République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université A.MIRA Bejaïa
Faculté de Technologie
Département de Génie Electrique

Mémoire fin d'étude

En vue de l'obtention du diplôme Master en Automatique
***Spécialité* : Automatique et systèmes**

Thème

Reconnaissance Automatique des chiffres avec le Deep Learning

Réalisé par :

M^r^ .SLIMANI Massinissa

M^r.^ KHALED Adel

Encadré par :

Mme GAGAOUA.M

Devant le jury d'examen composé de:

M^r^. ALLICHE Abdenour

M^r^ .MOKRANI Karim

Année universitaire2019/2020

Remerciements

En préambule à ce projet de fin d'étude on remercie ALLAH qui nous aide et nous donne la patience et le courage durant toutes ces années d'études.

Ces remerciements vont tout d'abord à mon encadreur M^me^.GAGAOUA d'avoir acceptée de nous encadrer dans ce travail de thèse. Nos échanges courts mais efficaces, vos encouragements sont toujours venus à point.

On tient aussi à témoigner nos reconnaissances à toute l'équipe de département d'ATE, pour leur accueil sympathique et leur collaboration.

On exprime notre gratitude et notre reconnaissance à nos parents et nos amis pour leurs contributions, leurs soutiens, leurs patiences qui nous ont entourés de leurs affections et de leurs encouragements pour l'accomplissement de ce travail.

On adresse nos plus sincères remerciements aux membres du jury qui ont accepté d'examiner notre projet.

Merci à tous et à toutes

Sommaire

CHAPITRE II : Deep Learning

CHAPITRE III : Implémentation et résultats

CONCLUSION GENERALE

BIBLIOGRAPHIE

ANNEXES

Liste des figures

Chapitre I

Chapitre II

Chapitre III

Liste des tableaux

Chapitre I

Chapitre II

Chapitre III

Liste des abréviations

OCR : Optical Character Recognition

SVM : Support Vector Machines

KNN : K Nearest Neighbors

DL : Deep Learning

IA : Artificial Intelligence

ML : Machine learning

CNN : Convolutional Neural Network

RNN : Reccurent Neural Network

FC : Fully Connected

RELU : REctified Linear Unit

AVG : AVeraGe

API : Application Programming Interface

Introduction générale

Depuis son invention il y a plus de 5300 ans, l'écriture reste un moyen de communication privilégié entre les êtres humains. Bien que l'imprimerie créée il y a plus de 550 ans puis l'informatique aient permis son automatisation, l'écriture manuscrite est loin d'avoir disparu de notre société et les individus émettent et reçoivent une grande quantité de documents manuscrits. Le traitement de masse de ces documents apparaît alors incontournable. C'est ainsi que les recherches concernant la lecture automatique des documents manuscrits ont débuté il y a plus de 55 ans. Les premiers OCR (Optical Character Recognition) ont fait leur apparition dans les années 1950 et les premiers systèmes de lecture d'adresses postales utilisés pour le tri du courrier sont apparus en 1965. La lecture automatique de l'écriture manuscrite dans les formulaires a fait son apparition dans les années 1980 grâce au développement de moyens de calculs toujours plus puissants.

La Reconnaissance de Chiffres Manuscrits (RCM), fait l'objet d'un nombre important de travaux de recherche, grâce à ses applications diverses et potentielles. Le recours à la RCM s'impose dans la plupart des domaines de la vie courante. A titre d'exemple, nous citons le tri postal qui est l'une des premières applications où tous les jours des milliers d'enveloppes sont automatiquement triés. A l'instar de cette application, on distingue la lecture du montant numérique des chèques bancaires et l'identification du numéro de sécurité sociale. Toutefois, malgré le progrès impressionnant des techniques utilisées ainsi que l'explosion dans la puissance de calcul des ordinateurs, la recherche sur la RCM avance avec une performance de reconnaissance qui reste loin de celle de l'œil humain.

Dans notre cas nous nous sommes concentrés sur l'utilisation des réseaux neurones convolutionnels pour les reconnaissances des chiffres manuscrits.

Le travail présenté dans ce mémoire est organisé en trois chapitres structurés comme suit :

Dans le premier chapitre nous allons traiter la problématique de la reconnaissance des chiffres manuscrits ainsi que les étapes et les méthodes utilisées à cet effet.

Dans le deuxième chapitre, nous allons présenter les réseaux de neurones, en particulier les réseaux de neurones convolutionnels. Ces réseaux sont capables d'extraire des caractéristiques d'images présentées en entrée et de classifier ces dernières.

Dans le dernier chapitre, nous allons élaborer une implémentation pour la reconnaissance des chiffres manuscrits en utilisant une approche de classification basée sur le réseau de neurone convolutionnel, et nous allons présenter et discuter les résultats obtenus.

ChapitreI
Reconnaissance
de l’écriture

I.1 La reconnaissance de l'écriture manuscrite :

I.1.1 Introduction :

L'écriture manuscrite est un moyen naturel de communication qui présente l'avantage d'être familier à la majorité des gens. De ce fait elle présente un moyen d'interaction facile avec l'ordinateur. L'un des objectifs les plus recherchés est de doter les ordinateurs de capacités de l'être Humain (Figure I.1).

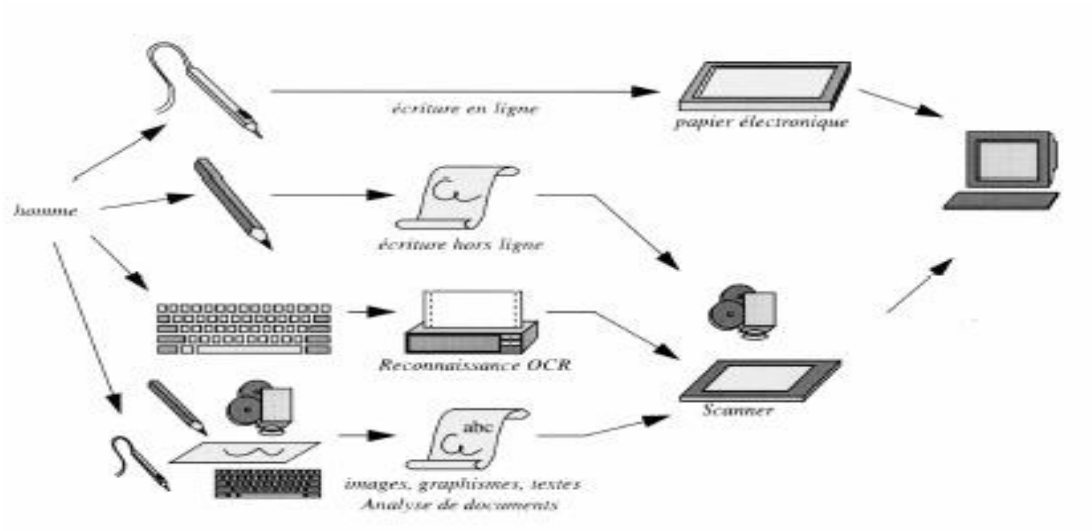

Figure I.1 types de communication Homme machine.

Le but de la reconnaissance automatique de l'écriture est de transformer un texte écrit en une représentation compréhensible par une machine et facilement reproductible par un traitement de texte. Cette tâche n'est pas triviale, car l'écriture possède une infinité de représentations dues au fait que chaque personne possède une écriture qui lui est propre avec de nombreux styles.

La reconnaissance de l'écriture est mieux connue sous le nom d'O.C.R (Optical Character Recognition). Les premières tentatives remontent aux années 1900 par TYURIN, au cours desquelles on inventa le scanner à balayage pour la télévision ; et en l'an 1912 par ALBE et en l'an 1925 par Thomas, qui ont mimé l'interprétation Humaine de l'informatique visuelle , un point de transformation est apparue avec l'invention du premier ordinateur en 1946 par

MAUCHLY et ECKERT , quelques années plus tard les premières expériences en reconnaissance de caractères ont pu être réalisées pendant les année soixante, et soixante-dix, les premiers systèmes de l'écriture automatique du texte imprimé ont vu le jour. [01]

Aussi, en 1975 les japonais utilisaient couramment les lecteurs qui déchiffrent le code postal inscrit à la main ou tapé à la machine. Dans la même période, le Français CONTER construisait un système de lecture automatique de texte imprimé destiné aux non-voyants, par la suite la compagnie américaine KURZWELL a amélioré le système précédent en proposant les machines à lire pour aveugle formulant le texte à haute voix par synthèse vocale. [02]

Durant cette phase les chercheurs se sont heurtés à de nombreuses difficultés outre la complexité du problème de la reconnaissance due à la grande variabilité de l'écriture manuscrite, la non disponibilité de mémoire et de puissance de calcul pour la réalisation des systèmes concrets opérant en temps réel, par contre depuis 1980 où les récents progrès électroniques et plus particulièrement l'avènement de calculateurs puissants à faible coût ont permis de résoudre ce type de problème et les recherches en reconnaissance manuscrite se sont multipliées de manière spectaculaire, et de nombreuses nouvelles techniques ont vu le jour.

I.1.2 Les différents aspects de la reconnaissance optique de l'écriture :

I.1.2.1 type d'écriture :

Elle peut être sous forme : imprimé ou manuscrite :

- **Manuscrite** : L'écriture manuscrite comme la parole fait partie de nos modes d'expression les plus évolués en tous les cas les deux sont très complexes, 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 possède deux caractéristiques : Un liée au scripteur par exemple vérifier une signature d'un auteur. L'autre liée au sens de ce qui est écrit (contextuelle). [03]
- **Imprimé** : L'invention de procédés d'impression par Gutenberg vers le milieu du quinzième siècle a transformé notre vie par une diffusion plus large et plus rapide de connaissance, la majorité de patrimoine culturel et technique de l'humanité n'est encore disponible que sous forme de document papier ; les entreprises et les collectivités sont

ainsi confrontées à un besoin de retraitement, c'est-à-dire conversion rétrospective pour passer à un format électronique. [04]

I.1.2.2 Le mode d'acquisition des données :

- **Reconnaissance en ligne** : ce mode de reconnaissance s'opère en temps réel (pendant l'écriture) les caractères sont reconnus au fur et à mesure de leurs écritures à la main. Ce mode est réservé généralement à l'écriture manuscrite, c'est une approche "signal" où la reconnaissance est effectuée sur des données à une dimension. L'écriture est représentée comme un ensemble de points dont les coordonnées sont en fonction du temps. L'acquisition de l'écrit est généralement assurée par une tablette graphique munie d'un stylo électronique. [05]
- **Reconnaissance hors ligne :** elle convient aux documents imprimés et les manuscrits déjà rédigés. Les données en entrée sont acquises via un scanner. Elle a comme tâche de d´déterminer quelles lettres ou mots sont présents dans l'image du texte scanné. [06]

Comparaison de la reconnaissance des caractères en ligne et hors ligne :

Le tableau I.1 présente une comparaison entre de la reconnaissance de l'écriture en ligne et hors ligne :

Reconnaissance en ligne	**Reconnaissance hors ligne**
-Utilisation de stylos numériques	Utilisation de papier
Exigence d'échantillons	Exigence de points
Taux de reconnaissance élevé	Taux de reconnaissance bas
Grande précision	Faible précision

Tableau I.1 Comparaison de la reconnaissance en ligne et hors ligne

I.1.2.3 Approches de reconnaissance :

- **Approche global** : elle se base sur une description unique du mot et tente de le reconnaitre en utilisant les caractéristiques du mot entier, cette approche est envisageable pour les vocabulaires réduits (chèques bancaires). [07]

- **Approche Analytique :** elle traite le mot comme une collection de sous unités simples et procédé en segmentant ces mots en unités Cette approche est la seule applicable dans le cas de grands vocabulaires. [07]

I.1.3 L'organisation générale d'un système de reconnaissance de l'écriture :

Les systèmes de reconnaissance de l'écriture manuscrite sont généralement basés sur les étapes principales suivantes :

Acquisition, prétraitements, segmentation, extraction des caractéristiques, suivis éventuellement d'une phase de reconnaissance.

La figure I.2, représente un schéma général d'un système de reconnaissance de l'écriture.

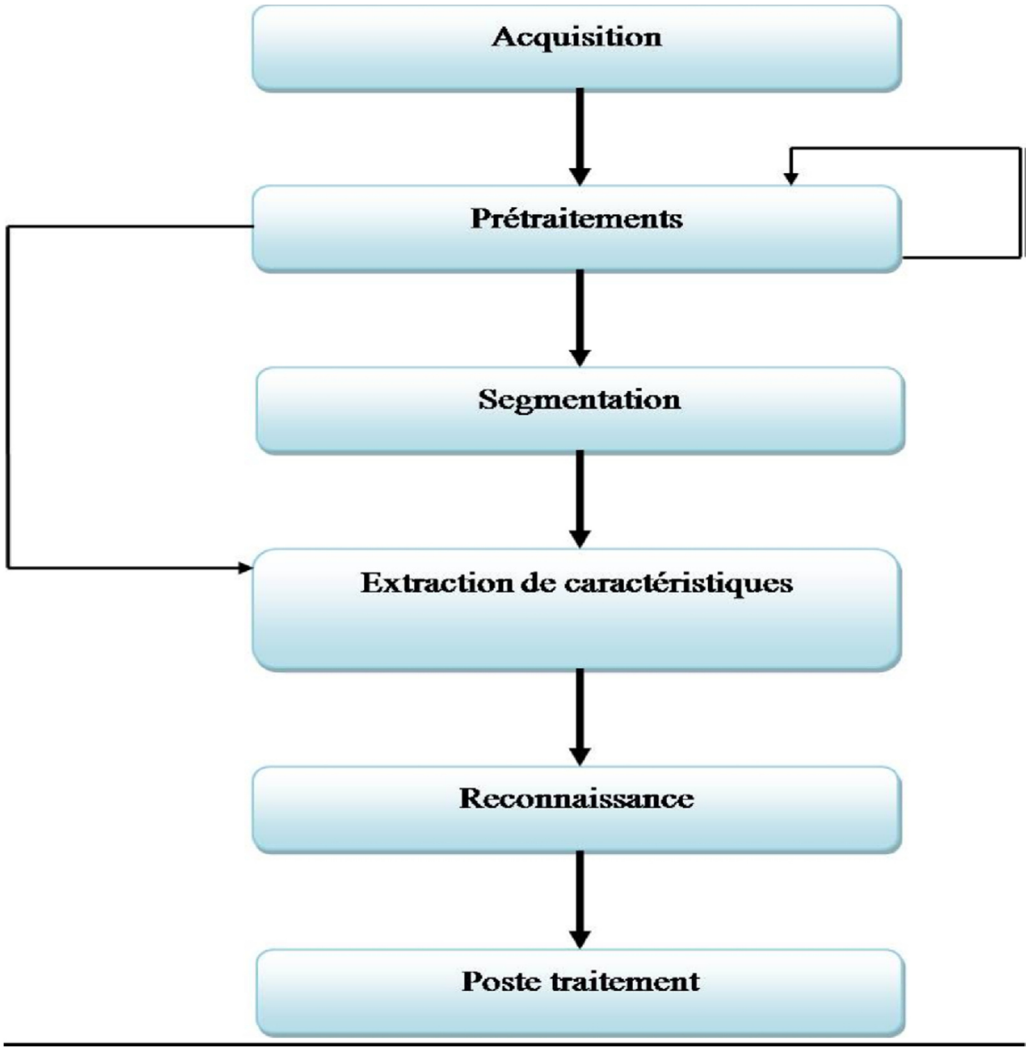

Figure I.2 Organisation générale d'un système de reconnaissance d'écriture

Pour assurer une bonne reconnaissance, après l'acquisition des données, ces dernières passent par un ensemble d'opérations de prétraitement, puis une étape de segmentation aura lieu. Dans le cas de reconnaissance de mot en utilisant l'approche holistique le système passe directement à l'´étape d'extraction de caractéristiques. Ces dernières seront utilisées dans l'étape de classification, puis à la fin une étape d'amélioration des résultats pourra avoir lieu.

I.1.4 Domaines d'application de l'écriture manuscrite :

L'écriture manuscrite connaît plusieurs applications pratiques dans plusieurs domaines parmi lesquels on cite :

- **Domaine bancaire :** Pour l'authentification des chèques par les banques.
- **Assistance à l'éducation :** Pour la reconnaissance et la traduction de textes en braille et l'apprentissage de l'écriture (lecture, écriture) des photo-senseurs, Et des simulateurs tactiques sont utilisés.
- **Lecture des adresses postales :** la lecture des codes postaux manuscrits associés à la lecture des noms de villes a permis d'atteindre le développement des machines de tri automatique de courrier (lettres et objets plats).
- **La police et la sécurité :** pour la reconnaissance des numéros minéralogiques pour le contrôle routier, l'authentification de manuscrits et l'identité des scripteurs.

Enfin on trouve encore des applications pour Bureautique, télécommunication … etc
Il n'existe pas un système universel de reconnaissance qui peut traiter tous les cas de l'écriture, mais plutôt des voies d'approche dépendant du type de données traitées et bien évidemment de l'application visée. [08]

I.2 Reconnaissance des chiffres manuscrits :

C'est la tâche la plus basique d'un système de reconnaissance de chaîne numérique. L'effort d'analyse est concentré sur un seul élément à la fois (vue comme une forme globale). Les méthodes de reconnaissance sont nombreuses, elles dépendent du choix du type des indices visuels ou de paramètres, ou encore de primitives. Ces paramètres peuvent être soit topologiques (éléments ou parties), soit géométriques ou métriques (de type distance, taille, courbure et angle), soit enfin statistiques relatives à des observations de points.

La difficulté du problème de la reconnaissance vient de la variabilité des formes de chiffres lorsqu'on se situe en contexte omni-scripteur. On peut constater ces différences sur la figure I.3.

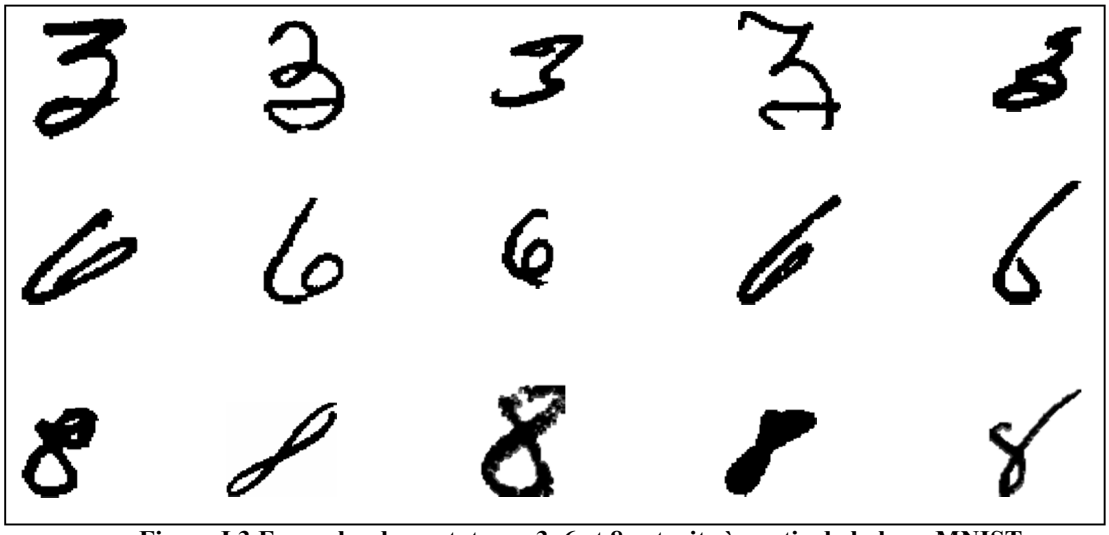

Figure I.3 Exemples de prototypes 3, 6 et 8 extraits à partir de la base MNIST

Dans **la figure I.3**, on remarque les variations de taille, de structure, d'inclinaison et de trait au sein d'une même classe.

I.2.1 Processus de système de reconnaissance :

Les systèmes de reconnaissance des chiffres manuscrits sont généralement basés sur les étapes principales suivantes :

Acquisition, prétraitements, segmentation, extraction des caractéristiques, classification.

I.2.1.1 Acquisition:

L'acquisition des données peut être faite par une tablette en cas de reconnaissance en ligne (temps réel) où la numérisation des scripts dépend de la résolution de la tablette et de la vitesse d´échantillonnage. Dans le cas de la reconnaissance hors ligne, l'acquisition est effectuée par un scanner où la qualité des données acquises dépend de la résolution de ce dernier. [09]

I.2.1.2 Prétraitement :

Lorsque l'acquisition est effectuée, la plupart des systèmes comportent une étape de prétraitement. Généralement, ces prétraitements ne sont pas spécifiques à la reconnaissance de l'écriture, mais sont des prétraitements classiques en traitement d'image. Le prétraitement a pour but de préparer l'image du tracé à la phase suivante d'analyse [10]. Il s'agit essentiellement de réduire le bruit superposé aux données et ne garder, autant que possible, que l'information significative de la forme présentée. Le bruit peut être dû au dispositif d'acquisition, aux

conditions d'acquisition (éclairage, mise incorrecte du document…), ou encore à la qualité du document d'origine Parmi les opérations de prétraitements généralement utilisées, citons : la binarisation, la normalisation.

- **La binarisation :**

Dans un système de reconnaissance d'écriture, la numérisation des images est la première étape de traitement. Elle permet de passer d'une image de niveaux de gris à une image binaire composée de deux valeurs 0 et 1, plus simple à traiter. En général, on utilise un seuil de binarisation approprié qui traduit la limite des contrastes fort et faible dans l'image. Selon la méthode de calcul du seuil de binarisation, on distingue deux types de binarisation : le seuillage globale et le seuillage adaptatif. Le premier type consiste à prendre un seuil ajustable mais identique pour toute l'image, cette méthode convient pour les documents de contraste fort et de bonne qualité. Dans le cas contraire, le seuillage adaptatif est plus approprié, dans ce cas le calcul du seuil se fait localement en fonction du voisinage du pixel traité. [11]

- **La normalisation :**

L'un des problèmes rencontrés en la reconnaissance des chiffres manuscrits est la taille d'image et l'inclinaison des lignes, qui introduit des difficultés pour la segmentation donc l'étape de normalisation est nécessaire Dans un système de reconnaissance d'écriture quand la taille varie avec le style de chiffre manuscrit pour chaque scripteur.

I.2.1.3 Phase de Segmentation :

Dans un système de reconnaissance de l'écriture manuscrite, les données à traiter sont des images. La mise en œuvre d'une étape de segmentation permet de diviser l'image en différentes régions (imagettes) connexes présentant une homogénéité selon certain critère, et de taille moins importantes qui peuvent être des graphèmes, des lettres ou bien des sous-mots. Cependant une imagette reste une matrice de pixels. [12]

L'objectif le plus potentiel de la segmentation des chaine de chiffres, c'est la séparation des chiffres l'un de l'autre.

I.2.1.4 Phase d'extraction de caractéristiques :

Dans un système de reconnaissance des chiffres manuscrits Le but ultime d'extraction de caractéristiques est obtenir le volume d'informations la plus pertinentes qui sera fourni au système. C'est une étape critique lors de la construction d'un système de reconnaissance. L'une des raisons pour laquelle cette phase pose un problème est qu'une plusieurs techniques d'extraction s'accompagne d'une perte d'information. De ce fait, il faut effectuer un compromis entre la quantité et la qualité de l'information [13].

La réduction du nombre de caractéristiques a de nombreux avantages : elle permet d'améliorer la visualisation et la compréhension des données, de réduire les temps d'apprentissage et de classification des systèmes, d'améliorer les performances en classification, et permet de réduire la taille des bases d'apprentissage.

I.2.1.5 Phase de Classification :

Après la segmentation de caractères et l'extraction des attributs caractéristiques, une étape de reconnaissance basée sur la classification de caractères est employée.

Les techniques de la classification reposent sur une stratégie de décision qui permet de catégoriser un objet le mieux possible selon certains critères d'optimisation. Et qui transforme les attributs caractérisant les formes en appartenance à une classe (passage de l'espace de codage vers l'espace de décision).

Le type d'une méthode de classification se décline généralement en deux familles : Le mode supervisé et le mode non supervisé. [14]

Classification supervisée : cette technique est basée sur l'étiquetage des observations en affectant chaque observation à une classe (supervisés où la sortie correcte doit être fournie à l'avance).

Classification non supervisée : aucune des observations n'est étiquetée (non supervisés où la sortie correcte n'est pas exigée à l'avance, elle résulte après une étape d'apprentissage).

Généralement, pour évaluer la performance d'un classificateur, on poursuit deux étapes Principales qui sont l'apprentissage et le test (Reconnaissance et décision).

Étape d'apprentissage: Les données d'apprentissage sont les données utilisées pour construire un classificateur capable de reconnaître des formes inconnues pour caractériser les classes. [15]

Étape de Reconnaissance : et décision : Elle cherche parmi les modèles de référence en présence, ceux qui lui sont les plus proches. Le problème revient à affecter une forme inconnue à l'une des classes obtenues pendant l'apprentissage. [16]

Si toutes les données sont employées pour l'apprentissage et le test en même temps, il est possible que le classificateur soit incapable de reconnaître d'autres formes inconnues. C'est pourquoi il est important d'avoir deux ensembles de données pour améliorer la généralisation d'un classificateur : le premier ensemble est spécifié pour l'apprentissage, le deuxième pour le test.

I.3 Quelques travaux réalisés :

Pour concevoir un système de reconnaissance des chiffres manuscrits, il faut avoir les différents travaux ultérieurs qu'ils ont été réalisés sur la base données MNIST, nous allons présenter quelques travaux dans **le tableau I.2** :

Année	Editeur	Base donnée	Méthode utilisée	Précision obtenue(%)	Références
2002	Mayraz et Hinton	MNIST	Produit expert	98.3	[17]
2004	Kussul et Baidyk	MNIST	LIRA (Limited Réceptive Area)	99.59	[18]
2010	Wu et Zhang	MNIST	Caractéristique de direction + SVM(Les Machines à Vecteur Support)	98.81	[19]
2014	Ebrahimzadeh et Jampour	MNIST	HOG (Histogramme gradient Orienté) + SVM	97.25	[20]
2014	Wakahara et Yamashita	MNIST	GAT (Global affine transformation) NNDEGD (Nearest-Neighbor Distance of Equi-Gradient Direction)	99.51	[21]
2015	Lee et al	MNIST	Réseaux neurones convolutionnels	99.71	[22]
2016	Wakahara et Yamashita	MNIST	NNDEGD (Nearest-Neighbor Distance of Equi-Gradient Direction) +GPT (global projection transformation)	92.5	[23]
2016	Dundar et al	MNIST	Réseaux convolutionnels profond+Kmeans	98.6	[24]

Tableau I.2 Quelque travaux réalisés sur la reconnaissance des chiffres.

Conclusion :

Dans ce chapitre, nous avons présenté brièvement la reconnaissance de l'écriture manuscrite et ses différents aspects ainsi que les approches utilisés et son domaine d'application.

Après, nous avons introduit la notion de la reconnaissance des chiffres manuscrits et son processus de reconnaissance qui est basé généralement sur : acquisition, prétraitement, segmentation, extraction des caractéristiques, classification.

Enfin, nous avons présenté quelques travaux déjà effectués sur la reconnaissance des chiffres manuscrits sous forme d'un tableau qui contient les noms des réalisateurs et leurs méthodes utilisés ainsi que la précision obtenue.

Dans le chapitre suivant nous allons détailler la méthode de classification basée sur les réseaux de neurones convolutionnels, voir le taux de précision qu'ils ont déjà obtenu auparavant, alors cette méthode sera l'objet principal de ce mémoire.

Chapitre II

Deep Learning

II.1 Introduction :

Le Deep Learning est un nouveau domaine de recherche du Machine Learning, qui a été introduit dans le but de rapprocher le ML de son objectif principal : l'intelligence artificielle. Il concerne les algorithmes inspirés par la structure et le fonctionnement du cerveau. Ils peuvent apprendre plusieurs niveaux de représentation dans le but de modéliser des relations complexes entre les données.

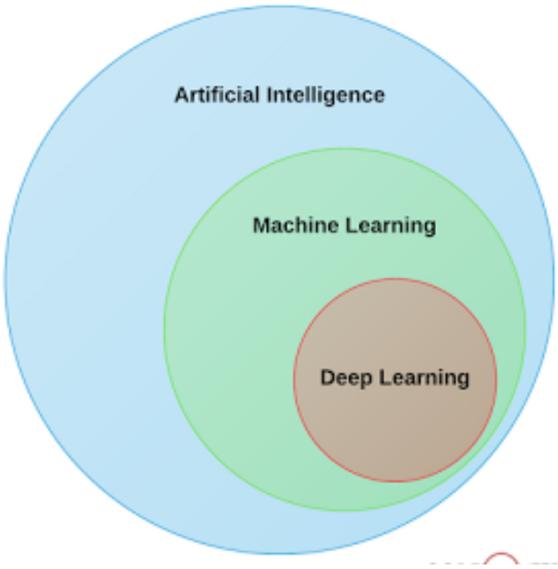

Figure II.1 la relation entre IA et ML et DL

II.1.1 Définition :

Le Deep Learning est basé sur l'idée des réseaux de neurones artificiels et il est taillé pour gérer de larges quantités de données en ajoutant des couches au réseau. Un modèle de deep learning a la capacité d'extraire des caractéristiques à partir des données brutes grâce aux multiples couches de traitement composé de multiples transformations linéaires et non linéaires et apprendre sur ces caractéristiques petit à petit à travers chaque couche avec une intervention humaine minime.[30]

Le terme "Deep Learning" a été introduit pour la première fois au ML par Dechter (1986) [31], et aux réseaux neuronaux artificiels par Aizenberg et al (2000).

II.1.2 Histoire du Deep Learning :

Le Deep Learning a connu une évolution énorme dans la dernière décennie, il a été inspiré par le mécanisme du cerveau humain, le **tableau II.1** présente l'historique des recherches effectuées sur ce domaine.

Année	Contributeur	Contribution
300 AC	Aristotle	introduction de l'associationnisme, début de l'histoire des humains qui essayent de comprendre le cerveau
1873	Alexander Bain	introduction du Neural Groupings comme les premiers modèles de réseaux de neurones
1943	McCulloch and Pitts	introduction du McCulloch–Pitts (MCP) modèle considérer comme L'ancêtre des réseaux de neurones artificiels
1949	Donald Hebb	considérer comme le père des réseaux de neurones, il introduit la règle d'apprentissage de Hebb qui servira de fondation pour les réseaux de neurones modernes
1958	Frank Rosenblatt	introduction du premier perceptron
1974	Paul Werbos	introduction de la retro propagation
1980	Teuvo Kohonen	introduction des cartes auto organisatrices
1980	Kunihiko Fukushima	introduction du Neocognitron, qui a inspiré les réseaux de neurones convolutif
1982	John Hopfield	introduction des réseaux de Hopfield
1985	Hilton and Sejnowski	introduction des machines de Boltzmann
1986	Paul Smolensky	introduction de Harmonium, qui sera connu plus tard comme machines de Boltzmann restreintes
1986	Michael I. Jordan	définition et introduction des réseaux de neurones récurrents
1990	Yann LeCun	introduction de LeNet et montra la capacité des réseaux de neurones profond
1997	Schuster and Paliwal	introduction des réseaux de neurones récurrents bidirectionnels
1997	Hochreiter and Schmidhuber	introduction de LSTM, qui a résolu le problème du vanishing gradient dans les réseaux de neurones récurrent
2006	Geoffrey Hinton	introduction des Deep belief Network
2009	Salakhutdinov and Hinton	introduction des Deep Boltzmann Machines
2012	Alex Krizhevsky	introduction de AlexNet qui remporta le challenge ImageNet

Tableau II.1 Les étapes majeures du Deep Learning [32]

II.1.3 Domaines d'application de Deep Learning :

Des applications de Deep Learning sont utilisées dans divers secteurs, de la conduite automatisée aux dispositifs médicaux. [33]

Conduite automatisée : Les chercheurs du secteur automobile ont recours au Deep Learning pour détecter automatiquement des objets tels que les panneaux stop et les feux de circulation. Le Deep Learning est également utilisé pour détecter les piétons, évitant ainsi nombre d'accidents**.** [34]

Aérospatiale et défense : Le Deep Learning sert à identifier des objets à partir de satellites utilisés pour localiser des zones d'intérêt et identifier quels secteurs sont sûrs ou dangereux pour les troupes au sol. [35]

Recherche médicale : À l'aide du Deep Learning, les chercheurs en cancérologie peuvent dépister automatiquement les cellules cancéreuses. Des équipes de l'Université de Californie à Los Angeles (UCLA) ont conçu un microscope qui génère un ensemble de données de grande dimension afin d'entraîner une application de Deep Learning à identifier avec précision des cellules cancéreuses. [36]

Automatisation industrielle : Le Deep Learning participe à l'amélioration de la sécurité des employés travaillant à proximité d'équipements lourds, en détectant automatiquement les situations dans lesquelles la distance de sécurité qui sépare le personnel ou les objets des machines est insuffisante. [37]

Électronique : Le Deep Learning est utilisé pour la reconnaissance audio et vocale. Par exemple, les appareils d'assistance à domicile qui répondent à votre voix et connaissent vos préférences fonctionnent grâce à des applications de Deep Learning. [37]

II.2 Les différents types d'architectures :

Il existe un grand nombre de variantes d'architectures. La plupart d'entre elles sont dérivées de certaines architectures parentales originales. Il n'est pas toujours possible de comparer les performances de toutes les architectures, car elles ne sont pas toutes évaluées sur les mêmes ensembles de données. Le Deep Learning est un domaine à croissance rapide, et de nouvelles

architectures, variantes ou algorithmes apparaissent toutes les semaines, on cite parmi eux : Réseaux de neurones à convolution (CNN) et Réseaux de neurones récurrents (RNN).

II.2.1 Réseau de neurones récurrents :

L'idée derrière les RNN est d'utiliser des informations séquentielles. Dans un réseau neuronal traditionnel, nous supposons que toutes les entrées (et les sorties) sont indépendantes les unes des autres. Mais pour de nombreuses tâches, si on veut prédire le prochain mot dans une phrase, il faut connaître les mots qui sont venus avant. Les RNN sont appelés récurrents car ils exécutent la même tâche pour chaque élément d'une séquence, la sortie étant dépendante des calculs précédents. Une autre façon de penser les RNN est qu'ils ont une « mémoire » qui capture l'information sur ce qui a été calculé jusqu'ici. En théorie, les RNN peuvent utiliser des informations dans des séquences arbitrairement longues, mais dans la pratique, on les limite à regarder seulement quelques étapes en arrière. [38]

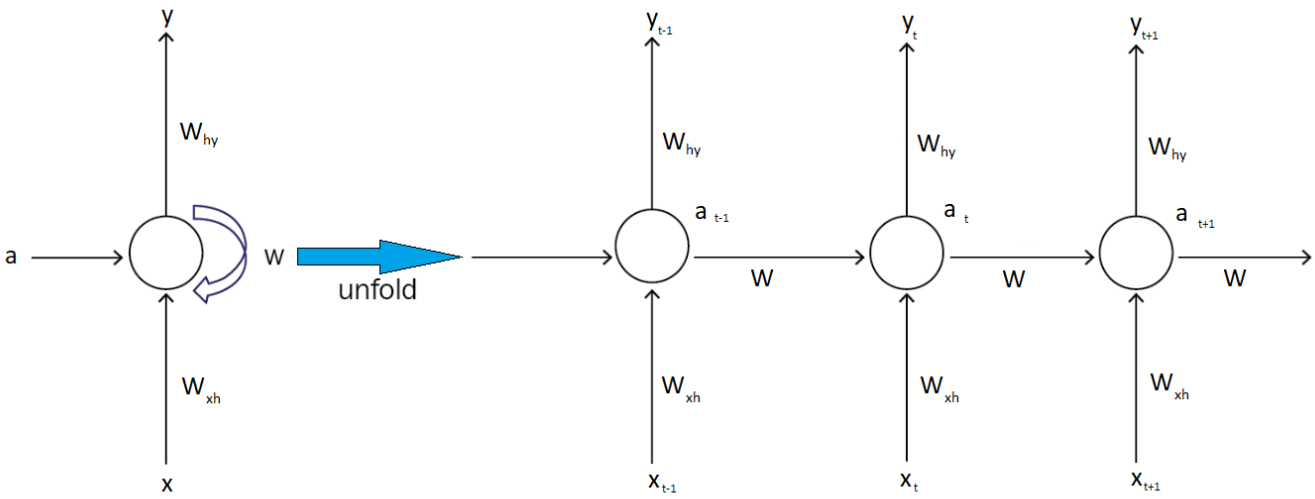

Figure II.2 Architecture de RNN

II.2.1.1 Domaines d'application :

Les RNN ont connu un grand succès dans de nombreuses tâches parmi eux :

- **Traduction automatique**: La traduction automatique est semblable à la modélisation du langage en ce sens que l'entrée est une séquence de mots dans une langue source (par exemple, l'arabe). Nous voulons générer une séquence de mots dans une langue cible (par exemple, l'anglais). Une différence majeure est que notre sortie ne commence que lorsque nous avons vu l'entrée complète, car le premier mot de la phrases traduites nécessite des informations capturées à partir de la séquence d'entrée complète. [39]
- **La reconnaissance vocale**: La reconnaissance vocale est une technique informatique qui permet d'analyser la voix humaine pour la transcrire sous la forme d'un texte exploitable par une machine. [40]
- **Description des images**: Ensemble avec les réseaux de neurones convolutifs, les RNN ont été utilisés pour générer des descriptions pour des images non étiquetées. Il est tout à fait incroyable de voir à quel point cela semble fonctionner. Le modèle combiné aligne même des mots générés avec des caractéristiques trouvées dans les images. [41]

II.2.2 Réseaux de neurones convolutifs :

II.2.2.1 Introduction :

L'une des capacités fondamentales de l'humain est celle d'analyser son environnement. Dans la majorité des cas, cela passe par la reconnaissance des éléments de notre champ de vision : trouver les autres personnes, identifier les voitures, les animaux…

Jusqu'à l'émergence des réseaux de neurones convolutifs, en 2012 avec *Alex Krizhevsky*, la tâche était difficile pour un ordinateur. Heureusement, l'approche de ces réseaux inspirés de notre œil (plus particulièrement du fait que certains neurones de notre aire visuelle ne réagissent qu'aux bordures verticales et d'autres aux horizontales/diagonales) a ouvert de nombreuses applications, que ce soit en imagerie médicale, véhicules autonomes, reconnaissance faciale, et même analyse de textes.

II.2.2.2 Définition :

Convolutional Neural Network (CNN). Est un algorithme de Deep Learning (apprentissage profond en français) doué dans l'analyse d'images. En effet, il s'appuie sur des filtres Les réseaux convolutifs ont connu un succès considérable dans les applications pratiques. Le nom « réseau de neurones convolutif » indique que le réseau emploie une opération mathématique appelée convolution. La convolution est une opération linéaire spéciale. Les réseaux convolutifs

sont simplement des réseaux de neurones qui utilisent la convolution à la place de la multiplication matricielle dans au moins une de leurs couches. [42]

Ils ont de larges applications dans la reconnaissance de l'image et de la vidéo, les systèmes de recommandations et le traitement du langage naturel. [43]

Parmi les réseaux convolutifs célèbres on trouve :

- **LeNet** : Les premières applications réussies des réseaux convolutifs ont été développées par Yann LeCun dans les années 1990. Parmi ceux-ci, le plus connu est l'architecture LeNet utilisée pour lire les codes postaux, les chiffres, etc. [44]
- **AlexNet** : Le premier travail qui a popularisé les réseaux convolutifs dans la vision par ordinateur était AlexNet, développé par Alex Krizhevsky, Ilya Sutskever et Geoff Hinton. AlexNet a été soumis au défi ImageNet ILSVRC en 2012 et a nettement surpassé ses concurrents. Le réseau avait une architecture très similaire à LeNet, mais était plus profond, plus grand et comportait des couches convolutives empilées les unes sur les autres (auparavant, il était commun de ne disposer que d'une seule couche convolutifs toujours immédiatement suivie d'une couche de pooling). [45]
- **ZFnet**[46] : Le vainqueur de ILSVRC challenge 2013 était un réseau convolutif de Matthew Zeiler et Rob Fergus. Il est devenu ZFNet (abréviation de Zeiler et Fergus Net). C'était une amélioration d'AlexNet en ajustant les hyper-paramètres de l'architecture, en particulier en élargissant la taille des couches convolutifs et en réduisant la taille du noyau sur la première couche.

- **GoogLeNet**[47] : Le vainqueur de ILSVRC challenge 2014 était un réseau convolutif de Szegedy et al. De Google. Sa principale contribution a été le développement d'un module inception qui a considérablement réduit le nombre de paramètres dans le réseau (4M, par rapport à AlexNet avec 60M). En outre, ce module utilise le global AVG pooling au lieu du PMC à la fin du réseau, ce qui élimine une grande quantité de paramètres. Il existe également plusieurs versions de GoogLeNet, parmi elles, Inception-v4 [48]
- **ResNet**[49] : Residual network développé par Kaiming He et al. A été le vainqueur de ILSVRC 2015. Il présente des sauts de connexion et une forte utilisation de la batch normalisation. Il utilise aussi le global AVG pooling au lieu du PMC à la fin.

II.2.2.3 Construction d'un réseau CNN :

La première partie d'un CNN est la partie convolutive à proprement parler. Elle fonctionne comme un extracteur de caractéristiques des images. Une image est passée à travers une succession de filtres, ou noyaux de convolution, créant de nouvelles images appelées cartes de convolutions. Certains filtres intermédiaires réduisent la résolution de l'image par une opération de maximum local. Au final, les cartes de convolutions sont mises à plat et concaténées en un vecteur de caractéristiques, appelé code CNN. [50]

Une architecture de réseau de neurones convolutifs est formée par un empilement de couches de traitement : la couche de convolution (CONV), la couche de pooling (POOL), la couche de correction (ReLU) et la couche « entièrement connectée » (FC), comme présentée dans la figureII.3.

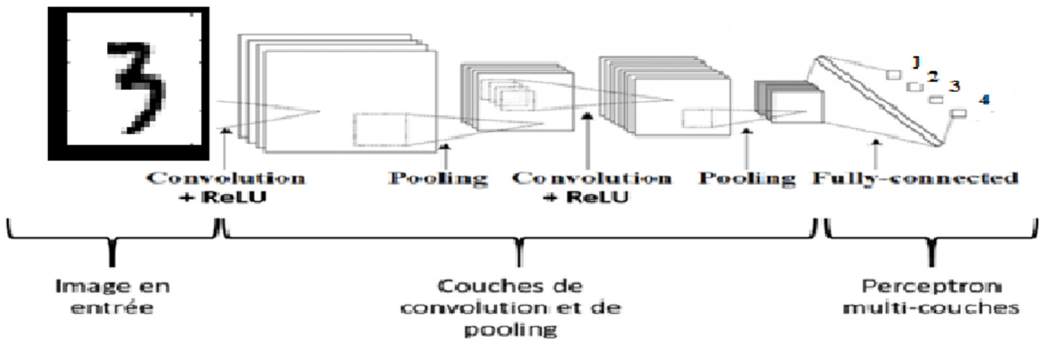

Figure II.3 Architecture standard de CNN

La figure II.3 présente l'empilement de couches dans le réseau de neurone covolutif, afin de reconnaitre le chiffre 3.

❖ La couche de convolution (CONV) :

La convolution est un outil mathématique simple qui est très largement utilisé pour le traitement d'image, ce qui explique que les réseaux de neurones à convolution soient particulièrement bien adaptés à la reconnaissance d'image.

La figure **II.4** représente la convolution qui agit comme un filtrage. On définit une taille de fenêtre qui va se balader à travers toute l'image (kernel_size).

Au tout début de la convolution, la fenêtre sera positionnée tout en haut à gauche de l'image puis elle va se décaler d'un certain nombre de cases (c'est ce que l'on appelle le pas) vers la droite et lorsqu'elle arrivera au bout de l'image, elle se décalera d'un pas vers le bas ainsi de-suite jusqu'à ce que le filtre est parcourue la totalité de l'image. [51]

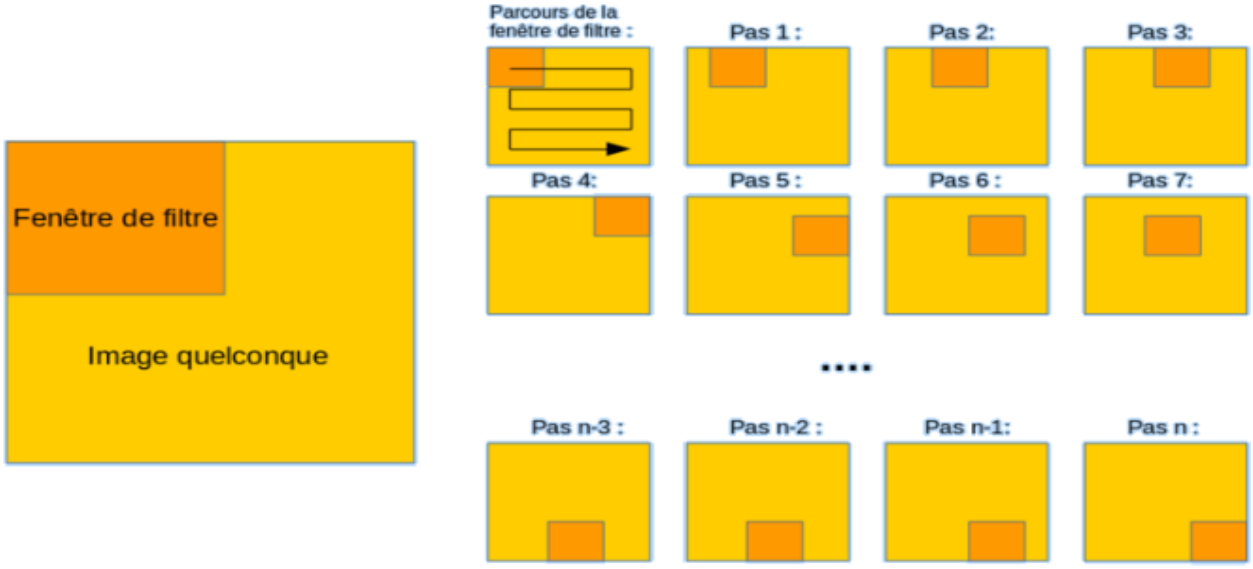

Figure II.4 schéma du parcours de la fenêtre de filtre sur l'image

Le but est de se servir des valeurs présentes dans le filtre à chaque pas. Par exemple si l'on définit une fenêtre 3 par 3, cela représentera 9 cases du tableau (c'est à dire 9 pixels). La convolution va effectuer une opération avec ces 9 pixels.

La couche de convolution contient deux paramètres principaux : le nombre de filtres et la forme de filtre.

Nombre de filtres : Comme la taille des images intermédiaires diminue avec la profondeur du traitement, les couches proches de l'entrée ont tendance à avoir moins de filtres tandis que les couches plus proches de la sortie peuvent en avoir davantage. Pour égaliser le calcul à chaque couche, le produit du nombre de caractéristiques et le nombre de pixels traités est généralement choisi pour être à peu près constant à travers les couches. Pour préserver l'information en entrée, il faudrait maintenir le nombre de sorties intermédiaires (nombre d'images intermédiaire multiplié par le nombre de positions de pixel) pour être croissante (au sens large) d'une couche

à l'autre. Le nombre d'images intermédiaires contrôle directement la puissance du système, dépend du nombre d'exemples disponibles et la complexité du traitement. [51]

Forme du filtre : Les formes de filtre varient grandement dans la littérature. Ils sont généralement choisis en fonction de l'ensemble de données. Les meilleurs résultats sur les images de MNIST (28x28) sont habituellement dans la gamme de 3x3 sur la première couche, tandis que les ensembles de données d'images naturelles (souvent avec des centaines de pixels dans chaque dimension) ont tendance à utiliser de plus grands filtres de première couche de 12x12, voire 15x15. Le défi est donc de trouver le bon niveau de granularité de manière à créer des abstractions à l'échelle appropriée et adaptée à chaque cas. [51]

❖ La couche de pooling (POOL) *:*

L'opération de pooling consiste à réduire la taille des images (réduire le nombre de paramètres et de calculs dans le réseau), on améliore ainsi l'efficacité du réseau et on évite le sur-apprentissage, tout en préservant leurs caractéristiques importantes. [51]

Pour cela, on découpe l'image en cellules régulière, puis on garde au sein de chaque cellule la valeur maximale. En pratique, on utilise souvent des cellules carrées de petite taille pour ne pas perdre trop d'informations. Les choix les plus communs sont des cellules adjacentes de taille (2 × 2) pixels qui ne se chevauchent pas.
Il existe plusieurs types de pooling :

- Le **"*max pooling*",** qui revient à prendre la valeur maximale de la sélection. C'est le type le plus utilisé car il est rapide à calculer (immédiat), et permet de simplifier efficacement l'image

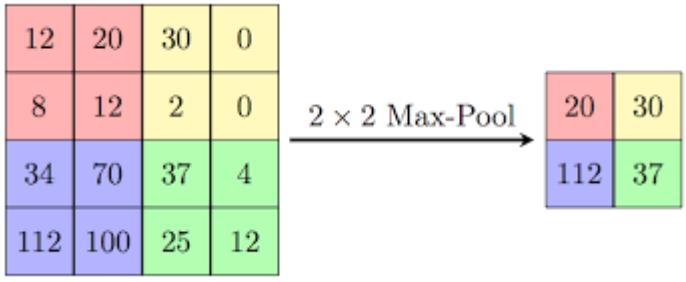

Figure II.5 représentation de maxpooling

- Le **"*mean pooling*"** (ou average pooling), soit la moyenne des pixels de la sélection: on calcule la somme de toutes les valeurs et on divise par le nombre de valeurs. On obtient ainsi une valeur intermédiaire pour représenter ce lot de pixels

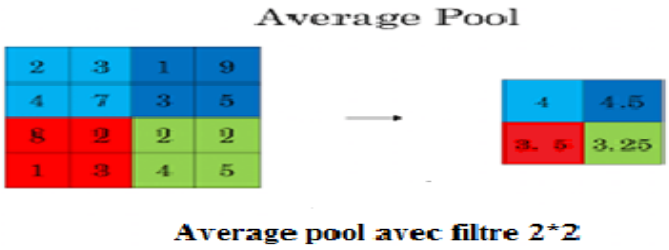

Figure II.6 représentation de meanpooling

- Le **"*sum pooling*",** c'est la moyenne sans avoir divisé par le nombre de valeurs (on ne calcule que leur somme)

De manière générale, il est recommandé d'utiliser max-pooling, car il se distingue de mean-pooling sur les cas extrêmes mais il est quasiment équivalent à mean-pooling dans les autres cas.

❖ La couche de correction (ReLU) (REctified Linear Unit) :

La fonction d'activation **ReLU** est une fonction dite "rectifier" très utilisée en Deep Learning. [29] Dans les réseau de neurones convolutionnels, elle est appliquée très souvent en sortie d'une couche de convolution, elle est définie par ReLU(x)=max(0,x)

ReLU par sa définition, est une fonction qui vient briser une partie de la linéarité en supprimant les valeurs négatives, ce qui permet également d'accélérer les calculs. D'autre part, elle n'influence pas les caractéristiques misent en évidence par la convolution. [52]

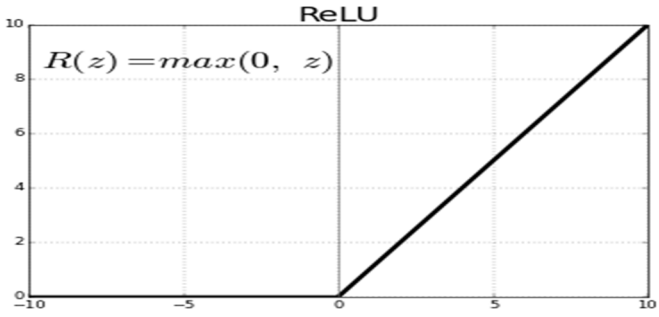

Figure II.7 Représentation de la fonction RELU

Il existe d'autres formes de correction :

- La correction par *tangente hyperbolique f(x)=tanh(x)*
- La correction par la *tangente hyperbolique saturante*: *f(x)=|tanh(x)|*
- La correction par la *fonction sigmoïde* {f(x)=(1+e^{-x})^{-1}}. *f(x)= $(1+e^{-x})^{-1}$*

❖ La couche fully-connected « entièrement connectée » (FC) :

Après plusieurs couches de convolution et de max-pooling, le raisonnement de haut niveau dans le réseau neuronal se fait via des couches entièrement connectées. Les neurones dans une couche entièrement connectée ont des connexions vers toutes les sorties de la couche précédente. La couche fully-connected permet de classifier l'image en entrée du réseau : elle renvoie un vecteur de taille N, où N est le nombre de classes dans notre problème de classification d'images. Chaque élément du vecteur indique la probabilité pour l'image en entrée d'appartenir à une classe. [52]

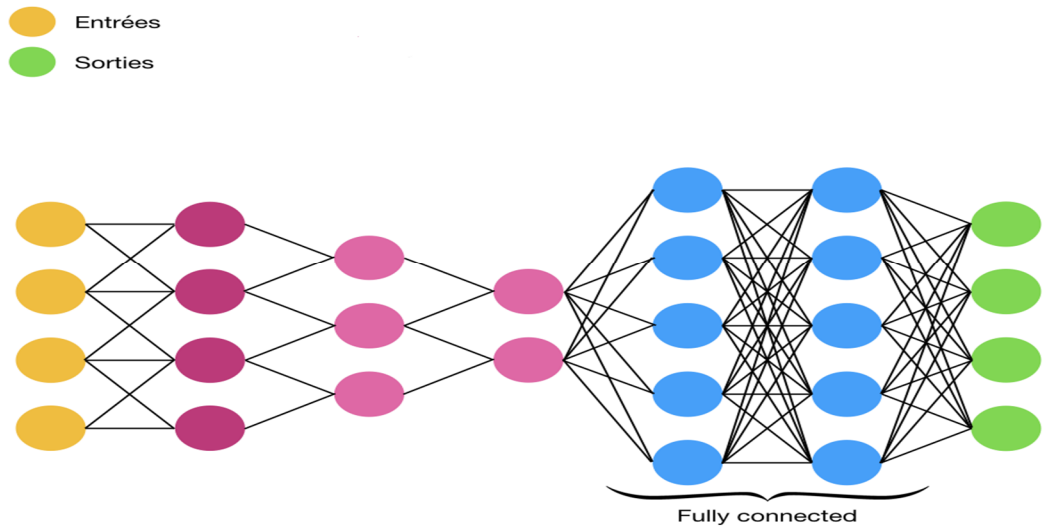

Figure II.8 Représentation de la couche fully-connected

II.2.2.4 Apprentissage des réseaux de neurones convolutifs :

L'apprentissage profond consiste à entrainer un réseau de neurones constitué d'une série de modules, chacun représentant une étape de traitement. Chaque module comporte des paramètres similaires aux poids des réseaux de neurones classiques. L'ensemble de ces paramètres sont ajustés de manière à rapprocher la sortie estimée par le système de la sortie de la vérité terrain.

Cela peut se faire par l'intermédiaire d'un algorithme d'optimisation basé sur la descente de gradient stochastique par lot (ou mini-batch).

Un élément important dans l'algorithme de descente de gradient est le pas (ou taux) d'apprentissage. Si le pas trop petit, l'algorithme devra effectuer un grand nombre d'itérations pour converger et prendra beaucoup de temps. Inversement, si le pas est trop élevé, l'algorithme risque de diverger et de s'éloigner ainsi de la bonne solution. De plus, ce pas d'apprentissage est global, ce qui signifie que tous les neurones utilisent le même taux, alors que toutes les données ne suivent pas forcément la même distribution et donc ne nécessitent pas d'adapter le réseau de la même manière. Pour pallier ces problèmes, de nombreuses variantes de l'algorithme de descente de gradient stochastiques ont été proposées. Parmi elles, on cite la méthode Adaptative Adam [53] que nous avons utilisée dans nos tests. Adam est l'un des algorithmes les plus récents et les plus efficaces pour l'optimisation par descente de gradient. Il calcule un taux d'apprentissage adaptatif pour chaque paramètre. En outre, le gradient dépend des estimations adaptatives des moments de premier et second ordre. [41]

Conclusion :

Dans ce chapitre, nous avons présenté les réseaux de neurones récurrents et les réseaux de neurones convolutionnels. Ce dernier est capable d'extraire des caractéristiques d'images présentées en entrée et de les classifier. Les réseaux de neurones convolutionnels présentent cependant un certain nombre de limitations, en premier lieu, les hyper paramètres du réseau sont difficiles à évaluer a priori. En effet, le nombre de couches, les nombre de neurones par couche ou encore les différentes connexions entre couches sont des éléments cruciaux et essentiellement déterminés par une bonne intuition ou par une succession de tests/calcul d'erreurs (ce qui est coûteux en temps). Le nombre d'échantillons d'apprentissage est également un élément déterminant.

Chapitre III
Implémentation et résultats

III.1 introduction :

Les systèmes de reconnaissance des chiffres manuscrits font face à plusieurs défis, y compris la variation illimitée de l'écriture humaine et des grandes bases des données publiques.

Dans ce chapitre on va définir notre architecture de CNN mise en point pour la reconnaissance des chiffres manuscrits sur la base MNIST, via le langage de programmation python.

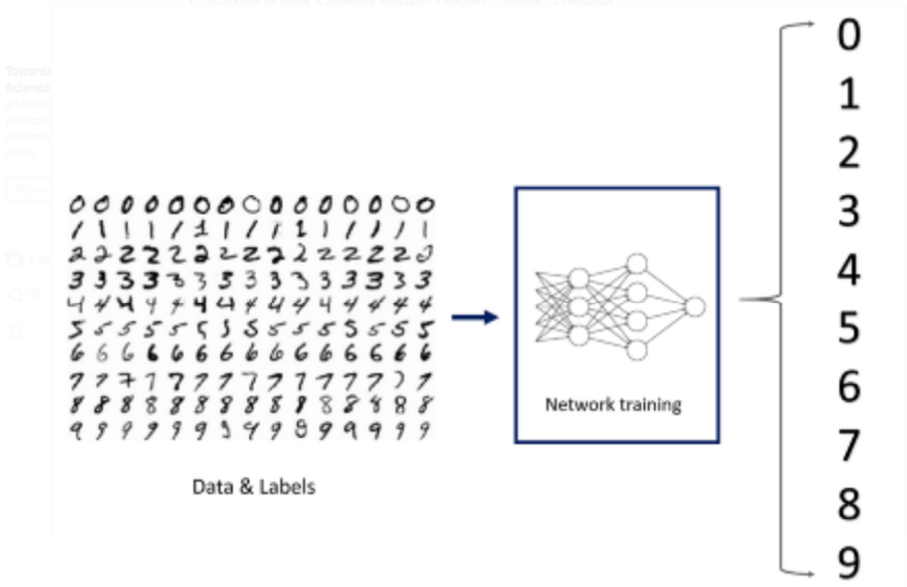

Figure III.1 Reconnaissance des chiffres manuscrits de la base MNIST.

III.2 Python :

III.2.1 Définir le python :

Python est un langage de programmation puissant et facile à apprendre. Il a des structures de données de haut niveau efficaces et une approche simple mais efficace de la programmation orientée objet. La syntaxe élégante et le typage dynamique de Python, ainsi que sa nature interprétée, en font un langage idéal pour les scripts et le développement rapide d'applications dans de nombreux domaines sur la plupart des plates-formes.

L'interpréteur Python et la bibliothèque standard étendue sont disponibles gratuitement sous forme source ou binaire pour toutes les principales plates-formes à partir du site Web

Python, https://www.python.org/ , et peuvent être librement distribués. Le même site contient également des distributions et des pointeurs vers de nombreux modules, programmes et outils Python tiers gratuits, ainsi que de la documentation supplémentaire. [55]

III.2.2 Librairies utilisées dans l'implémentation :

TensorFlow :

TensorFlow est un Framework de programmation pour le calcul numérique qui a été rendu Open Source par Google en Novembre 2015. Depuis son release, TensorFlow n'a cessé de gagner en popularité, pour devenir très rapidement l'un des Framework les plus utilisés pour le Deep Learning et donc les réseaux de neurones. Aujourd'hui, les principaux produits de Google sont basés sur TensorFlow: Gmail, Google Photos, Reconnaissance de voix. [56]

Keras :

Keras est l'outil le plus utilisé en Python dans le monde pour l'apprentissage profond (deep Learning). Cette librairie open-source, créée par François Chollet, Permet de créer facilement et rapidement des réseaux de neurones, en se basant sur les principaux frameworks (Tensorflow, Pytorch). Keras permet de diminuer de 30% le temps de développement d'un prototype de réseaux de neurones). [57]

Matplotlib :

Matplotlib est une bibliothèque du langage de programmation Python destinée à tracer et visualiser des données sous formes de graphiques. Elle peut être combinée avec les bibliothèques python de calcul scientifique Keras, NumPy et SciPy. Matplotlib est distribuée librement et gratuitement.

III.2.3 Configuration Utilisé dans l'implémentation :

La configuration du matériel utilisé dans notre implémentation est :

- Un PC portable HP Core i5 CPU 2.60 GHZ
- RAM de taille 4 GO
- Disque dur de taille 500 GO
- Système d'exploitation Windows 64 bits

III.3 La base donnée :

La base de données MNIST de chiffres manuscrits, contient un ensemble de formation de 60 000 exemples et un ensemble de test de 10 000 exemples (**Tableau III.1**). Il s'agit d'un sous-ensemble d'un ensemble plus large disponible auprès de la base donnée NIST. Les chiffres ont été normalisés en taille et centrés dans une image de taille fixe (28, 28, 1). [58]

Classes	*APPRENTISSAGE*	*TEST*	*TOTALE*
0	5923	980	6903
1	6742	1135	7877
2	5958	1032	6990
3	6131	1010	7141
4	5842	982	6824
5	5421	892	6313
6	5918	958	6876
7	6265	1028	7293
8	5851	974	6825
9	5949	1009	6958
Totale	60000	10000	70000

Tableau III.1 : Répartition de la base MNIST pour les chiffres.

Figure III.2 : Exemples de la base de données MNIST

III.4 Construction de l'architecture CNN :

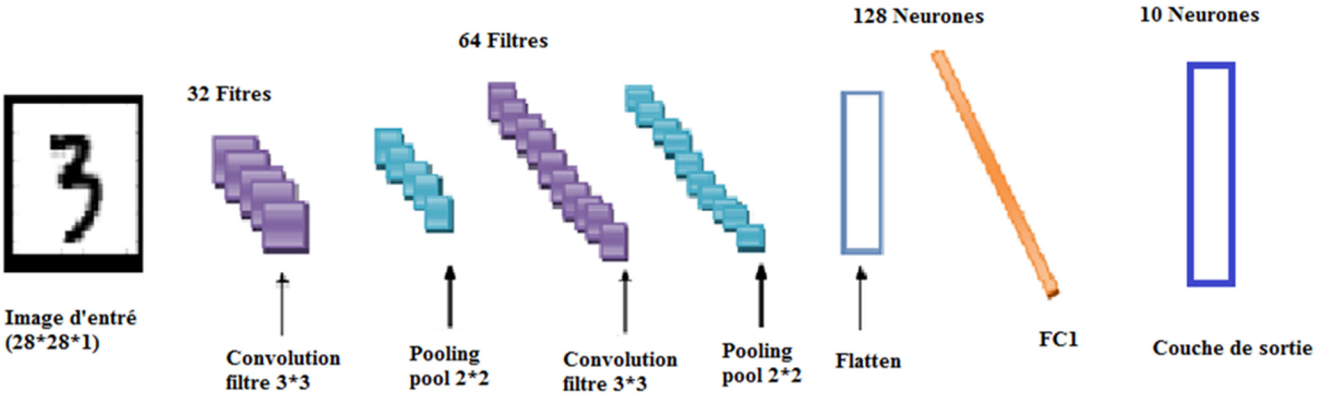

Figure III.3 Architecture de CNN

Le modèle que nous présentons dans la **figure III.3** est composé de deux couches de convolution et deux couches de maxpooling et d'une couche de fully connected, enfin une couche de sortie.

L'image en entrée est de taille 28*28, l'image passe d'abord à la première couche de convolution. Cette couche est composée de 32 filtres de taille 3*3, Chacune de nos couches de convolution est suivie d'une fonction d'activation ReLU. Cette fonction force les neurones à retourner des valeurs positives, après cette convolution, 32 features maps (images caractéristiques) de taille 28*28 seront créés. Ensuite on applique Maxpooling pour réduire la taille de l'image ainsi la quantité de paramètres et de calcul. À la sortie de cette couche, nous aurons 32 feature maps de taille 13*13. On répète la même chose avec la deuxième couche de convolution cette couche est composée de 64 filtres, la fonction d'activation ReLU est appliquée toujours sur chaque convolution. Une couche de Maxpooling est appliquée après la deuxième couche de convolution. À la sortie de cette couche, nous aurons 64 feature maps de taille 5*5. Pour ne pas tomber dans le problème de sur apprentissage il faut utiliser l'instruction dropout elle est très efficace pour les réseaux de neurones, elle permet de désactiver un nombre de neurones selon notre configuration, cette dernière sera utilisée aussi à la sortie de la couche de fully connected. Le vecteur de caractéristiques issu des convolutions a une dimension de 1600.

Après ces couches de convolution, nous utilisons un réseau de neurones composé d'une couche fully connected, elle possède 128 neurones où la fonction d'activation utilisée est le ReLU, pour la couche de sortie, sa fonction d'activation est softmax qui permet de calculer la distribution de probabilité des 10 classes (nombre de classe dans la base donnée MNIST.

III.5 Implémentation :

Pour la partie implémentation nous allons présenter les différentes étapes de notre code source qui sont :

- Importation des bibliothèques nécessaires.
- Importation de la base donnée.
- Traitement des images de la base donnée.
- Initialisation des hyper paramètres.
- Construction de l'architecture CNN.
- Compilation et ajustement du modèle.
- Évaluation du modèle.

III.5.1 Importation des bibliothèques nécessaires :

```
#  importation des biblio nécessaires
import keras
import tensorflow as tf
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten
from keras.layers import Conv2D, MaxPooling2D
import matplotlib.pyplot as plt
```

- **from keras.models import Sequential**

  La méthode la plus facile pour construire un modèle dans keras, il permet de construire couche par couche, chaque couche a un poids qui correspond à la couche suivante.

- **from keras.layers import Dense, Dropout, Flatten**

- flatten : ça permet de rendre un modèle sous forme d'une colonne.
- Dense : régulateur de couche, chaque couche reçoit une entré de chaque neurone de la couche précédente.
- Dropout : elle permet d'éviter le sur apprentissage « overfitting » en désactivant une partie de neurones.

❖ **from keras.layers import Conv2D, MaxPooling2D**

- Conv2D : agit comme un filtrage. On définit une taille de fenêtre qui va se balader à travers toute l'image (kernel_size).
- Max pooling: est utilisé pour réduire les dimensions spatiales du volume de sortie.

❖ **import matplotlib.pyplot as plt**

Est une bibliothèque du langage de programmation Python destinée à tracer et visualiser des données sous formes de graphiques.

III.5.2 Importation de la base donnée :

```
# importer notre base de donnée MNIST
(x_train, y_train), (x_test, y_test) = tf.keras.datasets.mnist.load_data()
```

Cette ligne permet de récupérer les données de la base MNIST déjà téléchargée à partir de Keras via ce folder : 'https://storage.googleapis.com/tensorflow/tf-keras-datasets/'

III.5.3 Traitement des images de la base donnée :

```
# remodeler le vecteur a 4 dim pour fonctionner avec l'interface de keras
x_train = x_train.reshape(x_train.shape[0], 28, 28, 1)
x_test = x_test.reshape(x_test.shape[0], 28, 28, 1)
input_shape = (28, 28, 1)

# convertir classe vecteur en classe matrice binaire
y_train = keras.utils.to_categorical(y_train, 10)
y_test = keras.utils.to_categorical(y_test, 10)
```

Pour pouvoir utiliser l'ensemble de données dans l'API Keras, nous avons besoin de tableaux à 4 dimensions. Cependant, comme nous le voyons dans la figure précédente, notre tableau est de 3 dimensions.

III.5.4 Initialisation des hyper paramètres :

```
#initialisation
num_classes = 10
epochs = 10
```

Cette instruction permet de définir le nombre de classe (0 à 9) ainsi que le nombre des itérations.

III.5.5 Construction du modèle CNN :

Nous allons procéder à construire notre réseau CNN par l'ensemble des instructions illustrés ci-dessous :

```
# construction de notre architecture CNN
model = Sequential()
model.add(Conv2D(32, kernel_size=(3, 3),activation='relu',input_shape=input_shape))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(64, kernel_size=(3, 3), activation='relu',input_shape=input_shape))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(num_classes, activation='softmax'))
```

Maintenant, on va visualiser notre modèle en utilisant l'instruction suivante :

```
#visualisation de notre model
model.summary()
```

Et on obtient :

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 26, 26, 32)        320
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 13, 13, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 11, 11, 64)        18496
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 5, 5, 64)          0
_________________________________________________________________
dropout_1 (Dropout)          (None, 5, 5, 64)          0
_________________________________________________________________
flatten_1 (Flatten)          (None, 1600)              0
_________________________________________________________________
dense_1 (Dense)              (None, 128)               204928
_________________________________________________________________
dropout_2 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_2 (Dense)              (None, 10)                1290
=================================================================
Total params: 225,034
```

La figure ci-dessus nous présente les différentes couches de notre architecture de CNN ainsi que leurs paramètres, le calcul des paramètres s'effectue comme suit :

- Pour les couches de convolutions : le nombre de paramètres est égal à la taille du filtre multiplié par le nombre de filtres plus le biais.
- Pour les couches dense : le nombre paramètres est égal aux entrés multiplié par le nombre de neurones dans la cette couche plus les biais.

- la première couche : Conv2d_1, contient 320 paramètres qui sont calculés comme suit : (3*3*32) +32=320.
- la couche max_pooling2d_1 : elle ne retourne pas de paramètres.
- La couche conv2d_2 : elle contient 18496 paramètres qui sont calculés comme suit (3*3*64*32) +64=18496.
- la couche max_pooling2d_2 : elle ne retourne pas de paramètres.
- Les deux couches dropout_1 et flatten_1 ne retournent pas de paramètres.

- La couche dense_1 : contient 204928 paramètres qui sont calculés comme suit : (128*1600) + 128=204928.
- Enfin la couche dense_2 : contient 1290 paramètres qui sont calculés comme suit : (10*128) +10

Finalement, nous avons un total de paramètres qui vaut 225034 qui est la somme de tous les paramètres calculés auparavant.

III.5.6 Compilation et ajustement du modèle :

Avec le code ci-dessus, nous avons créé un CNN vide non optimisé. Il est maintenant temps de définir un optimiseur « Adam» et une fonction de perte « Loss». Nous allons ajuster le modéle avec l'instruction suivante :

```
# définir un optimiseur
model.compile(loss=keras.losses.categorical_crossentropy,optimizer=keras.optimizers.Adadelta(),metrics=['accuracy'])

# ajuster le model
model.fit(x_train, y_train,batch_size=batch_size,epochs=epochs,verbose=1,validation_data=(x_test, y_test))
```

III.5.7 Évaluation du modèle :

Maintenant, nous allons évaluer le modèle entraîné avec x_test et y_test en utilisant l'instruction suivante :

```
#Évaluation du modèle
score = model.evaluate(x_test, y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

Après l'exécution on obtient :

```
Train on 60000 samples, validate on 10000 samples
Epoch 1/10
60000/60000 [==============================] - 61s 1ms/step - loss: 0.3195 - accuracy: 0.8992 - val_loss: 0.0634 -
val_accuracy: 0.9795
Epoch 2/10
60000/60000 [==============================] - 58s 962us/step - loss: 0.1081 - accuracy: 0.9677 - val_loss: 0.0421 -
val_accuracy: 0.9862
Epoch 3/10
60000/60000 [==============================] - 55s 920us/step - loss: 0.0793 - accuracy: 0.9759 - val_loss: 0.0338 -
val_accuracy: 0.9876
Epoch 4/10
60000/60000 [==============================] - 54s 907us/step - loss: 0.0657 - accuracy: 0.9805 - val_loss: 0.0311 -
val_accuracy: 0.9890
Epoch 5/10
60000/60000 [==============================] - 54s 907us/step - loss: 0.0572 - accuracy: 0.9826 - val_loss: 0.0314 -
val_accuracy: 0.9888
Epoch 6/10
60000/60000 [==============================] - 58s 974us/step - loss: 0.0510 - accuracy: 0.9851 - val_loss: 0.0260 -
val_accuracy: 0.9908
Epoch 7/10
60000/60000 [==============================] - 57s 955us/step - loss: 0.0464 - accuracy: 0.9859 - val_loss: 0.0269 -
val_accuracy: 0.9904
Epoch 8/10
60000/60000 [==============================] - 57s 946us/step - loss: 0.0429 - accuracy: 0.9872 - val_loss: 0.0267 -
val_accuracy: 0.9907
Epoch 9/10
60000/60000 [==============================] - 57s 948us/step - loss: 0.0409 - accuracy: 0.9879 - val_loss: 0.0251 -
val_accuracy: 0.9916
Epoch 10/10
60000/60000 [==============================] - 59s 975us/step - loss: 0.0375 - accuracy: 0.9887 - val_loss: 0.0253 -
val_accuracy: 0.9910
```

```
The model has successfully trained
Test loss: 0.02534664944172618 6
Test accuracy: 0.9909999966621399
```

Notre modèle a été entrainé avec succès, voir qu'il a un taux de précision qui vaut 99,09% et un taux d'erreur qui vaut 2,53%.

III.5.8 sauvegarder le modèle :

Afin de sauvegarder notre modèle, on fait appel à l'instruction suivante :

```
#sauvegarder le model
model.save('mnist.h5')
print("Saving the model as mnist.h5")
```

Maintenant on va prendre un index d'image au hasard de la base MNIST pour faire un test en utilisant les instructions suivantes :

```
image_index = 800
plt.imshow(x_test[image_index].reshape(28, 28),cmap='Greys')
pred = model.predict(x_test[image_index].reshape(1, 28, 28, 1))
print('le chiffre reconnu est:   ' ,pred.argmax())
```

Le résultat donné est :

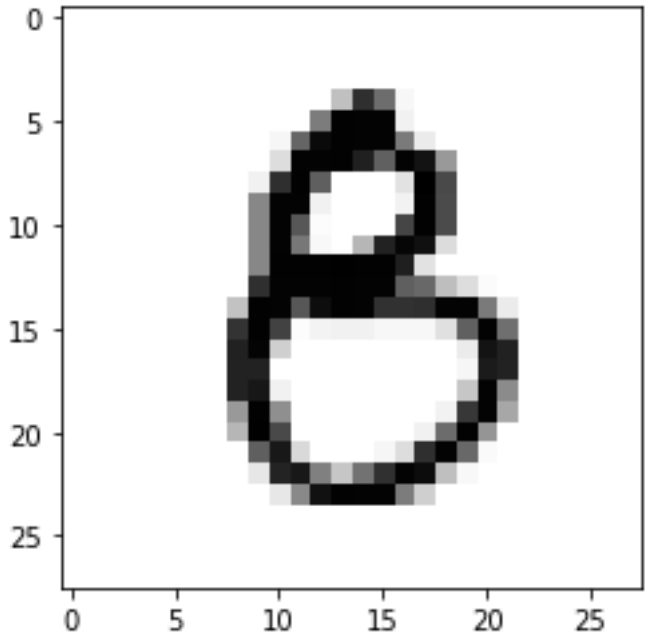

Figure III.4 Représente l'image ayant l'index 800 de la base MNIST

```
le chiffre reconnu est:   8
```

Le programme arrive à reconnaitre le chiffre huit '8'

On va essayer avec un autre index pour voir si le programme arrive toujours à avoir un bon résultat :

```
image_index = 4738
plt.imshow(x_test[image_index].reshape(28, 28),cmap='Greys')
pred = model.predict(x_test[image_index].reshape(1, 28, 28, 1))
print('le chiffre reconnu est:   ' ,pred.argmax())
```

Le résultat donné est :

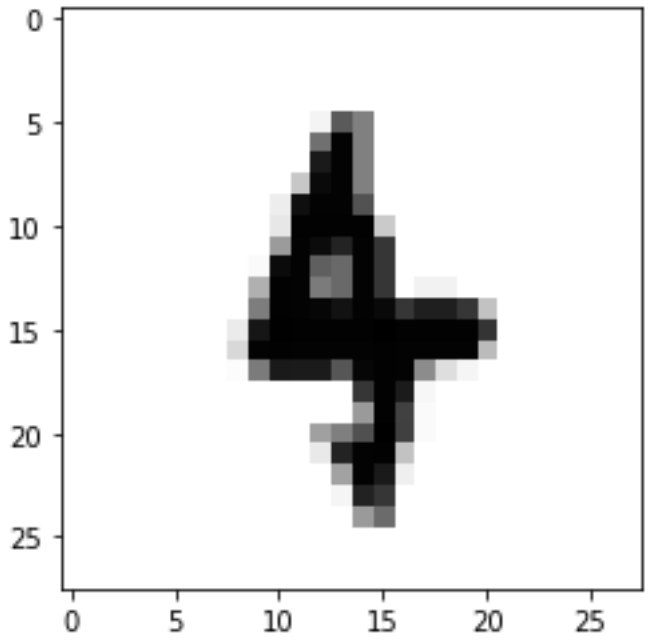

Figure III.5 Représente l'image ayant l'index 4738 de la base MNIST

```
le chiffre reconnu est:   4
```

Le programme arrive à reconnaitre le chiffre 4 malgré la médiocrité de l'écriture.

III.6 Résultats obtenus et discussion :

On illustre dans ce qui suit les résultats en termes de précision et d'erreur.

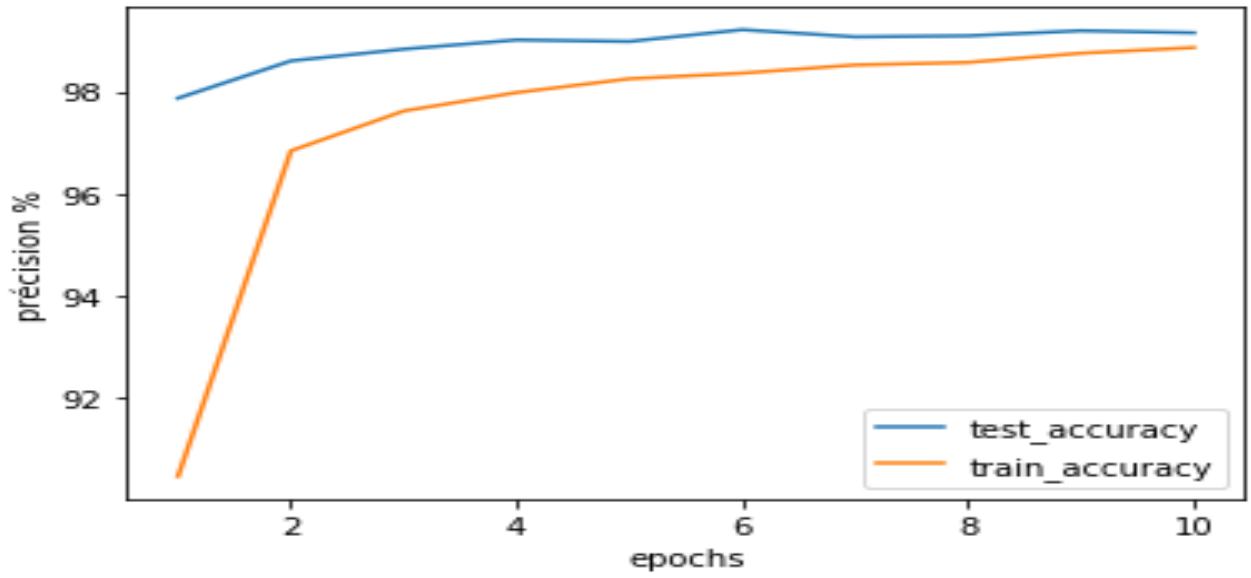

Figure III.6 Représentation de la précision en fonction des epochs

D'après **la Figure III.6** La précision de l'apprentissage et de test augmente avec le nombre d'epochs, ceci reflète qu'à chaque époch le modèle apprend plus d'informations.

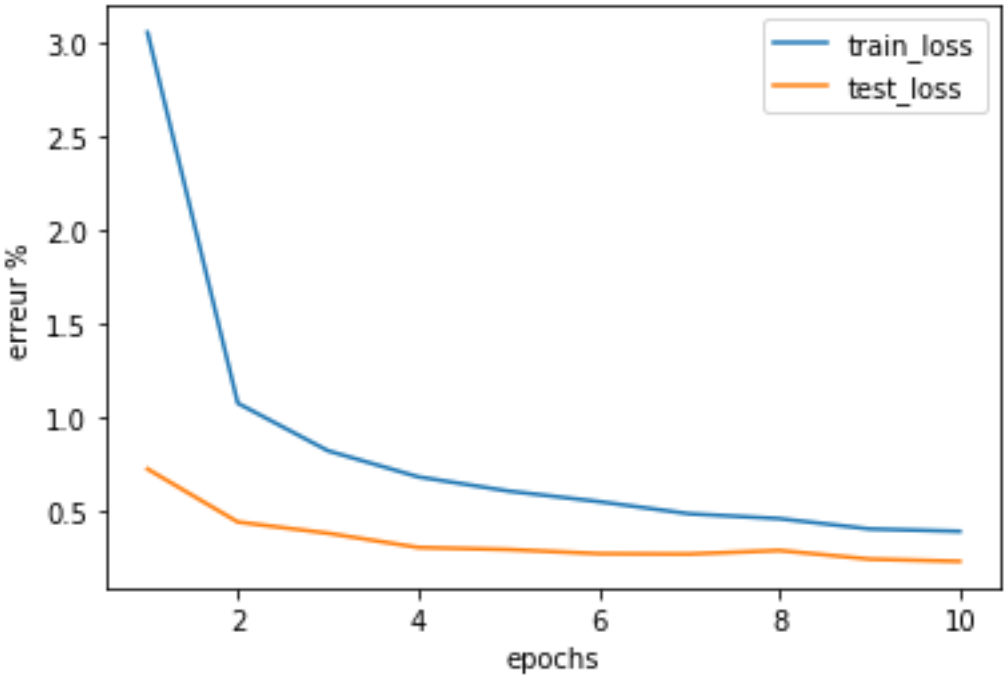

Figure III.7 Représentation de l'erreur en fonction des epochs

Dans **la figure III.7**, l'erreur d'apprentissage et de la validation diminue avec l'augmentation du nombre d'époque.

Nous remarquons aussi, un taux d'erreur de 2.53%, ce qui signifie qu'il y'a 253 images sur 10000 mal classées.

Conclusion :

Dans ce chapitre, nous avons élaboré une implémentation pour la reconnaissance des chiffres manuscrits en utilisant une approche de classification basée sur le réseau de neurone convolutionnel, nous avons détaillé les caractéristiques de ce dernier ainsi que toutes les techniques utilisées, La comparaison des résultats trouvés a montré que le nombre d'epochs, la taille de la base et la profondeur de réseaux, sont des facteurs importants pour l'obtention de meilleurs résultats.

Conclusion générale

Au cours de ce mémoire, nous avons rappelé les différentes techniques de reconnaissance de l'écriture manuscrite. Nous avons pu constater que la reconnaissance d'écriture a connu ces dernières années des progrès très importants, permettant désormais de faire face à la variabilité de l'écriture, et ainsi de reconnaître de manière satisfaisante des entités manuscrites : mots et chiffres. Dans ce travail, nous nous sommes focalisé sur la reconnaissance des chiffres manuscrits par les réseaux de neurones convolutifs.

Nous avons discuté des notions fondamentales sur le Deep Learning et les réseaux de neurones en générale et les 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: la couche convolutionnelle, la couche de rectification, la couche de pooling et la couche fully connected. Nous avons parlé aussi sur les méthodes de régularisation (dropout) utilisées pour éviter le problème de sur apprentissage.

Ainsi, après avoir présenté les différents modules constituant notre réseau convolutionnel, nous avons détaillé chaque module ainsi que ses avantages et son utilité, pour enfin passer à l'implémentation, dans cette partie, nous avons introduit la base donnée MNIST et le langage de programmation PYTHON utilisé ainsi que ses librairies, et nous avons aussi expliqué chaque instruction de notre code ainsi que ses paramètres.

Enfin, nous avons élaboré quelques tests sur des images de la base donnée avec un index au hasard, et ainsi nous avons établi des relations entre l'erreur et la précision via des graphes, et pour cela nous avons obtenu de très bons résultats avec une précision de 99.09% avec une erreur qui est presque nulle.

Bibliographie :

[01] Nedjem. Eddine. Ayat, sélection de modèle automatique des machines à vecteurs de suports: application à la reconnaissance d'images de chiffres manuscrits, thèse doctorat en génie. P. H. D. Université du Québec, Montréal, Canada, 2004.

[02] Matougui. Farida, système interactif pour la reconnaissance des Caractères arabes Manuscrits, SIRCAM, Thèse de Magistère, C, D, T, A, 1990.

[03] Fahmy, S.Al Ali Automatic recognition of handwritten Arabic characters using their geometrical features Studies in informatics and control journal (SIC journal), vol. 10, No 2, 2001

[04] N. Ben Amara, Utilisation des modèles de Markov caches planaires en reconnaissance de l'´écriture arabe imprimée Thèse de doctorat, Tunis, 1999

[05] B. Al-Badr, S.A. Mahmoud Survey and bibliography of Arabic optical text recognition Signal processing, vol. 41, Elsevier, 1995.

[06] L. M. Lorigo, V. Govindaraju, Offline Arabic Handwriting Recognition:A Survey,ieee transactions on pattern analysis and machine intelligence, vol. 28, no. 5, may 2006

[07] S. Madhvanath , V. Govindaraju "'The Role of Holistic Paradigms in Handwritten Word Recognition"',ieee transactions on pattern analysis and machine intelligence, vol. 23, no. 2, february 2001

[08] A. Belaid, Y. Belaid, reconnaissance des formes: méthodes et applications, interedition, paris, 1992.

[09] Jean-Pierre « reconnaissance de l'écriture manuscrite », Département Images, ENSTParis et Guy LORETTE RISA, CNRS UPRES-A 6074, Université de Rennes 1.

[10] D.Abdelhakim, 2011, «La reconnaissance des chiffres manuscrits par les machines à vecteurs de support(SVMs)»; Thèse de Master, Université de Tébessa

[11] G.Abdeldjalil, 2011, «Segmentation automatique pour la reconnaissance numérique des chèques bancaires Algériens » ; Thèse de MAGISTER, centre Universitaire de Khanchela

[12] F. Menasri, «Contributions à la reconnaissance de l'´ecriture arabe manuscrite»,Thèse de doctorat, universite´ paris des cartes, juin 2008.

[13] Frédéric Grandidier doctorat en génie ph.d. « un nouvel algorithme de sélection de caractéristiques – application à la lecture automatique de l'écriture manuscrite »
montréal, le 24 janvier 2003

[14] N. Benahmed, 2002. « Optimisation des réseaux de neurones pour la reconnaissance de chiffres manuscrits isolés : Sélection et pondération des primitives par algorithmes génétiques» Thèse de Master, École de technologie supérieure, MONTRÉAL.

[15] A. El-Yacoubi, R. Sabourin, M. Gilloux, and C.Y. Suen. Improved model architecture and training phase in an off-line HMM-based word recognition system. In Proc. of the 13th International Conference on Pattern Recognition, Brisbane, Australia, 1998

[16] P.M Lallican, C. Viarp-Gaudin and S. Knerr,. « From off-line to on-line handwriting recognition ». Proc. 7th workshop on frontiers in handwriting recognition, Amsterdam, 2000

[17] IEEE Transaction pn pattern analysis and machine intelligence, VOL 24, NO. 2, february 2002

[18] Ernest Kussul, Tatyana Baidyk : Improved Method of Handwritten Digit

[19] Ming Wu Zhen Zhang, Handwritten Digit Classification using the MNIST Data Set, ResearchGate publications, 2010.

[20] International Journal of Computer Applications (0975 – 8887) Volume 104 – No.9, October 2014

[21] Yukihiko Yamashita ; Toru Wakahara; k-NN Classification of Handwritten Characters Using a New Distortion-Tolerant Matching Measure, Pattern Recognition (ICPR), 2014, 2nd International Conference on; Stockholm.

[22] Lee et al, Generalizing Pooling Functions in Convolutional Neural Networks: Mixed, Gated, and Tree, NSF awards, 2015.

[23] Yukihiko Yamashita, Toru Wakahara: Affine-transformation and 2D-projection invariant k-NN classification of handwritten characters via a new matching measure

[24] Dundar et al, (2016).

[30] Y. Bengio, A. Courville, and P. Vincent, “Representation learning: A review and new perspectives,” IEEE transactions on pattern analysis and machine intelligence, vol. 35, no. 8, 2013.

[31] R. Dechter and J. Pearl, The cycle-cutset method for improving search performance in AI applications. University of California, Computer Science Department, 1986.

[32] H.Wang, B. Raj, and E. P. Xing, “On the origin of deep learning,” arXiv preprint arXiv: 1702.07800,2017.

[33] https://fr.mathworks.com/discovery/deep-learning.html

[34]https://www.usine-digitale.fr/article/le-deep-learning-ouvre-a-nvidia-le-marche-des-voitures-autonomes-des-drones-des-robots.N389240

[35] https://fr.mathworks.com/solutions/aerospace-defense.html

[36] https://medium.com/epidemium/nvidia-ia-et-deep-learning-en-santé- 28024c95e1c4
[37] https://www.cognex.com/fr-fr/what-is/deep-learning/for-factory-automation
[38] P. J.Werbos, "Generalization of backpropagation with application to a recurrent gas market model," Neural networks, vol. 1, no. 4, 1988.
[39] Yunbei ZHANG, «Adaptation des systèmes de traduction automatique neuronale aux domaines spécialisés», master, Institut National des Langues et Civilisations Orientales Département Textes, Informatique, Multilinguisme, 2018.
[40] M. Gregory Gelly, «Réseaux de neurones récurrents pour le traitement automatique de la parole», Thèse de doctorat de l'Université Paris-Saclay, Septembre 2017.
[41] Moualek Djaloul Youcef, «Deep Learning pour la classification des images», Master, Université Abou Bakr Belkaid Tlemcen Faculté des Sciences Département d'informatique, juillet 2017.
[42] A. Van den Oord, S. Dieleman, and B. Schrauwen, "Deep content-based music recommendation," in Advances in neural information processing systems, 2013.
[43] R. Collobert and J. Weston, "A unified architecture for natural language processing: Deep neural networks with multitask learning," in Proceedings of the 25th international conference on Machine learning, ACM, 2008.
[44] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," Proceedings of the IEEE, vol. 86, no 11, 1998.
[45] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in Advances in neural information processing systems, 2012.
[46] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in European conference on computer vision, Springer, 2014.
[47] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015.
[48] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. Alemi, "Inception-v4, inception-resnet and the impact of residual connections on learning," arXiv preprint arXiv :1602.07261, 2016.
[49] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.
[50] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, « Gradient-based learning applied to document recognition » ,Proceedings of the IEEE, vol. 86, no. 11, 1998.
[51] https://datasciencetoday.net/index.php/en-us/deep-learning/173-les-reseaux-de-neurones-convolutifs

[52] https://penseeartificielle.fr/focus-reseau-neurones-convolutifs/

[53] D.P. Kingma and J.L. Ba. ADAM: A method for stochastic optimization. ICLR 2015. juillet 2017.

[55] https://docs.python.org/3/tutorial/index.html

[56] https://www.tensorflow.org/

[57] https://keras.io/

[58] http://yann.lecun.com/exdb/mnist/

Annexe

Code source :

```
#  importation des biblio nécessaires
import keras
import tensorflow as tf
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten
from keras.layers import Conv2D, MaxPooling2D
import matplotlib.pyplot as plt
# importer notre base de donnée MNIST
(x_train, y_train), (x_test, y_test) = tf.keras.datasets.mnist.load_data()
# afficher le model d'entrainement
print(x_train.shape, y_train.shape)
# remodeler le vecteur a 4 dim pour fonctionner avec l'interface de keras
x_train = x_train.reshape(x_train.shape[0], 28, 28, 1)
x_test = x_test.reshape(x_test.shape[0], 28, 28, 1)
input_shape = (28, 28, 1)
# convertir class vecteur en  class matrices binaire
y_train = keras.utils.to_categorical(y_train, 10)
y_test = keras.utils.to_categorical(y_test, 10)
#  S'assurer que les valeurs sont de type float afin que nous puissions obtenir des points décimaux après la division
x_train = x_train.astype('float32')
x_test = x_test.astype('float32')
#Normaliser les codes RVB en les divisant par la valeur RVB maximale.
```

```
x_train /= 255

x_test /= 255

print('x_train shape:', x_train.shape)

print(x_train.shape[0], 'train samples')

print(x_test.shape[0], 'test samples')

#initialisation

num_classes = 10

epochs =10

# Construction de notre architecture CNN

model = Sequential()

model.add(Conv2D(32, kernel_size=(3, 3),activation='relu',input_shape=input_shape))

model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(64, kernel_size=(3, 3), activation='relu',input_shape=input_shape))

model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Dropout(0.25))

model.add(Flatten())

model.add(Dense(128, activation='relu'))

model.add(Dropout(0.5))

model.add(Dense(num_classes, activation='softmax'))

#visualisation de notre model

model.summary()

# définir un optimiseur

model.compile(loss=keras.losses.categorical_crossentropy,optimizer=keras.optimizers.Adadelta(),metrics
=['accuracy'])

# ajuster le model

model.fit(x_train, y_train,batch_size=128,epochs=epochs,verbose=1,validation_data=(x_test, y_test))
```

```
print("The model has successfully trained")

#Évaluation du modèle

score = model.evaluate(x_test, y_test, verbose=0)

print('Test loss:', score[0])

print('Test accuracy:', score[1])

print('the accuracy on the test set is :',score[1]*100,'%')

#sauvegarder le model

model.save('mnist.h5')

print("Saving the model as mnist.h5")

#Partie test

image_index = 4738

plt.imshow(x_test[image_index].reshape(28, 28),cmap='Greys')

pred = model.predict(x_test[image_index].reshape(1, 28, 28, 1))

print('le chiffre reconnu est:  ' ,pred.argmax())
```

Résumé

Notre travail consiste à faire la reconnaissance de l'écriture manuscrite, en particuliers les chiffres manuscrits, et pour cela nous avons procédé à l'utilisation des réseaux de neurones convolutionnels, cette méthode réduit l'empreinte mémoire, améliore les performances et sa profondeur a une grande influence pour avoir des meilleurs résultats.

Abstract

Our work consists of recognizing handwriting, in particular handwritten digits, and for this we have proceeded the use of convolutional neural networks, this method reduces the memory footprint, improves performance and its depth has a great influence to have better results.