République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université Akli Mohand Oulhadj de Bouira

Faculté des Sciences et des Sciences Appliquées

Département d'Informatique

Mémoire de Master

En Informatique

***Spécialité** : Génie Système Informatique*

Thème

Classification d'objets avec le Deep Learning

Encadré par

- OUAHABI Abdeldjlil

Réalisé par

- OULMI Mehdi
- KALOUNE Salim

2017/2018

Remerciements

Nous remercions tout d'abord Dieu le tout puissant de nous avoir donné la volonté et la persévérance pour réaliser ce travail.

Nous tenons à saisir cette occasion et adresser nos profonds remerciements et nos profondes reconnaissances à ABDELDJALIL OUAHABI, notre encadrant de mémoire, pour ses précieux conseils et son orientation ficelée tout au long de notre recherche.

Nos vifs remerciements vont également aux membres du jury pour l'intérêt qu'ils ont porté à notre recherche en acceptant d'examiner notre travail Et de l'enrichir par leurs propositions.

Nous souhaitons adresser nos remerciements les plus sincères aux personnes qui nous ont apporté leur aide et qui ont contribué à l'élaboration de ce mémoire ainsi qu'à la réussite de cette formidable année universitaire.

Nous remercions tous nos enseignants pour leur dévouement, leur patience et leur contribution à notre formation.

Enfin, nous adressons nos plus sincères remerciements à tous nos proches et amis, qui nous ont toujours encouragés au cours de la réalisation de ce mémoire.

Merci à tous et à toutes.

Dédicaces

Je dédie ce travail à mes parents qui ont fait en sorte que l'amour du savoir soit un véritable crédo pour leurs enfants.

A ma sœur Manel et mon frère Nabil qui m'ont encouragé sans cesse et cru en moi.

A ma fiancée Katia qui a été toujours près de moi et m'a soutenu tout au long de mon projet, que Dieu te garde pour moi mon amour.

Je tiens à remercier particulièrement Mr.Zerrouki pour ses énormes qualités d'homme de sciences, cela suscite respect et admiration.

A mon très cher cousin Lotfi que la mort a arraché au printemps de sa vie (que Dieu lui accorde le jardin d'Eden).

Je dédie aussi ce travail à mes défunts grands-parents (que Dieu leur accorde son vaste paradis).

OULMI Mehdi.

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.

Mes frères et sœurs qui n'ont cessé d'être pour moi des exemples de persévérance, de courage et de générosité.

Mes professeurs de département informatique qui doivent voir dans ce travail la fierté d'un savoir bien acquis.

KALOUNE Salim

Table des matières

Liste des figures

Liste des tableaux

Liste des abréviations

IA	**I**ntelligence **A**rtificielle
SVM	**S**upport **V**ector **M**achine
DNN	**D**eep **N**eural **N**etwork
SGD	**S**tochastique **G**radient **D**escente
CNN	**C**onvolutional **N**eural **N**etwork
FC DNN	**F**ully-**C**onnected **D**eep **N**eural **N**etwork
HTM	**H**ierarchical **T**emporal **M**emory
RBM	**R**estricted **B**oltzmann **M**achines
DBN	**D**eep **B**elief **N**etworks
RNN	**R**ecurrent **N**eural **N**etwork
LSTM	**L**ong **S**hort-**T**erm **M**emory
MNIST	**M**ixed **N**ational **I**nstitute of **S**tandards and **T**echnology
NORB	**N**yu **O**bject **R**ecognition **B**enchmark
CIFAR	**C**anadian **I**nstitute **F**or **A**dvanced **R**esearch
RELU	**RE**ctified **L**inear **U**nit
ANN	**A**rtificial **N**eural **N**etwork
CPU	**C**entral **P**rocessing **U**nit
GPU	**G**raphics **P**rocessing **U**nit
TPU	**T**ensor **P**rocessing **U**nit
CUDA	**C**ompute **U**nified **D**evice **A**rchitecture
CNTK	**M**icrosoft **C**ognitive **T**ool**K**it
API	**A**pplication **P**rogramming **I**nterface
SDK	**S**oftware **D**evelopment **K**it
GPS	**G**lobal **P**ositioning **S**ystem

Introduction générale

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

Les robots, les moteurs de recherche, les voitures autonomes, les agences de surveillance et bien d'autres ont des applications qui incluent la classification d'objets. On peut dire que cette dernière est un sous-champ de la vision par ordinateur, qui repose actuellement sur l'apprentissage automatique. Au cours de la dernière décennie, le domaine de cet apprentissage a été dominé par les réseaux de neurones profonds, qui tirent partie 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), convient bien aux tâches liées à l'image.

Le réseau est formé pour 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 projet de master, nous passerons en revue, des notions théoriques en commençant par le perceptron, s'étendant aux réseaux de neurones et enfin aux réseaux convolutifs.

Après nous allons implémenter différents modèles de réseaux neurones convolutifs avec différentes architectures et par la suite, on va appliquer ces modèles sur un jeu de données en modifiant à chaque fois différents hyper-paramètres comme par exemple le nombre d'époques et on évaluera au fur et à mesure les résultats de nos tests.

- On utilisera un jeu de données qui est couramment utilisé pour former des algorithmes d'apprentissage automatique et de vision artificielle.
- On fera appel à plusieurs bibliothèques pour faciliter l'implémentation et accélérer le processus d'apprentissage.
- On essaiera aussi de créer une petite application qui reçoit une image et des k classes possibles. La tâche consiste à décider à quelle classe appartient l'image. Par exemple, une image provenant d'une caméra embarquée dans une voiture autonome, contient soit une route pavée, soit une route non pavée, soit aucune d'elles : Laquelle de ces trois classes est dans l'image ?

Chapitre 1

1. Intelligence artificielle, machine learning et deep learning

1.1. Introduction

En premier lieu, nous devons définir clairement de quoi nous parlons lorsque nous mentionnons L'IA.

Qu'est ce que l'intelligence artificielle, l'apprentissage automatique et l'apprentissage en profondeur (voir la figure)

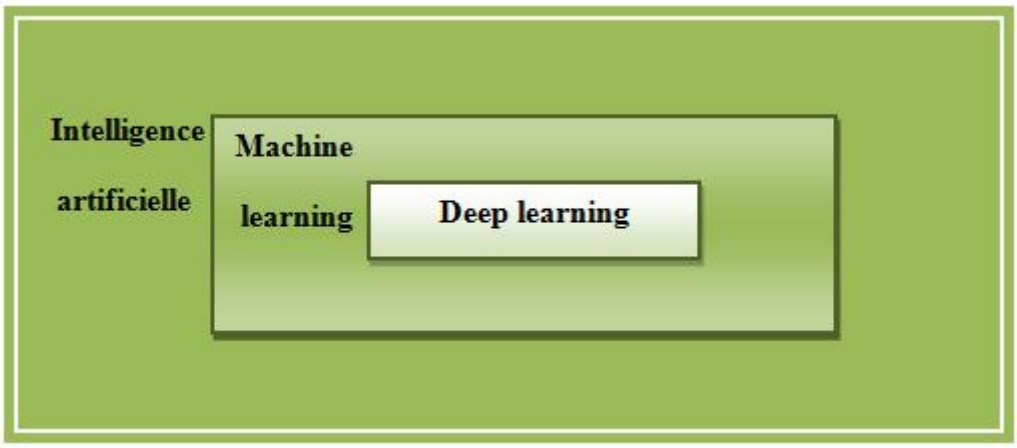

Figure 1- 1 Intelligence artificielle, machine learning et deep learning

1.2. Intelligence artificielle

L'intelligence artificielle est née dans les années 50, quand une poignée de pionniers du domaine naissant de l'informatique, ont commencé à se demander si les ordinateurs pouvaient être amenés à « penser », une question dont nous explorons encore aujourd'hui les ramifications. Une définition concise du champ serait la suivante : l'effort d'automatiser les

tâches intellectuelles normalement effectuées par les humains. En tant que tel, l'IA est un domaine général qui englobe l'apprentissage automatique et l'apprentissage en profondeur, mais qui comprend également beaucoup plus d'approches qui n'impliquent aucun apprentissage. Les programmes d'échecs initiaux, par exemple, ne concernaient que des règles codées en dur élaborées par des programmeurs et ne se qualifiaient pas comme apprentissage automatique. Pendant un temps assez long, de nombreux experts ont estimé que l'intelligence artificielle au niveau humain, pouvait être obtenue en faisant en sorte que les programmeurs fabriquent à la main un ensemble suffisamment large de règles explicites pour manipuler les connaissances. Cette approche est connue sous le nom d'IA symbolique et elle était le paradigme dominant de l'IA des années 50 et à la fin des années 80. Elle a atteint son pic de popularité durant le boom des systèmes experts des années 80. Bien que l'IA symbolique se soit révélée appropriée pour résoudre des problèmes logiques bien définis, comme jouer aux échecs, il était difficile de trouver des règles explicites pour résoudre des problèmes flous plus complexes, tels que la classification des images, la reconnaissance de la parole et la traduction. Une nouvelle approche est apparue pour prendre la place de l'IA symbolique : l'apprentissage automatique. [1]

1.3. Machine learning

L'apprentissage automatique (Machine learning) 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 domaine a souvent été associé à l'intelligence artificielle et plus spécifiquement l'intelligence computationnelle.

L'intelligence computationnelle est une méthode d'analyse de données qui pointe vers la création automatique de modèles analytiques. Autrement dit, permettant à un ordinateur d'élaborer des concepts, d'évaluer, prendre des décisions et prévoir les options futures. [2]

L'ensemble du processus d'apprentissage nécessite un ensemble de données comme suit :

- Ensemble de données pour l'entrainement : c'est la base de connaissance utilisée pour entrainer, notre l'algorithme d'apprentissage, pendant cette phase, les paramètres du modèle peuvent être réglés (ajustés) en fonction des performances obtenues.
- Ensemble de données pour le test : cela est utilisé juste pour évaluer les performances du modèle sur les données non-vues.

La théorie de l'apprentissage utilise des outils mathématiques dérivés de la théorie des probabilités et de la théorie de l'information.

Cela vous permet d'évaluer l'optimalité de certaines méthodes par rapport aux autres.

On peut citer trois types d'algorithme d'apprentissage automatique :

- Apprentissage supervisé.
- Apprentissage non supervisé.
- Apprentissage par renforcement.

1.3.1. Apprentissage supervisé

L'apprentissage supervisé est la tache 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.
Dans ce cas, la question cruciale est le problème de généralisation, après l'analyse d'un échantillon d'exemples, le système devrait produire un modèle qui devrait fonctionner pour toutes les entrées possibles.
L'ensemble de données pour l'entrainement, est constitué de données étiquetées, c'est-à-dire d'objets et de leurs classes associées. Cet ensemble d'exemples étiquetés constitue donc l'ensemble d'apprentissage.
Afin de mieux comprendre ce concept, prenons un exemple : un utilisateur reçoit chaque jour un grand nombre d'e-mails, certains sont des e-mails d'entreprises importants et d'autres sont des e-mails indésirables non sollicités ou des spam.

Un algorithme supervisé sera présenté avec un grand nombre d'e-mails qui ont déjà été étiquetés par l'utilisateur comme spam ou non spam. L'algorithme fonctionnera sur toutes les données étiquetées, faire des prédictions sur l'e-mail et voir si c'est un spam ou non.

Cela signifie que l'algorithme examinera chaque exemple et fera une prédiction pour chacun pour savoir si l'e-mail est un spam ou pas. La première fois, l'algorithme fonctionne sur toutes les données non étiquetées, la plupart des e-mails seront mal étiquetés car il peut fonctionner assez mal au début. Cependant, après chaque exécution, l'algorithme compare sa prédiction au résultat souhaité (l'étiquette). Au fur et à mesure, l'algorithme apprendra à améliorer ses performances et sa précision.

Dans l'exemple que nous avons utilisé, nous avons décrit un processus dans lequel un algorithme apprend à partir de données étiquetées (emails qui ont été catégorisés comme spam ou non-spam).

Dans certains cas, le résultat n'est pas nécessairement discret et il se peut que nous n'ayons pas un nombre fini de classes dans lesquelles classer nos données. Par exemple, nous essayons peut-être de prédire l'espérance de vie d'un groupe de personnes en fonction de paramètres de santé préétablis. Dans ce cas, comme le résultat est une fonction continue (nous pouvons spécifier une espérance de vie comme un nombre réel exprimant le nombre d'années que la personne devrait vivre), nous ne parlons pas d'une tâche de classification mais plutôt d'un problème de régression.

Dans un problème de régression, l'ensemble d'apprentissage est une paire formée par un objet et une valeur numérique associée. Il existe plusieurs algorithmes d'apprentissage supervisé qui ont été développés pour la classification et la régression. Parmi tous, les arbres de décision, les règles de décision, les réseaux de neurones et les réseaux bayé siens. [2] [3] [4]

1.3.2. Apprentissage non supervisé

La deuxième classe d'algorithmes d'apprentissage automatique est appelée apprentissage non supervisé, dans ce cas, nous n'étiquetons pas les données au préalable, nous laissons plutôt l'algorithme arriver à sa conclusion.

Ce type d'apprentissage est important car il est beaucoup plus commun dans le cerveau humain que l'apprentissage supervisé.

Les algorithmes d'apprentissage non supervisé sont particulièrement utilisés dans les problèmes de clustering, dans lesquels, étant donné une collection d'objets, nous voulons être en mesure de comprendre et de montrer leurs relations. Une approche standard consiste à définir une mesure de similarité entre deux objets, puis à rechercher tout groupe d'objets plus similaires les uns aux autres, par rapport aux objets des autres clusters. Par exemple, dans le cas précédent des e-mails spam/ non spam, l'algorithme peut être capable de trouver des éléments communs à tous les spam (par exemple, la présence de mots mal orthographiés). Bien que cela puisse fournir une classification meilleure qu'aléatoire, il n'est pas clair que les spam/non spam puissent être facilement séparés. [2] [3] [4]

1.3.3. Apprentissage par renforcement

L'apprentissage par renforcement est une approche de l'intelligence artificielle qui met l'accent sur l'apprentissage du système à travers ses interactions avec l'environnement. Avec l'apprentissage par renforcement, le système adapte ses paramètres en fonction des réactions reçues de l'environnement, qui fournit ensuite un retour d'information sur les décisions prises. Par exemple, un système qui modélise un joueur d'échecs qui utilise le résultat des étapes précédentes pour améliorer ses performances, est un système qui apprend avec le renforcement. La recherche actuelle sur l'apprentissage avec renforcement est hautement interdisciplinaire et comprend des chercheurs spécialisés dans les algorithmes génétiques, les réseaux de neurones, la psychologie et les techniques de contrôle. [2] [4]

La figure suivante résume les trois types d'apprentissage avec les problèmes connexes à résoudre :

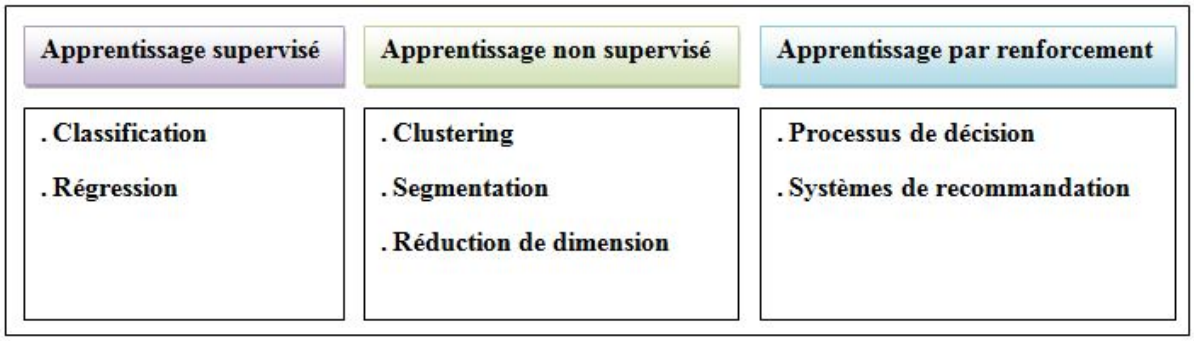

Figure 1- 2 Types d'apprentissage

1.4. Deep learning

L'apprentissage en profondeur (deep learning) est un domaine de recherche sur l'apprentissage automatique basé sur un type particulier de mécanisme d'apprentissage.

Il est caractérisé par l'effort de créer un modèle d'apprentissage à plusieurs niveaux, dans lequel les niveaux les plus profonds prennent en compte les résultats des niveaux précédents, les transformant et en faisant toujours plus d'abstraction. Cet aperçu des niveaux d'apprentissage est inspiré par la façon dont le cerveau traite l'information et apprend en réagissant aux stimuli externes. Chaque niveau d'apprentissage correspond, par hypothèse, à l'une des différentes zones qui composent le cortex cérébral. [2]

1.5. Brève description des algorithmes populaires

1.5.1. Random forest

Random Forest est un algorithme d'apprentissage supervisé. Comme vous pouvez déjà le voir à partir de son nom, il crée une forêt et la rend aléatoire. La «forêt» qu'il construit est un ensemble d'arbres de décision, la plupart du temps formés avec la méthode de «bagging». L'idée générale de la méthode bagging n'est qu'une combinaison de modèles d'apprentissage améliorant le résultat global.

Pour le dire en termes simples : La forêt aléatoire construit plusieurs arbres de décision et les fusionne pour obtenir une prédiction plus précise et plus stable.

Un grand avantage de la forêt aléatoire est qu'elle peut être utilisée pour les problèmes de classification et de régression, qui constituent la majorité des systèmes d'apprentissage automatique actuels. Ci-dessous, vous pouvez voir à quoi ressemble une forêt aléatoire avec deux arbres :

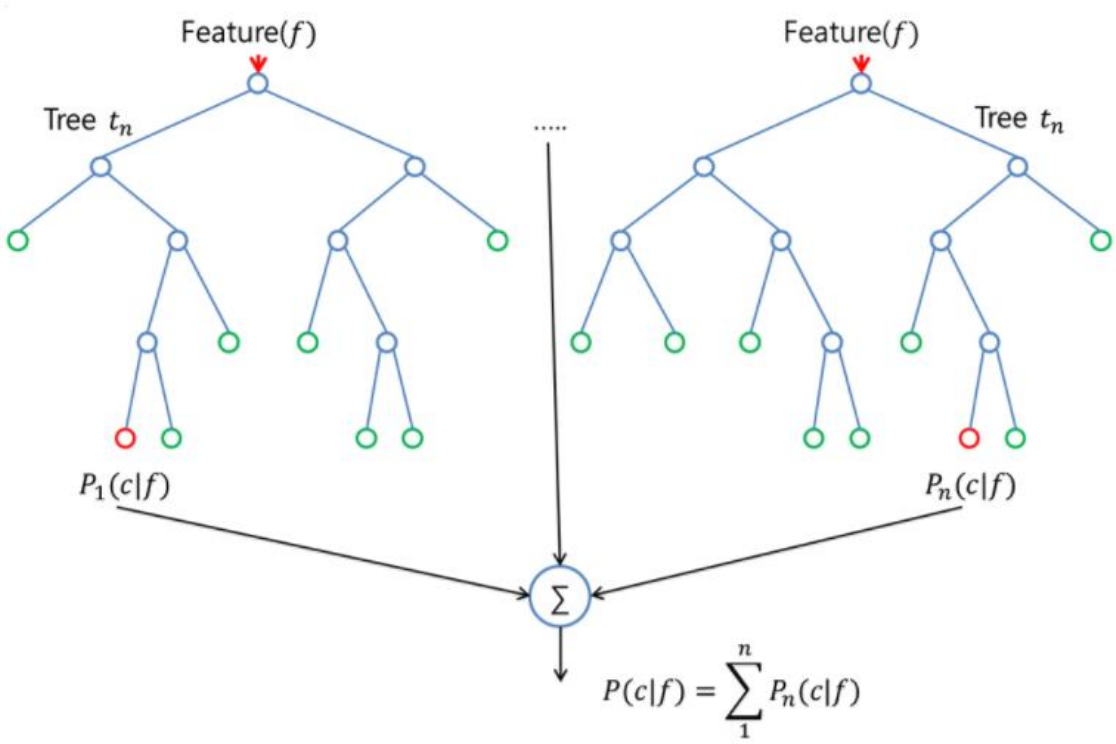

Figure 1- 3 Random forest

Quelques exceptions près, un classificateur à forêt aléatoire possède tous les hyper paramètres d'un classificateur d'arbres de décision et également tous les hyper paramètres d'un

classificateur d'ensachage, pour contrôler l'ensemble lui-même. Au lieu de construire un classificateur bagging et de le passer dans un classeur de décision-arbre, vous pouvez simplement utiliser la classe de classificateur random-forest qui est plus pratique et optimisée pour les arbres de décision. Notez qu'il existe également un régresseur de forêt aléatoire pour les tâches de régression.

L'algorithme de la forêt aléatoire apporte un caractère aléatoire supplémentaire dans le modèle, quand il fait pousser les arbres. Au lieu de rechercher la meilleure fonctionnalité lors de la division d'un nœud, il recherche la meilleure fonctionnalité parmi un sous-ensemble aléatoire de fonctionnalités. Ce processus crée une grande diversité, ce qui aboutit généralement à un meilleur modèle.

Par conséquent, lorsque vous créez un arbre dans une forêt aléatoire, seul un sous-ensemble aléatoire des entités est considéré pour diviser un nœud. Vous pouvez même rendre les arbres plus aléatoires, en utilisant des seuils aléatoires par-dessus pour chaque caractéristique plutôt que de chercher les meilleurs seuils possibles (comme un arbre de décision normal). [5]

1.5.2. SVM

La machine à vecteurs de support est un algorithme d'apprentissage automatique supervisé, principalement utilisé pour la classification. L'avantage de cet algorithme, par rapport aux autres algorithmes d'apprentissage automatique, est que non seulement il sépare les données en classes, mais il trouve aussi un hyper-plan de séparation (l'analogue d'un plan dans un espace avec plus de trois dimensions) qui maximise la marge séparant chaque point de l'hyper-plan. En outre, les machines vectorielles de support peuvent également traiter le cas des données qui ne sont pas linéairement séparables. Il y a deux façons de traiter ces données, l'une consiste à introduire des marges douces (soft margins) et l'autre consiste à introduire ce qu'on appelle l'astuce du noyau (kernel trick).

Les marges douces fonctionnent en permettant à quelques éléments classifiés de ne pas en tenir compte tout en conservant la capacité prédictive de l'algorithme.

La méthode du noyau contient ce que l'on appelle la fonction du noyau. Ces fonctions mappent l'espace d'entrée séparable non linéaire dans un espace de caractéristique séparable linéaire de plus grande dimension et dans ce nouvel espace linéaire séparable de dimension supérieure, les machines à vecteurs de support peuvent fonctionner normalement. La méthode

du noyau restitue ensuite les solutions, de sorte que dans l'espace d'entrée séparable non linéaire, vous disposez d'une solution non linéaire.

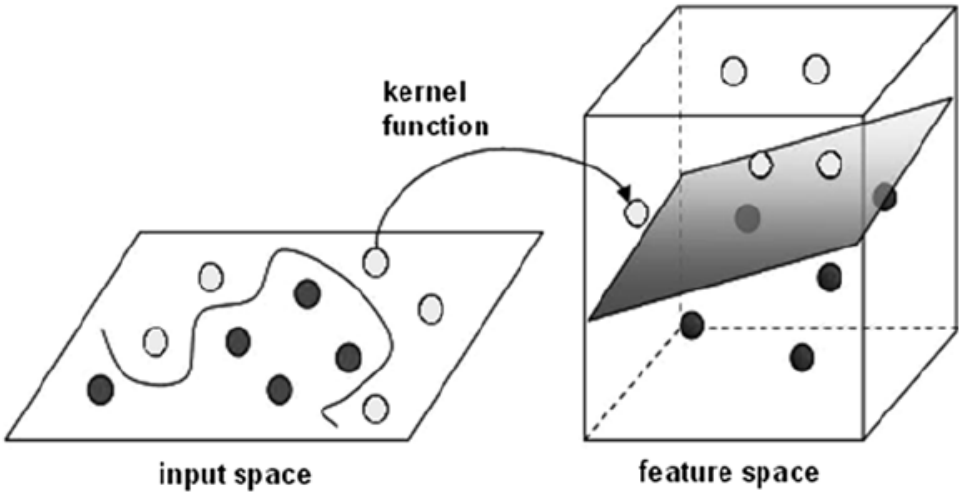

Figure 1- 4 SVM_karnel

Dans l'exemple ci-dessus, nous avons un espace de caractéristiques bi-dimensionnel non linéaire. Avec la fonction noyau, nous pouvons mapper l'espace d'entrée dans un espace à trois dimensions. Dans cet espace de fonctionnalités, nous pouvons alors séparer l'ensemble d'apprentissage linéaire. Lorsque nous restituons la solution à l'espace d'entrée, nous obtenons une solution non linéaire. [6]

1.5.3. Réseaux de neurones

Les réseaux de neurones sont un autre algorithme d'apprentissage automatique et ils ont connu des périodes de grande popularité et des périodes pendant lesquelles ils étaient rarement utilisés.

Le premier exemple d'un réseau neuronal a été appelé le perceptron, qui a été inventé par Frank Rosenblatt en 1957. Le perceptron est un réseau composé seulement d'une couche d'entrée et d'une couche de sortie. Dans le cas de classifications binaires, la couche de sortie n'a qu'un seul neurone ou unité. Le perceptron a semblé très prometteur dès le début, bien qu'il ait rapidement compris qu'il ne pouvait apprendre que des motifs linéairement séparables. Par exemple, Marvin Minsky et Seymour Papert ont montré qu'ils ne pouvaient pas apprendre la fonction logique XOR.

Dans ses représentations les plus basiques, les perceptrons ne sont que de simples représentations d'un neurone et son entrée (entrée qui peut être composée de plusieurs neurones). Etant donné les différentes entrées dans un neurone, nous définissons une valeur d'activation par la formule ($a(x) = \sum_i wi\ xi$) où xi est la valeur pour le neurone d'entrée, tandis que wi est la valeur de la connexion entre le neurone i et la sortie. Si la valeur d'activation, qui devrait être considérée comme l'état interne du neurone, est supérieure à un seuil fixe b, alors le neurone s'activera, c'est-à-dire qu'il se déclenchera, sinon ce ne sera pas le cas. Nous apprendrons ceci de façon plus détaillée dans le chapitre suivant, car nous devons juste remarquer que les perceptrons partagent de nombreuses similitudes avec les algorithmes de régression logistique et sont également contraints par des classificateurs linéaires. [6]

1.6. Conclusion

Nous avons consacré ce chapitre à la présentation des notions de base comme L'IA et son évolution, l'apprentissage automatique et ses types et on conclut avec les algorithmes les plus populaires et les plus utilisés.

Chapitre 2

2. Réseaux neurones artificiels

2.1. Introduction

Les réseaux neurones artificiels sont une nouvelle technologie informatique : Ce sont plusieurs processeurs parallèles, simples, fortement intégrés par un réseau de connexions qui créent un modèle distribué computationnel.

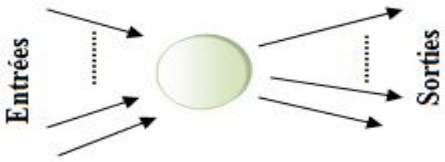

Figure 2- 1 Unité de traitement unique

Leur architecture était donc très novatrice dans les années 50, lorsqu'elles sont nées en parfaite analogie avec la structure du cerveau : de nombreux neurones fortement connectés par des synapses à travers lesquels les calculs se propagent en parallèle dans le cortex cérébral. Cela permet d'atteindre de nouvelles performances : trouver la solution en temps réel à des problèmes complexes, auto-apprentissage, résistance aux fautes et erreurs, etc. [7] [8]

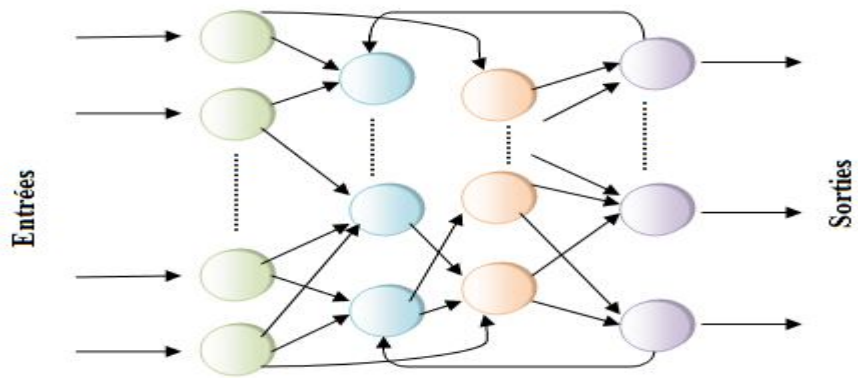

Figure 2- 2 Réseaux de neurones

2.2. Fonctionnement des réseaux de neurones

Un réseau neuronal se compose de nombreuses unités de traitement homogènes, fortement interconnectées par des liens d'intensité variable. L'activité de l'unité unique est simple et la puissance du modèle réside dans la configuration des connexions (topologie et intensité). A partir des unités d'entrée, auxquelles les données sont envoyées pour résoudre un problème, le calcul se propage en parallèle dans le réseau jusqu'aux unités de sortie, qui fournissent le résultat. Un réseau de neurones n'est pas programmé pour exécuter une certaine activité unique, mais entraîné (en utilisant un algorithme d'apprentissage automatique) au moyen d'une série d'exemples de la réalité à modéliser.

Dans les années 50, les réseaux neuronaux avaient une structure simple avec peu d'unités internes, alors que les réseaux actuels impliquent des millions d'unités, capables d'apprendre des modèles incroyablement complexes, même s'ils nécessitent des ordinateurs beaucoup plus puissants et des techniques de formation plus sophistiquées.

Si nous devons décrire le fonctionnement d'un réseau de neurones artificiel dans une phrase, nous pourrons dire qu'il prend en entrée des données, puis leur donne un sens, trouvant des régularités, ou des modèles.

Pendant que vous lisiez un livre, votre cerveau organise les lettres en mots, et les mots en phrases, jusqu'à ce que le sens émerge. De même, un réseau de neurones artificiels peut analyser vos commentaires sur un site Web à la recherche de mots significatifs et de sujets pertinents. Cette volonté de donner un sens à l'information est si profondément inhérente à notre cerveau qu'elle s'applique aussi à des schémas dénués de sens. Quand nous regardons les nuages, nous les associons spontanément à des images et des catégories réelles. Ce même mécanisme est présent dans les réseaux de neurones artificiels, ce qui les rend extrêmement fascinants et en même temps effrayants. [4] [7] [8]

2.3. Neurone de McCulloch-Pitts

Jusqu'à présent, nous savons que les réseaux de neurones artificiels sont calculatoires et sont des systèmes inspirés par les processus biologiques qui se produisent dans le cerveau humain :

- ils sont formés par des millions d'unités de calcul (appelés neurones) capables d'exécuter une somme méditée.
- ils ont un nombre élevé de connexions pesées (synapses) parmi les unités.

- ils sont hautement parallèles et non linéaires.
- ils sont adaptatifs, entraînables et l'apprentissage se fait en changeant les poids des connexions.
- ils sont tolérants aux erreurs car le stockage est répandu.
- il n'y a pas de distinction entre mémoire et zone de calcul.
- ils ont des compétences de généralisation : ils peuvent produire des sorties raisonnables avec des entrées qu'ils n'ont jamais rencontrées avant et pendant le processus d'apprentissage.

Entrons dans les détails de la façon dont une seule unité fonctionne (le Neuron de McCulloch-Pitts) :

Un neurone est l'unité fondamentale de calcul d'un réseau de neurones et il est composé de 3 éléments de base dans ce modèle :

1. un ensemble de synapses ou de connexions dont chacune est caractérisée par un poids (efficacité synaptique); Contrairement au modèle humain, le modèle artificiel peut avoir à la fois des poids négatifs et positifs.
2. un agent sommateur qui résume les signaux d'entrée pesés par les synapses respectives, produisant une combinaison linéaire des entrées.
3. une fonction d'activation pour limiter la largeur de la sortie d'un neurone.

Par commodité, la largeur des sorties appartient à l'intervalle [0,1] ou [-1,1].

Le modèle neuronal comprend également une valeur de seuil qui a pour effet, en fonction de sa positivité ou de sa négativité, d'augmenter ou de diminuer l'entrée nette dans la fonction d'activation. [7]

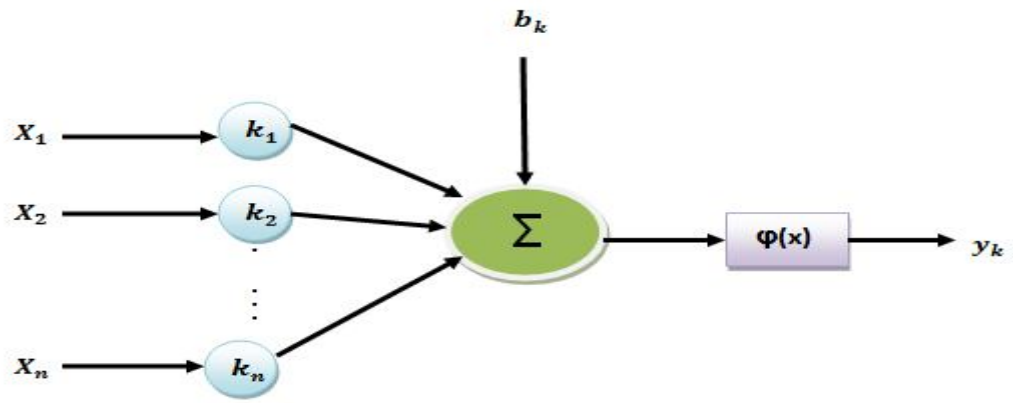

Figure 2- 3 Shema explicatif

En termes mathématiques, nous décrivons un k neurone avec les équations suivantes :

$$u_k = \sum_{j=1}^{m} w_{kj} \, . x_j$$

$$y_{\mathrm{k}} = \varphi(u_{\mathrm{k}} + b_k)$$

2.4. Types de fonction d'activation

Nous identifions 3 types de fonctions d'activation de base : fonctions de seuil ou fonctions Heaviside, fonctions linéaires par morceaux et fonctions sigmoïdes.

Les fonctions de seuil sont utilisées dans le modèle de neurones de McCulloch-Pitts.

$$\Phi(v) = \begin{cases} 1 \text{ si } v \geq 0 \\ 0 \text{ si } v < 0 \end{cases}$$

$$\Phi(v) = \begin{cases} 1 \text{ si } v \geq 2 \\ v \text{ si } -2 < v < 2 \\ 0 \text{ si } v \leq -2 \end{cases}$$

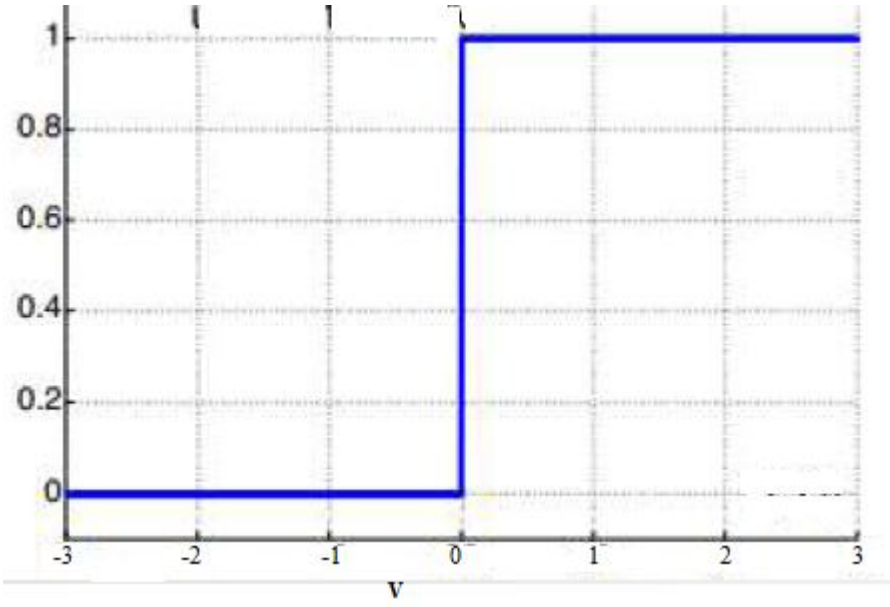

Figure 2- 4 Fonctions de seuil

Les fonctions linéaires par morceaux sont les suivantes :

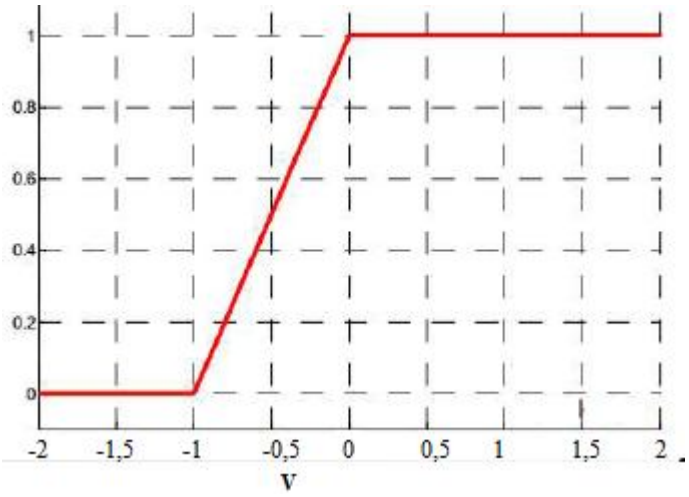

Figure 2- 5 Fonctions linéaires

Les fonctions sigmoïdes sont les fonctions les plus utilisées dans la création de réseaux neurones artificiels.

C'est une fonction strictement croissante qui présente un équilibre entre le comportement linéaire et non linéaire. Par exemple : $\varphi(v) = \frac{1}{1+e^{-av}}$ « a » est un paramètre qui indique la pente de la fonction. [4] [7] [8]

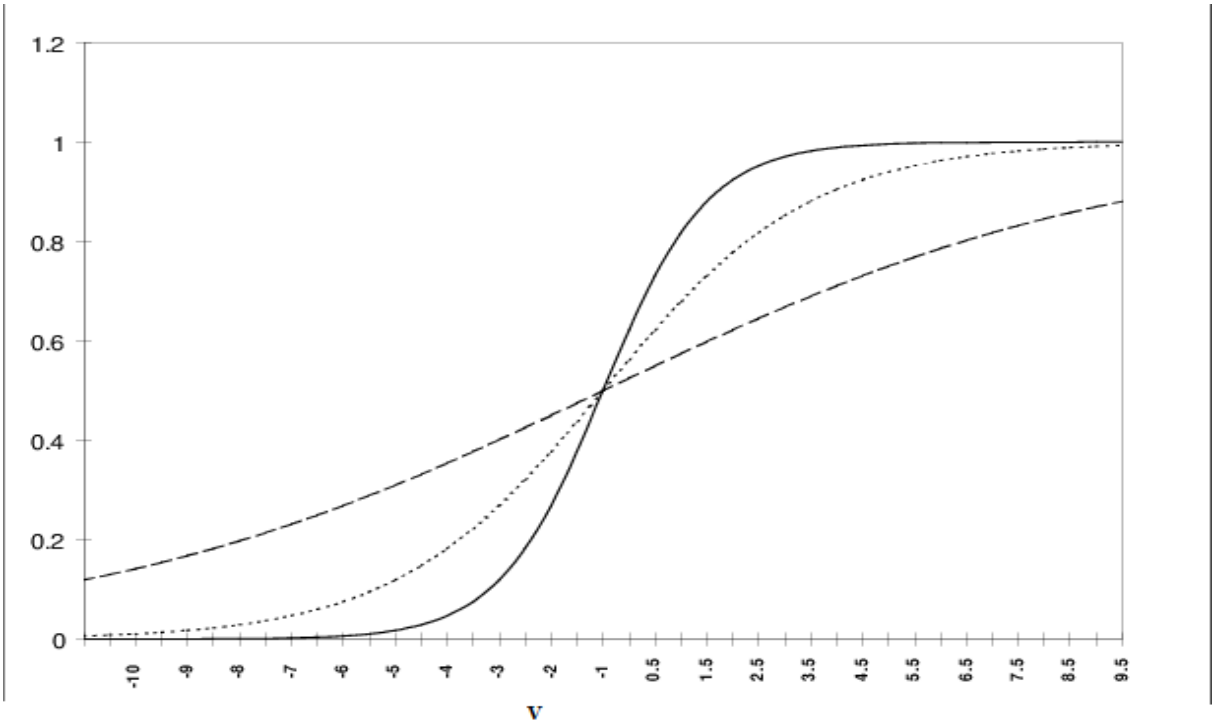

Figure 2- 6 Fonctions sigmoides

2.5. Types d'architectures de réseaux :

La structure d'un réseau dépend de l'algorithme d'apprentissage que vous avez l'intention d'utiliser. En général, nous pouvons identifier 3 classes de réseaux :

2.5.1. Réseaux à une couche (feed-forward) :

Dans cette forme simple de réseau en couches, nous avons des nœuds d'entrée et une couche de neurones (couche de sortie). Le signal se propage dans le réseau de manière linéaire, en commençant par la couche d'entrée et en se terminant par la couche de sortie. Il n'y a pas de connexions qui reviennent et pas de connexions transversales dans la couche de sortie. [7]

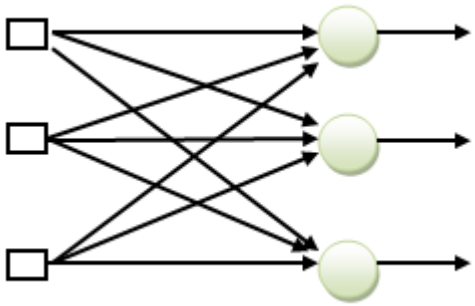

Figure 2- 7 Une couche

2.5.2. Réseaux multi-couches (feed-forward) :

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, donc la propagation du signal se fait de façon linéaire sans cycles et sans connexions transversales. Ce type d'architecture fournit au réseau une perspective globale car il augmente les interactions entre les neurones. [4] [7] [8]

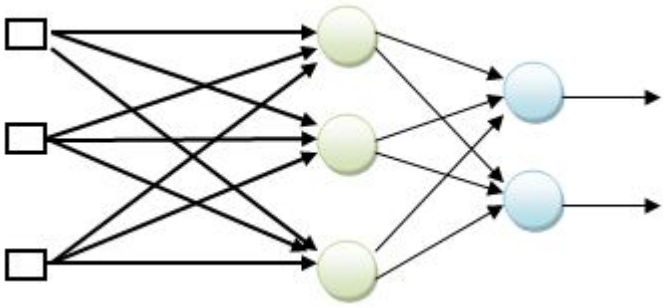

Figure 2- 8 Multi-couche

2.5.3. Réseaux récurrents (feed-back) :

Un réseau récurrent diffère des précédents par le fait qu'il est cyclique. La présence de cycles a un impact profond sur les capacités d'apprentissage du réseau et sur ses performances, notamment rendre le système dynamique. [4] [7]

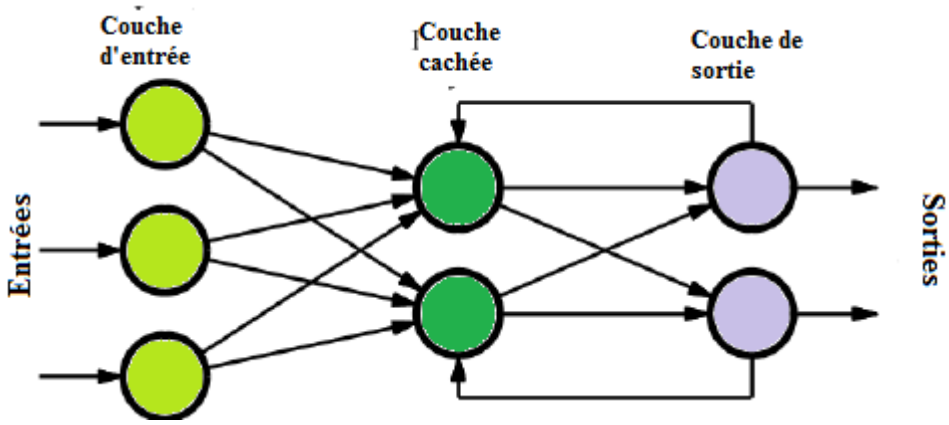

Figure 2- 9 Réseaux récurrents

2.6. Processus d'apprentissage

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 manière dont ces adaptations ont lieu.

Un algorithme d'apprentissage est un ensemble de règles bien définies qui résolvent un certain problème d'apprentissage. [4] [7]

Comme nous l'avons expliqué précédemment, les deux principaux types d'algorithmes d'apprentissage sont :

- Supervisé : il y a un «enseignant» qui connaît l'environnement et fournit des correspondances entrées / sorties correctes, tandis que le réseau devra ajuster ses paramètres libres : émuler l'enseignant d'une manière statistiquement optimale.
- Non supervisé : le réseau apprend indépendamment de toute intervention de «l'enseignant».

Voyons quelques types d'apprentissage existants.

2.6.1. Apprentissage avec correction d'erreur

Chaque neurone k reçoit une entrée du stimulus x (n) et génère une réponse yk (n), n étant un temps discret. Nous indiquons aussi avec dk (n) la réponse désirée. Par conséquent, un signal d'erreur ek (n) est généré.

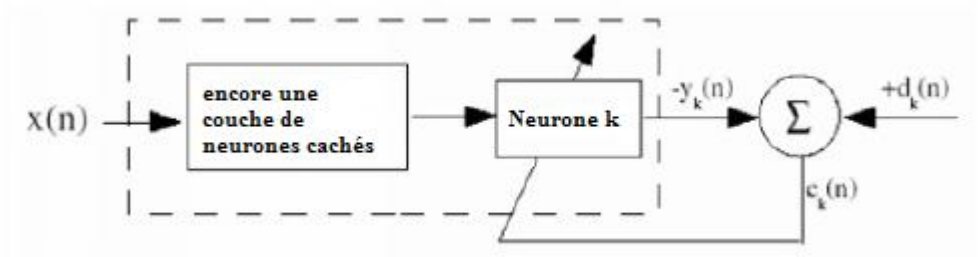

Figure 2- 10 Apprentissage avec correction d'erreur

Le signal d'erreur ek (n) met en œuvre un mécanisme de contrôle dans le but d'appliquer une séquence d'ajustements aux charges synaptiques du neurone k afin de rapprocher la réponse obtenue de celle souhaitée. [4] [7]

2.6.2. Apprentissage basé sur la mémoire

Dans l'apprentissage basé sur la mémoire, plusieurs ou toutes les expériences passées sont stockées dans une grande mémoire de paires entrée-sortie correctement classées.

Lorsque la classification d'un exemple n'est jamais atteinte avant que xtest ne soit demandé, le système répond en trouvant et en analysant les exemples stockés entourant xtest.

Toutes les méthodes d'apprentissage basées sur la mémoire impliquent 2 ingrédients de base :

- Le critère utilisé pour définir l'environnement d'un vecteur de test xtest.
- La méthode d'apprentissage appliquée aux exemples entourant xtest.

Un exemple de cette méthode est la méthode du plus proche voisin dans laquelle le plus proche de l'exemple test x'N est celui qui possède une distance euclidienne minimale.

$$x'_{\mathrm{N}} \epsilon\{x_1, x_2, \ldots\ldots x_N),\ \ min_t[\mathrm{d}(x_t, x_{test})] = \mathrm{d}(x'_N, x_{test})$$

Où d est la distance euclidienne. La classe affectée à xtest est la même que pour x'N. Une variante de cette méthode est le k-plus proche voisin dans lequel le voisinage de l'exemple de test n'est plus un, mais c'est l'ensemble des k exemples stockés les plus rapprochés. [4] [7]

La classe affectée est celle qui a la fréquence la plus élevée entourant l'exemple de test.

2.6.3. Hebbian apprentissage

Ceci est basé sur le postulat Hebb sur l'apprentissage, selon lequel lorsque l'axone d'un neurone A (sa ligne de transmission de sortie) est suffisamment proche pour exciter un neurone B et ceci, de façon répétitive et persistante, envoie un potentiel d'action, un processus de croissance commence dans un ou deux neurones, ce qui augmente l'efficacité de A. De cela, nous pouvons dériver deux règles :

- si 2 neurones reliés par une synapse sont activés simultanément, alors le poids de la synapse est progressivement augmenté.
- si 2 neurones connectés sont activés de manière asynchrone, alors le poids de la synapse est progressivement diminué ou éliminé.

Une synapse de ce type s'appelle la synapse hebbienne. Si la corrélation des signaux conduit à une augmentation de l'efficacité synaptique, nous appellerons cela une modification hebbienne, si elle conduit à une réduction, nous l'appellerons anti-hebbienne. [4] [7] [8]

2.6.4. Apprentissage compétitif

Avec l'apprentissage compétitif, les neurones d'un réseau se font concurrence pour devenir actifs. Seul un neurone peut être actif à un certain moment. Il y a 3 éléments de base dans une méthode d'apprentissage compétitif :

- un ensemble de neurones identiques, à moins que les poids synaptiques générés au hasard ne réagissent différemment à un ensemble donné d'entrée.
- une limite à la "force" de chaque neurone.
- un mécanisme qui permet aux neurones d'entrer en compétition pour obtenir le droit de répondre à un sous-ensemble donné d'entrées, de sorte qu'un seul neurone ou un seul neurone par groupe soit actif à un certain moment. Le neurone gagnant est appelé le gagnant-prend-tout ("winner-take-all").

De cette façon, les neurones ont tendance à se spécialiser sur un ensemble de modèles similaires et dans la reconnaissance des caractéristiques de différentes classes de modèles d'entrée. Dans la forme la plus simple, un réseau neuronal a seulement une couche de neurones de sortie, complètement connectée aux nœuds d'entrée (connexions excitatrices vers l'avant). Le réseau peut inclure des connexions entre des neurones qui conduisent à des inhibitions latérales (connexions de rétroaction inhibitrices). [4] [7]

2.7. Avantages et inconvénients

Nous avons vu un aperçu des caractéristiques de base d'un réseau de neurones et compris comment cela fonctionne essentiellement.

Nous allons donc souligner quels sont leurs principaux avantages et inconvénients dans le domaine de l'innovation.

Avantages de l'utilisation d'un RN sont :

- Ils conviennent aux problèmes qui ne nécessitent pas de réponses précises, mais des réponses approximatives avec un degré d'erreur ou de variation.
- Leur capacité à généraliser : ils peuvent produire de bonnes réponses même avec une contribution qui n'a pas été prise en compte lors de leur création et de leur entrainement.
- Ils sont faciles à implémenter, vous pouvez simplement définir un neurone, puis le copier et créer des connexions entre les neurones.
- Il fournit des opérations rapides car il fonctionne en parallèle, chaque neurone utilise seulement son entrée.
- La stabilité de la sortie par rapport aux valeurs d'entrée : les valeurs d'entrée peuvent être incomplètes, bruyantes, mal connues ou accepter un degré d'erreur ou de changement
- Ils peuvent déterminer le résultat en tenant compte de toutes les entrées en même temps.

Inconvénients de l'utilisation des réseaux neuronaux sont :

- Neural Network (RN) fonctionne comme une boîte noire : généralement, vous ne serez pas en mesure de comprendre pourquoi il a produit ce résultat spécifique.
- les connaissances mémorisées ne peuvent être décrites et localisées dans le réseau.
- ils impliquent souvent des techniques de formation sophistiquées qui prennent beaucoup de temps pour les calculs.
- il n'y a pas toujours un réseau capable de résoudre un problème spécifique, car il n'y a pas toujours un algorithme d'apprentissage qui converge en donnant une sortie réseau à faible erreur.
- les valeurs de sortie ne sont pas précises, mais ont une marge qui peut varier.

- nous avons besoin d'une très grande série d'exemples pour avoir un bon processus d'apprentissage et une faible erreur de rendement. [7]

2.8. Conclusion

Dans ce chapitre, nous avons expliqué que les réseaux de neurones artificiels sont calculatoires et sont des systèmes inspirés par les processus biologiques qui se produisent dans le cerveau humain. Nous avons montré que l'apprentissage est le processus par lequel les paramètres libres d'un réseau de neurones s'adaptent et nous avons identifié 3 types de fonctions d'activation de base ainsi que les architectures des réseaux neurones et enfin on a cité les différents avantages et inconvénients de ces réseaux neuronaux.

Chapitre 3

3. Réseaux neurones convolutionnels

3.1. Introduction

Comme nous l'avons également appris dans le chapitre précédent, les réseaux de neurones sont implémentés dans des machines capables d'apprendre à partir de mauvaises prédictions. Ceux-ci doivent en effet être connus (et donc utilisés en entrée) par le système qui doit analyser, comprendre et corriger ses résultats. Pendant le processus d'initialisation d'un réseau de neurones, les valeurs des matrices constituant les poids, sont grossièrement assignées de manière aléatoire, puis ajustées à mesure que la machine apprend. En effet, l'objectif principal d'un réseau de neurones est de disposer, après itération, les valeurs du poids, de sorte que la prédiction suivante soit plus précise que celle qui le précède immédiatement.

Cela peut sembler simple, mais en réalité ce n'est pas le cas, surtout dans les scénarios les plus compliqués. Former un réseau de neurones pour apprendre est une tâche très complexe, d'autant plus si nous pensons à des domaines tels que la reconnaissance de la parole, la vision par ordinateur ou les voitures autonomes. Sans aucun doute, la puissance de calcul est primordiale pour la qualité du réseau de neurones et seulement ces dernières années, l'utilisation de cartes graphiques a permis de nouveaux résultats, en les exploitant en parallèle pour accélérer de manière significative le processus de prédiction.

Ce qu'un réseau neuronal doit avoir pour fonctionner, c'est l'analyse des données brutes dérivant de l'entrée. Une fois l'intrant analysé et emballé, les réseaux de neurones sont capables d'extraire ses propriétés caractérisantes qui sont la base de l'apprentissage.

Avec l'approche conventionnelle de l'apprentissage automatique, ces propriétés ont été identifiées manuellement, en s'appuyant sur l'enregistrement en mémoire des caractéristiques des motifs d'entrée : en substance, les entrées connues ont été introduites dans le réseau neuronal et pour chacune d'entre elles, ce qui en fait un très long travail.

Lorsque le système a reçu une entrée déjà analysé dans le passé, il a reconnu la similitude et a pu extraire les caractéristiques d'intérêt, comme c'était déjà le cas. Clairement, plus les entrées sont variées, plus le réseau de neurones devient flexible.

L'apprentissage en profondeur élimine le besoin de ce type d'intervention humaine et par conséquent, la nécessité de placer des entrées connues afin d'apprendre à la machine à identifier les propriétés les plus «fréquentes».

Dans l'apprentissage en profondeur, des quantités croissantes de données sont utilisées pour améliorer la capacité des réseaux de neurones à «penser» et à «apprendre» grâce à l'augmentation de la quantité de données traitées. "Profondeur" se réfère aux différents niveaux d'apprentissage que le réseau de neurones accumule au fil du temps, améliorant les performances avec l'augmentation de la profondeur du réseau.

Bien que la majeure partie de l'apprentissage en profondeur actuel soit effectuée sous la supervision d'un être humain, l'objectif est de créer des réseaux de neurones capables de s'entraîner et d '«apprendre» indépendamment.

L'apprentissage en profondeur est un développement relativement récent, puisque nous avons récemment atteint la puissance de traitement et la capacité de stockage de données avancées qui permettent d'utiliser l'apprentissage en profondeur pour créer de nouvelles technologies passionnantes.

Ce type d'apprentissage automatique est à la base de la technologie «intelligente», qui va du logiciel de reconnaissance de la parole / image aux voitures sans conducteur. Les progrès de l'apprentissage en profondeur et de la robotique pourraient bientôt déboucher sur des technologies d'imagerie médicale intelligentes capables d'établir des diagnostics fiables, de piloter des drones et de gérer la maintenance automatisée des machines et des infrastructures de toutes sortes.

L'innovation révolutionnaire apportée par l'apprentissage en profondeur est celle qui consiste à permettre aux machines d'accéder au deuxième niveau d'apprentissage. Nous avons enseigné aux ordinateurs comment apprendre.

On va essayer de mieux éclaircir cela avec quelques exemples :

- Au cours de sa vie professionnelle, un radiologue examine des milliers de radiographies, avec lesquelles il acquiert l'expérience nécessaire pour dire si une radiographie représente ou non une initiation tumorale. Alors que (malheureusement) un radiologue peut faire une erreur,

un logiciel qui a examiné des millions (et non des milliers) de radiographies thoraciques pourrait être en mesure de donner une réponse beaucoup plus précise au même problème.

- Si vous envisagez de travailler dans le monde des transports, il est préférable de savoir que l'apprentissage en profondeur existe car dans quelques années, les voitures pourraient être pilotées par un logiciel formé pour reconnaître les bords de la route, les feux de signalisation, les panneaux et les obstacles. Avec le GPS, ils seront sûrement meilleurs que les humains en toute sécurité sur les routes.

- L'apprentissage en profondeur pourrait être utilisé pour construire de nouveaux logiciels anti-spam qui pourraient être formés chaque jour avec des millions de spam devenant meilleurs que n'importe quel humain pour reconnaître un faux email.

- Un logiciel capable de lire pourrait devenir meilleur que vous ne le pourriez en rédigeant des résumés, des revues de presse et en analysant la réputation de la marque. Même maintenant, les traducteurs ont un avenir plutôt difficile, car les logiciels de traduction automatique des dix dernières années ont presque égalé les capacités des êtres humains.

- Facebook utilise déjà des techniques d'apprentissage en profondeur pour permettre aux ordinateurs de reconnaître un visage dans une photographie, afin de suggérer des balises sur une image. Dans chaque secteur, il y a des startups qui approfondissent l'utilisation du deep learning dans une zone restreinte pour résoudre des problèmes spécifiques et peu à peu, la révolution sera bientôt terminée. [4] [7] [8]

3.2. Comment ça fonctionne ?

Le Deep Learning est un domaine d'apprentissage automatique basé sur un type particulier d'apprentissage des données. Il est caractérisé par l'effort de créer un modèle d'apprentissage automatique à plus d'un niveau, dans lequel les niveaux les plus profonds prennent des contributions des niveaux précédents, les transformant et les abstrayant de plus en plus. Cet aperçu des niveaux d'apprentissage donne le nom à l'ensemble du domaine (apprentissage en profondeur) et s'inspire de la façon dont le cerveau des mammifères traite l'information et apprend en réagissant aux stimuli externes.

Chaque niveau d'apprentissage correspond dans ce parallèle hypothétique, à l'une des différentes zones qui composent le cortex cérébral. Par exemple, le cortex visuel, responsable de la reconnaissance des images, montre une séquence de secteurs hiérarchisés. Chacun de ces secteurs reçoit une représentation d'entrée, au moyen des signaux de flux qui le relient aux

autres secteurs. Chaque niveau de cette hiérarchie représente un niveau d'abstraction différent, les caractéristiques les plus abstraites étant définies par rapport à celles du niveau inférieur. Lorsque le cerveau reçoit des images, il les traite à travers différentes phases, par exemple la détection des contours, la perception des formes (des primitives aux plus complexes graduellement). C'est pourquoi nous parlons de représentation hiérarchique de l'image au niveau de l'abstraction croissante. On peut expliquer ceci différemment :

Au fur et à mesure que le cerveau apprend par essais et erreurs et active de nouveaux neurones apprenant de l'expérience, même dans les architectures responsables de l'apprentissage profond, les étapes d'extraction peuvent être modifiées en fonction des informations reçues à l'entrée.

Les réseaux neurones sont l'un des principaux outils qui bénéficient du Deep Learning.

En fait, même si un réseau à trois couches (couche d'entrée, couche cachée et couche de sortie) possède un certain degré de capacité à distinguer entre des régions arbitrairement complexes, une architecture plus grande apporte certains avantages.

L'un des principaux avantages, par exemple, est lié au nombre de nœuds par couche : en effet, dans un réseau à trois niveaux, le nombre de nœuds dans chaque couche limite la complexité de la région à reconnaître. En conséquence, nous risquons en limitant la profondeur, d'avoir besoin d'un grand nombre de nœuds. Les résultats théoriques montrent qu'il existe des situations dans lesquelles le nombre de nœuds nécessaires augmente exponentiellement avec la taille de l'entrée. Une explosion de nœuds sur les couches entraîne un coût de calcul élevé et une utilisation importante de la mémoire, pour cette raison, une planification minutieuse de l'architecture de réseau à utiliser doit toujours être effectuée. [4] [7] [8]

3.3. Architectures et algorithmes

Le terme DNN désigne des réseaux «profonds» composés de plusieurs couches (dont au moins deux sont cachées) organisées hiérarchiquement. L'organisation hiérarchique permet de partager et de réutiliser l'information (un peu comme la programmation structurée). Le long de la hiérarchie, vous pouvez sélectionner des caractéristiques spécifiques et éliminer les détails inutiles (afin de maximiser l'invariance). La structure la plus simple que nous connaissions est celle des réseaux de feed-back formés par des algorithmes de rétro propagation.

La mise à jour des poids au cours de la phase d'apprentissage est réalisée à l'aide de la méthode SGD, c'est-à-dire dans les formules : $w_{ij}(t+1) = w_{ij}(\mathrm{t}) + \eta \frac{\partial \mathrm{C}}{\partial w_{ij}}$

A titre d'exemple, le système visuel du corps humain fonctionne selon une hiérarchie de niveaux (profondeur) : Les DNN les plus utilisés comprennent un nombre de couches compris entre 7 et 50. Des réseaux plus profonds (100 niveaux et plus) ont montré qu'ils pouvaient garantir une performance légèrement meilleure, mais au détriment de l'efficacité. La profondeur (nombre de niveaux) n'est que l'un des facteurs de complexité : le nombre de neurones, de connexions et de poids caractérisent également la complexité d'un DNN.

Plus le nombre de poids est élevé (c'est-à-dire les paramètres à apprendre), plus la complexité de la formation sera grande. En même temps, un nombre élevé de neurones et de connexions rend la propagation vers l'avant et vers l'arrière plus coûteuse.

Ces réseaux sont notamment utilisés dans la modélisation de langages, la reconnaissance d'objets et plus généralement dans la modélisation de relations complexes non-linéaires. Certaines limites de ce type de réseau sont le temps d'exécution élevé dans la phase d'apprentissage et le risque de sur-apprentissage. [4] [7] [8]

3.4. Types du DNN

Les principaux types de Deep Neural Network sont :

1. Modèles prédictifs «discriminatoires» pour la classification (ou la régression) avec une formation principalement supervisée :
    - CNN (ou ConvNet)
    - FC DNN : DNN entièrement connecté (MLP avec au moins deux niveaux cachés)
    - HTM : est une théorie (ou modèle) de l'intelligence biologiquement contrainte, décrite à l'origine dans le livre de 2004 sur l'intelligence de Jeff Hawkins avec Sandra Blakeslee. HTM est rigoureusement basé sur la neuroscience et la physiologie et l'interaction des neurones pyramidaux dans le néocortex du cerveau des mammifères (en particulier de l'homme). [9]
2. Formation non supervisée (modèles "génératifs" formés pour reconstruire l'entrée, utiles pour la préformation d'autres modèles et pour produire des caractéristiques saillantes) :

- RBM : inventée sous le nom d'Harmonium en 1986 par Maul Smolenski, c'est un réseau neurones artificiels utilisé pour avoir une estimation de la distribution d'un jeu de données. [10]
- DBN : est un modèle graphique génératif, ou bien une classe de réseaux de neurones profonds, composée de plusieurs couches de variables latentes ("unités cachées"), avec des connexions entre les couches mais pas entre chaque couche. [11]

3. Modèles récurrents (utilisés pour les séquences, la reconnaissance de la parole, l'analyse des sentiments, le traitement du langage naturel, ...) :
    - RNN : est une classe de réseau neuronal artificiel où les connexions entre les nœuds forment un graphe orienté le long d'une séquence. Ils peuvent utiliser leur état interne (mémoire) pour traiter des séquences d'entrées. Cela les rend applicables à des tâches telles que la reconnaissance d'écriture manuscrite non segmentée, connectée ou la reconnaissance vocale. [12]
    - LSTM : sont des unités d'un réseau neuronal récurrent (RNN). Un RNN composé d'unités LSTM est souvent appelé un réseau LSTM. Une unité LSTM commune est composée d'une cellule, d'une porte d'entrée, d'une porte de sortie et d'une porte d'oubli. La cellule se souvient de valeurs sur des intervalles de temps arbitraires et les trois portes régulent le flux d'informations entrant et sortant de la cellule.
4. Apprentissage par renforcement (pour apprendre les comportements) :
    - Q-Learning profond : est une technique d'apprentissage par renforcement utilisée dans l'apprentissage automatique. Le but de Q-Learning est d'apprendre une politique qui indique à un agent les mesures à prendre dans des circonstances. Il ne nécessite pas de modèle d'environnement et peut gérer des problèmes de transitions stochastiques et de récompenses, sans nécessité d'adaptations.

3.5. Réseaux neurones convolutionnels (CNN)

Les réseaux convolutifs ont été introduits pour la première fois par Fukushima [13], il a dérivé une architecture du réseau nerveux hiérarchique inspirée par le travail de recherche de Hubel [14]. Lecun [15], les a généralisés pour classer les chiffres avec succès et pour reconnaître les numéros de contrôle manuscrit par LeNet-5 qui est montré à la Fig.3-1. Ciresan [16] a utilisé les réseaux convolutifs et réalisé les meilleures performances dans la

littérature pour la reconnaissance d'objets multiples pour des bases de données d'images : MNIST, NORB, CIFAR10 et l'ensemble de données ImageNet.

Les réseaux CNN se concentrent principalement sur le fait que l'entrée sera composée d'images. Cela permet de centrer l'architecture à mettre en place pour répondre au mieux à la nécessité de traiter un type de données spécifiques.

Les réseaux neuronaux convolutifs diffèrent des autres formes de réseaux neuronaux artificiels en ce sens. Qu'au lieu de se concentrer sur l'intégralité du domaine problématique, les connaissances sur le type spécifique d'entrées sont exploitées, cela permet à son tour de mettre en place une architecture réseau beaucoup plus pérformante. [17] [18] [19]

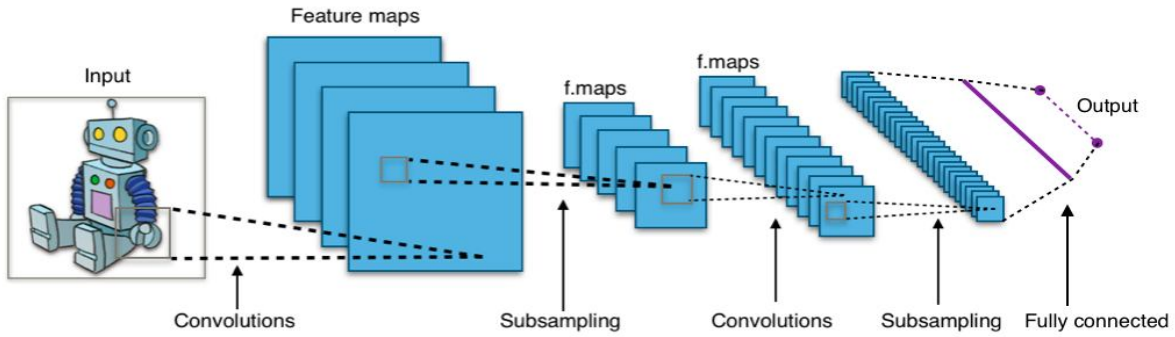

Figure 3- 1 Réseau neuronal convolutif

3.6. Architecture globale de CNN

Les CNN sont composés de trois types de couches : des couches convolutives, des couches de regroupement et des couches entièrement connectées. Lorsque ces couches sont empilées, une architecture CNN a été formée. Une architecture CNN simplifiée pour la classification MNIST est illustrée à la figure 3-3. [17] [18]

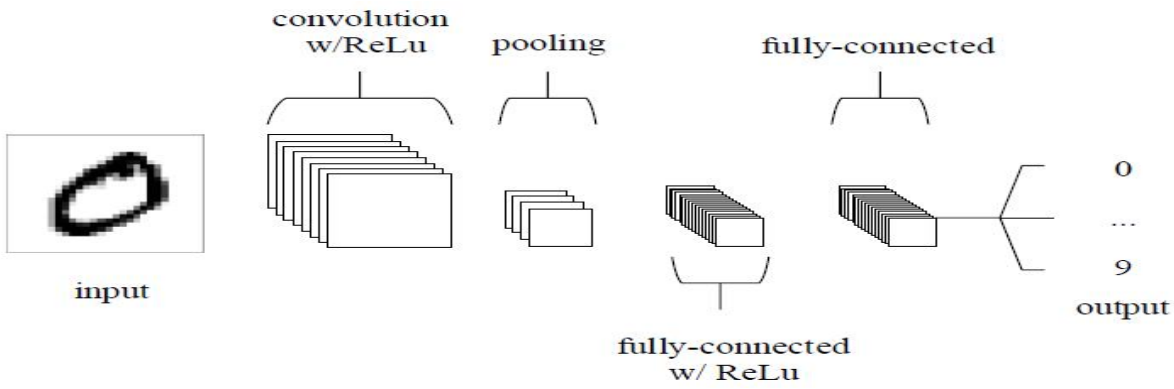

Figure 3- 2 Architecture simple de CNN

3.6.1. Couche convolutive

Les couches convolutives constituent le noyau du réseau convolutif. Ces couches se composent d'une grille rectangulaire de neurones qui ont un petit champ réceptif étendu à travers toute la profondeur du volume d'entrée. Ainsi, la couche convolutive est juste une convolution d'image de la couche précédente, où les poids spécifient le filtre de convolution.

La couche convolutive déterminera la sortie des neurones qui sont connectés aux régions locales de l'entrée par le calcul du produit scalaire entre leurs poids et la région connectée au volume d'entrée. ReLu vise à appliquer une fonction d'activation «élémentaire» telle qu'une fonction sigmoïde à la sortie de l'activation produite par la couche précédente. [17] [18] [19]

3.6.2. Couche de pooling

Après chaque couche convolutive, il peut y avoir une couche de pooling. Cette couche sous échantillonne le long de la dimensionnalité spatiale de l'entrée donnée, ce qui réduira davantage le nombre de paramètres au sein de cette activation. Il y a plusieurs façons de faire cette mise en commun, comme prendre la moyenne ou le maximum, ou une combinaison linéaire prise par des neurones dans le bloc. Par exemple, la Fig. 3 montre le max pooling sur une fenêtre 2 × 2. [17] [18] [19]

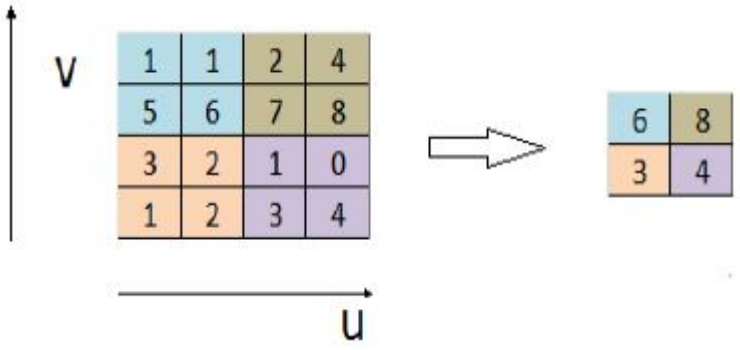

Figure 3- 3 Max pooling

3.6.3. Couche totalement connectée

Enfin, après les couches de convolution et pooling, le raisonnement de haut niveau dans le réseau neuronal se fait via des couches totalement connectées.

Dans les réseaux de neurones convolutifs, chaque couche agit comme un filtre de détection pour la présence de caractéristiques spécifiques ou de motifs présents dans les données d'origine. Les premières couches d'un réseau convolutif détectent des caractéristiques qui peuvent être reconnues et interprétées facilement. Les couches ultérieures détectent de plus en plus des caractéristiques plus abstraites. La dernière couche du réseau convolutif est capable

de faire une classification ultra-spécifique en combinant toutes les caractéristiques spécifiques détectées par les couches précédentes dans les données d'entrée.

Les couches totalement connectées font les mêmes tâches que celles des ANN standard et tenteront de produire des notes de classe à partir des activations, pour les utiliser pour la classification. Il est également suggéré d'utiliser ReLu entre ces couches pour améliorer les performances. [17] [18] [19]

3.6.4. Couche de correction (ReLu)

C'est une couche pour 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 : **F(x)=max (0, x)** Cette fonction force les neurones à retourner des valeurs positives. [17] [18]

3.7. Conclusion

Dans ce chapitre, on a commencé par déterminer ce qu'est l'apprentissage profond, ensuite, nous avons montré le rôle des réseaux neurones convolutifs. Ces réseaux sont capables d'extraire des caractéristiques d'images présentées en entrée et de classifier ces caractéristiques. Ils sont fondés sur la notion de « champs récepteurs » (receptive fields), ils implémentent aussi l'idée de partage des poids qui permettant de réduire beaucoup le nombre de paramètres libres de l'architecture. Ce partage des poids permet en outre de réduire les temps de calcul, l'espace mémoire nécessaire et également d'améliorer les capacités de généralisation du réseau.

Chapitre 4

4. Implémentation

4.1. Introduction

Dans ce chapitre, on mettra en avant notre conception et on va montrer comment réaliser notre classificateur et créer notre modèle. Part la suite on va évaluer et créer deux modèles avec des architectures différente, par la suite, on appliquera ces modèles sur la base d'images CIFAR 10. Pour cela, on va travailler avec le langage de programmation python 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. Conception

Avant d'entrer dans l'implémentation, on va expliquer comment concevoir notre programme. Le tableau ci-dessus est la conception de notre classificateur :

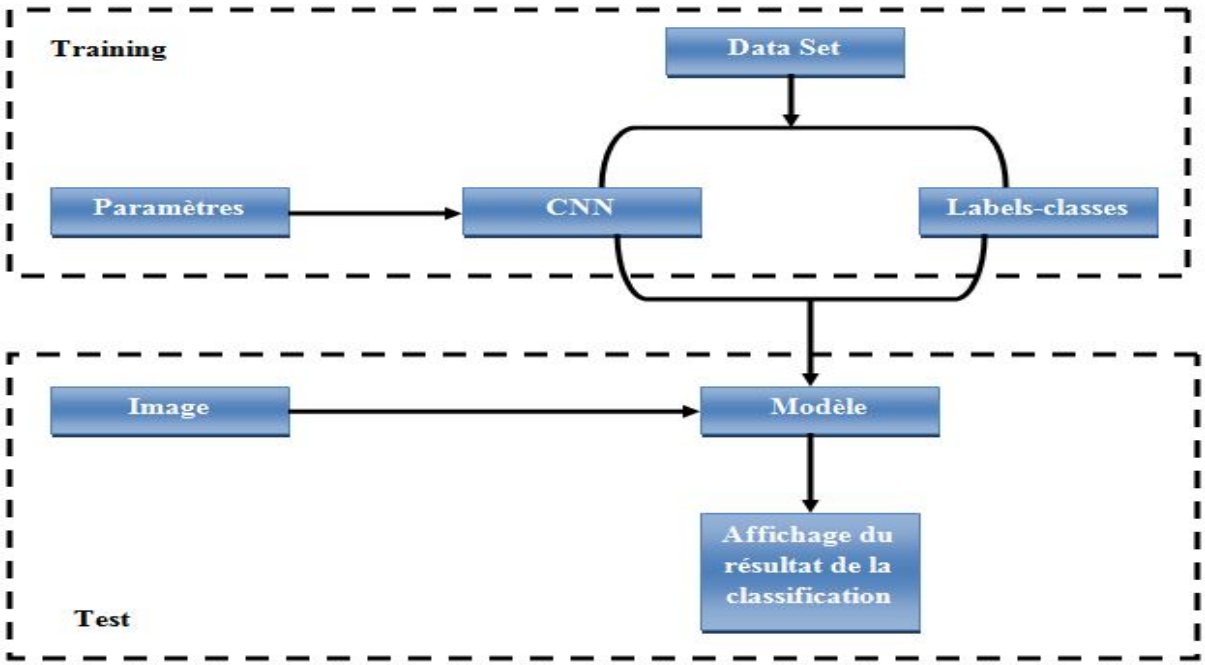

Figure 4- 1 Conception du classificateur

On remarque qu'il ya deux grands processus qui englobent notre conception :

a. Training : c'est le processus le plus important parce qu'on va créer notre modèle grâce à des configurations précises.

- La Dataset : c'est une base de données d'images répertoriées en classes. Par exemple, si on prend une dataset d'animaux domestiques, on peut trouver des classes comme 'chien', 'chat', 'coq', 'oiseau' ... etc.
- labels_classes : c'est un fichier texte qui portera les noms des classes de notre dataset. Si on prend notre dernier exemple, on trouvera que notre fichier comportera 'la classe chien' , 'la classe chat' ...etc.
- Cnn et paramètres : c'est notre algorithme de création d'un réseau neurones convolutionels qui sera configuré avec des paramètres, par exemple : nombre d'époques, nombre de filtre, nombre de couche ...etc.

On va exécuter la dataset sur notre algorithme CNN qui sera paramétrée pour générer un modèle puis on va utiliser ce modèle pour le Test.

b. Test :

Dans le processus Test on retrouve :

- L'image : c'est l'entrée pour les tests. Ce sont plusieurs images ou une seule.
- Le modèle : c'est un fichier généré dans notre training.
- L'affichage de la classification : son nom résume son travail, on va afficher le résultat sortie du modèle qui est le nom d'une classe.

4.3. Logiciels et librairies utilisés dans l'implémentation

4.3.1. Python

Python est un excellent langage de programmation orienté objet, interprété et interactif. Il est souvent comparé (favorablement bien sûr) à Lisp, Tcl, Perl, Ruby, C #, Visual Basic, Visual Fox Pro, Scheme ou Java ... etc.

Python combine un pouvoir remarquable avec une syntaxe très claire. Il comporte des modules, des classes, des exceptions, des types de données dynamiques de très haut niveau et le typage dynamique. Il existe des interfaces vers de nombreux appels systèmes et bibliothèques, ainsi que vers différents systèmes de fenêtrage.

Les nouveaux modules intégrés sont faciles à écrire en C ou C ++ (ou dans d'autres langages, selon l'implémentation choisie). Python est également utilisable comme langage d'extension pour les applications écrites dans d'autres langages nécessitant des interfaces de script ou d'automatisation facile à utiliser. [20]

4.3.2. Tensorflow

TensorFlow est une bibliothèque logicielle open source pour le calcul numérique de haute performance. Son architecture flexible permet un déploiement facile du calcul sur diverses plates-formes (CPUs ,GPUs ,TPUs), et des ordinateurs de bureau aux clusters de serveurs, aux périphériques mobiles. Initialement développé par des chercheurs et des ingénieurs de l'équipe de Google Brain au sein de l'organisation de l'IA de Google, il s'appuie sur l'apprentissage automatique et l'apprentissage en profondeur. [21]

4.3.3. Keras

Keras est une API de réseaux neuronaux de haut niveau, écrite en Python et capable de s'exécuter sur TensorFlow, CNTK ou Theano. Il a été développé dans le but de permettre une expérimentation rapide. Être en mesure de passer de l'idée au résultat le plus rapidement possible, est la clé pour faire de la recherche :

- Permet un prototypage facile et rapide (grâce à la convivialité, à la modularité et à l'extensibilité).
- Prend en charge les réseaux convolutionnels et les réseaux récurrents ainsi que les combinaisons des deux.
- Fonctionne de manière transparente sur le processeur et le processeur graphique. [22]

4.3.4. CUDA

CUDA est une plate-forme de calcul parallèle et un modèle de programmation développé par NVIDIA pour l'informatique générale sur les unités de traitement graphique (GPU). Avec CUDA, les développeurs peuvent accélérer considérablement les applications informatiques en exploitant la puissance des GPU.

Dans les applications accélérées par GPU, la partie séquentielle de la charge de travail s'exécute sur le processeur, optimisée pour les performances mono-thread, tandis que la partie intensive de calcul de l'application s'exécute en parallèle sur des milliers de cœurs de GPU.

Lors de l'utilisation de CUDA, les développeurs programment dans des langages populaires tels que C, C ++, Fortran, Python et MATLAB et expriment le parallélisme à travers des extensions sous la forme de quelques mots-clés de base. [23]

4.3.5. CuDNN

La bibliothèque NVIDIA CUDA Deep Neural Network (cuDNN) est une bibliothèque de primitives accélérée par GPU pour les réseaux de neurones profonds. cuDNN fournit des implémentations hautement adaptées pour les routines standard telles que les couches de convolution, de pooling, de normalisation et d'activation avant et arrière. cuDNN fait partie du kit SDK NVIDIA Deep Learning.

Les chercheurs en apprentissage en profondeur et les développeurs de framework du monde entier font confiance à cuDNN pour une accélération GPU de haute performance. Cela leur permet de se concentrer sur la formation de réseaux de neurones et le développement d'applications plutôt que de passer du temps sur le réglage des performances des GPU de bas niveau. cuDNN accélère les cadres d'apprentissage profond largement utilisés, y compris Caffe2, MATLAB, Microsoft Cognitive Toolkit, TensorFlow, Theano et PyTorch. [24]

4.3.6. Configuration utilisé dans l'implémentation

La configuration du matériel utilisé dans notre implémentation est :

- Un PC portable Dell i5 CPU 2.20 GHZ
- Carte graphique Nvidia GeForce 920M
- RAM de taille 4 GO
- Disque dur de taille 500 GO
- Système d'exploitation 64 bits Windows 7

4.4. Base de données CIFAR-10

Le jeu de données CIFAR-10 se compose de 60000 images en couleurs 32x32 dans 10 classes, 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.

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

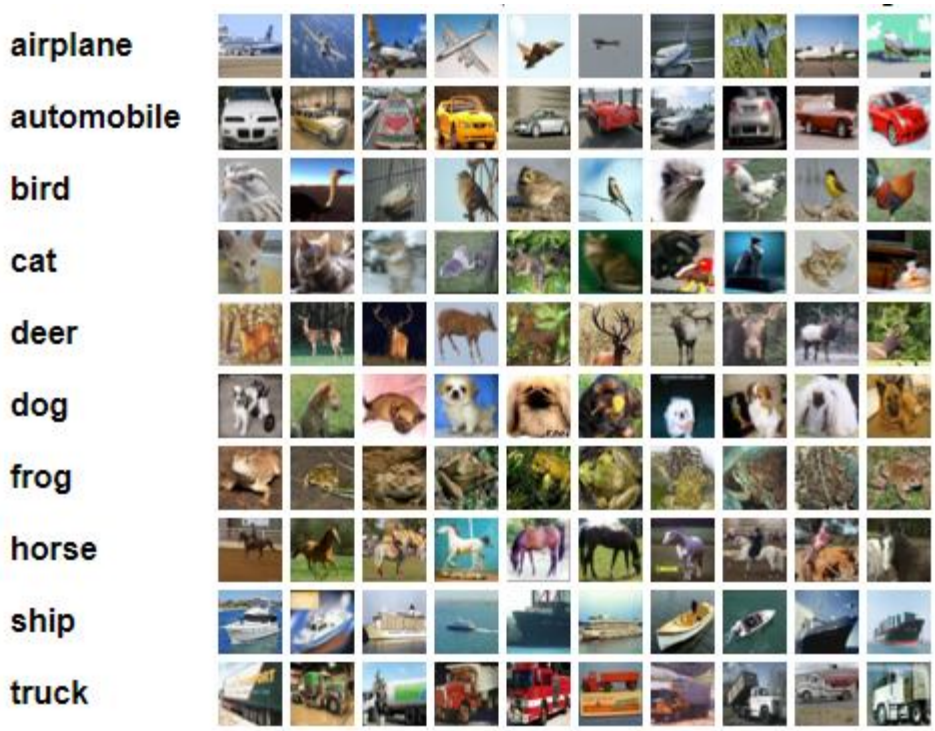

Figure 4- 2 Images de cifar-10-classes

4.5. Architecture des réseaux

Au cours de nos expérimentations, nous avons créé deux modèles avec différentes architectures où on a appliqué sur les deux modèles la base d'images CIFAR10 et pour chaque modèle, on a fait une évaluation sur le nombre d'époques.

Dans ce qui suit on présentera l'architecture des 2 modèles et le résultat obtenu à chaque fois :

4.5.1. Réseau neuronal convolutif à 4 couches

Le premier modèle que nous présentons est composé de quatre couches de convolution, deux couches de maxpooling et de deux couches de fully connected.

L'image en entrée a une taille de 32*32, chaque couche de convolution composée de plusieurs filtres (32 pour les 2 première couche), la taille de chaque filtre est de 3*3, 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.

Le Maxpooling est appliqué après pour réduire la taille de l'image. A la fin de la deuxième couche de convolution, on utilise Dropout qui un est une technique de régularisation pour réduire le sur ajustement dans les réseaux neurones.

Dans la troisième et quatrième couche, on change quelques paramètres comme le nombre de filtres qui devient 64 à la place de 32, la fonction d'activation reste la même (ReLu) et la même chose pour Maxpooling puis on refait un appel à la fonction Dropout.

Après les couches de convolution et de regroupement, notre partie classification se compose de quelques couches entièrement connectées. Cependant, ces couches ne peuvent accepter que des données à une dimension. Pour convertir nos données 3D en 1D, nous utiliserons la fonction Flatten, cela permettra essentiellement d'arranger notre volume 3D en un vecteur 1D.

Les dernières couches d'un réseau neurones convolutionnel sont des couches entièrement connectées. Ces dernières ont des connexions complètes avec toutes les activations de la couche précédente.

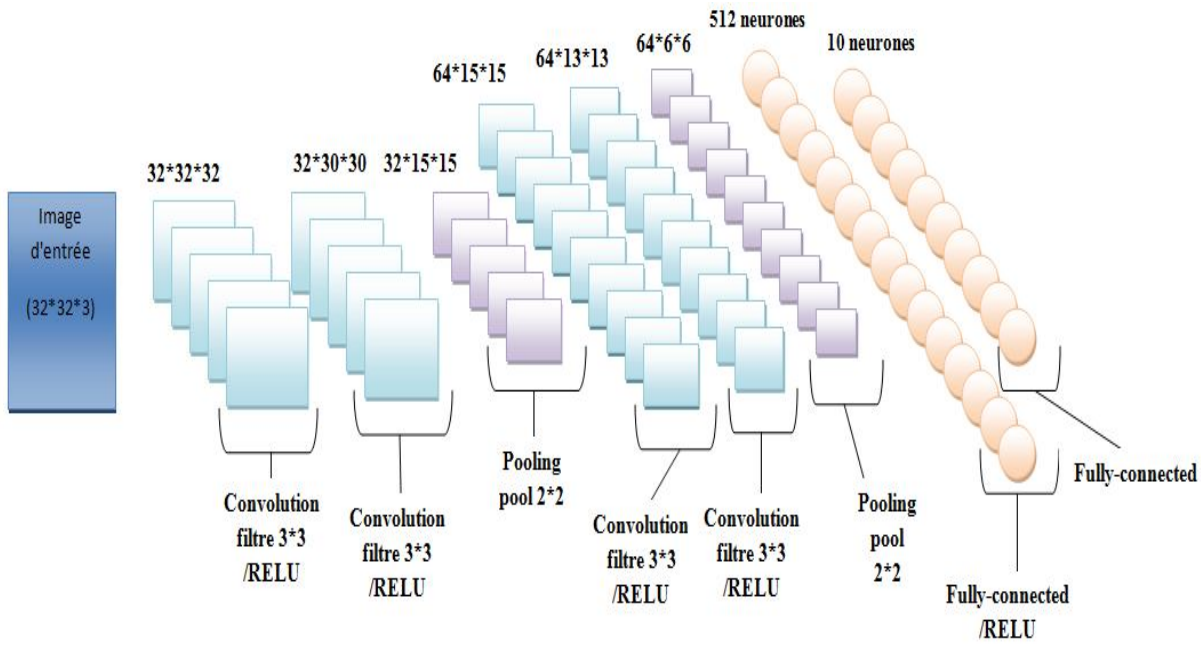

Figure 4- 3 Architecteur de modèle 1 (4 couches)

OPERATION		DATA DIMENSIONS			WEIGHTS(N)	WEIGHTS(%)
Input	#####	3	32	32		
Conv2D	|/	----------			896	0.1%
relu	#####	32	32	32		
Conv2D	|/	----------			9248	0.7%
relu	#####	32	30	30		
MaxPooling2D	Y max	----------			0	0.0%
	#####	32	15	15		
Dropout	| ||	----------			0	0.0%
	#####	32	15	15		
Conv2D	|/	----------			18496	1.5%
relu	#####	64	15	15		
Conv2D	|/	----------			36928	3.0%
relu	#####	64	13	13		
MaxPooling2D	Y max	----------			0	0.0%
	#####	64	6	6		
Dropout	| ||	----------			0	0.0%
	#####	64	6	6		
Flatten	|||||	----------			0	0.0%
	#####	2304				
Dense	XXXXX	----------			1180160	94.3%
relu	#####	512				
Dropout	| ||	----------			0	0.0%
	#####	512				
Dense	XXXXX	----------			5130	0.4%
softmax	#####	10				

Tableau 4- 1 Configuration de modèle 1 (4 couches)

4.5.2. Réseau neuronal convolutif à 6 couches

Sur ce deuxième modèle, on a modifié le nombre de couches pour la convolution et le maxpooling, maintenant notre architecture est composée de six couches convolution et trois couches de maxpooling et enfin les deux dernières couches de fully connected.

Comme pour le premier modèle, l'image en entrée a une taille de 32*32, chaque couche de convolution composée de plusieurs filtres (32 pour les 2 premières couches, 64 pour les 2 autres couches et 128 filtres pour les 2 dernière couches), la taille de chaque filtre est inchangeable pour les six couches de convolution (elle est de 3*3), la fonction d'activation utilisée pour toutes les couches de convolution est ReLU, à la fin des deux couches de convolution, on utilise le maxpooling afin de réduire la taille des images.

On va utiliser un Dropout pour réduire le sur ajustement avec un pourcentage de 20% à chaque fois après chaque couche de convolution et de regroupement, notre partie classification se compose de deux couches entièrement connectées. Cependant, ces couches ne peuvent accepter que des données à une dimension. On passera par la fonction Flatten qui convertira les données à une dimension.

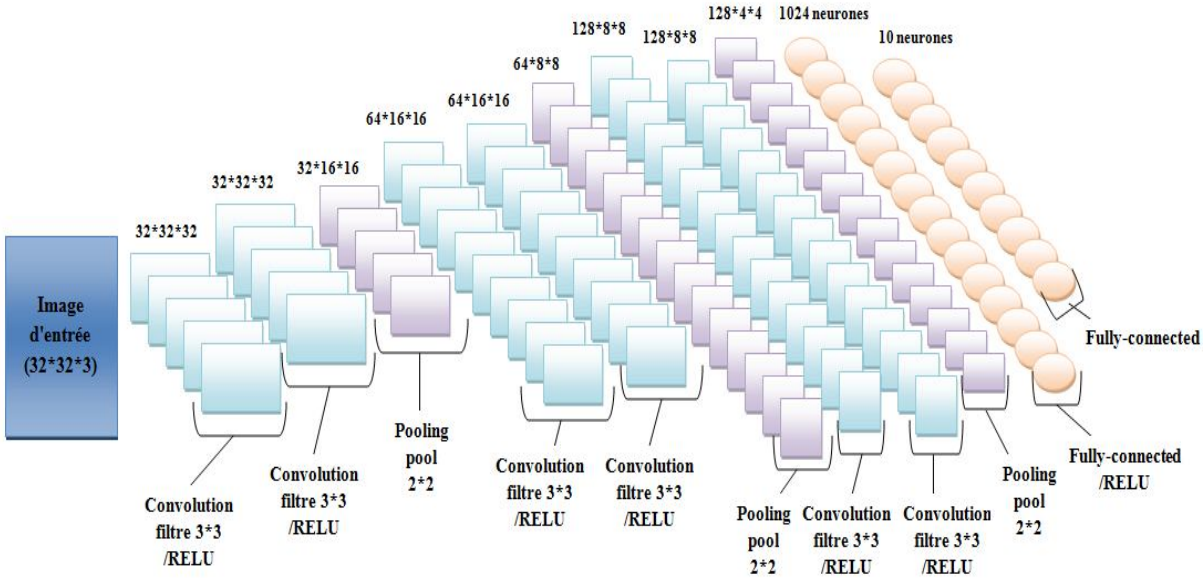

Figure 4- 4 Architecture de modèle 2 (6 couches)

OPERATION		DATA DIMENSIONS			WEIGHTS(N)	WEIGHTS(%)
Input	#####	3	32	32		
Conv2D	|/				896	0.0%
relu	#####	32	32	32		
Dropout	| ||				0	0.0%
	#####	32	32	32		
Conv2D	|/				9248	0.4%
relu	#####	32	32	32		
MaxPooling2D	Y max				0	0.0%
	#####	32	16	16		
Conv2D	|/				18496	0.8%
relu	#####	64	16	16		
Dropout	| ||				0	0.0%
	#####	64	16	16		
Conv2D	|/				36928	1.5%
relu	#####	64	16	16		
MaxPooling2D	Y max				0	0.0%
	#####	64	8	8		
Conv2D	|/				73856	3.1%
relu	#####	128	8	8		
Dropout	| ||				0	0.0%
	#####	128	8	8		
Conv2D	|/				147584	6.2%
relu	#####	128	8	8		
MaxPooling2D	Y max				0	0.0%
	#####	128	4	4		
Flatten	|||||				0	0.0%
	#####	2048				
Dropout	| ||				0	0.0%
	#####	2048				
Dense	XXXXX				2098176	87.6%
relu	#####	1024				
Dropout	| ||				0	0.0%
	#####	1024				
Dense	XXXXX				10250	0.4%
softmax	#####	10				

Tableau 4- 2 Configuration de modèle 2 (6 couches)

4.6. 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 ainsi que les matrices de confusion par rapport au nombre d'époques (Une époque décrit le nombre de fois que l'algorithme passe sur l'ensemble de données) pour chaque modèle.

A. Résultats obtenus pour le modèle 1 (4 couches)

A.1. Nombre d'époques = 10

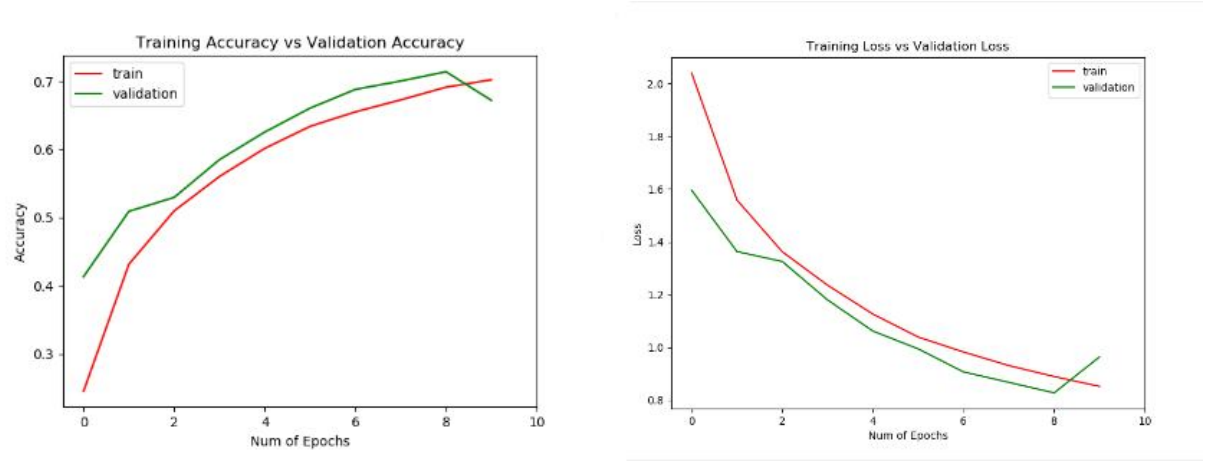

Figure 4- 5 Précision et erreur pour le modèle 1 (10 époques)

D'après la Figure 4-5 La précision de l'apprentissage et de la validation augmente avec le nombre d'époques jusqu'au nombre d'époque (8) où la validation commence à diminuer.

De même, l'erreur d'apprentissage et de la validation diminue avec le nombre d'époque jusqu'au nombre d'époque (8) où la validation commence à augmenter.

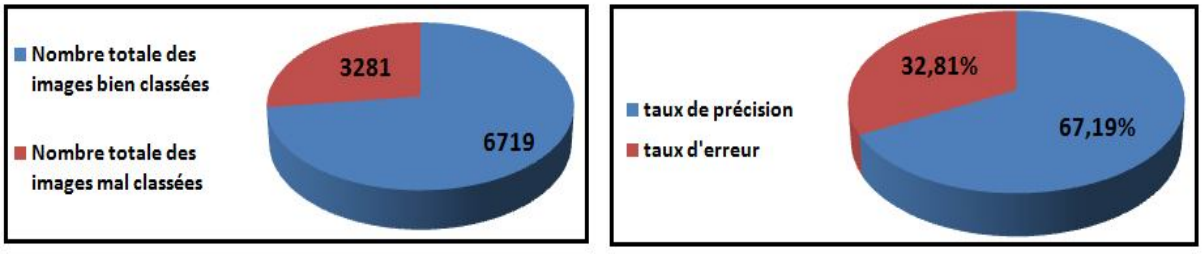

Figure 4- 6 Nombre total des images mal et bien classées et taux d'erreur et de précision de modèle 1 (10 époques)

D'après la figure 4-6 nous remarquons que la totalité des images mal classées est de 3281 images, un taux d'erreur de 32,81% et la totalité des images bien classées est de 6719 donc un taux de précision de 67,19%.

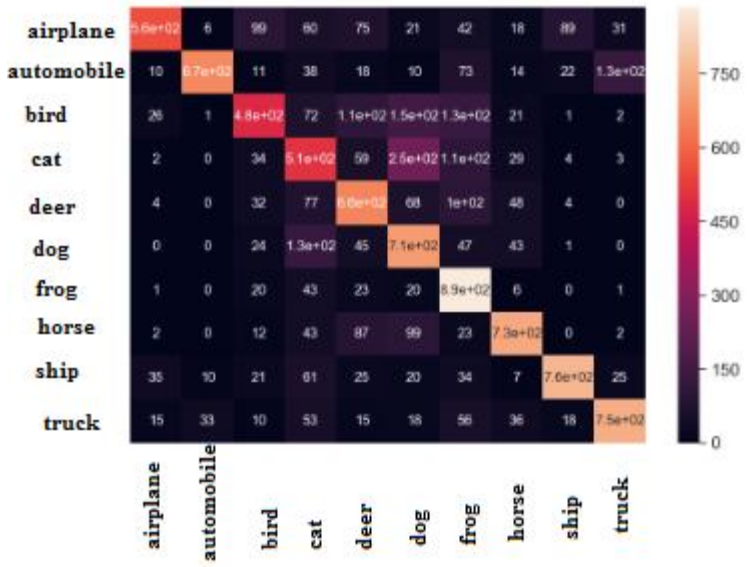

Figure 4- 7 Matrice de confusion pour le modèle1 (10 époques)

La matrice de confusion permet d'évaluer la performance de notre modèle, puisqu'elle reflète les métriques du Vrai positif, Vrai négatif, Faux positif et Faux négatif. La figure 4-7 illustre de près la position de ces métriques pour chaque classe. A titre d'exemple le modèle a bien classé les images frog et il a mal classé les images bird.

A.2. Nombre d'époques = 20

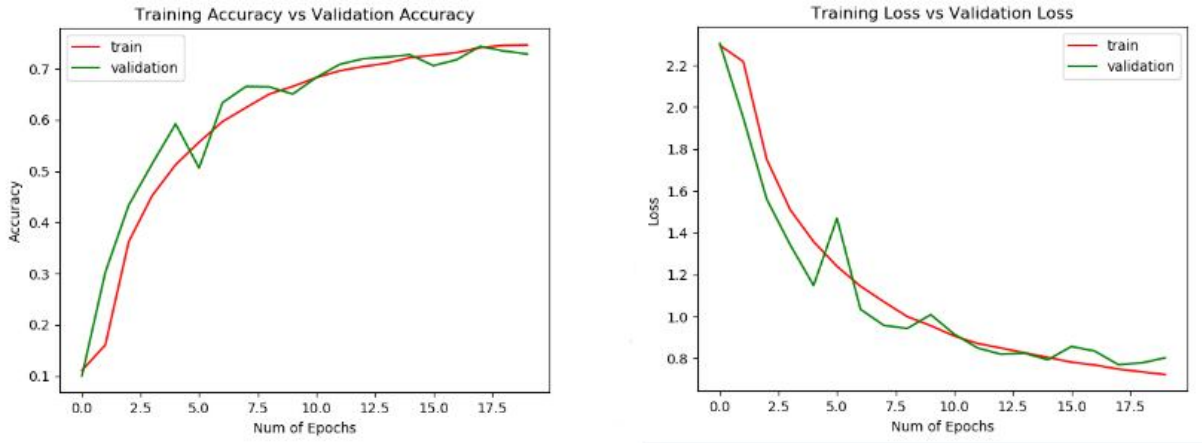

Figure 4- 8 Précision et erreur pour le modèle 1 (20 époques)

D'après la Figure 4-8 La précision de l'apprentissage et de la validation augmente avec le nombre d'époques.

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

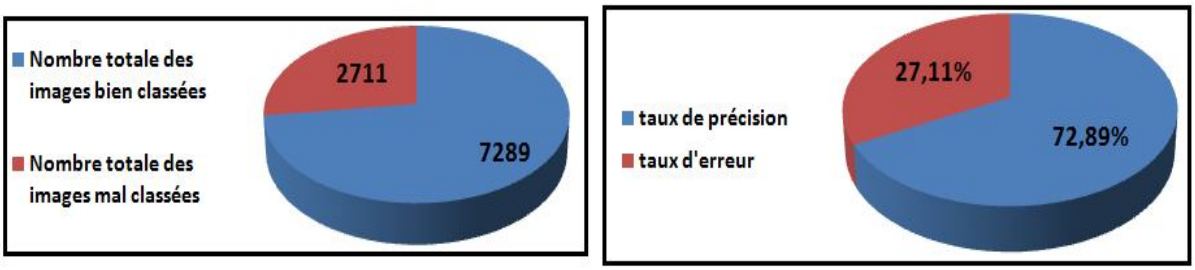

Figure 4- 9 Nombre total des images mal et bien classées et taux d'erreur et de précision de modèle 1 (20 époques)

D'après la figure 4-9 nous remarquons que la totalité des images mal classées est de 2711 images, un taux d'erreur de 27,11% et la totalité des images bien classées est de 7289 donc un taux de précision de 72,89%.

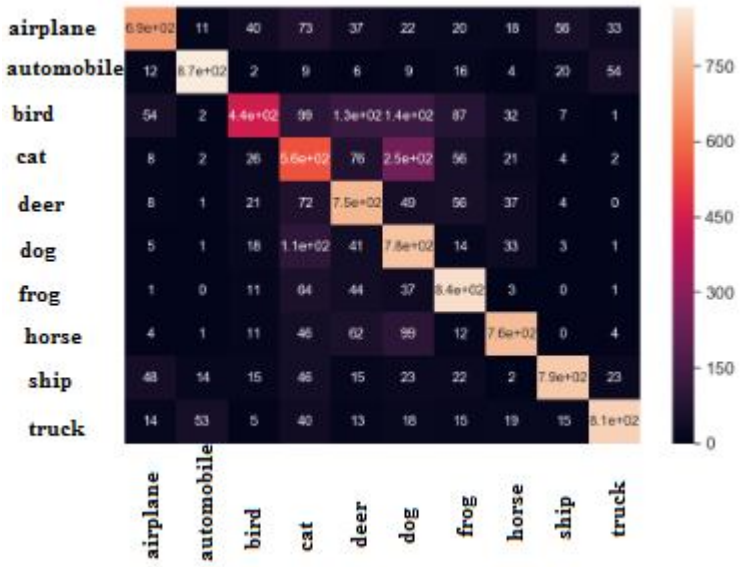

Figure 4- 10 Matrice de confusion pour le modèle 1 (20 époques)

La matrice de confusion permet d'évaluer la performance de notre modèle, puisqu'elle reflète les métriques du Vrai positif, Vrai négatif, Faux positif et Faux négatif. La figure 4-10 illustre de près la position de ces métriques pour chaque classe. A titre d'exemple le modèle a bien classé les images automobiles, truck et frog et il a mal classé les images bird.

A.3. Nombre d'époques = 50

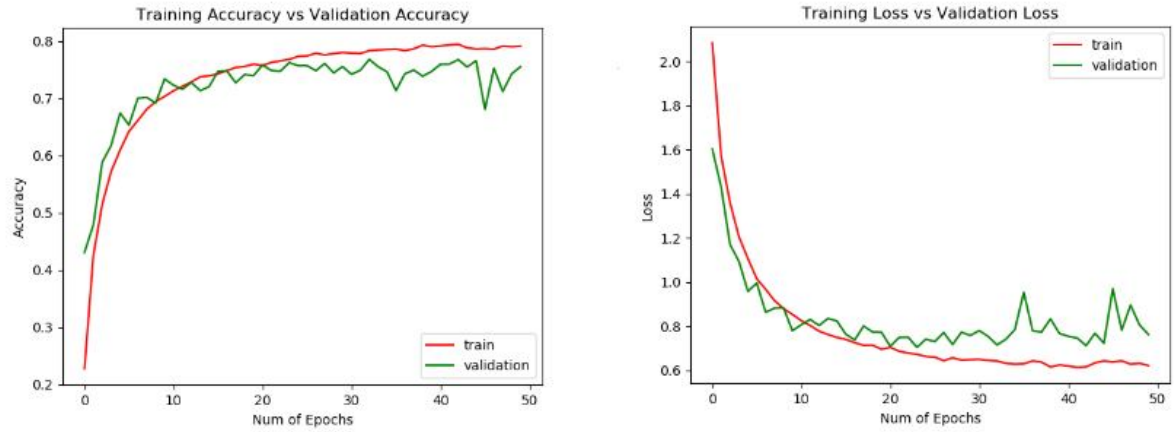

Figure 4- 11 Précision et erreur pour le modèle 1 (50 époques)

D'après la Figure 4-11 La précision de l'apprentissage et de la validation augmente dans l'intervalle du nombre d'époques [0-10], et dans l'intervalle [10-35] la validation se stabilise dans le niveau (0.7) et l'apprentissage augmente, mais dans l'intervalle [35-50] nous avons remarqué que la précision de l'apprentissage se stabilise au niveau (0.8) et la validation reste stable dans le même niveau.

De même, l'erreur d'apprentissage et de la validation diminue dans l'intervalle du nombre d'époques [0-10], et dans l'intervalle [10-30], la validation reste stable dans le niveau (0.8) et l'apprentissage diminue, mais dans l'intervalle [30-50], nous avons remarqué que la précision de l'apprentissage se stabilise au niveau (0.5) et celle de la validation reste stable aussi dans le même niveau.

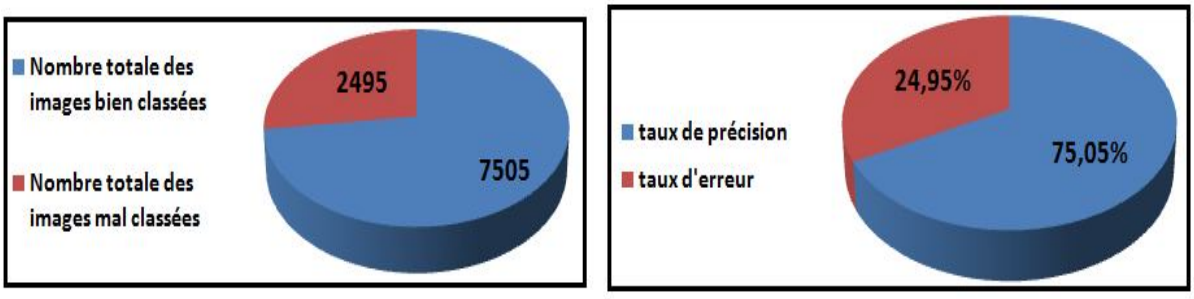

Figure 4- 12 Nombre total des images mal et bien classés et taux d'erreur et de précision du modèle 1 (50 époques)

D'après la figure 4-12, nous remarquons que la totalité des images mal classées est de 2495 images, un taux d'erreur de 24,95% et la totalité des images bien classées est de 7505 donc un taux de précision de 75,05%.

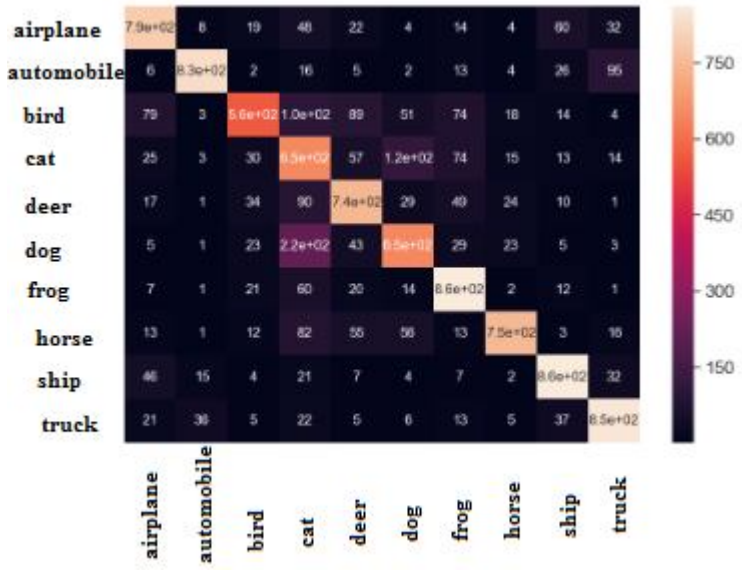

Figure 4- 13 Matrice de confusion pour le modèle 1 (50 époques)

La matrice de confusion permet d'évaluer la performance de notre modèle, puisqu'elle reflète les métriques du Vrai positif, Vrai négatif, Faux positif et Faux négatif. La figure 4-13 illustre de près la position de ces métriques pour chaque classe. A titre d'exemple ; le modèle a bien classé les images automobiles, ship, frog et truck et il a mal classé les images bird.

A.4. Nombre d'époques = 100

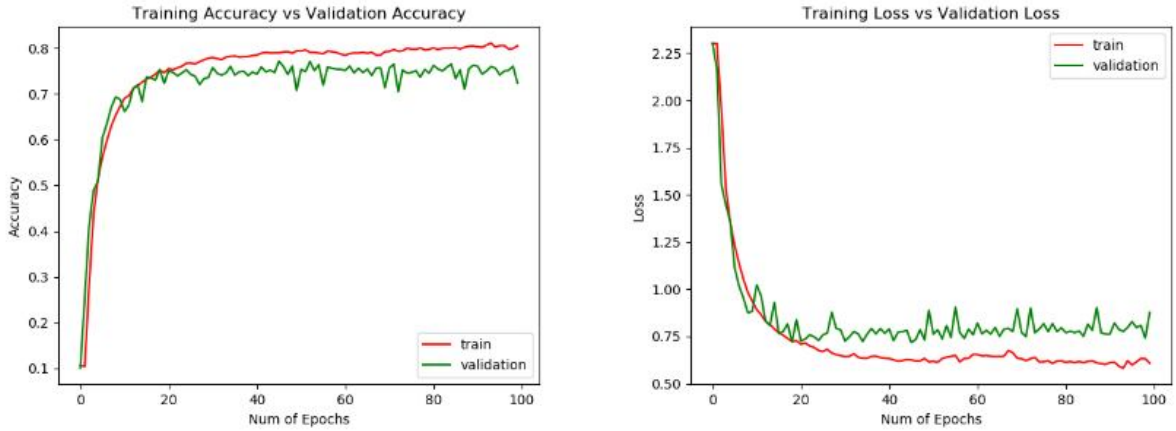

Figure 4- 14 Précision et erreur pour le modèle 1 (100 époques)

D'après la Figure 4-14, la précision de l'apprentissage et de la validation augmente dans l'intervalle du nombre d'époques [0-20] et dans l'intervalle [20-70], la validation reste dans le niveau (0.7) et l'apprentissage se stabilise au niveau (0.75) , mais dans l'intervalle [70-100] nous avons remarqué que la précision de l'apprentissage augmente et la validation reste stable dans le même niveau.

De même, l'erreur d'apprentissage et de la validation diminue dans l'intervalle du nombre d'époques [0-20] et dans l'intervalle [20-100], la validation se stabilise dans le niveau (0.75) et l'apprentissage reste stationnaire au niveau (0.6).

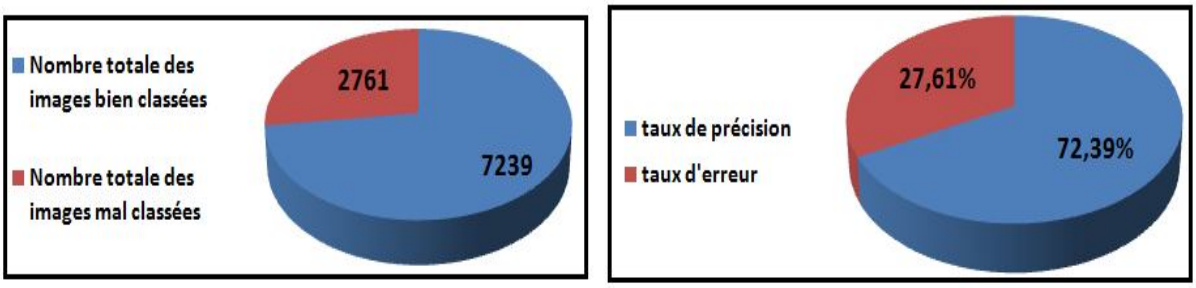

Figure 4- 15 Nombre total des images mal et bien classées et taux d'erreur et de précision de modèle 1 (100 époques)

D'après la figure 4-15 nous remarquons que la totalité des images mal classées est de 2761 images, un taux d'erreur de 27,61% et la totalité des images bien classées est de 7239 donc un taux de précision de 72,39%.

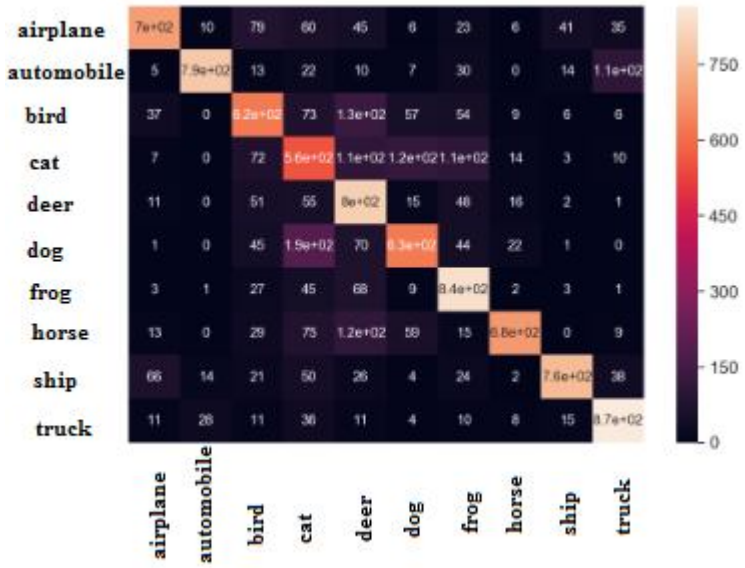

Figure 4- 16 Matrice de confusion pour le modèle 1 (100 époques)

La matrice de confusion permet d'évaluer la performance de notre modèle, puisqu'elle reflète les métriques du Vrai positif, Vrai négatif, Faux positif et Faux négatif. La figure 4-16 illustre de près la position de ces métriques pour chaque classe. A titre d'exemple le modèle a bien classé les images deer, truck et frog et il a mal classé les images cat.

B. Résultats obtenus pour le modèle 2 (6 couches)

B.1. Nombre d'époques = 10

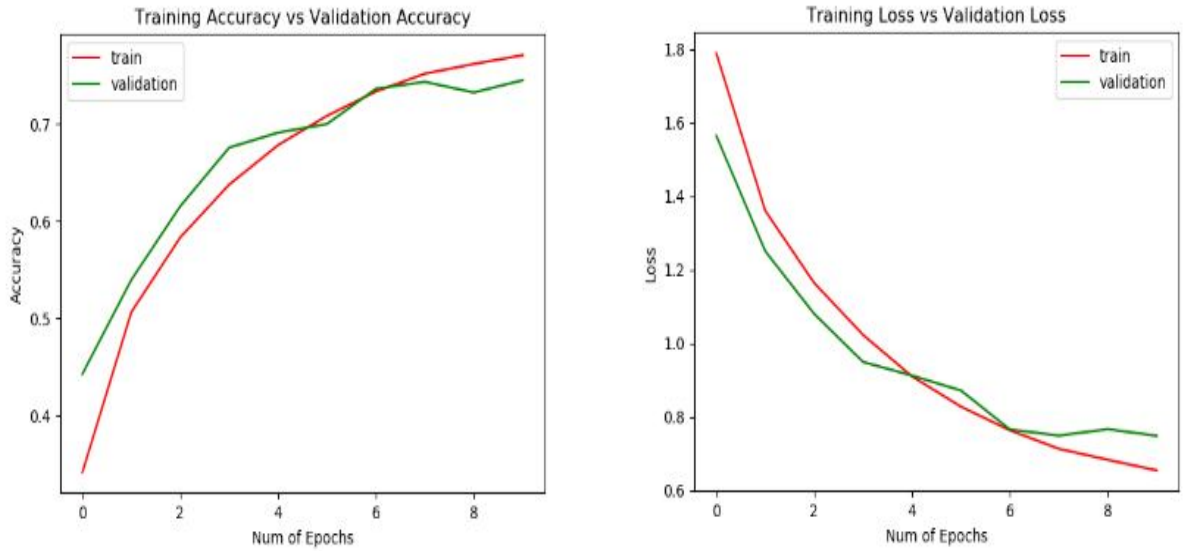

Figure 4- 17 Précision et erreur pour le modèle 2 (10 époques)

D'après la Figure 4-17 La précision de l'apprentissage et de la validation augmente avec le nombre d'époques.

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

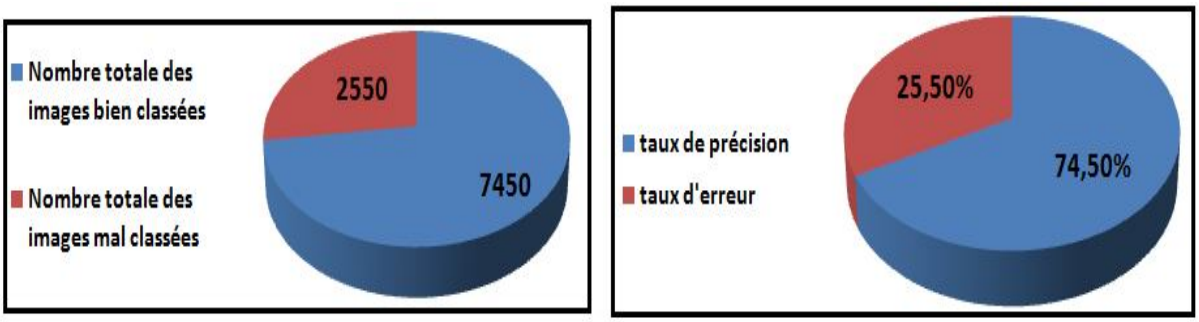

Figure 4- 18 Nombre total des images mal et bien classées et taux d'erreur et de précision de modèle 2 (10 époques)

D'après la figure 4-18 nous remarquons que la totalité des images mal classées est de 2550 images, un taux d'erreur de 25,50% et la totalité des images bien classées est de 7450 donc un taux de précision de 74,50%.

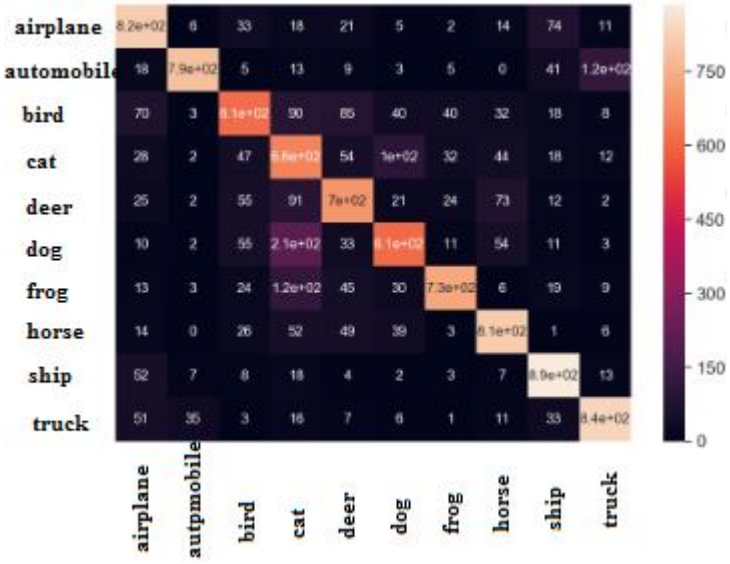

Figure 4- 19 Matrice de confusion pour le modèle 2 (10 époques)

La matrice de confusion permet d'évaluer la performance de notre modèle, puisqu'elle reflète les métriques du Vrai positif, Vrai négatif, Faux positif et Faux négatif. La figure 4-19 illustre de près la position de ces métriques pour chaque classe. A titre d'exemple le modèle a bien classé les images truck, horse, ship et airplane et il a mal classé les images bird et dog.

B.2 Nombre d'époques = 20

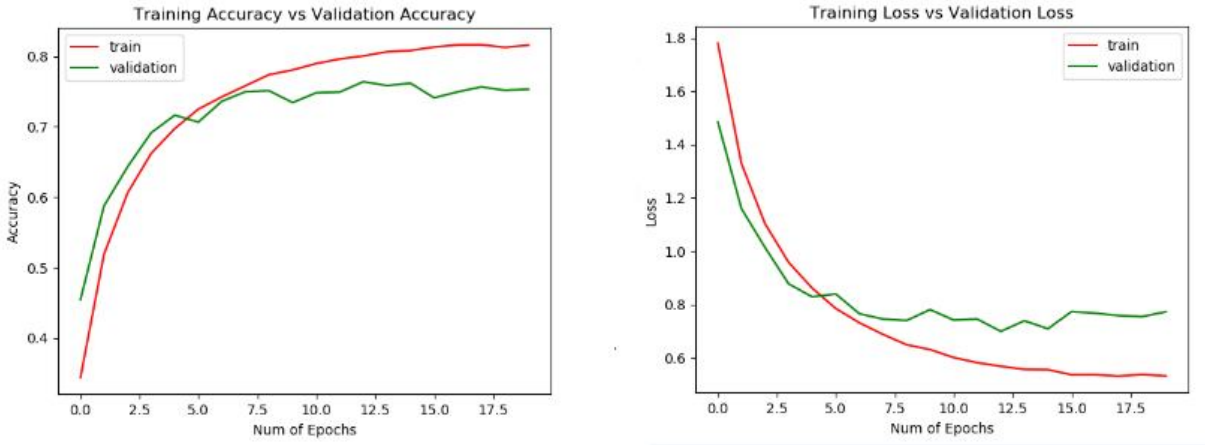

Figure 4- 20 Précision et erreur pour le modèl 2 (20 époques)

D'après la Figure 4-20 La précision de l'apprentissage et de la validation augmente dans l'intervalle du nombre d'époques [0-7.5], et dans l'intervalle [7.5-20], l'apprentissage augmente et la validation se stabilise au niveau (0.75).

De même, l'erreur d'apprentissage et de la validation diminue dans l'intervalle du nombre d'époques [0-7.5], et dans l'intervalle [7.5-20], l'apprentissage diminue et la validation se stabilise au niveau (0.8).

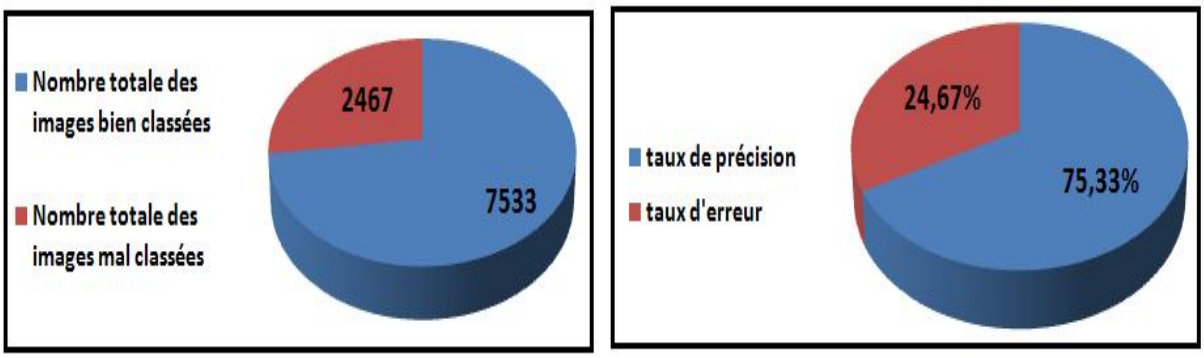

Figure 4- 21 Nombre total des images mal et bien classées et taux d'erreur et de précision de modèle 2 (20 époques)

D'après la figure 4-21 nous remarquons que la totalité des images mal classées est de 2467 images, un taux d'erreur de 24,67% et la totalité des images bien classées est de 7533 donc un taux de précision de 75,33%.

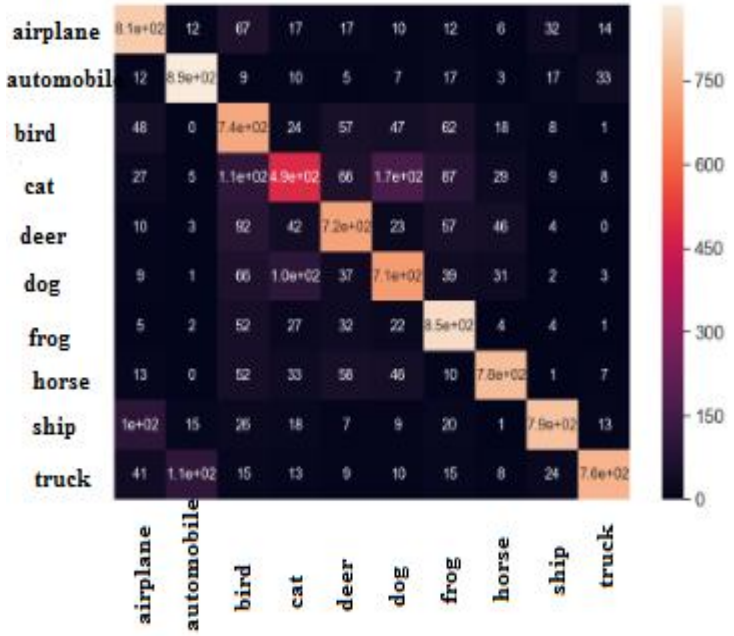

Figure 4- 22 Matrice de confusion pour le modèle 2 (20 époques)

La matrice de confusion permet d'évaluer la performance de notre modèle, puisqu'elle reflète les métriques du Vrai positif, Vrai négatif, Faux positif et Faux négatif. La figure 4-22 illustre de près la position de ces métriques pour chaque classe. A titre d'exemple le modèle a bien classé les images automobiles, airplane et frog et il a mal classé les images cat.

B.3. Nombre d'époques = 50

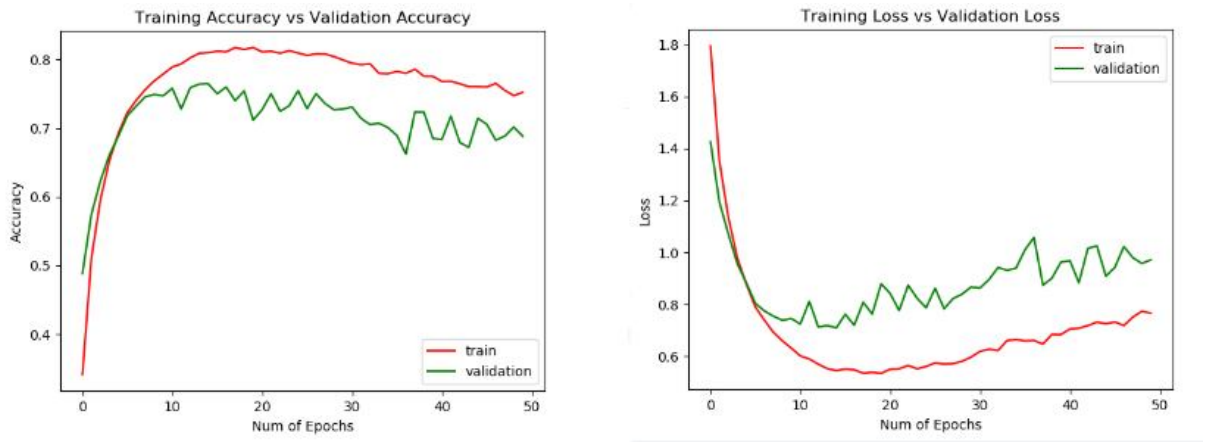

Figure 4- 23 Précision et erreur pour le modèle 2 (50 époques)

D'après la Figure 4-23 La précision de l'apprentissage et de la validation augmente dans l'intervalle du nombre d'époques [0-10] et dans l'intervalle [10-30], la validation reste stationnaire dans le niveau (0.75) et l'apprentissage se stabilise au niveau (0.8), mais dans l'intervalle [30-50], nous avons remarqué que la précision de l'apprentissage et de la validation diminue.

De même, l'erreur d'apprentissage et de la validation diminue dans l'intervalle du nombre d'époques [0-10] et dans l'intervalle [10-20], la validation reste stationnaire dans le niveau (0.8) et l'apprentissage se stabilise au niveau (0.6), mais dans l'intervalle [20-50] nous avons remarqué que la précision de l'apprentissage et de la validation augmente.

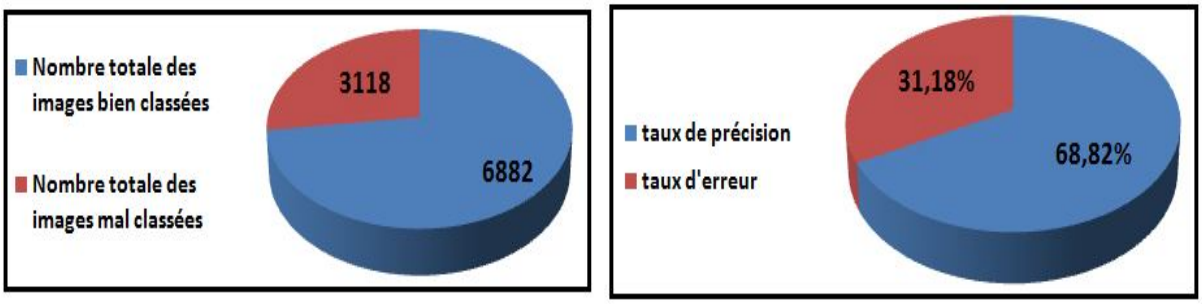

Figure 4- 24 Nombre total des images mal et bien classées et taux d'erreur et de précision de modèle 2 (50 époques)

D'après la figure 4-24 nous remarquons que la totalité des images mal classées est de 3118 images, un taux d'erreur de 31,18% et la totalité des images bien classées est de 6882 donc un taux de précision de 68,82%.

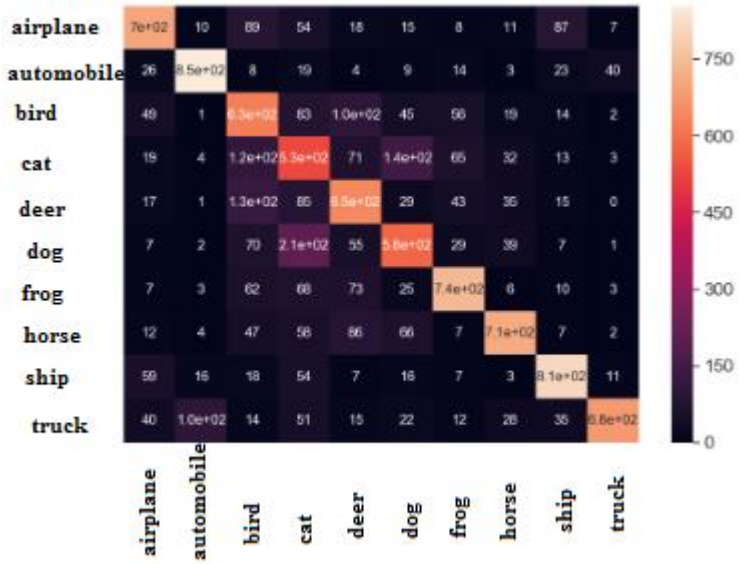

Figure 4- 25 Matrice de confusion pour le modèle 2 (50 époques)

La matrice de confusion permet d'évaluer la performance de notre modèle, puisqu'elle reflète les métriques du Vrai positif, Vrai négatif, Faux positif et Faux négatif. La figure 4-25 illustre de près la position de ces métriques pour chaque classe. A titre d'exemple le modèle a bien classé les images automobiles et ship et il a mal classé les images cat et dog.

B.4. Nombre d'époques = 100

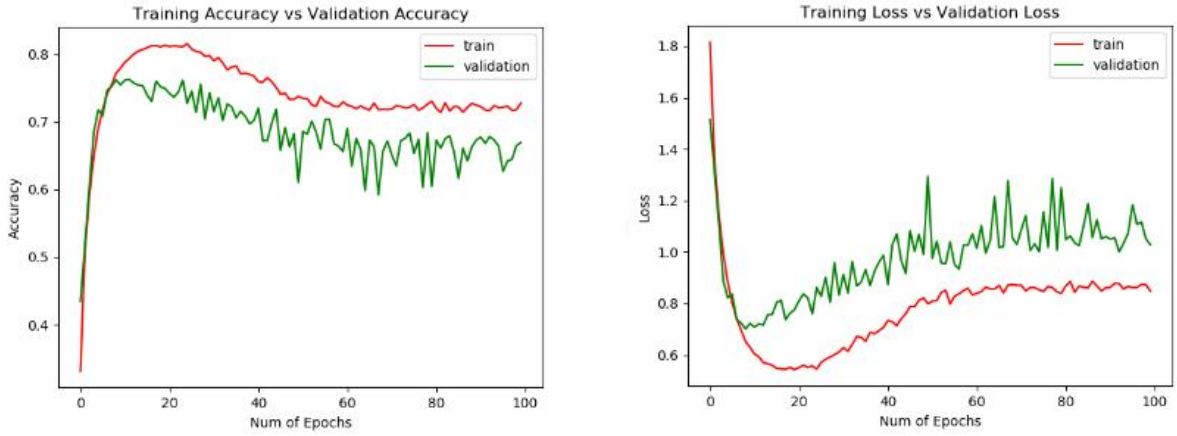

Figure 4- 26 Précision et erreur pour le modèle 2 (100 époques)

D'après la Figure 4-26, la précision de l'apprentissage et de la validation augmente dans l'intervalle du nombre d'époques [0-10] et dans l'intervalle [10-30], la validation reste stationnaire dans le niveau (0.75) et l'apprentissage se stabilise au niveau (0.8), mais dans l'intervalle [30-50], nous avons remarqué que la précision de l'apprentissage et de la validation diminue et dans l'intervalle [50-100], la validation reste stationnaire dans le niveau (0.65) et l'apprentissage se stabilise au niveau (0.75).

De même, l'erreur d'apprentissage et de la validation diminue dans intervalle de nombre d'époque [0-15] et dans l'intervalle [15-25], la validation reste stationnaire dans le niveau (0.75) et l'apprentissage se stabilise au niveau (0.5), mais dans l'intervalle [25-55], nous avons remarqué que la précision de l'apprentissage et de la validation augmente et dans l'intervalle [55-100], la validation reste stationnaire dans le niveau (1.0) et l'apprentissage se stabilise au niveau (0.8).

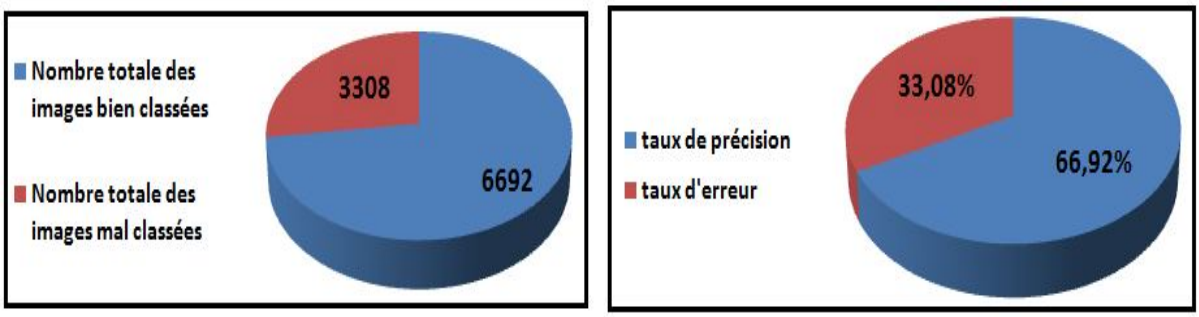

Figure 4- 27 Nombre total des images mal et bien classées et taux d'erreur et de précision de modèle 2 (100 époques)

D'après la figure 4-27, nous remarquons que la totalité des images mal classées est de 3308 images, un taux d'erreur de 33,08% et la totalité des images bien classées est de 6692 donc un taux de précision de 66,92 %.

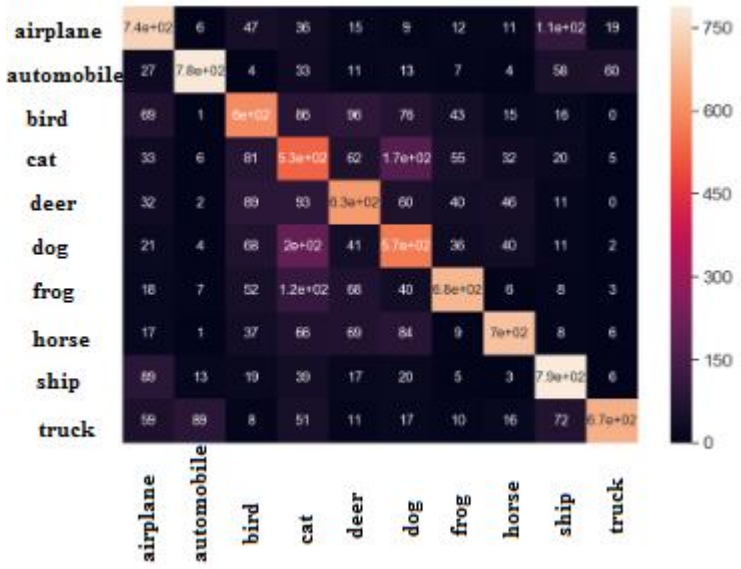

Figure 4- 28 Matrice de confusion pour le modèle 2 (100 époques)

La matrice de confusion permet d'évaluer la performance de notre modèle, puisqu'elle reflète les métriques du Vrai positif, Vrai négatif, Faux positif et Faux négatif. La figure 4-28 illustre

de près la position de ces métriques pour chaque classe. A titre d'exemple le modèle a bien classé les images automobiles et ship et il a mal classé les images cat et dog.

C. Tableau de comparaison des résultats

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

	Architecture utilisé			Nombre d'époques	Précision	Erreur	Temps d'exécution	
	Couche de convolution	Couche de pooling	Couche de fully connected				CPU	GPU
Modèle1 (4couches)	4	2	2	10	67,19%	32,81%	10h	18min
				20	72,89%	27,11%	20h	37min
				50	75,05%	24,95%	2j et 2h	1h et 34min
				100	72,39%	27,61%	4j et 4h	3h et 9min
Modèle2 (6couches)	6	3	2	10	74,50%	25,50%	20h	30min
				20	75,33%	24,67%	1j et 16h	1h
				50	68,82%	31,18%	4j et 4h	2h et 30min
				100	66,92%	33,08%	8j et 8h	5h

Tableau 4- 3 Tableau de comparaison des résultats

Le tableau montre l'architecture utilisée dans chaque modèle ainsi que le nombre d'époques utilisé. Les résultats obtenus sont exprimés en termes de précision, erreur et enfin de temps d'exécution. Ce dernier est partagé en deux sections, la première est "CPU" qui consiste à exécuter notre modèle sur le processeur de notre machine et la deuxième est "GPU" qui consiste à utiliser notre carte graphique à la place du processeur pour lancer notre modèle.

Pour notre premier modèle (4 couches), on remarque qu'à chaque fois qu'on augmente le nombre d'époques, le taux de précision augmente et le taux d'erreur diminue, on remarque aussi que ceci n'est pas proportionnel car arrivé à un certain seuil d'époques, ça commence à se stabiliser et l'augmentation n'est pas aussi importante qu'au début, puis le taux de précision commence à diminuer.

En ce qui concerne notre deuxième modèle (6 couches), 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 1%), mais arrivé à un certain seuil d'époques, le taux de précision ainsi que le taux d'erreur diminuent fortement.

Si on compare nos deux modèles, on remarque que pour 10 époques le modèle 2 donne de meilleurs résultats (une différence de 7%) et ceci grâce à l'architecture plus profonde de ce modèle et quand le nombre d'époques passe à 20, on remarque que l'augmentation est significative en ce qui concerne le premier modèle et on enregistre un taux de 5% , le deuxième modèle n'a fait qu'1% d'augmentation mais reste encore meilleur que le premier.
A 50 époques, le premier modèle continue à augmenter en taux de précision et fait un +3%, tandis que le deuxième modèle baisse d'un seul coup de 7% et enfin quand on compare les 100 dernières époques pour les 2 modèles, on remarque que les deux derniers ont diminué significativement de 2% presque.

En ce qui concerne le temps d'exécution, que ça soit pour CPU ou GPU, il augmente proportionnellement en augmentant le nombre d'époques.
Le temps d'exécution du CPU est très couteux par rapport à celui du GPU, ceci revient à la grande dimension de la base de données, ce qui nécessite l'utilisation d'un GPU au lieu d'un CPU.

D'une manière générale, un réseau neurones convolutionnel important et profond, donne des bons résultats et la performance de notre réseau se dégrade si une couche convolutive ou plusieurs sont supprimées. Par exemple, d'après le tableau, l'élimination de l'une des deux couches intermédiaires entraîne une perte d'environ 5% pour la performance du réseau, mais aussi il faut prendre en compte le nombre d'époques utilisé à chaque fois. Donc la profondeur est primordiale pour atteindre des bons résultats et avec un minimum d'époques.

4.7. Conclusion

Nous avons présenté dans ce chapitre une approche de classification basée sur les réseaux neurones convolutionnels, pour cela, on a utilisé deux modèles avec différentes architectures et nombre d'époques puis on a interprété les différents résultats obtenus en termes de précision, d'erreur et en temps d'exécution entre CPU et GPU. 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.

Conclusion générale

Dans ce projet, nous avons discuté des notions fondamentales de l'apprentissage automatique, des algorithmes les plus populaires, 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 correction, couche de pooling et couche fully connected).

On a implémenté deux modèles de réseaux neurones convolutifs avec différentes architectures; le premier qui a 4 couche de convolution et le second a 6 couche de convolution et par la suite ,on a appliqué pour chacun de ces modèles des tests qui consistent à changer le nombre d'époques à chaque fois, calculer le temps d'exécution pour chacun des tests et afficher à la fin la matrice de confusion pour récupérer le résultat des image bien et mal classées .

L'implémentation a été faite avec le langage de programmation python et on a utilisé des bibliothèques pour faciliter la tâche de création de nos modèles et pour l'accélération du training et enfin on a terminé avec un tableau récapitulatif et comparatif des deux modèles.

Finalement, on peut voir les choses en plus loin quant à l'utilisation des CNN, on peut par exemple créer une application en temps réel, ou utiliser un modèle pré entrainé pour avoir un meilleur résultat.

Bibliographie

a. Bibliographie :

[1] CHOLLET Francois. Deep learning with python. 2017

[2] ZACCONE Giancarlo, MD REZAUL Karim, MENSHAWY Ahmed. Deep learning with tensorflow. 2017

[3] DJOKHRAB Ala eddine. Planification et optimisation de trajectoire d'un robot manipulateur à 6 ddl par des techniques neuro_floues. 2015

[4] PARIZEAU Marc. Réseaux de neurones. 2004

[6] ZOCCA Valentino, SPACAGNA Gianmario, SLATER Daniel, ROELANTS Peter. Python deep learning. 2017

[7] NAKAMOTO Pat. Neural networks and deep learning : deep learning explained to your granny a visual introduction for beginners who want to make their own deep learning neural network. 2017

[8] TOUZET Claude. Les réseaux de neurones artificiels, introduction au connexionnisme. 1992

[9] HAWKINS Jeff, BLAKESLEE Sandra. On intelligence : how a new understanding of the brain will lead to the creation of truly intelligent machines. 2007

[10] SMOLENSKY Paul. Information processing in dynamical systems: foundations of harmony theory. 1986

[11] HINTON Geoffrey E. Deep belief networks. 2009

[12] GRAVES Alex, LIWICKI Marcus, FERNANDEZ Santiago, BERTOLAMI Roman, BUNKE Horst, SCHMIDHUBER Jurden. A novel connectionist system for unconstrained handwriting recognition. 2009

[13] FUKUSHIMA Kunihiko, MAYAKE Sei. Neocognitron: a self-organizing neural network model for a mechanism of visual pattern recognition. 1982

[14] HUBEL David hunter, WIESEL Torsten nils. Ferrier lecture-functional architecture of macaque monkey visual cortex. 1977

[15] LECUN Yann, BOTTOU Leon, BENGIO Yoshua, HAFFNER Patrick. Gradient-based learning applied to document recognition. 1998

[16] CIRESAN Dan C, MEIER Ueli, MASCI Jonathan, MARIA GAMBARDELLA Luca, SCHIDHUBER Jurden. Flexible, high performance convolutional neural networks for image classification. 2011

[17] O'SHEA Keiron, NASH Ryan. An introduction to convolutional neural networks. 2015

[18] WANG Peng, XU Jiaming, XU Bo, LIU Chenglin, ZHANG Heng, WANG Fangyuan HAO Hongwei. Semantic clustering and convolutional neural networks for short text categorization. 2015

---

b. Webographie :

[5] https://towardsdatascience.com/the-random-forest-algorithm-d457d499ffc Consultez le 25/ juin/2018

[19] http://adventuresinmachinelearning.com/convolutional-neural-networks-tutorial-tensorflow/ Consultez le 2/aout /2018

[20] https://www.python.org/ Consultez le 15/aout/ 2018

[21] https://www.tensorflow.org/ Consultez le 19/aout /2018

[22] https://keras.io/ Consultez le 22/aout /2018

[23] https://developer.nvidia.com/cuda-zone Consultez le 28/aout /2018

[24] //https://developer.nvidia.com/cudnn Consultez le 29/aout /2018

Resumé

La classification automatique des images consiste à affecter automatiquement une classe à une image en utilisant un système de classification. On retrouve ainsi la classification des objets, des scènes, la reconnaissance des visages, des empreintes digitales et des personnages. Ce projet consiste à utiliser et créer un classificateur d'images avec deux modèles de réseaux neurones convolutionnels différents et à évaluer le meilleur d'entre eux en modifiant des hyper paramètres et en ajoutant plus de profondeur au réseau.

Abstract

Automatic image classification consists of automatically assigning a class to an image using a classification system. We thus find the classification of objects, scenes, recognition of faces, fingerprints and characters. This project involves using and creating an image classifier with two different convolutional neural network models and evaluating the best between them by modifying hyperparametre and adding more depth to the networks.

ملخص

التصنيف التلقائي للصور يصنف صورة في فئة تلقائيًا باستخدام نظام تصنيف. وهكذا نجد تصنيف الأشياء والمشاهد و تعرف على الوجوه وبصمات الأصابع والشخصيات. يتضمن هذا المشروع استخدام مصنّف للصورة وإنشاءه باستخدام نموذجين مختلفين للشبكة العصبية الالتفافية وتقييم أفضل ما بينهما من خلال تعديل إعدادات وإضافة المزيد من العمق إلى الشبكة.