

REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE  
MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE  
SCIENTIFIQUE  
UNIVERSITE AKLI MOHAND OULHADJ-BOUIRA



Faculté des Sciences et des Sciences Appliquées  
Département : **Génie Electrique**

**Mémoire de fin d'étude**

Présenté par :  
**Lamri Abd elbasset**  
**Kramdi Mustapha**

Filière : **Génie Electrique**  
Option : **Systèmes des Télécommunications**

**Thème :**

Reconnaissance des chiffres manuels par le  
réseau de neurones convolutifs (CNN)

**Devant le jury composé de :**

SAOUDI KAMAL	MCA	UAMOB	Président
KASMI REDA	MCB	UAMOB	Encadreur
SAOUD BELLAL	MAB	UAMOB	Examineur

Année Universitaire 2020/2021

# *Remerciements*

Nous rendons grâce à Dieu qui nous a donné l'aide, la patience, le courage pour accomplir ce travail et nous a maintenu en santé pour mener à bien cette année d'étude.

Nous tenons à remercier notre promoteur, M.Kasmi Reda , pour nous avoir proposé ce travail, pour l'aide qu'il a fourni et les connaissances qu'il a su nous transmettre. Nous le remercions également pour sa disponibilité et la qualité de ses conseils.

Nos remerciements s'adressent aussi aux Mrs les jurés pour l'intérêt qu'ils ont porté à ce travail en acceptant d'être examinateurs.

Nous tenons à saisir cette occasion et adresser nos profonds remerciements et nos profondes reconnaissances aux responsables et au personnel de l'AMOB. Nous désirons aussi remercier les professeurs de l'AMOB, qui nous ont fournis les outils nécessaires à la réussite de nos études universitaires.

Un grand merci à nos mamans et nos papas, pour leur amour, leurs conseils ainsi que leur soutien inconditionnel à la fois moral et économique, ainsi qu'à nos sœurs et frères et toutes nos familles, qui nous ont permis de réaliser les études que nous voulions et par conséquent ce mémoire.

Nous voudrions exprimer nos reconnaissances envers les amis et collègues qui nous ont apporté leur soutien moral et intellectuel tout au long de notre démarche.

# *Dédicace*

Merci Allah de m'avoir donné la capacité d'écrire et de réfléchir, la force d'y croire, la patience d'aller jusqu'au bout du rêve.

Je dédie ce modeste travail à ceux qui m'ont encouragé et soutenu, je cite :  
Les parents les plus chers au monde, papa et maman, que dieu les garde et les protège.

A ma sœur et mes frères qui je t'amie beaucoup  
Et surtout mes meilleurs amis, qui m'ont toujours encouragé, et à qui je souhaite plus de succès.

À mon binôme **Abd el basset**

A la promotion de département de Genie-Electrique  
2021/2022 Et à toute personne ayant participé à l'élaboration  
de ce travail

***Mustapha***

# Dédicace

Je dédie ce travail à tous ceux qui comptent  
pour moi, À mes très chers parents,

Aucun hommage ne saurait exprimer mon amour éternel, ma  
reconnaissance et ma considération pour les sacrifices que vous avez  
consentis pour mon éducation et mon bien être.

Que ce modeste travail soit l'exaucement de vos vœux tant formulés,  
le fruit de vos innombrables sacrifices, bien que je ne vous en  
acquitterai jamais assez.

Puisse Dieu, le Très Haut, vous accorder santé, bonheur et longue vie et  
faire en sorte que jamais je ne vous déçoive.

À mon cher frère « Abdelfettah », et mes chères sœurs « Melissa » et «  
Ahlam

», En témoignage de mon affection fraternelle et de ma profonde  
tendresse, je vous souhaite une vie pleine de bonheur et de succès et  
que Dieu, le tout puissant, vous protège et vous garde.

À mon binôme « **Mustapha** »,

À toutes les personnes qui ont participé à l'élaboration de ce travail.

1. À tous mes amis(es) de la promotion 2021/2022, en particulier  
« Yasmine ».

À tous ceux que j'ai omis de citer.

***Abd el basset***

# Table des matières

Remerciement .....	01
Dédicace .....	02
Table des matières.....	04
Liste des figures .....	07
Introduction générale .....	08
Chapitre 1 : Reconnaissance d'écriture .....	09
1.Introduction.....	09
1.1 La Reconnaissance de Chiffres Manuscrits (RCM) .....	09
2 Outils pour la reconnaissance automatique .....	09
3 Le type de l'écriture.....	09
4 Classification et apprentissage.....	10
4.1 Les réseaux de neurones .....	10
4.1.1 Architecture des reseau de neurone.....	11
4.2 SVM .....	12
4.2.1 Avantages.....	13
4.2.2 Inconvénients .....	14
4.3 Méthode k-plus proche voisins .....	14
4.3.1 Avantages.....	14
4.3.2 Inconvénients .....	15
5 Apprentissage Profond.....	15
5.1 L'objectif Principal.....	15
5.2Prétraitement des donnees .....	16
6Les réseaux de neurone convolutifs.....	17
6.1Architecture du CNN.....	18
6.2Les différents modules d'un réseau de neurones convolutifs.....	19

6.2.1 La convolution.....	19
6.2.2 Le Pooling.....	22
6.2.3 Les fonctions d'activations.....	23
6.2.4 Le dropout.....	24
6.2.5 La batch normalisation .....	24
6.2.6 La fonction de perte Softmax.....	25
7. Conclusion .....	25
Chapitre 2 .....	26
1. Introduction.....	26
2. Implémentation et résultats.....	26
3. Construire un premier réseau neurones .....	28
4. CNN :Réseau de neurones Convolutif .....	30
5. Conclusion .....	35
Conclusion général .....	36
Annexe .....	37
Bibliographie .....	41

## Liste de figure :

Figure 1 : Neurone artificiel avec une seule sortie [9]

Figure 2 : Le réseau de neurones à un seul niveau [9]

Figure 3: Le réseau de neurones multi-niveaux [9]

Figure 4 : illustration SVM cas linéaires [12]

Figure 5 : illustration méthode Kppv[19].

Figure 6 : L'évolution de l'intelligence artificielle [22]

Figure 7 : Architecture CNN [16]

Figure 8 : Architecture CNN 2[16]

Figure 9 : Illustration de la convolution [17]

Figure 10 : Illustration du zeropadding [17]

Figure 11 : illustration de la convolution avec le padding et le stride [17]

Figure12 : illustration du pooling.[17]

Figure 13 : Quelques fonctions d'activations [17]



## Introduction générale :

L'intelligence artificielle (IA) vise à comprendre le fonctionnement de la cognition humaine et à la reproduire. Il fait référence à des technologies basées sur l'utilisation d'algorithmes. Ces technologies, qui présentent de multiples variantes, se caractérisent souvent par leur capacité prédictive. Il s'agit de doter les machines d'une intelligence et d'une autonomie qui leur sont propres.

L'IA existe dans des secteurs tels que la production industrielle, la médecine, les transports ou la sécurité. Les technologies recherchées concernent l'informatique, l'électronique, les mathématiques, les neurosciences et les sciences cognitives.

Le déploiement de l'IA poursuit divers objectifs tels que l'amélioration des conditions de vie des populations, la personnalisation des soins médicaux, la stimulation de l'innovation et de la productivité, la reconnaissance des codes et des chiffres manuscrits par des méthodes développées comme le deep learning, ces technologies peuvent parvenir à modifier les limites entre l'homme et la machine.

L'objectif de notre travail est de développer un programme pour reconnaître les chiffres manuscrits plus précisément les chiffres en utilisant des techniques de Réseau de Neurones Convolutif CNN.

Par conséquent, notre mémoire est subdivisée comme suit :

Dans le premier chapitre, nous avons invoqué quelques notions sur écriture manuscrite et la reconnaissance de chiffres et de l'écriture manuscrite puis nous parlerons sur reconnaissance automatique et les différents classificateurs.

La deuxième partie de premier chapitre a été consacré à la méthode de l'apprentissage profond et leur Principe et nous nous sommes surtout intéressés sur les réseaux de neurone convolutif.

Dans le deuxième chapitre nous exposerons la présentation et la discussions des différents résultats obtenus lors de la phase d'expérimentation.

# Chapitre 1 : Reconnaissance d'écriture

## 1 Introduction :

La reconnaissance de l'écriture manuscrite est l'un des plus anciens problèmes, depuis son apparition dans les années 1950. Un vrai challenge aux nouveaux algorithmes d'apprentissage, il reste un véritable défi scientifique et technique.

La reconnaissance de l'écriture manuscrite est un traitement informatique qui vise à traduire un texte écrit en texte encodé numériquement. [1]

### 1.1 La Reconnaissance de Chiffres Manuscrits (RCM) :

La reconnaissance de chiffres manuscrits est la conversion d'images que les humains peuvent comprendre en codes qui peuvent être interprétés par des ordinateurs. [2]

## 2 Outils pour la reconnaissance automatique

L'homme rêve aussi de communiquer avec la machine de la même manière qui semble plus simple et plus facile pour l'accélération des opérations d'accès, d'échange et de traitement de l'information. D'où l'émergence de la reconnaissance automatique de l'écriture manuscrite (RAE). [4]

## 3 Le type de l'écriture :

### 3.1 Les systèmes de reconnaissance de l'écriture imprimée :

C'est les étapes de transition de formulaire papier au formulaire de codage électronique.

### 3.2 Les systèmes de reconnaissance de l'écriture manuscrite :

C'est un moyen particulier de codage de l'information qui concerne nos idées, pensées, nos sentiments, d'une autre façon la reconnaissance de l'écriture en général.

## 4 Classification et apprentissage :

La classification est l'acte de trier par classes, par catégories, des objets ayant des propriétés communes. Il existe deux catégories de classification : la classification supervisée et la classification non supervisée. Dans la première catégorie, les méthodes consistent à classer les objets dans une base de données dite d'apprentissage, tandis que dans l'autre, les méthodes classent les objets sans avoir besoin d'apprentissage au préalable.

Diverses méthodes de classification ont été présentées dans la littérature scientifique. Nous vous présenterons les plus célèbres.

### 4.1 Les réseaux de neurones :

Les réseaux de neurones ont été développés comme un modèle mathématique générique afin de modéliser les neurones biologiques. Ils comportent un certain nombre d'éléments de traitement d'information appelés neurones. [8]

Chaque neurone a son propre état interne interprété par la fonction d'activation. Il envoie son activation aux autres neurones sous forme de signaux. La connexion entre les neurones est réalisée via des liens orientés et pondérés. [8]

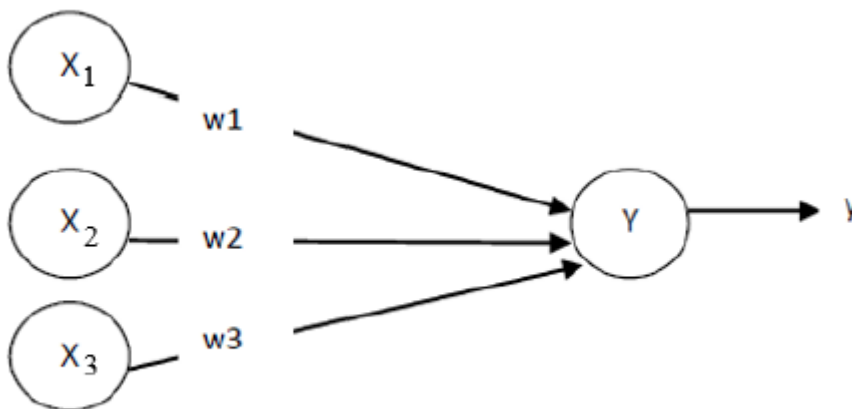
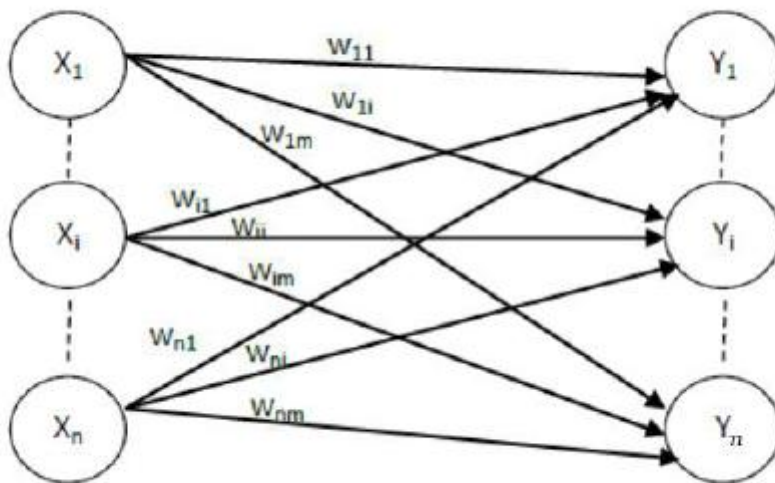


Figure 1: Neurone artificiel avec une seule sortie [9]

Sur la figure1 le neurone Y reçoit les entrées de X1, X2 et X3 qui ont comme valeurs de sortie  $x_1$ ,  $x_2$  et  $x_3$ . Les poids des liens de connexion de X1, X2 et X3 sont  $w_1$ ,  $w_2$  et  $w_3$ . La valeur d'entrée de neurone Y est :  $y = w_1x_1 + w_2x_2 + w_3x_3$ . Le signal de sortie  $y$  est déterminée par la fonction d'activation ( $y$ ). Les réseaux de neurones sont caractérisés par l'architecture (l'organisation des neurones), l'apprentissage (méthode de détermination des poids de connexions), et par leur fonction d'activation. [7]

#### 4.1.1 Architecture des réseaux de neurones :

Les réseaux de neurones sont souvent classés en deux architectures : les réseaux de neurones à un et les réseaux de neurones à plusieurs couches. [7]



Figur2. : Le réseau de neurones à un seul couche [9]

Les neurones de la couche d'entrée doivent uniquement passer et distribuer les entrées et ne pas effectuer de calcul. Ainsi, la seule vraie couche de neurones est celle de droite. Chacune des entrées  $X_1, X_2 \dots X_n$  est connectée à chaque neurone de la couche de sortie à travers le poids de lien.

Comme chaque valeur des sorties  $Y_1, Y_2 \dots Y_n$  est calculée à partir du même ensemble de valeurs d'entrée, chaque sortie est modifiée en fonction des poids de liens. [7, 10]

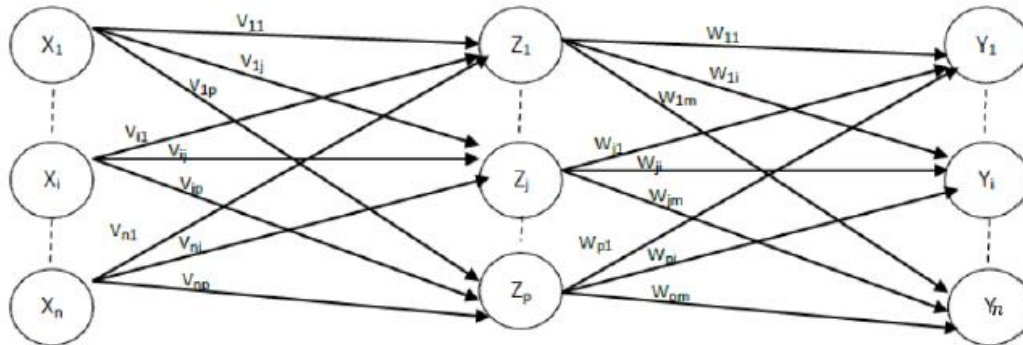


Figure3: Le réseau de neurones multi couches [9]

La figure 3 montre le réseau de neurones à plusieurs couches. Dans les deux structures, les nœuds d'entrée se transmettent les informations aux unités de la première couche cachée, puis les sorties de la première couche cachée sont transmises à la couche suivante, et ainsi de suite. [8, 10]

## 4.2 SVM

Cette technique est une méthode de classification à deux classes qui tente de séparer linéairement les exemples positifs des exemples négatifs dans l'ensemble des exemples. En voulant que la marge entre le plus proche des positifs et des négatifs soit maximale, le séparateur d'exemples positif et négatif appelé l'hyperplan est cherché par cette méthode. En tenant compte que de nouveaux exemples ne seront forcément pas similaires à ceux utilisés pour trouver l'hyperplan mais seront sûrement situés soit du côté négative ou positive ce qui garantirait une généralisation du principe. [11]

Le but des SVM est de trouver un séparateur entre deux classes qui soit au maximum éloigné de n'importe quel point des données d'entraînement. Si on arrive à trouver un séparateur linéaire c'est-à-dire qu'il existe un hyperplan séparateur alors le problème est dit linéairement séparable sinon il n'est pas linéairement séparable et il n'existe pas un hyperplan séparateur. [3]

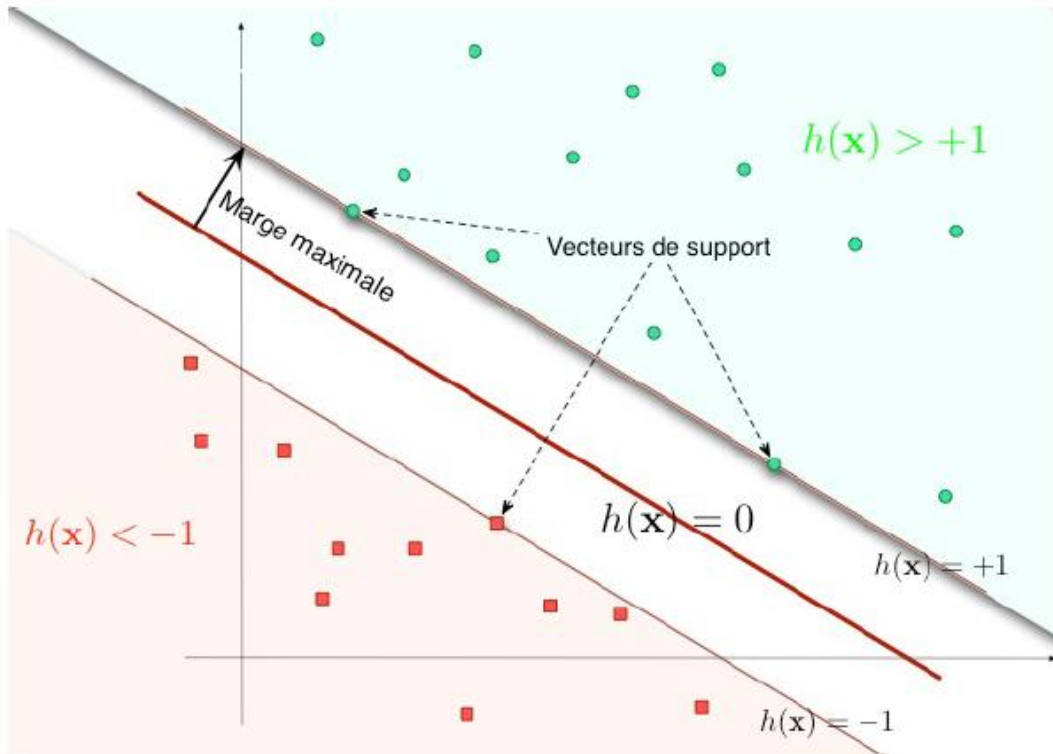


Figure4. Illustration SVM cas linéaires [12]

Les exemples utilisés lors de la recherche d'hypervoie ne seront plus utiles et pour classer un nouveau cas, seuls les vecteurs supports qui se représentent comme les vecteurs discriminants par lesquels l'hypervoie est déterminée seront utilisés.

#### 4.2.1 Avantages : [13]

- les exemples de test ne sont pas comparés avec tous les exemples d'apprentissage mais juste avec les supports vecteurs.
- Traitement des problèmes non linéaires avec le choix des noyaux.
- la capacité a traité des données de grandes dimensions (variable élève)
- Plus performant que la plupart des autres méthodes.

#### 4.2.2 Inconvénients : [13]

- Difficulté à identifier les bonnes valeurs des paramètres et sensibilité aux paramètres.
- Utilisation des points de support pour les noyaux non linéaire.
- Les classes bruitées posent des problèmes.
- Un calcul matriciel important dus aux grandes quantités. D'exemple en entrée.

#### 4.3 Méthode k-plus proche voisins :

Le classificateur des k plus proche voisin noté k-ppv (k-Nearest Neighbor ou k-NN), est l'un des classificateurs les plus simples, il prend un point de test sous forme vectorielle et trouve la distance euclidienne entre celle-ci et la représentation vectorielle de chaque d'exemple d'apprentissage. [14]

L'exemple d'entraînement le plus proche du point de test est appelé son voisin le plus proche, comme cet exemple et en quelque sorte le plus proche de notre point de test, il est logique d'attribuer son label de classe au point de test, cela exploite l'hypothèse de (régularité) selon laquelle les points proches les uns des autres sont susceptibles d'avoir la même classe. [18]

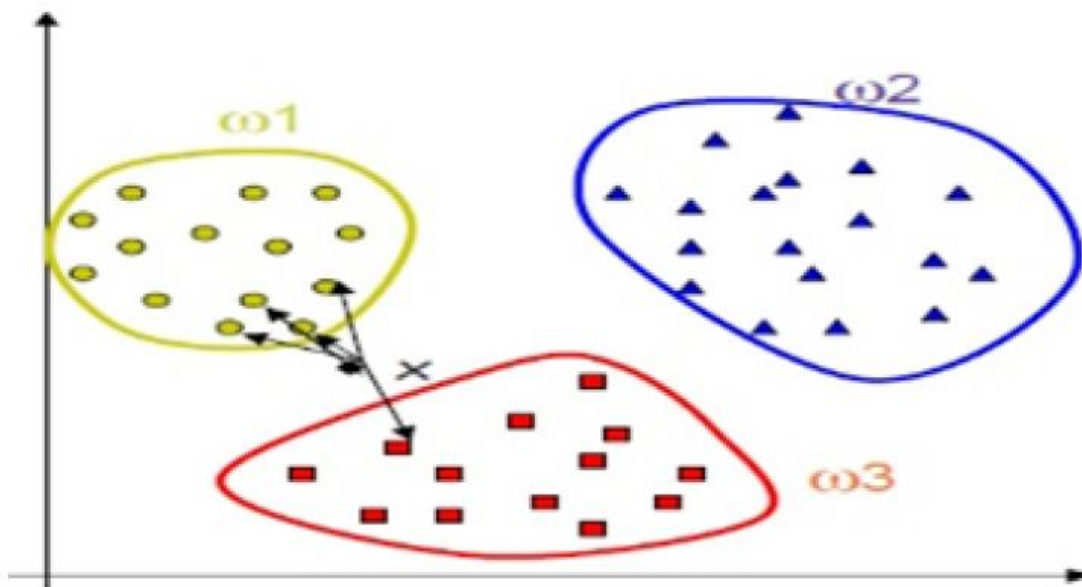


Figure5. illustration méthode Kppv[19].

#### 4.3.1 Avantages : [20]

- Fournit de bonne performance
- Apprentissage rapide

- Une facilité dans la conception et la compréhension de la méthode

#### 4.3.2 Inconvénients : [21]

-Faible vitesse de classification due au nombre important de distances a calculé

-Nécessite de garder la base de données sous la main ce qui induit à la lenteur de la prédiction méthode gourmande en place mémoire

- Sensibilité de la dimensionnalité et au attribut non pertinent

## 5 Apprentissage profond

Appelé aussi Deep learning, c'est un sous-ensemble de l'apprentissage automatique (AA) donc on peut dire que tout Apprentissage profond est apprentissage automatique mais pas le contraire. [15]

Sont architecture consiste en plusieurs couches de neurones.

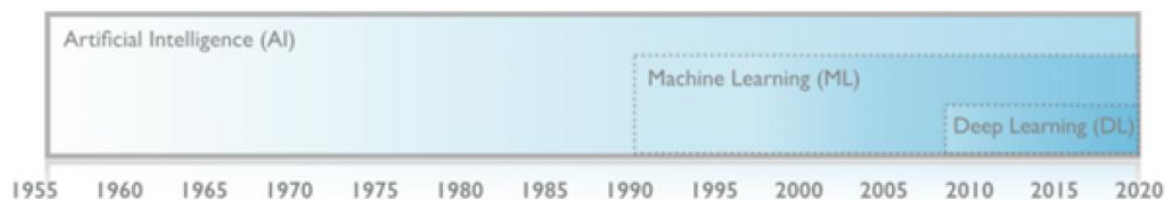


Figure 6. L'évolution de l'intelligence artificielle [22]

### 5.1 L'objectif principal :

le deep learning consiste à optimiser les poids des neurones dans chaque couche, c'est-à-dire que lorsqu'il reçoit une entrée cette dernière sera analysée puis il prendra une décision à ce sujet, si la prédiction est incorrecte l'algorithme ajuste les connexions entre les neurones, cela changera ces prédictions à l'avenir, même si le réseau donnera de mauvaises réponses plusieurs fois, mais au fur et à mesure que le réseau s'entraîne à travers des milliers d'exemples, les connexions entre les neurones s'ajusteront pour que le réseau de neurones prenne les bonnes décisions à chaque fois. [15]



## 5.2 Prétraitement des données :

Avant d'entrer des données dans des modèles d'apprentissage profond, ces données dites d'entrée passent généralement par une phase de pré-traitement pour rendre les procédures d'apprentissage plus faciles et plus efficaces.

Pour traiter les données d'entrée, nous nous appuyons sur différentes techniques de prétraitement, telles que les méthodes de débruitage et de transformation :

Le débruitage consiste en une suppression ou une réduction stochastique des données d'entrée, tandis que le traitement de transformation alloue l'espace de données d'entrée à un nouvel espace plus approprié.

Il existe plusieurs techniques de transformation de données, les plus utilisés sont :

La soustraction moyenne :

Il permet de centrer l'espace des caractéristiques des données autour d'une origine le long de chaque dimension, on soustrait la moyenne de chaque caractéristique individuelle des données.

La normalisation :

La normalisation des données d'entrée se fait de sorte que toutes les données aient approximativement la même échelle, de sorte que chaque dimension des données soit normalisée de sorte que les vecteurs des données finales tombent dans la plage  $[-1,1]$  ou  $[0, 1]$ .

La standardisation :

Cette technique consiste à configurer chaque dimension des données pour avoir une moyenne nulle et une unité de variance.

## 6 Les réseaux de neurone convolutif :

Convolutional Neural Networks a également souligné que CNN pour Convolutional Neural Network sont des réseaux multicouches d'anticipation qui ont été spécialement conçus pour la reconnaissance de caractéristiques dans des images bidimensionnelles.

Étant donné que les CNN sont utilisés pour la reconnaissance d'images 2D, l'image est composée de plusieurs pixels. Chaque pixel porte des informations sur la couleur, la couleur peut être représenté par des canaux RVB ou par un seul canal de niveaux de gris

Dans les réseaux convolutifs, les neurones travaillent en prenant en compte une petite partie de l'image appelée sous-image, afin de rechercher des caractéristiques reconnaissables par le réseau, les sous-images sont examinées, leurs caractéristiques sont captées par les cartes du caractéristiques respectives ( feature maps) des réseaux, ces feature maps sont le résultat d'un filtre appliqué sur la couche précédente.

Il existe deux types de couche, la couche convolutive (couche convolutive) et la couche de sous-échantillonnage (couche de regroupement ou de sous-échantillonnage). Identifier les caractéristiques de l'image d'entrée est l'objectif de la couche convolutive, cette couche est constituée de plusieurs cartes caractéristiques, pour la couche de sous-échantillonnage elle suit toujours la couche convolutive et se compose du même nombre de cartes caractéristiques ou de chaque carte. La couche convolutive est utilisée comme entrée pour la carte des caractéristiques de la couche de sous-échantillonnage..

Selon la profondeur du réseau, on alterne entre les couches de convolution et de sous échantillonnage jusqu'à ce que la dernière couche de cette dernière soit atteinte, il peut y avoir un certain nombre de couches entièrement connecté comme décrit dans la section précédente et en ont dernier lieux on a la couche de sortie. [15][16]

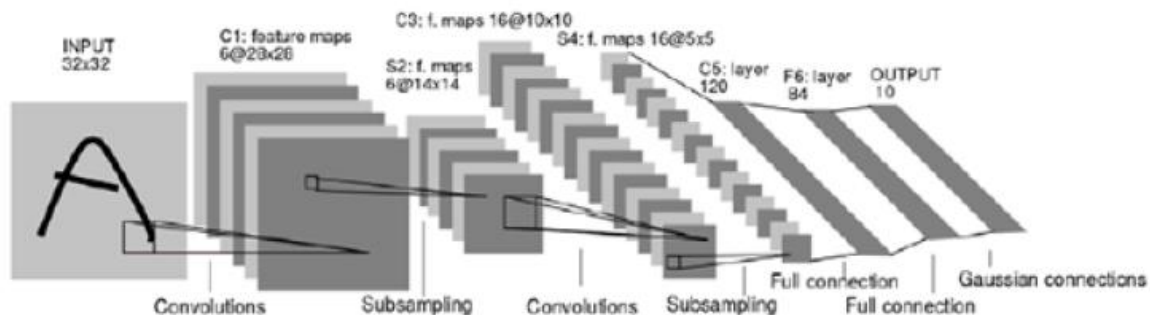


Figure 7. Architecture CNN [16]

## 6.1 Architecture du CNN :

Un réseau de neurones convolutifs se compose de deux parties essentielles, où chaque partie a un rôle à jouer et un objectif à atteindre, la première partie (extraction de caractéristiques) est responsable de l'extraction des caractéristiques, des couches de convolution et des sous-couches. L'échantillonnage alterne en Alors que la seconde partie effectue la classification selon les caractéristiques extraites dans la partie précédente, dans cette partie un perceptron multicouche entièrement connecté (MLP) est généralement utilisé.

Le processus d'extraction de caractéristiques est effectué comme suit : dans l'image capturée, chaque pixel est vu comme une entité pour les neurones regroupés dans les cartes de caractéristiques de la première couche convolutive qui sont disposées dans une grille à deux dimensions, avec afin d'optimiser la mise en oeuvre. En nécessitant moins de mémoire et en offrant de meilleures performances, tous les neurones qui se trouvent sur la même carte de caractéristiques partagent leur poids et garantissent que le même filtre est utilisé pour chaque pixel.

Dans chaque carte de caractéristiques donnée, chaque neurone doit reconnaître la même caractéristique. Notez que la couche d'entrée peut être considérée comme une couche de sous-échantillonnage avec une seule carte de caractéristiques, le but des couches de sous-échantillonnage est de réduire les cartes de caractéristiques.

La deuxième partie du réseau commence le processus de classification en utilisant un réseau MLP entièrement connecté. [15,16]

## 6.2 Les différents modules d'un réseau de neurones convolutif :

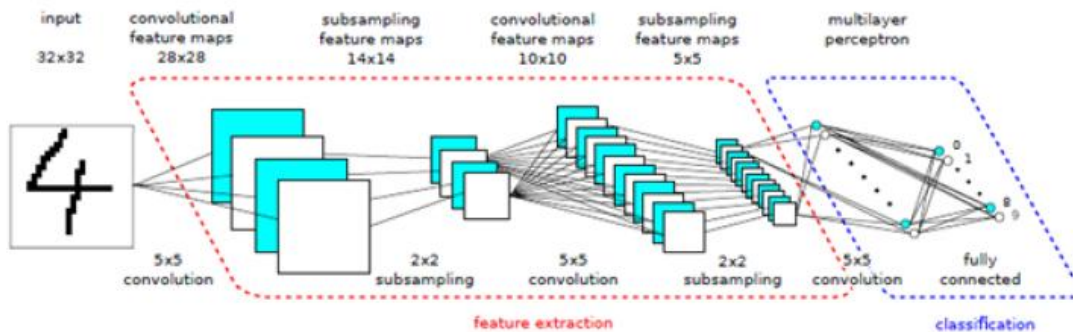


Figure 8. Architecture CNN 2[16]

Dans cette section nous allons présenter les différents modules utilisés dans les CNN :

### 6.2.1 La convolution :

La pierre maitresse d'un CNN est la couche convolutive, son but est de détecter les caractéristiques des données en entrée, donc cette opération se réside à appliquer un noyau ou un filtre de convolution sur une matrice d'entrée (une image ou carte de caractéristique précédente) a plusieurs emplacements de la donnée en entrée et tout en extrayant les caractéristiques abstraites de chaque partie de l'entrée.

Les filtres utilisés sont appliqués sur toute l'entrée, des caractéristiques plus complexes peuvent être extraite en utilisant plusieurs couches ainsi on aura des cartes de caractéristique qui seront prêtes à être utilisé par les couches de convolution suivantes.

Le nombre de cartes de caractéristique dans la couche convolutionnelle est une décision importante qui est prise en compte lors de la conception d'un CNN, il n'existe pas de valeur optimale, la meilleure façon de déterminer le nombre des cartes de caractéristiques est d'expérimenter, mais généralement un grand nombre de carte de caractéristique pour

reconnaitre des images très complexes tandis qu'un petit nombre serviraient pour les tâches dites simples.[15][16]

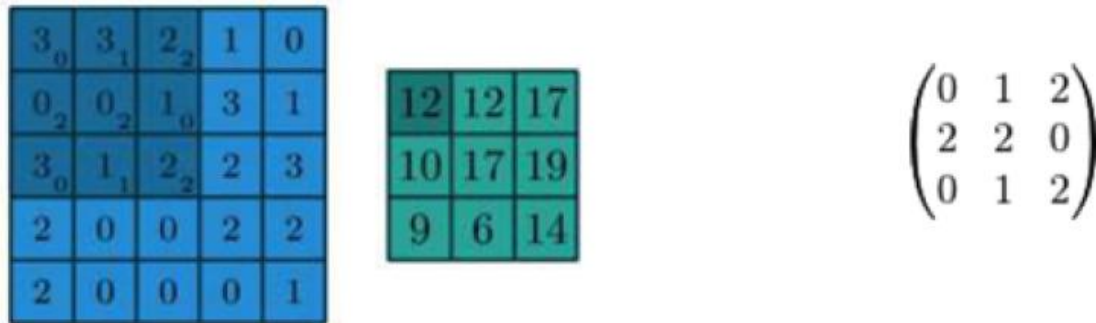


Figure 9. Illustration de la convolution [17]

Trois paramètres permettent de dimensionner le volume de la couche de convolution la profondeur, le pas et la marge.

-Profondeur de la couche :

Nombre de noyaux de convolution (ou nombre de neurones associés à un même champ récepteur).

-Le pas :

contrôle le chevauchement des champs récepteurs. Plus le pas est petit, plus les champs récepteurs se chevauchent et plus le volume de sortie sera grand.

- La marge (à 0) ou zeropadding :

Parfois, il est commode de mettre des zéros à la frontière du volume d'entrée. La taille de ce 'zero-padding' est le troisième hyper paramètre. Cette marge permet de contrôler la dimension spatiale du volume de sortie. En particulier, il est parfois souhaitable de conserver la même surface que celle du volume d'entrée. [17]

0 <sub>2</sub>	0 <sub>0</sub>	0 <sub>1</sub>	0	0	0	0
0 <sub>1</sub>	2 <sub>0</sub>	2 <sub>0</sub>	3	3	3	0
0 <sub>0</sub>	0 <sub>1</sub>	1 <sub>1</sub>	3	0	3	0
0	2	3	0	1	3	0
0	3	3	2	1	2	0
0	3	3	0	2	3	0
0	0	0	0	0	0	0

Figure 10. Illustration du zeropadding[17]

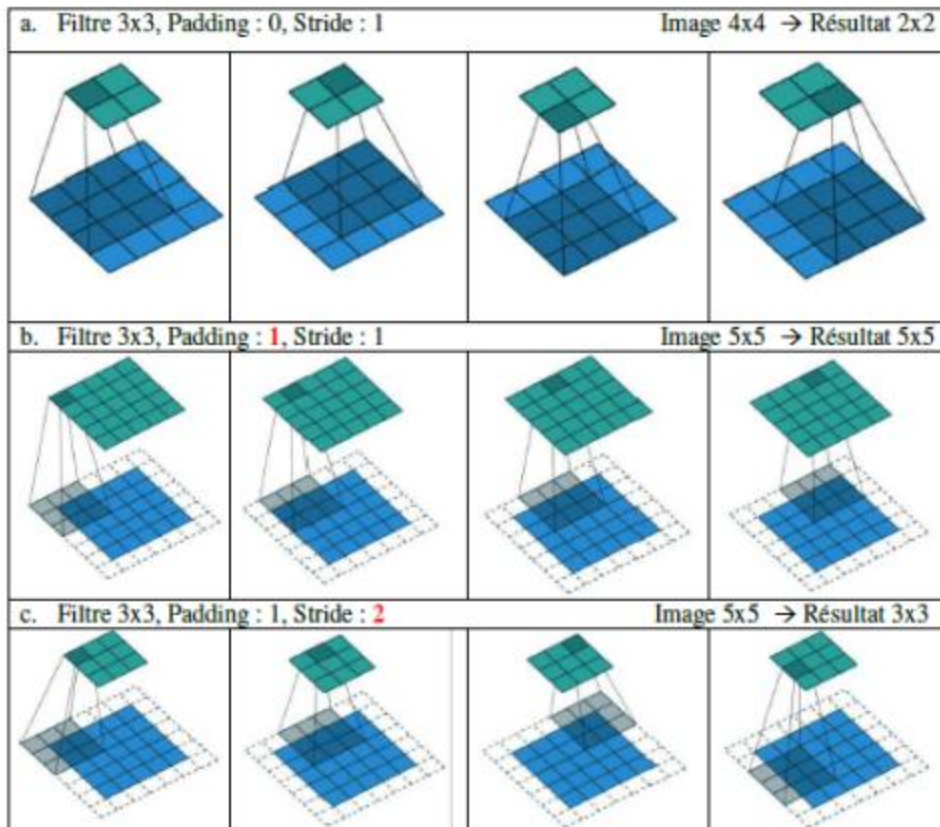


Figure 11. illustration de la convolution avec le padding et le stride[17]

### 6.2.2 Le pooling :

La couche d'agglomération (subsampling ou pooling) peut-être la couche d'entrée du réseau mais généralement elle vient après la couche de convolution, le but de cette couche est de réduire la dimension de ces entrées pour simplifier et généraliser les caractéristiques extraites, il existe différents types de pooling mais les plus utilisés sont le max pooling qui renvoie l'élément maximum sur une fenêtre de calcul et l'average pooling qui renvoie la moyenne des éléments sur une fenêtre de calcul.[17]

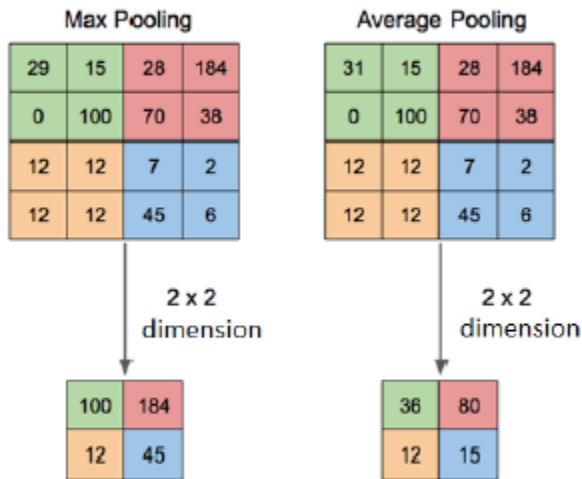


Figure 12. illustration du pooling[17]

### 6.2.3 Les fonctions d'activations :

Le choix de la fonction d'activation joue un rôle très important dans le comportement des réseaux de neurones, les fonctions d'activation les plus connues qui permettent la non-linéarité dans les couches de CNN sont le sigmoïde, la tangente hyperbolique et le RELU.

Sigmoïde :

C'est une fonction d'activation non linéaire simple à calculer et à différencier. Elle se définit en :

$$\text{sigmoid}(x) = \frac{1}{1 + e^{-x}} \quad (1)$$

-Hyperbolique tangente :

C'est une autre fonction non-linéaire, mais qui donne une convergence plus que la sigmoïde puisque la sortie de la fonction sigmoïde est centrée sur 0, ce qui favorise la fonction tanh pour l'entraînement des modèles profonds.

$$\text{tanh}(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \quad (2)$$



-RELU :

Abréviation de Unités Rectifié linéaires C'est la fonction la plus populaire et la plus utilisé dans les CNN profond c'est une fonction très simple définit par :

$$\text{relu}(x) = \max(0, x) \quad (3)$$

Fournir des réponses parcimonieuse (sparse) est sont point fort elle force les neurones à retourner des valeurs positives.

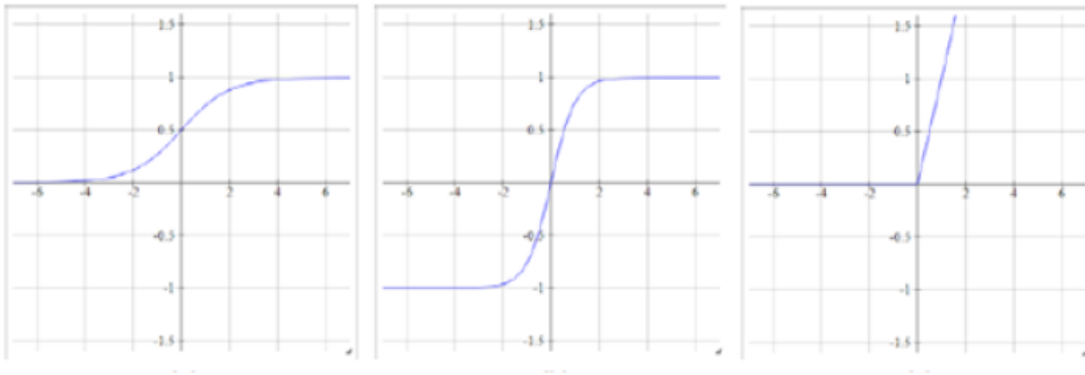


Figure 13. Quelques fonctions d'activation [17]

#### 6.2.4 Le dropout :

C'est une couche qui est utilisée pendant l'apprentissage, son but est de désactiver au hasard plusieurs neurones pendant les différentes itérations de l'apprentissage, l'objectif étant d'éviter le sur-apprentissage (overfitting).

En d'autres termes, le dropout forme de nombreuses architectures de réseaux de neurones différents contenant moins de paramètres.

Dans la phase de test, tous les neurones seront réactivés. [17]

#### 6.2.5 La batch normalisation :

Cette technique est née d'une observation pendant l'apprentissage où dans chaque itération, il y avait un changement sur la distribution des entrées des différentes couches du réseau.

L'idée de cette technique est de normaliser les entrées de chaque couche an que les distributions de celles-ci soient de moyenne nulle et de variance unitaire.

Durant l'apprentissage, les couches de batch normalisation apprennent des paramètres (un facteur d'échelle et un biais) permettant d'ajuster cette normalisation : ces paramètres permettent d'appliquer une transformation sur la distribution normalisée. [17]

### 6.2.6 La fonction de perte Softmax :

Généralement utilisée pour l'optimisation de réseau de classification d'images. Elle permet la maximisation de la probabilité qu'a une entrée d'appartenir à une classe plutôt qu'à une autre. Soit un vecteur de sortie prédit par le CNN. [17]

$$y^* = [y^* 1, \dots, y^* K]^T \quad (4)$$

## 7 Conclusion :

Ce chapitre a été consacré à la notion d'écriture manuscrite, ainsi qu'aux méthodes de reconnaissance et de classification automatiques, aux différents types d'apprentissage

Nous introduisons également l'apprentissage en profondeur, tous basés sur des réseaux de neurones convolutifs.

## Chapitres 2 : La reconnaissance des chiffres manuels de la base MNIST

### 1. Introduction :

L'objectif de notre travail est la reconnaissance des chiffres manuscrits pour la création d'un meilleur réseau de neurones convolutifs et il est très efficace en dataset et mnist.

Ce travail sera fait par le programme Python sous l'interface Spyder.

### 2. Implémentation et résultats :

Nous allons travailler sur la base de données MNIST qui contient 60 000 images en niveaux Grille de résolution 28x28, représentant les 10 chiffres 0 à 9, ainsi qu'un ensemble de test de 10 000 images. Tout d'abord, chargeons cet ensemble de données.

Pour ça on utilise les bibliothèques de « Keras » et « Tensorflow »

```

1 from keras.datasets import mnist
2 from keras.models import Sequential
3 from keras.layers.core import Dense, Dropout, Activation, Flatten
4 from keras.layers import Conv2D, MaxPooling2D
5 from keras.optimizers import SGD
6 from keras.utils import np_utils
7 import matplotlib
8 import matplotlib.pyplot as plt
9 import scipy
10 import scipy.ndimage
11 import numpy as np
12 from keras import backend as K
13 import keras
14 (X_train_base, y_train_base), (X_test_base, y_test_base) = mnist.load_data()
15

```

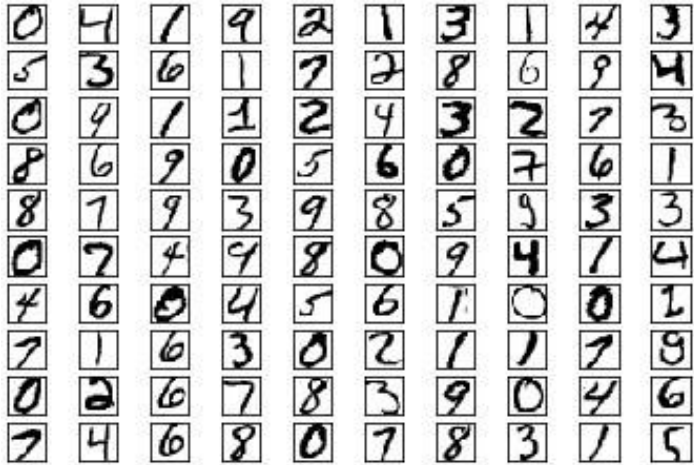
Visualisons quelques données.

```

def plot_10_by_10_images(images):
    """ Plot 100 MNIST images in a 10 by 10 table. Note that we crop
    the images so that they appear reasonably close together. The
    image is post-processed to give the appearance of being continued. """
    fig = plt.figure()
    images = [image[3:25, 3:25] for image in images]
    #image = np.concatenate(images, axis=1)
    for x in range(10):
        for y in range(10):
            ax = fig.add_subplot(10, 10, 10*y+x+1)
            ax.matshow(images[10*y+x+1], cmap = matplotlib.cm.binary)
            plt.xticks(np.array([]))
            plt.yticks(np.array([]))
    plt.show()

plot_10_by_10_images(X_train_base)

```



C'est en applique sur notre estrien on obtient la figure suivante de taille 10 par 10, en remarque chaque chiffre été crée par une façon très variée.

Maintenant que nous avons chargé les données, nous allons modifier la dimension de matrices, afin de les mettre sous une forme qui pourra être traitée par nos réseaux de neurones.

```
33 subset=10000 #size
34 nb_classes=10
35 X_train = X_train_base[:subset].reshape(subset, 784)
36 X_test = X_test_base.reshape(10000, 784)
37 X_train = X_train.astype("float32")
38 X_test = X_test.astype("float32")
39 X_train /= 255
40 X_test /= 255
41 y_train = np_utils.to_categorical(y_train_base[:subset], nb_classes)
42 y_test = np_utils.to_categorical(y_test_base, nb_classes)
43
44
```

```
45 print(X_train.shape)
46 print(X_test.shape)
47 print(y_train)
```

```
(10000, 784)
(10000, 784)
[[0. 0. 0. ... 0. 0. 0.]
 [1. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 ...
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 1.]
 [0. 0. 0. ... 1. 0. 0.]]
```

On remarque que on 'a pour x\_train 10000 image chaque une de taille 784 ,x\_test aussi ; et enfin pour y\_train en le retriever une vecteur de la baille .

### 3. Construire un premier réseau de neurones :

Pour cela, nous allons créer un modèle Keras en utilisant l'api

Sequential:Puis utiliser les méthodes de Keras pour ajouter des

couches à ce modèle.

```
48 model = Sequential()
49 model.add(Dense(12, input_shape=(784,),activation='sigmoid'))
50 model.add(Dense(12,activation='sigmoid'))
Model: "sequential_4"
```

Layer (type)	Output Shape	Param #
dense_10 (Dense)	(None, 12)	9420
dense_11 (Dense)	(None, 12)	156
dropout_6 (Dropout)	(None, 12)	0
dense_12 (Dense)	(None, 10)	130

```
Total params: 9,706
Trainable params: 9,706
Non-trainable params: 0
```

On obtient ce tableau que présent dans la premier colon le nombre des couches, dans la deuxième la taille de sortie de chaque couche et dans le troisième colon le nombre de paramètre assoie a chaque couche.

On obtient enfin un nombre de paramètre égale 9706.

Ensuite, nous allons lancer l'apprentissage des paramètres.

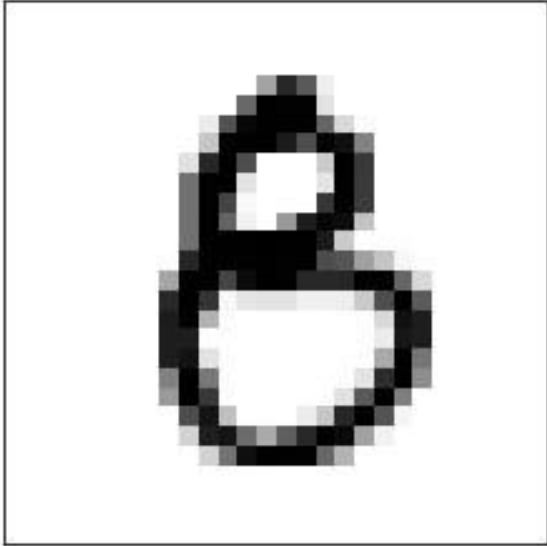
```
59 |
60 | batch_size = 256
61 | epochs=20
62 | model.fit(X_train, y_train, batch_size=batch_size, epochs=epochs, verbose=1, validation_data=(X_test, y_test))
63 |

Epoch 1/20
40/40 [=====] - 1s 8ms/step - loss: 2.3280 - accuracy: 0.1161 - val_loss: 2.2507 - val_accuracy: 0.2362
Epoch 2/20
40/40 [=====] - 0s 4ms/step - loss: 2.2041 - accuracy: 0.1791 - val_loss: 2.0525 - val_accuracy: 0.4003
Epoch 3/20
40/40 [=====] - 0s 4ms/step - loss: 1.9911 - accuracy: 0.2577 - val_loss: 1.7334 - val_accuracy: 0.5522
Epoch 4/20
40/40 [=====] - 0s 4ms/step - loss: 1.7578 - accuracy: 0.3437 - val_loss: 1.4692 - val_accuracy: 0.6033
Epoch 5/20
40/40 [=====] - 0s 4ms/step - loss: 1.5875 - accuracy: 0.4118 - val_loss: 1.2974 - val_accuracy: 0.6470
Epoch 6/20
40/40 [=====] - 0s 4ms/step - loss: 1.4587 - accuracy: 0.4711 - val_loss: 1.1643 - val_accuracy: 0.6870
Epoch 7/20
40/40 [=====] - 0s 4ms/step - loss: 1.3503 - accuracy: 0.5163 - val_loss: 1.0650 - val_accuracy: 0.7110
Epoch 8/20
40/40 [=====] - 0s 4ms/step - loss: 1.2796 - accuracy: 0.5553 - val_loss: 0.9737 - val_accuracy: 0.7513
Epoch 9/20
40/40 [=====] - 0s 4ms/step - loss: 1.2219 - accuracy: 0.5826 - val_loss: 0.9147 - val_accuracy: 0.7740
Epoch 10/20
40/40 [=====] - 0s 4ms/step - loss: 1.1805 - accuracy: 0.6053 - val_loss: 0.8635 - val_accuracy: 0.8148
Epoch 11/20
40/40 [=====] - 0s 4ms/step - loss: 1.1417 - accuracy: 0.6250 - val_loss: 0.8147 - val_accuracy: 0.8252
Epoch 12/20
40/40 [=====] - 0s 4ms/step - loss: 1.1113 - accuracy: 0.6309 - val_loss: 0.7766 - val_accuracy: 0.8283
Epoch 13/20
40/40 [=====] - 0s 4ms/step - loss: 1.0965 - accuracy: 0.6429 - val_loss: 0.7527 - val_accuracy: 0.8422
Epoch 14/20
40/40 [=====] - 0s 4ms/step - loss: 1.0370 - accuracy: 0.6584 - val_loss: 0.7271 - val_accuracy: 0.8421
Epoch 15/20
40/40 [=====] - 0s 3ms/step - loss: 1.0389 - accuracy: 0.6551 - val_loss: 0.7022 - val_accuracy: 0.8418
Epoch 16/20
40/40 [=====] - 0s 4ms/step - loss: 1.0285 - accuracy: 0.6688 - val_loss: 0.6838 - val_accuracy: 0.8459
Epoch 17/20
40/40 [=====] - 0s 4ms/step - loss: 1.0002 - accuracy: 0.6668 - val_loss: 0.6681 - val_accuracy: 0.8552
Epoch 18/20
40/40 [=====] - 0s 4ms/step - loss: 0.9879 - accuracy: 0.6767 - val_loss: 0.6481 - val_accuracy: 0.8606
Epoch 19/20
40/40 [=====] - 0s 4ms/step - loss: 0.9741 - accuracy: 0.6938 - val_loss: 0.6584 - val_accuracy: 0.8550
Epoch 20/20
40/40 [=====] - 0s 4ms/step - loss: 0.9617 - accuracy: 0.6975 - val_loss: 0.6236 - val_accuracy: 0.8630
The accuracy on the test set is 86.2999975681305 %
```

On obtient une loss égale a peu près un 1et un acc égale 0.64, donc les résultats sont moyen.

```
64 |
65 | def plot_mnist_digit(image):
66 |     """ Plot a single MNIST image. """
67 |     fig = plt.figure()
68 |     ax = fig.add_subplot(1, 1, 1)
69 |     ax.imshow(image, cmap = matplotlib.cm.binary)
70 |     plt.xticks(np.array([]))
71 |     plt.yticks(np.array([]))
72 |     plt.show()
73 | loss, acc = model.evaluate(X_test, y_test, verbose=0)
74 | index=800
75 | print('The accuracy on the test set is ', (acc*100), '%')
76 | plot_mnist_digit(X_test_base[index])
77 | cl=model.predict_classes(X_test[index].reshape((1,784)))
78 |
79 |
80 | print("le chiffre reconnu est: ", cl[0])
81 | print("le chiffre à reconnaître est: ", np.argmax(y_test[index]))
82 |

The accuracy on the test set is 86.2999975681305 %
```



```
le chiffre reconnu est: 5  
le chiffre à reconnaître est: 8
```

D'après les résultats en remarque donc notre modèle est n'est pas performant.

#### 4. CNN : réseaux de neurones convolutionnels :

```
83  
84 img_rows, img_cols = 28, 28  
85 (x_train, y_train), (x_test, y_test) = mnist.load_data()  
86 x_train = x_train[:subset].reshape(x_train[:subset].shape[0], img_rows, img_cols, 1)  
87 x_test = x_test.reshape(x_test.shape[0], img_rows, img_cols, 1)  
88 input_shape = (img_rows, img_cols, 1)  
89 x_train = x_train.astype('float32')  
90 x_test = x_test.astype('float32')  
91 x_train /= 255  
92 x_test /= 255  
93 y_train = keras.utils.to_categorical(y_train[:subset], nb_classes)  
94 y_test = keras.utils.to_categorical(y_test, nb_classes)  
95  
96  
97 print('x_train shape:', x_train.shape)  
98 print(x_train.shape[0], 'train samples')  
99 print(x_test.shape[0], 'test samples')  
100
```

```
x_train shape: (10000, 28, 28, 1)  
10000 train samples  
10000 test samples|
```

```

100
101 model = Sequential()
102
103 model.add(Conv2D(4, kernel_size=(3, 3),
104                 activation='relu',
105                 input_shape=input_shape))
106 model.add(MaxPooling2D(pool_size=(2, 2)))
107
108 model.add(Dropout(0.25))
109 model.add(Flatten())
110 model.add(Dense(10, activation='relu'))
111 model.add(Dropout(0.5))
112 model.add(Dense(nb_classes, activation='softmax'))
113
114 sgd = SGD(lr=0.1, decay=1e-6, momentum=0.9, nesterov=True)
115 model.compile(loss='categorical_crossentropy', metrics=['accuracy'], optimizer=sgd)
116
117 model.summary()

```

En rajoute une couche convolutionnel utiles des filtres ; (nous utilise un nombre de filtre égale a 4) chacune une taille de 3 par 3 et en choisies aussi une fonction d'activation égale relu, en précise la taille d'entres input shape , en suite en rajoute une couche max-pooling...

```

Model: "sequential_7"

```

Layer (type)	Output Shape	Param #
conv2d_5 (Conv2D)	(None, 26, 26, 4)	40
max_pooling2d_5 (MaxPooling2D)	(None, 13, 13, 4)	0
dropout_10 (Dropout)	(None, 13, 13, 4)	0
flatten_3 (Flatten)	(None, 676)	0
dense_18 (Dense)	(None, 10)	6770
dropout_11 (Dropout)	(None, 10)	0
dense_19 (Dense)	(None, 10)	110

```

Total params: 6,920
Trainable params: 6,920
Non-trainable params: 0

```

-Pour un filtre de taille 3 par 3 le modèle il vas chercher optimiser 9 valeurs alors le résultat obtenu de paramètre égale  $9 \times 4 + 4 = 40$ .

-pour la couche de max pooling en remarque et divisé de la taille de sortie de la couche de convolution par 2 et cette couche n'est pas d'un paramètre.

-le nombre de taille de paramètre est égale 6.920, est donc en remarque quand on 'a bien un nombre



de paramètre inférieur à celui obtenu avec le premier modèle.

```
118  
119     batch_size = 256  
120     epochs=20  
121     model.fit(x_train, y_train, batch_size=batch_size, epochs=epochs, verbose=1, validation_data=(x_test, y_test))  
122
```

```

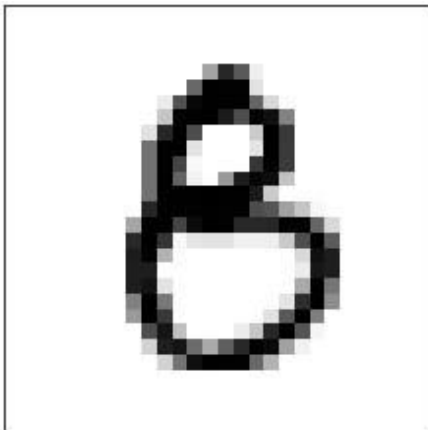
Epoch 1/20
40/40 [=====] - 4s 90ms/step - loss: 1.1583 - accuracy: 0.6100 - val_loss: 0.2677 - val_accuracy: 0.9251
Epoch 2/20
40/40 [=====] - 4s 88ms/step - loss: 0.3364 - accuracy: 0.8963 - val_loss: 0.1934 - val_accuracy: 0.9393
Epoch 3/20
40/40 [=====] - 4s 88ms/step - loss: 0.2281 - accuracy: 0.9303 - val_loss: 0.1087 - val_accuracy: 0.9661
Epoch 4/20
40/40 [=====] - 3s 88ms/step - loss: 0.1801 - accuracy: 0.9469 - val_loss: 0.1027 - val_accuracy: 0.9660
Epoch 5/20
40/40 [=====] - 4s 94ms/step - loss: 0.1618 - accuracy: 0.9514 - val_loss: 0.1294 - val_accuracy: 0.9590
Epoch 6/20
40/40 [=====] - 3s 85ms/step - loss: 0.1489 - accuracy: 0.9545 - val_loss: 0.0832 - val_accuracy: 0.9735
Epoch 7/20
40/40 [=====] - 3s 81ms/step - loss: 0.1170 - accuracy: 0.9647 - val_loss: 0.0731 - val_accuracy: 0.9766
Epoch 8/20
40/40 [=====] - 3s 79ms/step - loss: 0.1155 - accuracy: 0.9628 - val_loss: 0.0945 - val_accuracy: 0.9723
Epoch 9/20
40/40 [=====] - 3s 88ms/step - loss: 0.1090 - accuracy: 0.9675 - val_loss: 0.0650 - val_accuracy: 0.9788
Epoch 10/20
40/40 [=====] - 4s 92ms/step - loss: 0.1083 - accuracy: 0.9665 - val_loss: 0.0611 - val_accuracy: 0.9801
Epoch 11/20
40/40 [=====] - 4s 92ms/step - loss: 0.0857 - accuracy: 0.9725 - val_loss: 0.0592 - val_accuracy: 0.9817
Epoch 12/20
40/40 [=====] - 4s 98ms/step - loss: 0.0769 - accuracy: 0.9764 - val_loss: 0.0568 - val_accuracy: 0.9820
Epoch 13/20
40/40 [=====] - 4s 88ms/step - loss: 0.0761 - accuracy: 0.9747 - val_loss: 0.0543 - val_accuracy: 0.9830
Epoch 14/20
40/40 [=====] - 4s 103ms/step - loss: 0.0615 - accuracy: 0.9795 - val_loss: 0.0504 - val_accuracy: 0.9841
Epoch 15/20
40/40 [=====] - 4s 92ms/step - loss: 0.0732 - accuracy: 0.9778 - val_loss: 0.0598 - val_accuracy: 0.9818
Epoch 16/20
40/40 [=====] - 4s 95ms/step - loss: 0.0613 - accuracy: 0.9807 - val_loss: 0.0590 - val_accuracy: 0.9826
Epoch 17/20
40/40 [=====] - 4s 87ms/step - loss: 0.0580 - accuracy: 0.9818 - val_loss: 0.0531 - val_accuracy: 0.9835
Epoch 18/20
40/40 [=====] - 4s 99ms/step - loss: 0.0503 - accuracy: 0.9827 - val_loss: 0.0547 - val_accuracy: 0.9834
Epoch 19/20
40/40 [=====] - 4s 98ms/step - loss: 0.0458 - accuracy: 0.9855 - val_loss: 0.0543 - val_accuracy: 0.9842
Epoch 20/20
40/40 [=====] - 4s 98ms/step - loss: 0.0450 - accuracy: 0.9856 - val_loss: 0.0560 - val_accuracy: 0.9844

```

```

123
124 def plot_mnist_digit(image):
125     """ Plot a single MNIST image. """
126     fig = plt.figure()
127     ax = fig.add_subplot(1, 1, 1)
128     ax.imshow(image, cmap = matplotlib.cm.binary)
129     plt.xticks(np.array([]))
130     plt.yticks(np.array([]))
131     plt.show()
132     loss, acc = model.evaluate(x_test, y_test, verbose=0)
133     index=800
134     print('The accuracy on the test set is ', (acc*100), '%')
135     plot_mnist_digit(X_test_base[index])
136     cl=model.predict_classes(x_test[index].reshape((1,28,28,1)))
137
138
139     print("le chiffre reconnu est: ", cl[0])
140     print("le chiffre à reconnaître est: ", np.argmax(y_test[index]))
141

```



```
The accuracy on the test set is 94.2300021648407 %
le chiffre reconnu est: 5
le chiffre à reconnaître est: 8
```

-On va créer un modèle plus complexe qui contient deux couches de convolution suivies chacune par une couche de MaxPooling, et on va augmenter le nombre de paramètres dans les couches de convolution.

Première couche de convolution de 32 filtres en suite une couche de maxpooling et on rajoute une autre couche de convolution de 64 filtres et suivie aussi d'une couche de maxpooling, après en garde la même couche que tout alors mais en choisissant le nombre de neurones égale à 100.

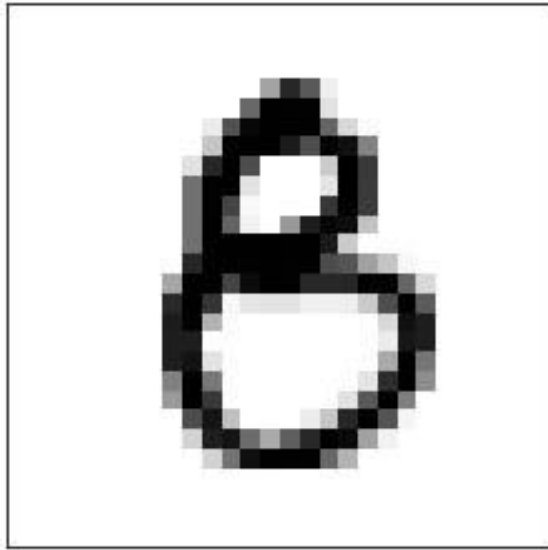
```
103 model = Sequential()
104
105 model.add(Conv2D(32, kernel_size=(3, 3),
106                 activation='relu',
107                 input_shape=input_shape))
108 model.add(MaxPooling2D(pool_size=(2, 2)))
109
110 model.add(Conv2D(64, kernel_size=(3, 3),
111                 activation='relu',
112                 input_shape=input_shape))
113 model.add(MaxPooling2D(pool_size=(2, 2)))
114
115 model.add(Dropout(0.25))
116 model.add(Flatten())
117 model.add(Dense(100, activation='relu'))
118 model.add(Dropout(0.5))
119 model.add(Dense(nb_classes, activation='softmax'))
120
121 sgd = SGD(lr=0.1, decay=1e-6, momentum=0.9, nesterov=True)
122 model.compile(loss='categorical_crossentropy', metrics=['accuracy'], optimizer=sgd)
123
124 model.summary()
```

Model: "sequential\_5"

Layer (type)	Output Shape	Param #
conv2d_3 (Conv2D)	(None, 26, 26, 32)	320
max_pooling2d_3 (MaxPooling2D)	(None, 13, 13, 32)	0
conv2d_4 (Conv2D)	(None, 11, 11, 64)	18496
max_pooling2d_4 (MaxPooling2D)	(None, 5, 5, 64)	0
dropout_7 (Dropout)	(None, 5, 5, 64)	0
flatten_2 (Flatten)	(None, 1600)	0
dense_13 (Dense)	(None, 100)	160100
dropout_8 (Dropout)	(None, 100)	0
dense_14 (Dense)	(None, 10)	1010

=====  
Total params: 179,926  
Trainable params: 179,926  
Non-trainable params: 0

```
The accuracy on the test set is 98.43999743461609 %
```



```
le chiffre reconnu est: 8  
le chiffre à reconnaître est: 8
```

-on remarque que on 'a très bonne score.

- on est bien a reconnaître l'image le chiffre reconnu est : 8

Et le chiffre à reconnaître est : 8

Donc voila on 'a arrive a crée un meilleur réseau de neurone convolutionnel et il est très performance sur le jeu de donne et mnist.

## 5. Conclusion :

Dans ce chapitre, nous avons fait un traitement d'image (lecture et affichage ) En différentes manières pour cela on a utiliser le programme python et travailler sur l'interface Spyder .

Notre travail a été consacré au traitement d'images, et la construction de réseau neurons convolutifs(CNN).

## Conclusion général :

Dans ce projet, nous avons brièvement introduit le domaine de l'intelligence artificielle, où nous avons réussi à expliquer et discuter les bases de certains algorithmes d'apprentissage automatique, tels que les algorithmes d'apprentissage en profondeur, tout en s'appuyant généralement sur les réseaux de neurones, en particulier les réseaux de neurones convolutifs. Nous avons couvert toutes les étapes nécessaires à la mise en œuvre de ces réseaux de neurones, qui sont à la pointe des algorithmes d'apprentissage en profondeur.

Notre application concerne la reconnaissance des chiffres manuels avec le réseau de neurone convolutif. En utilisant la base donnée MNIST, l'algorithme implémenté donne des résultats satisfaisants.

# Annex

## Definition :

### TensorFlow :

TensorFlow est un framework de programmation axé sur l'apprentissage automatique créée par Google, TensorFlow a été initialement développé par des ingénieurs et des chercheurs de l'équipe Google Brain, principalement pour un usage interne, il a été rendu Open Source par Google en Novembre 2015. Depuis ce temps il s'est imposé comme étant le framework le plus populaire et le plus utilisé pour le deep learning.

TensorFlow est considéré comme le successeur de l'application fermée DistBelief et est actuellement utilisé par Google à des fins de recherche et de production. TensorFlow est considéré comme la première mise en œuvre sérieuse d'un cadre axé sur l'apprentissage en profondeur. TensorFlow est également connu sous le nom de Google TensorFlow.

TensorFlow tire son nom des tableaux multidimensionnels connus sous le nom de tenseurs, qui sont utilisés par les réseaux de neurones pour différentes opérations, Un Tensor à deux dimensions est l'équivalent d'une matrice. Selon Google, par rapport à DistBelief, TensorFlow est plus rapide, plus intelligent et plus flexible et peut facilement être adapté à de nouveaux domaines et produits. Il a été principalement créé pour la recherche sur les réseaux neuronaux profonds et pour faciliter l'apprentissage automatique, bien que TensorFlow ait été utilisé dans un large éventail d'autres domaines.

TensorFlow est disponible sur différents systèmes d'exploitation tels que Linux, Windows, MacOS et également sur des plateformes d'exploitation mobiles comme iOS et Android. L'une des principales caractéristiques de TensorFlow est qu'il est capable de fonctionner sur plusieurs CPU et GPU. Les calculs dans TensorFlow sont signalés en tant que graphes de flux de données avec état. Actuellement, TensorFlow est utilisé dans plus de six mille dépôts en ligne gratuits. [25]

### KERAS :

Keras est une bibliothèque de réseau neuronal open source écrite en Python. Il est

capable de fonctionner sur TensorFlow, Microsoft Cognitive Toolkit, Theano ou MXNet. Il a été développé dans le but de permettre une expérimentation rapide avec les réseaux neuronaux profonds, Pouvoir passer de l'idée au résultat avec le moins de retard possible est la clé pour faire de bonnes recherches.

IL se concentre sur l'ergonomie, la modularité et l'extensibilité. Il a été développé dans le cadre de l'effort de recherche du projet ONEIROS (Système d'Exploitation de Robots Intelligents Neuroélectroniques Ouverts), et son principal auteur et mainteneur est François Chollet, un ingénieur de Google.

En 2017, l'équipe TensorFlow de Google a décidé de soutenir Keras dans la bibliothèque principale de TensorFlow. Chollet a expliqué que Keras a été conçu pour être une interface plutôt qu'un cadre autonome d'apprentissage automatique. Il offre un ensemble d'abstractions de niveau supérieur et plus intuitif qui facilite le développement de modèles d'apprentissage en profondeur, quel que soit le système de calcul utilisé. [23]

### *Python :*

Python est un langage de programmation objet, multi-paradigme et multiplateformes. Il favorise la programmation impérative structurée, fonctionnelle et orientée objet. Il est doté d'un typage dynamique fort, d'une gestion automatique de la mémoire par ramasse-miettes et d'un système de gestion d'exceptions ; il est ainsi similaire à Perl, Ruby, Scheme, Smalltalk et Tcl.

Le langage Python est placé sous une licence libre proche de la licence BSD4 et fonctionne sur la plupart des plates-formes informatiques, des supercalculateurs aux ordinateurs centraux<sup>5</sup>, de Windows à Unix avec notamment GNU/Linux en passant par macOS, ou encore Android, iOS, et aussi avec Java ou encore .NET. Il est conçu pour optimiser la productivité des programmeurs en offrant des outils de haut niveau et une syntaxe simple à utiliser.

Il est également apprécié par certains pédagogues qui y trouvent un langage où la syntaxe, clairement séparée des mécanismes de bas niveau, permet une initiation aisée aux concepts de base de la programmation<sup>6</sup>. Python est un langage simple, facile à apprendre et permet une bonne réduction du coût de la maintenance des codes. Les bibliothèques (packages) python encouragent la modularité et la réutilisabilité des codes. Python et ses



bibliothèques sont disponibles sans charges pour la majorité des plateformes et peuvent être redistribués gratuitement. [24]

## Bibliographie :

1. consulté le 13/10/2021

<https://www.google.com/search?q=la+reconnaissance+d%27%C3%A9criture&hl=fr&sxsrf=AOaemvIo7Zy1lfmFpTQqZDge1A7cwoxTAg%3A1634110434211&source=hp&ei=4otmYav1>.

2. consulé le 13/10/2021

<https://www.google.com/search?q=reconnaissance+%C3%A9criture+manuscrite+python&hl=fr&biw=1280&bih=520&sxsrf=AOaemvKCoIF1iwdTEfracWK5geaEppjJiQ%3A1634927506422&ei=kgNzYbKTGFCEjLsPkcWWqAc&oq=reconnaissance+%C3%A9criture+manuscrite>.

3. Hilali, h., application de la classification textuelle pour l'extraction des règles d'association maximales. thèse de maitrise en informatique, université du québec à trois-rivières, trois-rivières, 2009.

4. Consulté le 13/10/2021

<https://www.google.com/search?q=la+reconnaissance+d%27%C3%A9criture+automatique&hl=fr&sxsrf>.

5. R. Plamondon, S. Srihari, «On-line and Off-line Handwriting Recognition: A Comprehensive Survey», IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 22, n°. 1, pp. 63-84, 2000

6. A.Belaid, «Reconnaissance automatique de l'écriture et du document» , Pour la science, disponible sur le lien web :<http://webloria.loria.fr/~abelaid/Publications.html>, 2001

7. N.E. Ayat , «Selection de modele automatique des machines a vecteurs de support: application a la reconnaissance d'images de chiffres manuscrits» , Thèse de doctorat , 2004.

8. Gurney, k., an introduction to neural networks. 1997.

9. mémoire a présente à l'université de Québec à trois-révières.juin 2017.

10. Fausett, I.v., fundamentals of neural networks: architectures, algorithms and Applications. 1993
  
11. Marref 2013 Marref, Nadia. Apprentissage Incrémental Machines à Vecteurs Supports. Diss. Université Mustapha Ben Boulaid Batna 2, 2013.
12. Antoine. "Apprentissage Supervisé."
13. HAMZA CHERIF, Ikram. "Classification des tracés TocoGraphiques (CTG) d'un foetus à l'aide de classieurs multiples."
  
14. Nemouchi, S., and N. Farah. "Reconnaissance de l'écriture Arabe par Systèmes Flous." Département Informatique, Université Badji Mokhtar Laboratoire de Gestion Electronique du Document (LABGED), Algérie (2010).
  
15. Chafik, Sanaa. Machine learning techniques for content-based information retrieval. Diss. Université Paris-Saclay, 2017.
  
16. Bc. Ján Vojt, "Deep neural networks and their implementation", 2016
  
17. ] Chabot, Florian. Analyse ne 2D/3D de véhicules par réseaux de neurones profonds. Diss. Université Clermont Auvergne, 2017.
  
18. Burrow, Peter. "Arabic handwriting recognition." Report of Master of Science School of Informatics, University of Edinburgh (2004).
  
19. rifqi, "k plus proche voisins", 2002
  
20. Consulté le 27/03/2018 <https://fr.slideshare.net/wassimlahbib/algorithmeknn>

21. RAKOTOMALALA, Ricco. "Quelques approches pour rendre calculable  $P(Y/X)$ ."
22. Consulté le 02/04/2018 <https://siecledigital.fr/2016/12/22/machine-learningdeep-learning-ca-marche/>
23. Consulté le 20/08/2018 <https://en.wikipedia.org/wiki/Keras>
24. Consulté le 20/08/2018 [https://fr.wikipedia.org/wiki/Python\\_\(langage\)](https://fr.wikipedia.org/wiki/Python_(langage))
25. Consulté le 20/08/2018 [https://www.techopedia.com/denition/32862/tensorflow\]](https://www.techopedia.com/denition/32862/tensorflow)