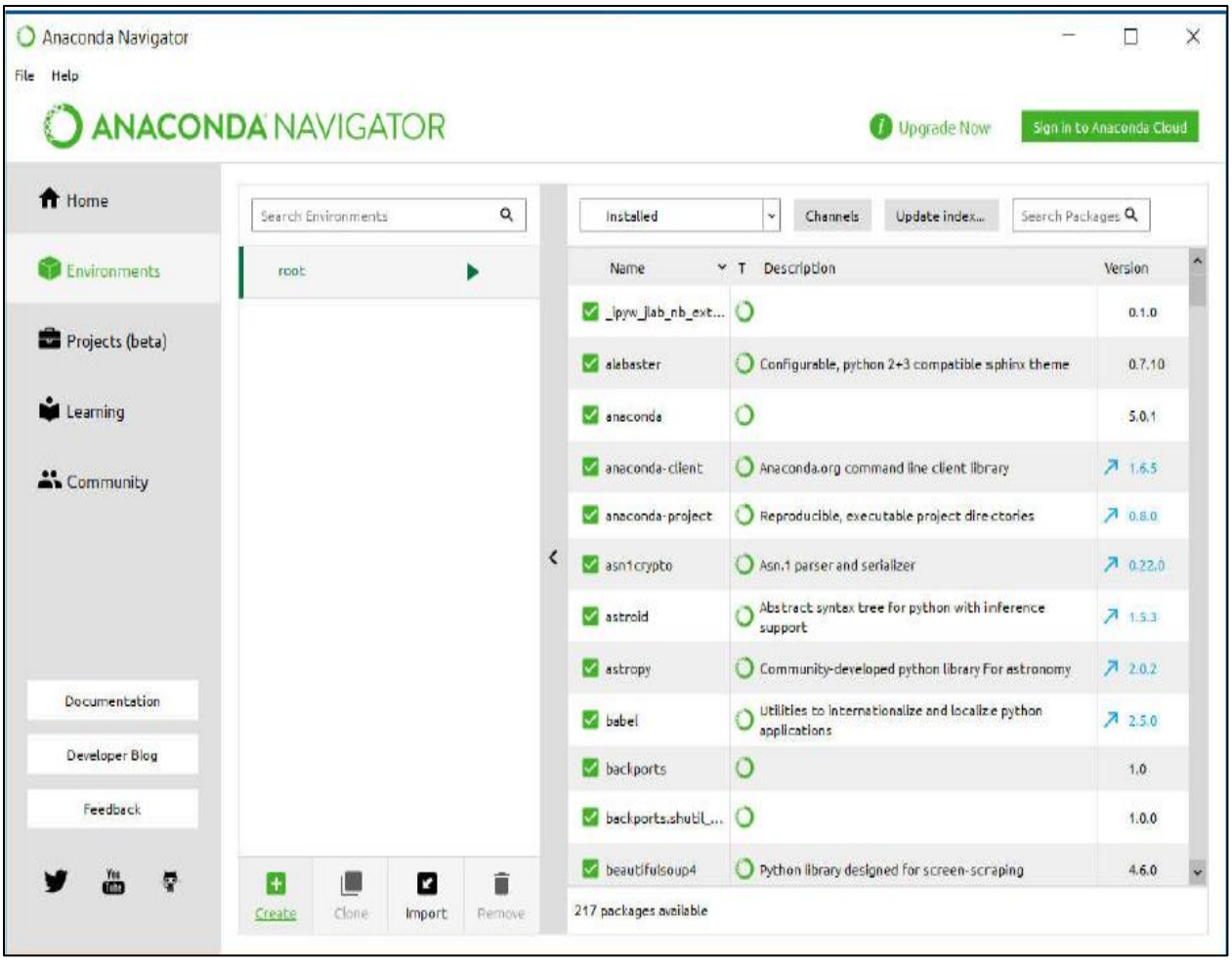


Figure 4.3. Interface Anaconda.

Spyder : est un environnement de développement pour python. Libre (Licence MIT) et multiplateforme (Windows, Mac OS, GNU/Linux), il intègre de nombreuse bibliothèque d'usage scientifique : Matplotlib, Numpy, SciPy et IPython**.** Créé et développé par Pierre Raybaut en 2008, Spyder est maintenu, depuis 2012, par une communauté de développeurs qui ont pour point commun d'apprentissage à la communauté Python scientifique.

En comparaison avec d'autres IDE pour le développement scientifique, Spyder a un ensemble unique de fonctionnalités multiplateforme, open source, écrit en Python et disponible sous une licence non copyleft. Spyder est extensible avec des plugins, comprend le support d'outils interactifs pour l'inspection des données et incorpore des instruments d'assurance de la qualité et d'introspection spécifiques au code Python.

4.3 Base de données

Notre projet s'appuie sur la classification des images, Les bases de données utilisées pour l'apprentissage sont appelées CIFAR-10 et MNIST.

CIFAR : est un acronyme qui signifie Canadian Institute For Advanced Research. Le jeu de données CIFAR-10 a été développé en même temps que le jeu de données CIFAR-100 par des chercheurs de l'institut CIFAR.

Le jeu de données CIFAR-10 se compose de 60000 images en couleurs 32x32 dans 10 classes, comme des grenouilles, des oiseaux, des chats, des bateaux, etc. Avec 6000 images par classe. Il y a 50000 images d'entraînement et 10000 images de test. L'ensemble de données est divisé en cinq lots de formation et un lot de tests, chacun contenant 10 000 images. Le lot de test contient exactement 1000 images sélectionnées au hasard dans chaque classe. Les lots d'entraînement contiennent les images restantes dans un ordre aléatoire, mais certains lots d'entraînement peuvent contenir plus d'images d'une classe que d'une autre. Entre eux, les lots de formation contiennent exactement 5000 images de chaque classe. Les étiquettes de classe et leurs valeurs entières standard associées sont énumérées ci-dessous :

0 : avion

1 : automobile

2 : oiseau

3 : chat

4 : cerf

5 : chien

6 : grenouille

7 : cheval

8 : bateau

9 : camion

Voici les classes dans le jeu de données, ainsi que 10 images aléatoires de chacune :

Figure 4.4. Images de CIFAR-10 classes.

MNIST : l'acronyme MNIST (Modified ou Mixed National Institute of Standards and Technology), est une base de données de chiffres écrits à la main. C'est un jeu de données très utilisé en Deep Learning.

La reconnaissance de l'écriture manuscrite est un problème difficile, et un bon test pour les algorithmes d'apprentissage. La base MNIST est devenue un test standard. Elle regroupe 60000 images d'apprentissage et 10000 images de test, ce sont des images en noir et blanc, normalisées centrées de 28 pixels de côté dans 10 classes.

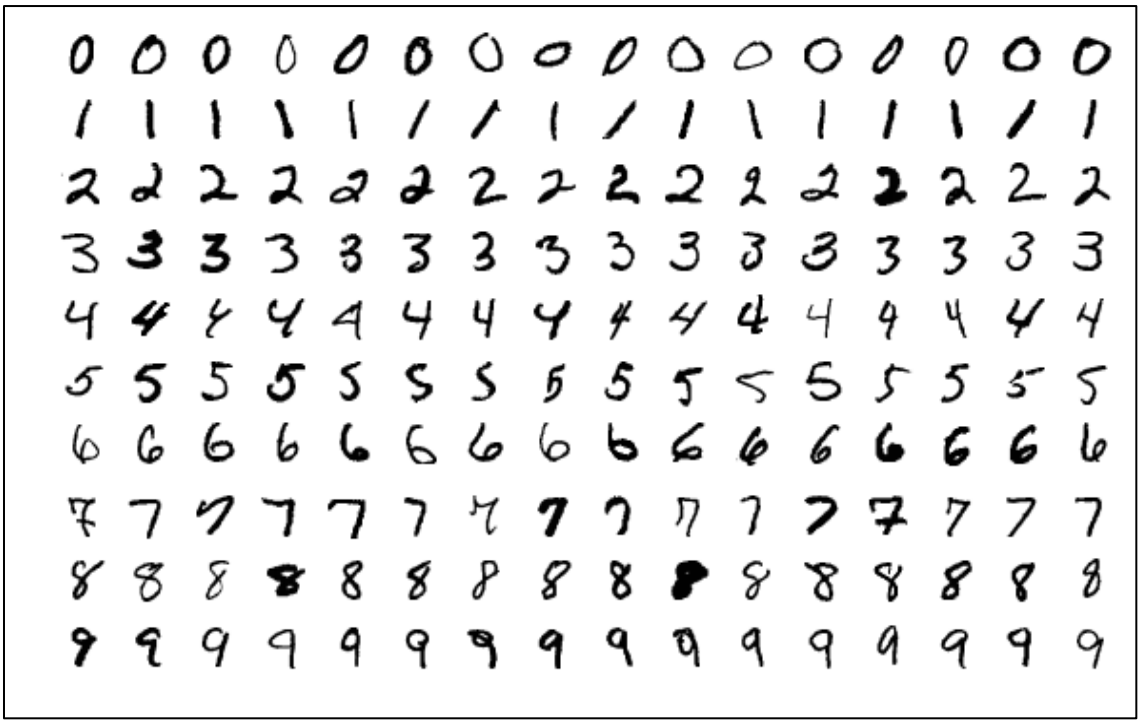


Figure 4.5. Images de MNIST classes.

Le tableau ci-dessous illustre la structure de notre base de données et l'apprentissage et le test utilisée.

Base données	Apprentissage	Test	Totale
CIFAR-10	50000	10000	60000
MNIST	60000	10000	70000

Tableau 4.1. Base de données utilisées.

4.4 Implémentation

4.4.1 L'architecture de notre réseau

Dans notre expérimentation, nous avons créé deux modèles avec des architectures différentes, nous avons appliqué sur la base de l'image CIFAR-10 et MNIST, l'algorithme utilisée CNN (Convolutional Neural Networks), le premier modèle était un CNN sur CIFAR-10 et le deuxième était un CNN sur MNIST.

Dans ce qui suit, nous présentons l'architecture de ces deux modèles et le résultat obtenu à chaque fois.

Le modèle CNN sur CIFAR-10

Le premier modèle dans la figure 4.6 représente les différentes couches (quatre couches de Convolution, deux couches de Maxpooling et de deux couches de Fully Connected).

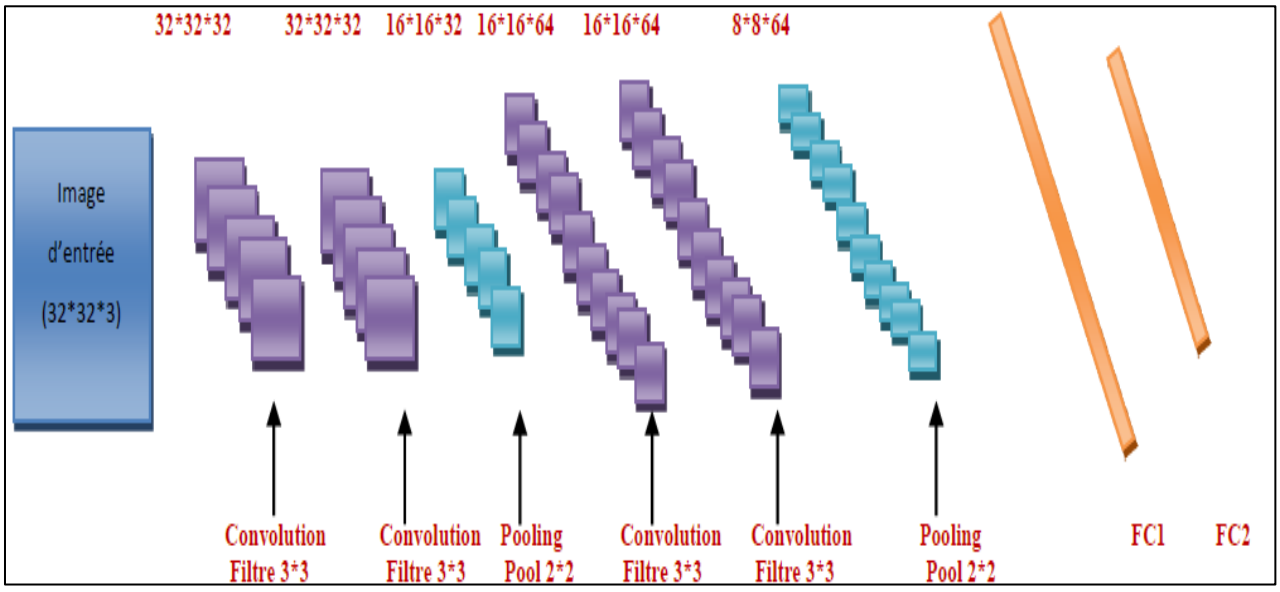


Figure 4.6. L'architecture du modèle CNN sur CIFAR-10.

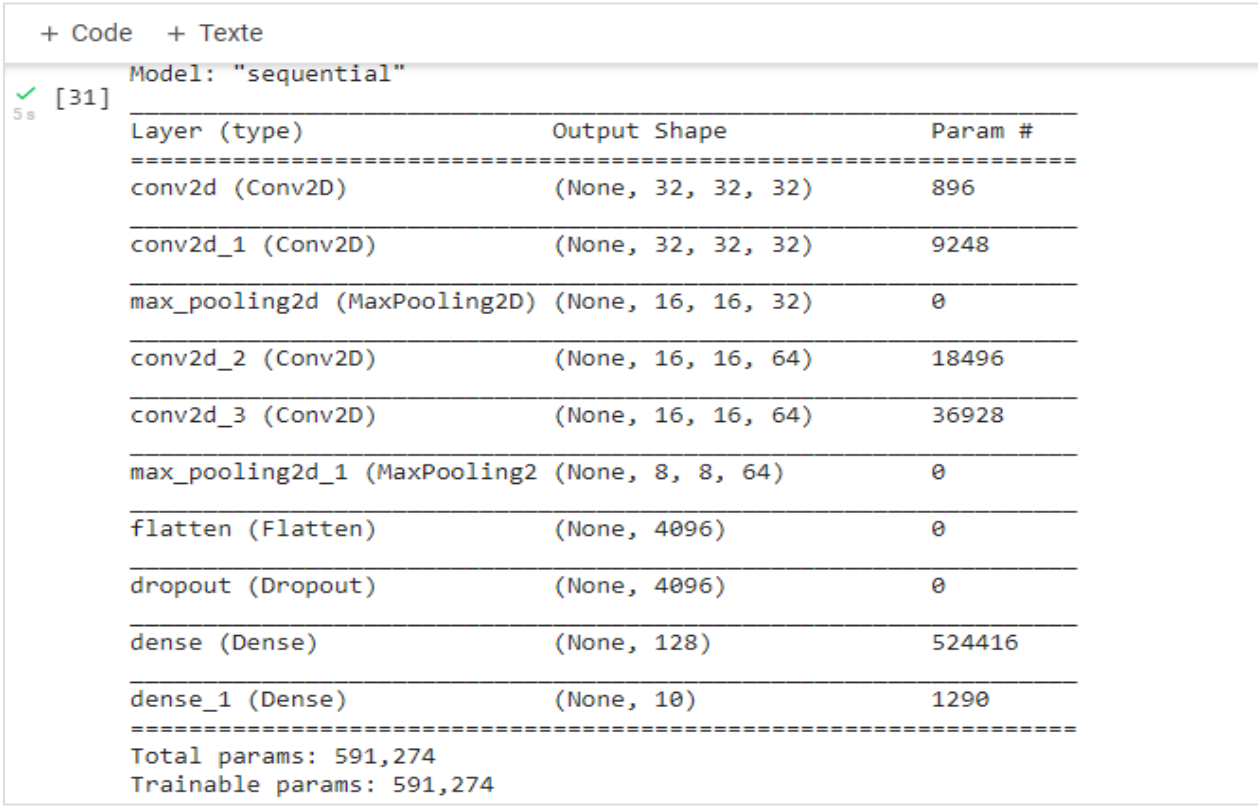


Figure 4.7. Configuration de modèle CNN sur CIFAR-10.

- L'image en entrée a une taille de 32*32*3 (32 de large, 32 de haut, 3 canaux de couleur).

- Les couches de Convolution : chaque couche composée de plusieurs filtres de taille 3*3 (32 pour les 2 premières couches et 64 pour les 2 dernières couches) après les 2 premières Convolutions seront créés 32 features maps de taille 32*32 et après 2 dernières 64 features maps de taille 16*16. La fonction d'activation ReLU est utilisée à chaque fois qu'on passe par une couche de Convolution, cette fonction d'activation force les neurones à retourner des valeurs positives.

- Couche de Maxpooling : est appliqué après pour réduire la taille de l'image. À la sortie de cette couche, nous aurons pour la première couche 32 features maps de taille 16*16 et pour la deuxième 64 features maps de taille 8*8.

- La couche Flatten : nous ajoutons une couche flatten dans le but de préparer le résultat de la couche Maxpooling à la couche entièrement connectée, la figure suivante [86] illustre l'idée :

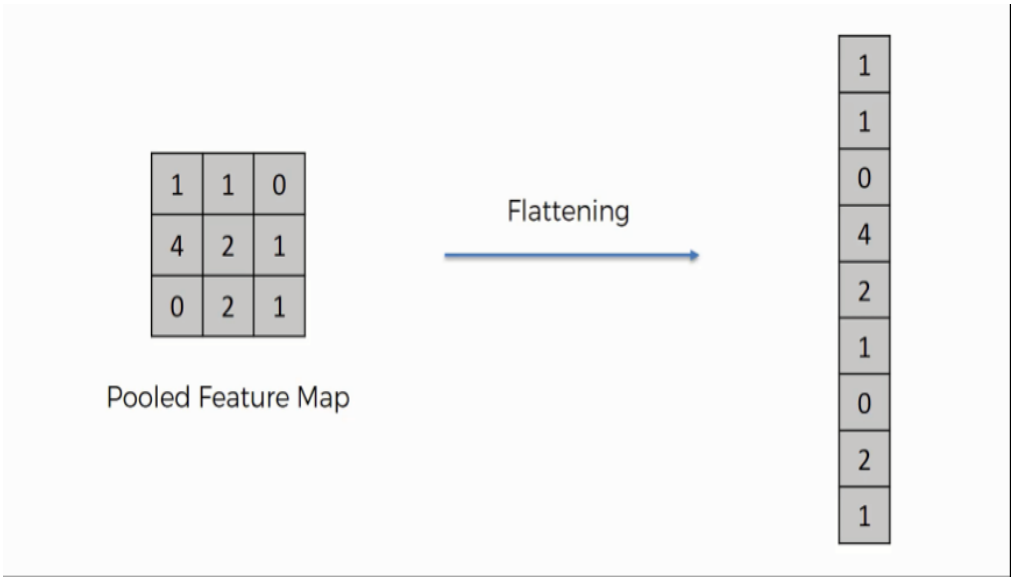


Figure 4.8. Illustration du rôle de la couche Flatten [86].

- La couche Dropout : nous avons appliqué une seul couche Dropout est une technique de régularisation pour réduire le sur ajustement dans les réseaux neurones.

- La couche entièrement connectée : La première couche est composée de 128 neurones où la fonction d'activation utilisée est le ReLU, la deuxième couche utilise la fonction Softmax qui permet de calculer la distribution de probabilité des 10 classes (nombre de classe dans la base d'image CIFAR-10).

Le modèle CNN sur MNIST :

Le deuxième modèle dans la figure 4.9 représente les différentes couches (deux couches de Convolution, une couche de Maxpooling et deux couches de Fully Connected).

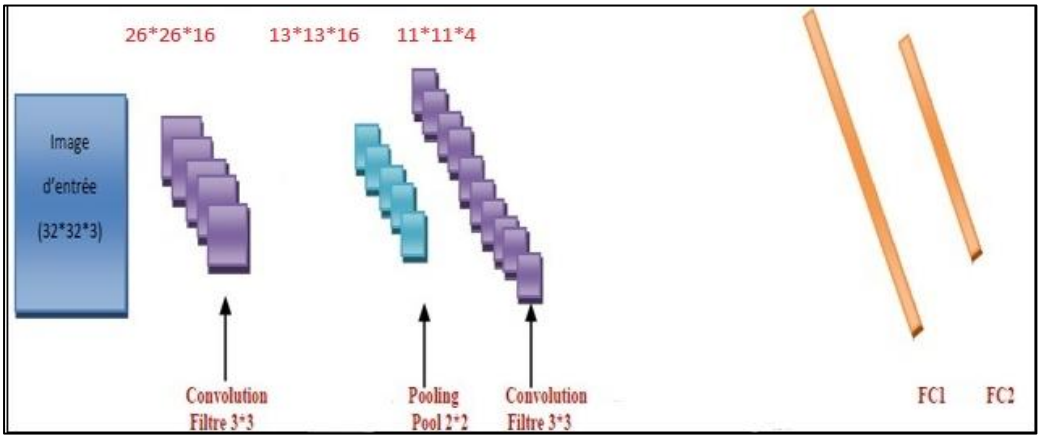


Figure 4.9. L'architecture du modèle CNN sur MNIST.

```
[28] Model: "sequential_6"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_9 (Conv2D)            (None, 26, 26, 16)        160
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 13, 13, 16)        0
_________________________________________________________________
conv2d_10 (Conv2D)           (None, 11, 11, 4)         580
_________________________________________________________________
dropout_1 (Dropout)          (None, 11, 11, 4)         0
_________________________________________________________________
flatten_2 (Flatten)          (None, 484)               0
_________________________________________________________________
dense_4 (Dense)              (None, 10)                4850
_________________________________________________________________
dense_5 (Dense)              (None, 10)                110
=================================================================
Total params: 5,700
Trainable params: 5,700
Non-trainable params: 0
_________________________________________________________________
```

Figure 4.10. Configuration de modèle CNN sur MNIST.

- L'image en entrée a une taille de 28*28*1.

- Les couches de Convolution : chaque couche composée de plusieurs filtres de taille 3*3 (16 pour la première couche et 4 pour la deuxième couche) après la première Convolution sera créé 16 features maps de taille 26*26 et après la dernière 4 features maps de taille 11*11. La fonction d'activation ReLU est utilisée à chaque fois qu'on passe par une couche de Convolution, cette fonction d'activation force les neurones à retourner des valeurs positives.

- Couche de Maxpooling : est appliqué après pour réduire la taille de l'image. À la sortie de cette couche, nous aurons pour une couche 16 features maps de taille 13*13.

- La couche Flatten : nous ajoutons une couche flatten dans le but de préparer le résultat de la couche Maxpooling à la couche entièrement connectée.

- La couche Dropout : nous avons appliqué une seule couche Dropout pour la régularisation.

- La couche entièrement connectée : La première couche est composée de 10 neurones où la fonction d'activation utilisée est le ReLU, la deuxième couche utilise la fonction

Softmax qui permet de calculer la distribution de probabilité des 10 classes (nombre de classe dans la base d'image MNIST).

4.5 Résultats obtenus et discussion

Afin de montrer les résultats obtenus pour les deux modèles, on illustre dans ce qui suit les résultats en termes de précision et d'erreur.

4.5.1 Résultats obtenus pour le modèle CIFAR-10

- Nombre d'époques =15

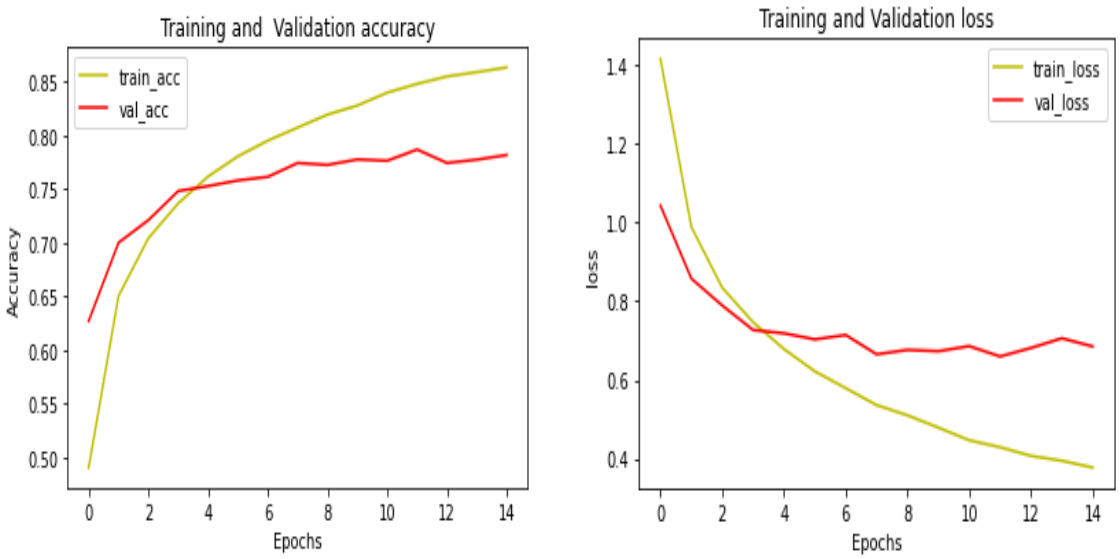


Figure 4.11. Précision et l'erreur de modèle CIFAR-10 (15 époques).

Après l'analyse des résultats obtenus, on constate les remarques suivantes :

D'après la figure 4.11 la précision de l'apprentissage et de la validation augmente avec le nombre d'époques jusqu'au nombre d'époques égale 15.

Cela reflète qu'à chaque époque, le modèle apprend plus d'informations. Si la précision diminue, nous aurons besoin de plus d'informations pour faire apprendre notre modèle, nous devons donc augmenter le nombre d'époques.

De même, l'erreur d'apprentissage et de la validation diminue avec le nombre d'époques jusqu'au nombre d'époques égale 15.

Nous remarquons que la totalité des images mal classées est de 6851 images, un taux d'erreur de 68,51% et la totalité des images bien classées est de 7817 un taux de précision de 78,17%.

- Nombre d’époques=30

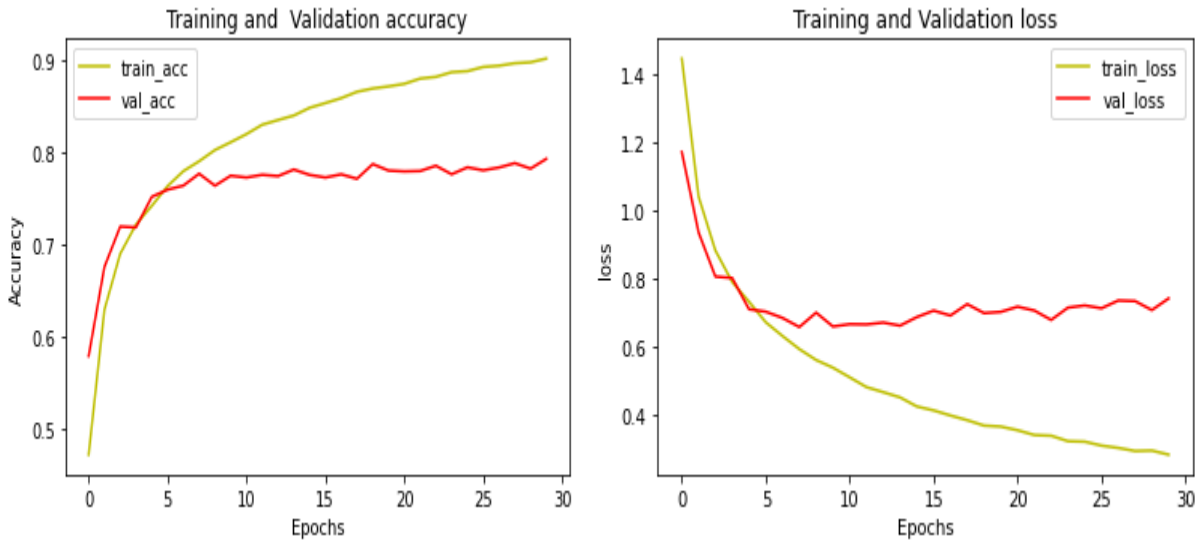


Figure 4.12. Précision et l’erreur de modèle CIFAR-10 (30 époques).

La figure 4.12 montre la performance de modèle CNN sur CIFAR-10 avec le nombre d’époques égale 30. Dans la précision l’apprentissage augmente avec le nombre d’époques jusqu’au nombre d’époques égale 30 et la validation dans intervalle de nombre d’époques [0- 7] augmente et dans l’intervalle [7-30] reste au niveau 0.8.

De même, dans l’erreur l’apprentissage diminue avec le nombre d’époques jusqu'au nombre d’époques égale 30 et la validation diminue dans l’intervalle de nombre d’époques [0-7] et reste stationnaire dans l’intervalle [7-30].

Nous remarquons que la totalité des images mal classées est de 7411 images, un taux d’erreur de 74,11% et la totalité des images bien classées est de 7924 un taux de précision de 79,24%.

4.5.2 Résultats obtenus pour le modèle MNIST

- Nombre d'époques =15

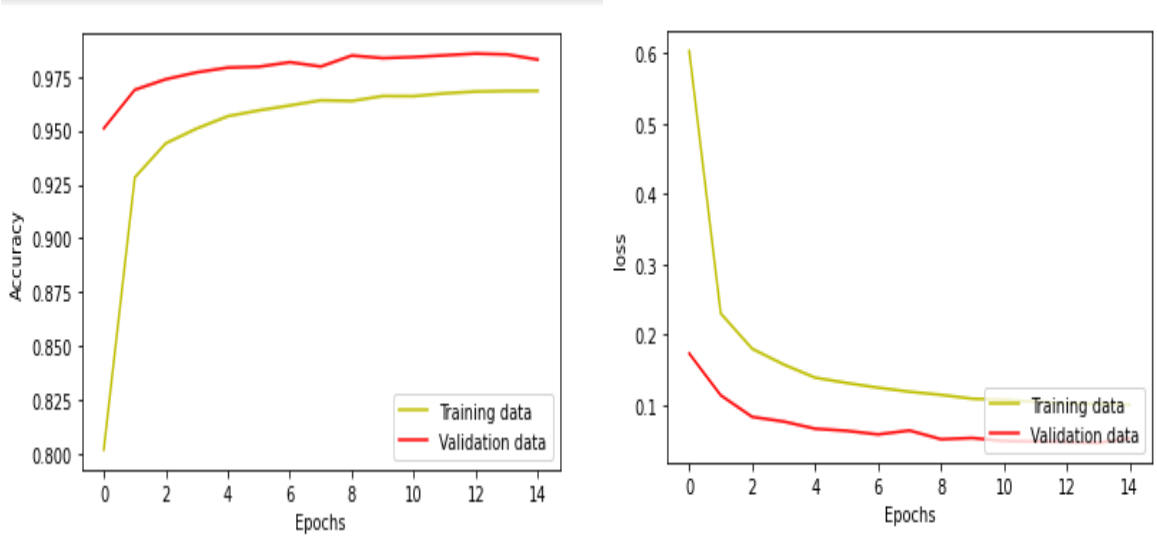


***Figure 4.13*.** Précision et l'erreur de modèle CNN sur MNIST (15 époques).

D'après la figure 4.13, la précision de l'apprentissage et de la validation augmente rapidement dans l'intervalle du nombre d'époques [0-1.8] au niveau 0.925 et continue à grandir doucement dans l'intervalle [1.8-15].

De même, dans l'erreur l'apprentissage et de la validation diminue rapidement avec le nombre d'époques jusqu'au nombre d'époques égale 2 et reste diminue dans l'intervalle de nombre d'époques [2-15].

Nous remarquons que la totalité des images mal classées est de 1003 images, un taux d'erreur de 10,03% et la totalité des images bien classées est de 9829 un taux de précision de 98 ,29%.

- Nombre d'époques =30

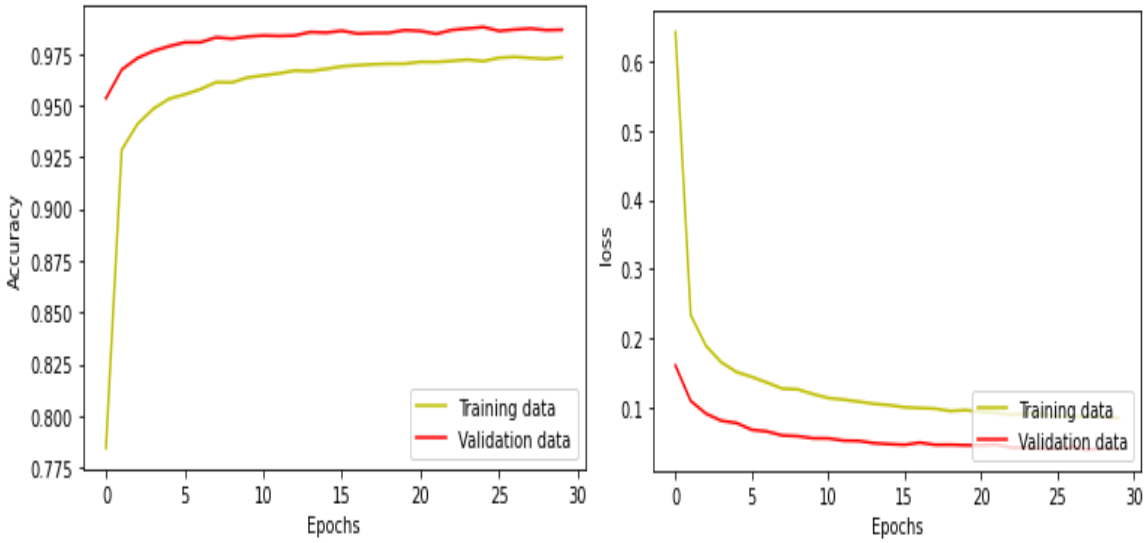


Figure 4.14. Précision et l'erreur de modèle CNN sur MNIST (30 époques).

La figure 4.14 montre la performance de modèle CNN sur MNIST avec le nombre d'époques égale 30. Au début de la précision de l'apprentissage et de la validation augmente rapidement avec le nombre d'époques égale 1 au niveau 0.925, et reste la croissance pendant les époques mais leurs valeurs sont différentes.

De même, pour l'erreur de l'apprentissage et de la validation diminue avec le nombre d'époques, il se stabilise à peu près après l'époque numéro 15.

Nous remarquons que la totalité des images mal classées est de 0855 images, un taux d'erreur de 08,55% et la totalité des images bien classées est de 9865 un taux de précision de 98,65%.

4.6 Tableau de comparaison

Le tableau ci-dessous montre les différents résultats obtenus sur les deux modèles :

Base de données	Architecture de modèle			Nombre d'époques	Taux de précision	Taux d'erreur
CIFAR-10	4 couches de Conv	2 couches de Maxpooling	2 couches de FC	15	78 ,17%	68,51%
				30	79,24%	74,11%
MNIST	2 couches de Conv	1 couche de Maxpooling	2 couches de FC	15	98,29%	10,03%
				30	98,65%	08,55%

Tableau 4.2. Comparaison des résultats.

Le tableau montre l'architecture utilisée pour chaque modèle ainsi que le nombre d'époques utilisé. Les résultats obtenus sont exprimés en termes de précision et erreur. Ce dernier est utilisé GPU, notre carte graphique à la place du processeur (CPU) pour lancer notre modèle.
Pour notre premier modèle (CNN sur CIFAR-10), on remarque qu'à chaque fois qu'on augmente le nombre d'époques, le taux de précision augmente et le taux d'erreur diminue, mais arrivé certain seuil d'époques, le taux d'erreur commence à augmente.
En ce qui concerne notre deuxième modèle (CNN sur MNIST), on remarque que l'augmentation des époques entraine une augmentation du taux de précision et une diminution du taux d'erreur qui n'est pas assez importante (presque 2%).

Si on compare nos deux modèles, on remarque que pour 15 et 30 époques le modèle MNIST donne de meilleurs résultats (une différence de 19%) et ceci grâce à l'architecture et la base de données, et pour le taux d'erreur reste le deuxième modèle le meilleur avec le nombre d'époques 15 et 30.
D'une manière générale les deux modèles donnent à peu près les bons résultats, mais la base d'image MNIST fait mieux en termes de toutes les mesure.
Le CNN important et profond, donne de bonnes performances si nous les utilisons pour le problème de classification d'images où la quantité des données de l'apprentissage joue un rôle important et peut affecter les résultats surtout si nous avons entrainé le CNN en faible quantité.

4.7 Conclusion

Dans ce chapitre, nous avons vu notre part de contribution au problème de classification d'image basée sur le réseau de neurone convolutionnel, pour cela on a utilisé deux modèles avec différentes architectures et on a montré les différents résultats obtenus en termes de précision et d'erreur. La comparaison des résultats trouvés a montré que le nombre d'époques, la profondeur de réseaux, sont des facteurs importants pour l'obtention de meilleurs résultats. A la fin, On conclut que la base d'image MNIST fait mieux en termes de toutes les mesures.

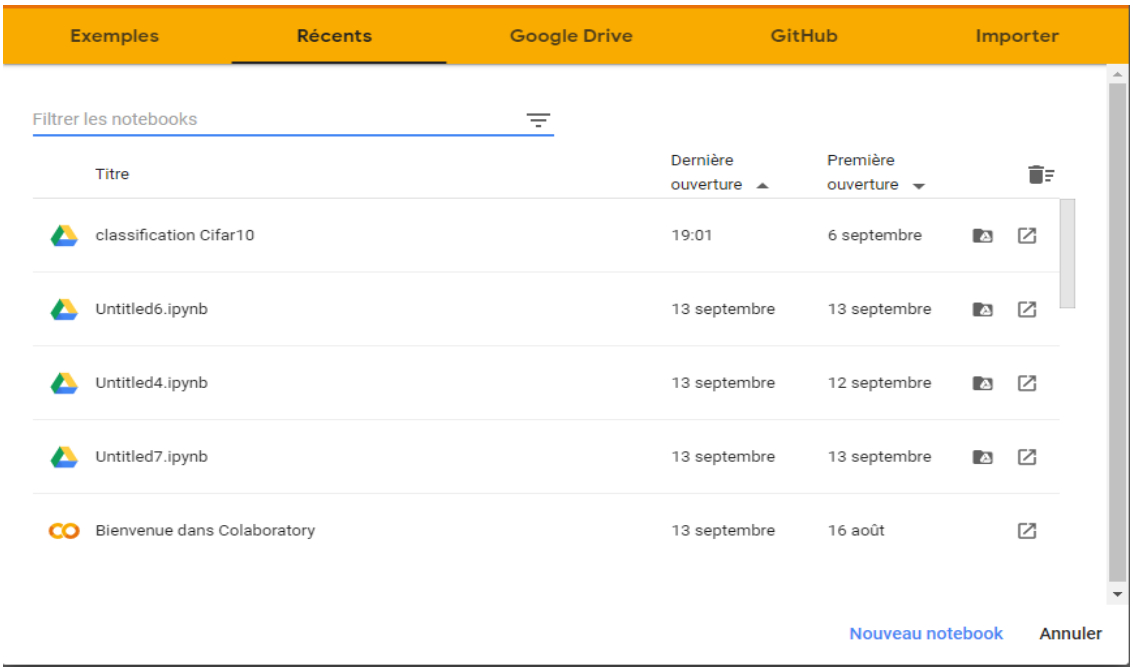
Conclusion générale

Dans ce mémoire, nous avons fait un état de l'art de Deep Learning en Télécommunication. Nous avons discuté l'apport important du Deep Learning dans le domaine de la télécommunication, des notions fondamentales du Deep Learning, des algorithmes les plus utilisés, des réseaux de neurones en général et des réseaux de neurones convolutionnels en particulier. Nous avons introduit ces réseaux de neurones convolutionnels en présentant les différents types de couches utilisées dans la classification (couche convolutionnelle, couche de Pooling et couche entièrement connectée), et nous avons implémenté ce principe dans la classification d'image qui est un sous-champ du traitement d'image où nous avons conclu que la base de données d'images MNIST donne de meilleurs résultats en termes de toutes les mesures.

Afin d'atteindre ces résultats, nous avons passé beaucoup de temps à lire et réviser des publications, des articles et des livres pour voir et comprendre les concepts et comment appliquer un modèle de Deep Learning à notre problématique.

Enfin, ce travail nous a permis de développer et de mettre en pratique nos connaissances et de les enrichir en abordant plusieurs domaines tels que le Deep Learning, et surtout le langage de programmation Python, et le plus important est que nous fassions le premier pas vers le Deep Learning, un des champs les plus importants de l'intelligence artificielle. En souhaitant que cela nous permettra d'ouvrir des horizons dans le futur.

L’annexe 1 : Google Colab



Pour visiter Google Colab en effectuant une recherche sur notre navigateur Web de préférence Chrome, il affichera automatiquement nos cahiers précédents et donnera une option pour créer un nouveau cahier. Ici, nous pouvons cliquer sur un nouveau bloc-notes et commencer à exécuter notre code. Google colab est un formidable Jupyter Notebook basé sur un navigateur en ligne qui nous permet d’entrainer gratuitement des modèles Machine et Deep Learning sur des CPU, GPU, TPU et on préfère toujours GPU à n’importe quel autre CPU en raison de la puissance de calcul et de la vitesse d’exécution.


Welcome To Colaboratory
File Edit View Insert Runtime Tools Help
Share
+ Code + Text
Copy to Drive
Connect
Editing
Getting Started
The document you are reading is a Jupyter notebook, hosted in Colaboratory. It is not a static page, but an interactive environment that lets you write and execute code in Python and other languages.
For example, here is a code cell with a short Python script that computes a value, stores it in a variable, and prints the result:
[ ] seconds_in_a_day = 24 * 60 * 60
seconds_in_a_day
86400
To execute the code in the above cell, select it with a click and then either press the play button to the left of the code, or use the keyboard shortcut "Command/Ctrl+Enter".
All cells modify the same global state, so variables that you define by executing a cell can be used in other cells:
[ ] seconds_in_a_week = 7 * seconds_in_a_day
seconds_in_a_week
604800
For more information about working with Colaboratory notebooks, see Overview of Colaboratory.

L’annexe 2 : Programmation de CIFAR-10

```
[1]  import tensorflow as tf
     import matplotlib.pyplot as plt
     from tensorflow.keras.datasets import cifar10
```

Loading the datasts

```
class_names=['airplane','automobile','bird','cat','deer','dog','frog','horse','ship','truck']
(x_train,y_train),(x_test,y_test)=cifar10.load_data()
```

```
Downloading data from https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz
170500096/170498071 [==============================] - 11s 0us/step
170508288/170498071 [==============================] - 11s 0us/step
```

Normalizing the images

```
[24] x_train=x_train/255.0
     x_train.shape
```

```
(50000, 32, 32, 3)
```

```
[25] x_test=x_test/255.0
     x_test.shape
```

```
(10000, 32, 32, 3)
```

```
[26] y_train.shape
```

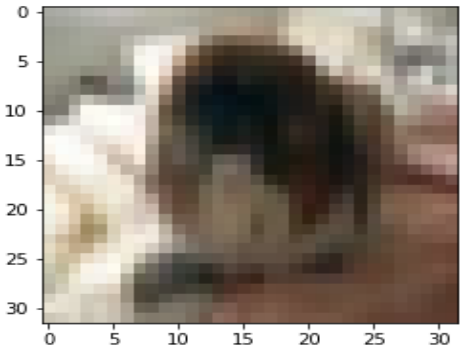
```
(50000, 1)
```

```
[27] y_test.shape
```

```
(10000, 1)
```

```
[28] plt.imshow(x_test[200])
```

```
<matplotlib.image.AxesImage at 0x7f6859aaf310>
```



Building a Convolutional Neural Network

```
[31] cifar10_model=tf.keras.models.Sequential()
     cifar10_model.add(tf.keras.layers.Conv2D(filters=32,kernel_size=3,padding="same",activation="relu",input_shape=[32,32,3]))
     cifar10_model.add(tf.keras.layers.Conv2D(filters=32,kernel_size=3,padding="same",activation="relu"))
     cifar10_model.add(tf.keras.layers.MaxPool2D(pool_size=2,strides=2,padding='valid'))
     cifar10_model.add(tf.keras.layers.Conv2D(filters=64,kernel_size=3,padding="same",activation="relu"))
     cifar10_model.add(tf.keras.layers.Conv2D(filters=64,kernel_size=3,padding="same",activation="relu"))
     cifar10_model.add(tf.keras.layers.MaxPool2D(pool_size=2,strides=2,padding='valid'))
     cifar10_model.add(tf.keras.layers.Flatten())
     cifar10_model.add(tf.keras.layers.Dropout(0.5,noise_shape=None,seed=None))
     cifar10_model.add(tf.keras.layers.Dense(units=128,activation='relu'))
     cifar10_model.add(tf.keras.layers.Dense(units=10,activation='softmax'))
     cifar10_model.summary()
```

+ Code + Texte RAM Disque Modification

```
[32] cifar10_model.compile(loss="sparse_categorical_crossentropy",optimizer="Adam",metrics=["accuracy"])
```

Training the Model

```
[33] history=cifar10_model.fit(x_train,y_train,validation_data=(x_test,y_test),epochs=15)

     Epoch 1/15
     1563/1563 [==============================] - 48s 12ms/step - loss: 1.4148 - accuracy: 0.4908 - val_loss: 1.0425 - val_accuracy: 0.6273
     Epoch 2/15
     1563/1563 [==============================] - 17s 11ms/step - loss: 0.9886 - accuracy: 0.6505 - val_loss: 0.8574 - val_accuracy: 0.7001
     Epoch 3/15
     1563/1563 [==============================] - 17s 11ms/step - loss: 0.8344 - accuracy: 0.7043 - val_loss: 0.7890 - val_accuracy: 0.7211
     Epoch 4/15
     1563/1563 [==============================] - 17s 11ms/step - loss: 0.7466 - accuracy: 0.7367 - val_loss: 0.7268 - val_accuracy: 0.7482
     Epoch 5/15
     1563/1563 [==============================] - 17s 11ms/step - loss: 0.6785 - accuracy: 0.7617 - val_loss: 0.7184 - val_accuracy: 0.7528
     Epoch 6/15
     1563/1563 [==============================] - 17s 11ms/step - loss: 0.6218 - accuracy: 0.7805 - val_loss: 0.7031 - val_accuracy: 0.7581
     Epoch 7/15
     1563/1563 [==============================] - 17s 11ms/step - loss: 0.5791 - accuracy: 0.7951 - val_loss: 0.7141 - val_accuracy: 0.7614
     Epoch 8/15
     1563/1563 [==============================] - 17s 11ms/step - loss: 0.5362 - accuracy: 0.8073 - val_loss: 0.6652 - val_accuracy: 0.7743
     Epoch 9/15
```

✓ 1 s terminée à 00:09

+ Code + Texte RAM Disque Modification

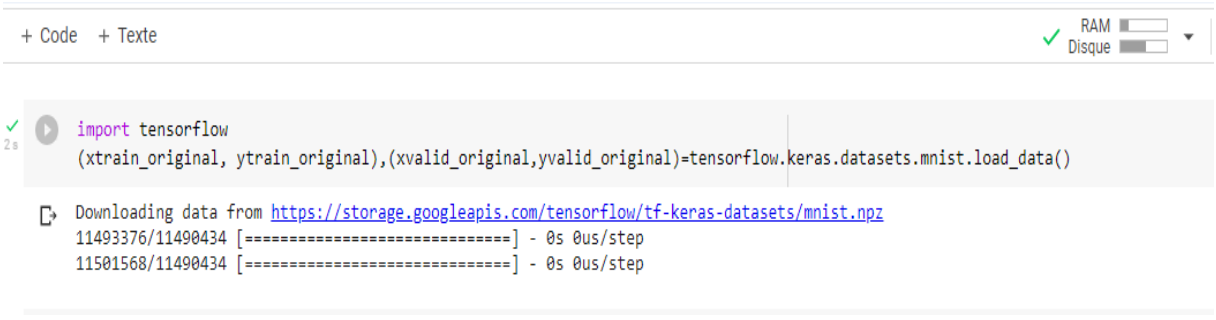
```
[33] 1563/1563 [==============================] - 17s 11ms/step - loss: 0.5362 - accuracy: 0.8073 - val_loss: 0.6652 - val_accuracy: 0.7743
     Epoch 9/15
     1563/1563 [==============================] - 17s 11ms/step - loss: 0.5102 - accuracy: 0.8194 - val_loss: 0.6762 - val_accuracy: 0.7725
     Epoch 10/15
     1563/1563 [==============================] - 17s 11ms/step - loss: 0.4790 - accuracy: 0.8278 - val_loss: 0.6730 - val_accuracy: 0.7776
     Epoch 11/15
     1563/1563 [==============================] - 17s 11ms/step - loss: 0.4468 - accuracy: 0.8398 - val_loss: 0.6858 - val_accuracy: 0.7764
     Epoch 12/15
     1563/1563 [==============================] - 17s 11ms/step - loss: 0.4292 - accuracy: 0.8479 - val_loss: 0.6596 - val_accuracy: 0.7868
     Epoch 13/15
     1563/1563 [==============================] - 17s 11ms/step - loss: 0.4073 - accuracy: 0.8549 - val_loss: 0.6812 - val_accuracy: 0.7743
     Epoch 14/15
     1563/1563 [==============================] - 17s 11ms/step - loss: 0.3948 - accuracy: 0.8589 - val_loss: 0.7057 - val_accuracy: 0.7774
     Epoch 15/15
     1563/1563 [==============================] - 17s 11ms/step - loss: 0.3776 - accuracy: 0.8632 - val_loss: 0.6851 - val_accuracy: 0.7817
```

```
[35] test_loss,test_accuracy=cifar10_model.evaluate(x_test,y_test)

     313/313 [==============================] - 2s 6ms/step - loss: 0.6851 - accuracy: 0.7817
```

L’annexe 3 : Programmation de MNIST

Step 0 : Load MNIST dataset



Step 1 : check the number of images and their dimensions

```
[4]  print("Training dataset")
     print(xtrain_original.shape)
     print(ytrain_original.shape)
     print('Validation dataset:')
     print(xvalid_original.shape)
     print(yvalid_original.shape)
```

```
Training dataset
(60000, 28, 28)
(60000,)
Validation dataset:
(10000, 28, 28)
(10000,)
```

Step 2 : visualize a random image and its label in the train set

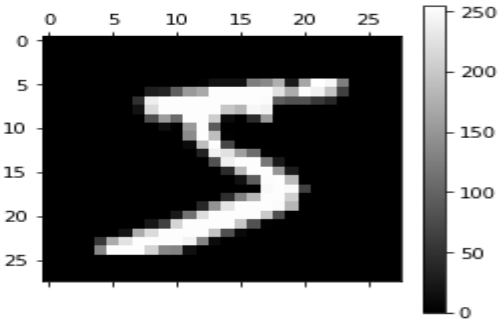
```
print(xtrain_original[0, 6:24, 6:24])
```

```
[[  0   0  30  36  94 154 170 253 253 253 253 253 225 172 253 242 195  64]
 [  0  49 238 253 253 253 253 253 253 253 253 251  93  82  82  56  39   0]
 [  0  18 219 253 253 253 253 253 198 182 247 241   0   0   0   0   0   0]
 [  0   0  80 156 107 253 253 205  11   0  43 154   0   0   0   0   0   0]
 [  0   0   0  14   1 154 253  90   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0 139 253 190   2   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0  11 190 253  70   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0  35 241 225 160 108   1   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0  81 240 253 253 119  25   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0  45 186 253 253 150  27   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0  16  93 252 253 187   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0 249 253 249  64   0   0   0]
 [  0   0   0   0   0   0   0   0  46 130 183 253 253 207   2   0   0   0]
 [  0   0   0   0   0   0  39 148 229 253 253 253 250 182   0   0   0   0]
 [  0   0   0   0  24 114 221 253 253 253 253 201  78   0   0   0   0   0]
 [  0   0  23  66 213 253 253 253 253 198  81   2   0   0   0   0   0   0]
 [ 18 171 219 253 253 253 253 195  80   9   0   0   0   0   0   0   0   0]
 [226 253 253 253 253 244 133  11   0   0   0   0   0   0   0   0   0   0]]
```

```
+ Code  + Texte

[6]  import matplotlib.pyplot as plt

     plt.matshow(xtrain_original[0],cmap= 'gray')
     plt.colorbar()
     plt.show()
```




```
[7]  print(ytrain_original[0])

     5
```

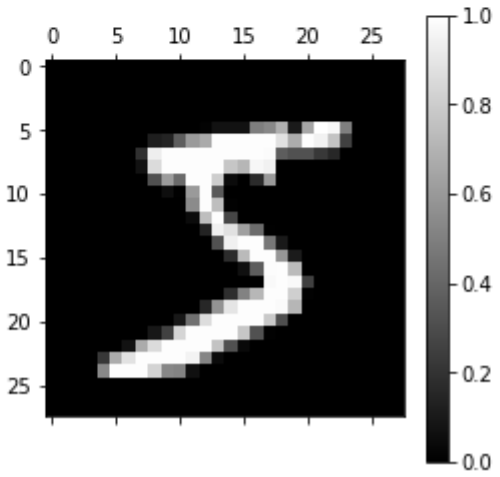
Step 3 : reshape and normalize the input

```
+ Code  + Texte

[8]  xtrain= xtrain_original.reshape( (60000, 28, 28, 1) )
     xvalid= xvalid_original.reshape( (10000, 28, 28, 1) )

     xtrain= xtrain / 255
     xvalid= xvalid / 255

[9]
     plt.matshow( xtrain[0, :, :, 0], cmap= 'gray')
     plt.colorbar()
     plt.show()
```



Step 4 : reformat our labels

+ Code + Texte

```
[10] print("Before:")
     print( ytrain_original.shape )
     print( ytrain_original[0] )
```

```
Before:
(60000,)
5
```

```
[11]
     ytrain=tensorflow.keras.utils.to_categorical( ytrain_original )
     yvalid=tensorflow.keras.utils.to_categorical( yvalid_original )
```

```
[12] print("After:")
     print( ytrain.shape )
     print( ytrain[0] )
```

```
After:
(60000, 10)
[0. 0. 0. 0. 0. 1. 0. 0. 0. 0.]
```

Step 5 : create a neural network with the following architecture

```
[23] from tensorflow.keras.models import Sequential
     from tensorflow.keras.layers import Dense,Conv2D, Flatten, MaxPooling2D, Dropout

     model= Sequential()
     model.add( Conv2D( 16, ( 3, 3), activation ='relu', input_shape = xtrain[0, :, :, :].shape) )
     model.add( MaxPooling2D( 2, 2))
     model.add( Conv2D( 4, ( 3, 3), activation ='relu') )
     model.add( Dropout( 0.5))
     model.add( Flatten() )
     model.add( Dense( 10, activation ='relu') )
     model.add( Dense( 10, activation ='softmax') )
     model.summary()
```

Step 6 : compile and train the model

```
[29] model.compile( optimizer = 'adam', loss='categorical_crossentropy', metrics = [ 'accuracy' ] )
     history=model.fit( xtrain, ytrain, validation_data = (xvalid, yvalid), epochs = 15, batch_size = 64 )
```

```
[29] Epoch 1/15
     938/938 [==============================] - 6s 6ms/step - loss: 0.6028 - accuracy: 0.8020 - val_loss: 0.1729 - val_accuracy: 0.9510
     Epoch 2/15
     938/938 [==============================] - 5s 5ms/step - loss: 0.2300 - accuracy: 0.9283 - val_loss: 0.1137 - val_accuracy: 0.9689
     Epoch 3/15
     938/938 [==============================] - 5s 5ms/step - loss: 0.1797 - accuracy: 0.9441 - val_loss: 0.0829 - val_accuracy: 0.9738
     Epoch 4/15
     938/938 [==============================] - 5s 5ms/step - loss: 0.1574 - accuracy: 0.9510 - val_loss: 0.0763 - val_accuracy: 0.9770
     Epoch 5/15
     938/938 [==============================] - 5s 6ms/step - loss: 0.1387 - accuracy: 0.9567 - val_loss: 0.0661 - val_accuracy: 0.9792
     Epoch 6/15
     938/938 [==============================] - 5s 6ms/step - loss: 0.1312 - accuracy: 0.9593 - val_loss: 0.0631 - val_accuracy: 0.9796
     Epoch 7/15
     938/938 [==============================] - 5s 5ms/step - loss: 0.1245 - accuracy: 0.9617 - val_loss: 0.0580 - val_accuracy: 0.9817
     Epoch 8/15
     938/938 [==============================] - 5s 5ms/step - loss: 0.1187 - accuracy: 0.9640 - val_loss: 0.0636 - val_accuracy: 0.9797
     Epoch 9/15
     938/938 [==============================] - 5s 5ms/step - loss: 0.1143 - accuracy: 0.9637 - val_loss: 0.0511 - val_accuracy: 0.9848
     Epoch 10/15
     938/938 [==============================] - 5s 6ms/step - loss: 0.1087 - accuracy: 0.9660 - val_loss: 0.0529 - val_accuracy: 0.9836
     Epoch 11/15
     938/938 [==============================] - 5s 5ms/step - loss: 0.1073 - accuracy: 0.9660 - val_loss: 0.0485 - val_accuracy: 0.9841
     Epoch 12/15
     938/938 [==============================] - 5s 6ms/step - loss: 0.1044 - accuracy: 0.9673 - val_loss: 0.0480 - val_accuracy: 0.9849
     Epoch 13/15
     938/938 [==============================] - 5s 5ms/step - loss: 0.1018 - accuracy: 0.9681 - val_loss: 0.0474 - val_accuracy: 0.9857
      Epoch 13/15
      938/938 [==============================] - 5s 5ms/step - loss: 0.1018 - accuracy: 0.9681 - val_loss: 0.0474 - val_accuracy: 0.9857
      Epoch 14/15
      938/938 [==============================] - 5s 5ms/step - loss: 0.1011 - accuracy: 0.9683 - val_loss: 0.0461 - val_accuracy: 0.9853
      Epoch 15/15
      938/938 [==============================] - 5s 5ms/step - loss: 0.1003 - accuracy: 0.9684 - val_loss: 0.0523 - val_accuracy: 0.9829
```

```
[ ]
```

```
[30] print(history.params)

     {'verbose': 1, 'epochs': 15, 'steps': 938}
```

Bibliographie

[1] Cisco., «Global 2021 Forcast Highlights».

[2] P. T. E. D. S. J. K. Guto Leoni Santos, «When 5G Meets Deep Learning: A Systematic Review,» 2020.

[3] Z. Y. K. Z. P. C. K. Y. W. X. Kan Zheng, «Big data-driven optimization for mobile networks,» *IEEE network,* 2016.

[4] IEEE, «IEEE Network special issue: Exploring Deep Learning for Efficient».

[5] P. P. H. h. Chaoyun Zhang, «Deep learning in Mobile and Wireless Networking: A survey,» *IEEE Communications Surveys & Tutorials,* 2019.

[6] H. L. W. L. M. Morocho-Cayamcela, «Machine Learning for 5G/B5G Mobile and Wireless Communications: Potential,Limitations,andFuture Directions,» *IEEE Access,* vol. vol. 7, pp. pp. 137184-137206,2019.

[7] A. M. E. O. A. L. P. Ez-Zazi I, «A Hybrid Adaptive Coding and Decoding Scheme for Multi-hop Wireless Sensor Networks,» *Wireless Personal Communications,* 2016.

[8] L. X. Y. N. W. A. Jiang L, «Deep Learning-Aided Constellation Design for Downlink,» chez *15th International Wireless Communications & Mobile Computing*, Tangier, Morocco, 2019.

[9] K. I. C. C. Kang J.M, «Deep Learning-Based MIMO-NOMA With Imperfect SIC Decoding,» *IEEE Syst. J,* 2019.

[10] K. N. L. W. C. D. Kim M, «Deep learning-aided SCMA,» *IEEE Commun. Lett.,* 2018.

[11] M. Y. Y. N. T. R. Xue S, «Unsupervised deep learning for MU-SIMO joint transmitter and,» *IEEE Wirel. Commun. Lett.,* 2018.

[12] L. P. W. T. V. B. D. T. F. Santos J, «Anomaly detection for smart city applications,» chez *the 2018 IEEE/IFIP Network Operations and*, Taipei, Taiwan.

[13] L. Maimó, Á. Gómez, F. Clemente, M. Pérez et G. Pérez, «self-adaptive deep learning-based,» *IEEE Access,* 2018.

[14] B. Hussain, Q. Du, S. Zhang, A. Imran et M. Imran, «Mobile Edge Computing-Based Data-Driven Deep Learning Framework for Anomaly Detection,» *IEEE Access ,* 2019.

[15] R. Li, Z. Zhao, J. Zheng, C. Mei, Y. Cai et H. Zhang, «The learning and prediction of application-level traffic data in cellular networks,» *IEEE Trans. Wirel. Commun,* 2017.

[16] Q. Guo, R. Gu, Z. Wang, T. Zhao, Y. Ji, J. Kong, R. Gour et J. Jue, «Proactive Dynamic Network Slicing with Deep Learning Based Short-Term Traffic Prediction for 5G Transport Network,» chez *2019 Optical Fiber Communications Conference and Exhibition (OFC)*, 2019.

[17] D. Bega, M. Gramaglia, M. Fiore, A. Banchs et C.-P. X, «DeepCog: Cognitive network management in sliced 5G networks with deep learning,» chez *IEEE INFOCOM 2019—IEEE Conference on Computer Communications*, Paris, France, 2019.

[18] C. Huang, C. Chiang et Q. Li, «study of deep learning networks on mobile traffic forecasting,» chez *IEEE 28th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, Montreal, QC, Canada, 2017.

[19] Y. B. a. G. H. Yann LeCun, «Deep learning,» *Nature,* 2015.

[20] D. N. S. L. H.-P. T. Z. H. Mohammad Abu Alsheikh, «Mobile big data analytics using deep learning and Apache Spark,» *IEEE network,* 2016.

[21] C. Francois, Deep Learning with Python, 2017.

[22] M. R. K. M. A. ZACCONE Giancarlo, Deep Learning with Tensorflow, 2017.

[23] J.-G. GANASCIA, «Universalis.fr,» [En ligne]. Available: https://www.universalis.fr/encyclopedie/apprentissage-profond-deep-learning/2-reseaux-de-neurones-formels/. [Accès le 2021].

[24] M. Lefkowitz, "Cornell Chronicle," 2019. [Online]. Available: https://news.cornell.edu/stories/2019/09/professors-perceptron-paved-way-ai-60-years-too-soon.

[25] H. J. Kelley, «Gradient Theory of Optimal Flight Paths,» *American Rocet Society Journal,* 1960.

[26] S. J. Farlow, «The GMDH Algorithm of Ivakhnenko,» *The American Statistician,* pp. 210-215, 1981.

[27] S. P. Marvin Minsky, Perceptrons: an introduction to computational geometry, 1969.

[28] Journal Du Net, «AI winter : qu'est-ce que l'hiver de l'intelligence artificielle ?,» Mai 2021. [En ligne]. Available: https://www.journaldunet.fr/web-tech/guide-de-l-intelligence-artificielle/1501295-ai-winter-qu-est-ce-que-l-hiver-de-l-intelligence-artificielle/.

[29] The Franklin Institute, «KUNIHIKO FUKUSHIMA,» Mars 2020. [En ligne]. Available: https://www.fi.edu/laureates/kunihiko-fukushima. [Accès le Juillet 2021].

[30] J. J. HOPFIELD, «Neural networks and physical systems with emergent collective,» *Proceedings of the National Academy of Sciences,* 1982.

[31] P. Werbos, *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Science,* Cambridge, Massachusetts, 1974.

[32] G. H. T. S. David H. Ackley, *Boltzmann Machines Constraint Satisfaction Networks That Learn,* 1984.

[33] T. J. S. a. C. R. Rosenberg, «NETtalk: a parallel network that learns to read aloud,» The Johns Hopkins University Electrical Engineering and Computer Science, 1986.

[34] G. E. H. &. R. J. W. David E. Rumelhart, «Learning representations by back-propagating errors,» *Nature,* 1986.

[35] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard et L. D. Jackel, «Backpropagation Applied to Handwritten Zip Code Recognition,» *Neural Computation.*

[36] G. Cybenkot, «Approximation by Superpositions of a Sigmoidal Function,» *Mathematics of Control, Signals, and Systems,* 1989.

[37] J. S. Sepp Hochreiter, «Long Short-term Memory,» *Neural Computation,* 1997.

[38] A. M. A. Y. N. Rajat Raina, «Large-scale Deep Unsupervised Learning using Graphics Processors,» Computer Science Department, Stanford University, 2008.

[39 J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li et L. Fei-Fei, «ImageNet: A large-scale
] hierarchical image database,» chez *IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

[40 J. Wei, «AlexNet: The Architecture that Challenged CNNs,» towards data science, Juillet
] 2019. [En ligne]. Available: https://towardsdatascience.com/alexnet-the-architecture-that-challenged-cnns-e406d5297951. [Accès le Juillet 2021].

[41 J. P.-A. M. M. B. X. D. W.-F. S. O. A. C. Y. B. Ian J. Goodfellow, *Generative Adversarial
] *Networks,* Departement d'informatique et de recherche opérationnelle, 2014.

[42 France24, «Le champion du monde du jeu de Go encore battu par un ordinateur,» Mars
] 2016. [En ligne]. Available: https://www.france24.com/fr/20160312-champion-monde-jeu-go-encore-battu-ordinateur-troisieme-defaite-coree-sud-google. [Accès le Juillet 2021].

[43 C. Bichler, «Prix Turing : le Français Yann LeCun couronné avec deux autres pionniers de
] l'intelligence artificielle,» France Culture, Mars 2019. [En ligne]. Available: https://www.franceculture.fr/sciences/le-prix-turing-couronne-trois-pionniers-de-lintelligence-artificielle. [Accès le Juillet 2021].

[44 D. S. G. S. P. R. Z. Ivan Vasilev, Python Deep Learning, 2019.
]

[45 H. M. W. M. W. P.] J. Lettvin, «What the frog's Eye Tells The Frig's Brain,» *Proceedings
] of the IRE,* 1959.

[46 N. Buduma, Fundamentals of Deep Learning Designing Next-Generation Machine
] Intelligence Algorithms, 2017.

[47 G. S. D. S. P. R. Valentino Zocca, Python Deep Learning, 2017.
]

[48 A. B. a. Y. B. X. Glorot, «Deep Learning rectifier neural networks,» chez *In Proceedings of
] the fourteenth international conference of artificial intelligence and statistics*, 2011.

[49 P. Marc, Réseaux de neurones, 2004.
]

[50 N. Pat, Neural networks and deep learning : deep learning explained to your granny,
] 2018.

[51 T. Claude, Les réseaux de neurones artificiels, introduction au connexionnisme, 1992.
]

[52 B. R. Reza Bosagh Zadeh, TensorFlow for Deep Learning, 2017.
]

[53 «Recurrent neural networks tutorial part 1: introducton to to RNNs,» wildml, [En ligne].
] Available: http ://www.wildml.com/2015/09/recurrent-neural-networks-tutorial-part-1-introduction-to-rnns/.

[54 P. Marc, Réseaux de neurones, 2004.
]

[55 h. Hilali, *Application de la classification textuelle pour l'extraction des règles*
] *d'association maximales,* Québec, Trois-rivières: Université Québec à Trois-rivières, 2009.

[56 l. b. k. a. l. a. Amira, *Identification et commande des systèmes non linéaires,* 2011.
]

[57 a. Daria Amodei, «Deep speech : End-to-End recognition in English and Mandarin,» chez
] *Proceedings of 33rd International Conference on Machine Learning*, New York, 2016.

[58 a. A. WAIBEL, «Phoneme recognition using time-delay neural networks,» *IEEE*
] *Transactions on acoustics, Speech and signal processing,* 1989.

[59 J. W. Ronan COLLOBERT, «A unified architecture for natural language processing : Deep
] neural networks with multitask learning,» chez *Proceedings of the 25th international conference on machine learning*, 2008.

[60 A. V. A. Z. Omkar M.Parkhi, «Deep Face Recognition,» chez *proceedings of the British*
] *Machine Vision*, 2015.

[61 X. Z. S. R. J. S. Kaiming He, «Deep Residual Learning for Image Recognition,» 2015.
]

[62 C. Szegedy, «Going deeper with Convolutions,» chez *Ieee conference on computer vision*
] *and pattern recognition*, 2015.

[63 [En ligne]. Available: http://www.psychomedia.qc.ca/lexique/definition/apprentissage-
] profond.

[64] G. D. Serge Abiteboul, Le temps des algorithmes, 2017.

[65] «evaluer un modle en apprentissage automatique,» microsoft, [En ligne]. Available: https://msdn.microsoft.com/big_data_france/2014/06/17/evaluer-un-modle-en-apprentissageautomatique/.

[66] M. S. Fukushima Kunihiko, «Neocognitron : a self-organizing neural network model for mechanism of visual pattern recognition,» *NHK Broadcasting Science Research Laboratories,* 1981.

[67] M. M. Zakaria, *Classification d'image avec les réseaux de neurones convolutionnels, mémoire de projet de fin d'étude.*

[68] «convolutional neural networks,» Super data science, [En ligne]. Available: https://www.superdatascience.com/blogs/convolutional-neural-networks-cnn-step-1b-relu-layer/.

[69] N. R. O'SHEA Keiron, «An introduction to convolutional neural networks,» 2015.

[70] X. J. X. B. L. C. Z. H. W. F. H. H. Wang Peng, «Semantic clustering and convolutional neural networks for short text categorization,» 2015.

[71] «Convolutional neural networks tutorial tensorflow,» [En ligne]. Available: http://adventuresinmachinelearning.com/convolutional-neural-networks-tutorial-tensorflow/Consultez_le_2/aout/2018.

[72] «Convolutional neural network,» Towards data science, [En ligne]. Available: https://towardsdatascience.com/covolutional-neural-networkcb0883dd6529?fbclid=IwAR0UWoPkFYTEAqFitcR4fuuQUcNBvxV8ig1oJGZ3EhbRypu8Qf9pk9CXdy4.

[73] «neural network zoo,» [En ligne]. Available: http://www.asimovinstitute.org/neural-network-zoo/.

[74] M. Mignotte, *Traitement d'image : Introduction.*

[75] L. Laouamer, *Approche Exploratoire Sur La Classification Appliquée Aux Images,* Québec, 2006.

[76] [En ligne]. Available: https://www.python.org/.

[77] Keras.io, [En ligne]. Available: http ://KERAS.io.

[78] [En ligne]. Available: https ://www.linkedin.com/in/fchollet.

[79] [En ligne]. Available: https ://www.tensorflow.org/.

[80] [En ligne]. Available: https ://www.anaconda.com/distribution/.

[81] S. Hochreiter, «The vanishing Gradient problem during learning Recurrent Neural Nets problem solutions,» *International Journal of Uncertainty, Fuzziness and Knowledge-ed Systems,* 1998.

[82] S. M. D. J. R. M. W. Diederik P Kingma, «Semi-supervised learning with deep generative models,» chez *Advances in Neural Information Processing Systems*, 2014.

[83] «Réseaux de neurones : avantages et possibilités,» Wikiversité, mai 2016. [En ligne]. Available: https://fr.wikiversity.org/wiki/r%c3%a9seaux_de_neurones/avantages_et_possibilit%c3%a9s.

[84] Cunn. Y. Le, Quand la machine apprend, la révolution des neurones artificiels et de l'apprentissage profond, Odile Jacob, 2019.

[85] S. B. N. D. L. a. C. M. Petko Georgiev, «Low-resource multi-task audio sensing for mobile and embedded devices via shared deep neural network representations,» *Proceedings ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies,* 2017.

[86] hgfrd, «hhkjhjkh,» nb;nk,lm, 2012. [En ligne]. [Accès le 2015].

[87] J. P.-A. M. M. B. X. D. W.-F. S. O. A. C. a. Y. B. Ian Goodfellow, «Generative adversarial nets,» chez *Advances in neural information processing systems*, 2014.

[88] L. Maimó, A. Celdrán, M. Pérez, F. Clemente et G. Pérez, «Dynamic management of a,» *J. Ambient Intell. Humaniz. Comput,* 2019.

[89] A. B. a. Y. B. X. Glorot, «Deep Sparse Rectifier Neural Networks,» chez *In Proceedings of the Fourteenth International Conference on Artificial Intelligence a Statistics*, 2011.

[90] M. Parwez, D. Rawat et M. Garuba, «Big data analytics for user-activity analysis and user-anomaly,» *IEEE Trans. Ind. Inform,* 2017.

[91] O. Campesato, Artificial Intelligence, Machine Learning and Deep Learning, MERCURY LEARNING AND INFORMATION, 2020.

[92] Y. Ouyang, Z. Li, L. Su, W. Lu et Z. Lin, «APP-SON: Application characteristics-driven SON to optimize 4G/5G network performance and quality of experience,» chez *IEEE International Conference on Big Data (Big Data)*, Boston, MA, USA, 2017.

[93] Y. Ouyang, Z. Li, L. Su, W. Lu et Z. Lin, «Application behaviors Driven Self-Organizing Network (SON) for 4G LTE networks,» *IEEE Trans. Netw. Sci. Eng,* 2018.

[94] S. Saha, «A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way,» towards data science, Decembre 2018. [En ligne]. Available: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53.