République Algérienne Démocratique et Populaire

Ministère de l'enseignement supérieur et de la recherche scientifique

Université Larbi Tébessi – Tébessa

Faculté des Sciences Exactes et des Sciences de la Nature et de la Vie

Département : Mathématiques et Informatique

Mémoire de fin d'étude

Pour l'obtention du diplôme de MASTER

Domaine : Mathématiques et Informatique

Filière : Informatique

Option : Systèmes d'information

Thème

La reconnaissance des chiffres manuscrits isolés en utilisant l'apprentissage profond

Présenté Par :

Guedri marouane

Devant le jury :

Mr. M. Amroun	*MCA*	*Univérsité Larbi Tébessa*	*Président*
Mr. I. Bendib	*MCB*	*Univérsité Larbi Tébessa*	*Examinateur*
Mr. A. Gattal	*MCA*	*Univérsité Larbi Tébessa*	*Encadreur*

Date de soutenance : 22/06/2019

Résumé

La reconnaissance des chiffres manuscrits est le premier problème de recherche de la communauté de l'analyse et de la reconnaissance de documents depuis plus de trois décennies.

Les sous-problèmes de reconnaissance des chiffres manuscrits comprennent principalement la segmentation, la reconnaissance des caractères isolés et la reconnaissance des chaînes des chiffres et des chiffres isolés.

Parmi ces différentes modalités, nous nous intéresserons à la reconnaissance des chiffres manuscrits isolés en utilisant l'apprentissage profond pour connaître le contenu d'une image de la base de données non normalisé CVL basé sur le modèle de « Convolutional Neural Networks-CNNs ».

Mots clés : chiffres manuscrits isolés, binarisation, rembourrage (Padding), apprentissage profond, réseau de neurones convolution (CNNs).

Abstract

Handwriting recognition has been the premier research problem of the document analysis and recognition community for over three decades now.

The sub problems in handwriting recognition mainly include line, recognition of isolated characters, and recognition of numerical strings and isolated digits.

Among these different modalities of handwriting recognition, this work focuses on recognition of isolated handwritten digits using deep learning to know the content of the non-normalised CVL database image based on the « Convolutional Neural Networks-CNNs » model.

Key words: isolated handwritten digits, Binarization, Padding, Deep learning, Convolutional neural network (CNNs).

ملخص

يعد التعرف على الأرقام المكتوبة بخط اليد المشكلة البحثية الأولى في مجتمع تحليل المستندات والاعتراف بها لأكثر من ثلاثة عقود.

تتضمن المشاكل الفرعية في التعرف على الأرقام المكتوبة بخط اليد بشكل أساسي تجزئة مستوى الخط والتعرف على الأحرف المعزول والتعرف على السلاسل الرقمية والأرقام المعزولة.

من بين هذه الطرائق المختلفة، سوف نركز على التعرف على الأرقام المعزولة باستخدام التعلم العميق لمعرفة محتوى صورة قاعدة بيانات CVL غير موحدة الحجم بناءً على نموذج الشبكات العصبية « Convolution Neural Networks- CNNs».

الكلمات المفتاحية: الأرقام المعزولة المكتوبة بخط اليد، الترميز الثنائي، الحشو، التعلم العميق، الشبكات العصبية، الشبكة العصبية التلافيفية (CNNs).

Dédicaces

*Avant tout, Je remercie **ALLAH** qui m'aide*

*Je dédie ce modeste travail à mes très chers parents **TAHER** et **DRIFA** qui n'ont pas cessé de m'encourager durant toutes mes études et que dieu me les garde ;*

*À mes frères, mes sœurs et ses enfants **SALAH, WALID, ABD-ELKARIM, SAIDA, NADIA** et **ZAHRA ;***

*Je le dédie à notre regretté, mon grand chagrin de ne pas le voir présent, mon grand frère **MOHI ELDIN ;***

*Mes grand-mères **KHAIRA** et **TEBRA**, que dieu les bénisse et mes regretté grands pères ;*

À mes cousins et cousines et ses enfants et tous ceux que j'aime ;

*À toutes les **personnes** qui m'ont aidés à la réalisation de mon travail.*

*À tous les étudiants de branche **informatique** 2018-2019*

Guedri marouane

Remerciements

Nous exprimons notre profonde gratitude à toute personne qui, de près ou de loin, a contribué à la réalisation de ce travail.

Nos remerciements s'adressent plus particulièrement à Mr Gattal Abdeljalil qui a bien accepté de diriger ce mémoire. Ses remarques, ses précieux conseils et ses corrections nous ont été d'une grande utilité. Nous lui disons « Merci ».

Nous remercions également les jurys d'avoir accepté l'évaluation de ce travail.

Nous aimerions témoigner notre gratitude aux professeurs et enseignants de la Faculté des Mathématiques et Informatiques qui ont assuré notre formation universitaire pour les conseils prodigués depuis la première année.

Nos remerciements s'adressent également à tous ceux qui nous ont soutenu tant moralement que matériellement.

Que les camarades de classe trouvent eux aussi nos remerciements les plus distingués pour leur assistance et fraternité au cours de notre formation universitaire.

Les derniers mots sont consacrés au Dieu le tout persistant de

Nos avoir donné la puissance d'en arriver là.

Table des matières

Chapitre 01

I. Introduction générale 10
1 Introduction 12
2 Le système de reconnaissance des chiffres manuscrits 12
2.1.1 Phase de prétraitement 12
2.1.1.1 Éliminer le bruit 12
2.1.1.2 Binarisation 13
2.1.1.3 Squelettisation 13
2.1.1.4 Normalisation de la taille 14
2.1.1.5 Zonage 14
2.1.2 Phase de segmentation 15
2.1.2.1 Segmentation explicite 15
2.1.2.2 Segmentation implicite 15
2.1.3 Phase d'extraction des caractéristiques 16
2.1.3.1 caractéristiques structurelles 16
2.1.3.2 caractéristiques texturale 16
2.1.3.3 caractéristiques globale 17
2.1.3.4 caractéristiques morphologique 17
2.1.4 Phase de classification 17
2.1.4.1 l'apprentissage 17
2.1.4.1.1 Apprentissage supervisé « Classification » 17
2.1.4.1.2 Apprentissage non supervisé 18
2.1.4.1.3 Apprentissage semi-supervisé 18
2.1.4.2 la reconnaissance et décision 18
2.1.4.2.1 Approche statistique 19
2.1.4.2.2 Approche structurelle ou syntaxique 19
2.1.4.2.3 Approche stochastique 19
2.1.4.2.4 Approche hybride 20
3 Conclusion 20

Chapitre 02

1 Introduction 21

2 Système de reconnaissance basé sur l'apprentissage profond « Deep Learning »....21
2.1 Réseaux de neurones artificiels....21
2.1.1 C'est quoi un perceptron ?....21
2.1.1.1 perceptron linéaire à seuil....21
2.1.1.2 Perceptron avec entrée supplémentaire....22
2.1.1.3 Perceptrons qui calculent le OÙ....22
2.1.1.4 Fonction d'activations....22
2.1.2 perceptrons multicouche....23
2.1.3 Structure d'interconnexion....24
2.1.3.1 Réseau à connexions locales :....24
2.1.3.2 Réseau à connexions récurrentes :....24
2.1.3.3 Réseau à connexion complète :....25
3 Réseaux de neurone profond....25
4.1 Réseau de neurones convolutifs « CNNs »....26
4.1.1 Couche de convolution « CONV » :....26
4.1.2 Couches de regroupement (Pooling)....27
4.1.3 Couches non linéaires « Non-linear layers »....28
4.1.5 Couches entièrement connectées « *Fully connected layers (FC)* »....29
4.2 Boltzmann Machines « *BMs* »....29
4.2.1 Apprentissage dans les machines Boltzmann....30
4.2.2 La vitesse d'apprentissage....30
4.3 Réseau Hopfield....30
4.3.1 Réseaux asynchrones....30
4.3.2 Mise à jour et paramètres....30
4.5 Longue mémoire à court terme (LMCT)....34
4.6 réseaux de croyance profond....35
5 Conclusion....36

Chapitre 03

3.1 Introduction....37
3.2 Description de différents travaux ultérieurs....37
3.2.1 Reconnaissance des chiffres manuscrits isolés :....37
3.2.1.1 F. Menasri et N. Vincent. (2008) [50] :....37
3.2.1.2 N. V. Rao et al. (2011) [51] :....37

3.2.1.3 A. Gattal et al. (2014) [52] 38
3.2.1.4 L. B. Saldanha et C. Bobda. (2015) [53] 38
3.2.1.5 A. Gattal et al. (2016) [54] 39
3.2.1.6 H. Zhan et al. (2017) [55] 39
3.2.1.7 S. R. Kulkarni et B. Rajendran. (2018) [56] 40
3.2.1.8 J. Qiao et al (2018) [57] 40
3.2.1.9 S. M. Shamim et al. (2018) [58] : 41
3.2.2 Reconnaissance une chaine de chiffres manuscrits : 41
3.2.2.1 A. Gattal et al. (2017) [59] 41
3.2.2.2 G. Andre et al. (2018) [78] : 42
3.2.2.3 A. G. Hochuli et al. (2018) [79] : 42
3.2.3 Compétition HDR : 43
3.2.3.1 Markus Diem et al. (2013) [80] 43
3.2.3.1.1 Système de François Rabelais : 43
3.2.3.1.2 Système de Hannover : 43
3.2.3.1.3 Système de Jadavpur : 43
3.2.3.1.5 Système de Orand : 44
3.2.3.1.6 Système de Salzburg I : 44
3.2.3.1.7 Système de Salzburg II : 44
3.2.3.1.8 Système de Tébessa I : 44
3.2.3.1.9 Système de Tébessa II : 44
3.2.3.2 Résultats 44
3.3 Comparaison entre les travaux ultérieurs 45
3.4 Conclusion 46

Chapitre 04

4.1 Introduction 47
4.2 Outils de développement 47
4.2.1 Matériel 47
4.2.2 Environnement de développement 47
4.2.2.1 Matlab 47
4.2.2.2 Python 48
4.2.2.3 Google colab (Colaboratory) 48
4.2.3 La bases de données utilisées 49

4.3 Expérimentation 50
4.3.1 Protocole d'expérimentation 1 51
4.3.1.1 les résultats expérimentaux 51
4.3.1.2 Discussion 53
4.3.2 Protocole d'expérimentation 2 53
4.3.2.1 les résultats expérimentaux 53
4.3.2.2 Discussion 55
4.3.3 Protocole d'expérimentation 3 55
4.3.3.1 les résultats expérimentaux avec le protocole 1 56
4.3.3.2 les résultats expérimentaux avec le protocole 2 57
4.3.3.3 Discussion 58
4.4 Résultats & Discussion 58
4.5 Comparaison des résultats : 59
4.6 Conclusion 59
II. Conclusion générale 60
III. Références Bibliographiques 61
IV. Annexes 68
IV.1 Code sources : 68

Liste des tableaux

Chapitre 03

Table 3.1 : La performance du logiciel 39
Table 3.2 : Taux de reconnaissance de différents modèles sur les ensembles de données. 40
Table 3.3 : Taux de reconnaissance obtenus. 41
Table 3.4 : Réparation des données utilisées pour l'apprentissage et le test de classifieur. Les échantillons sont uniformément repartis entre les classes. 42
Table 3.5 : Performance des approches sans segmentation sur les données synthétiques 42
Table 3.6 : Taux de reconnaissance de diffèrent méthodes de la compétition 45
Table 3.7 : Classifieurs utilisés dans les systèmes de reconnaissance des chiffres manuscrits. 46

Chapitre 04

Table 4. 1 : Répartition des chiffres dans la base de données CVL. 50
Table 4. 2 : Résultats obtenus dans le protocole un. 53
Table 4. 3 : Les résultats obtenus dans le protocole deux. 55
Table 4. 4 : Résultats obtenus avec le protocole un. 57
Table 4. 5 : Résultats obtenus avec le protocole deux. 58
Table 4. 6 : Les résultats expérimentaux 59
Table 4. 7 : Comparaison de notre travaille aux travaux existants dans la littérature . 59

Liste des figures

Chapitre 01

Figure 1.1 : Exemple de Suppression du bruit. 12
Figure 1.2 : Exemple de la binarisation. 13
Figure 1.3 : Exemple de squelettisation. 14
Figure 1.4 : Exemple d'un chiffre manuscrit normalisé 14
Figure 1.5 : Exemple d'un chiffre zoné de taille (3 * 3). 14
Figure 1.6 : Exemple de segmentions des chiffres manuscrits (explicite). 15
Figure 1.7 : Exemple de segmentions des chiffres manuscrits « implicite » [78] 16
Figure 1.8 : Exemple d'hybridation entre deux approches. 20

Chapitre 02

Figure 2. 1 : Le perceptron avec seuil 21
Figure 2. 2 : Le perceptron avec entrée supplémentaire 22
Figure 2. 3 : perceptrons qui calculent le OÙ 22
Figure 2. 4 : Example d'un perceptron multicouche 23
Figure 2. 5 : Définition des couches d'un réseau multicouche 24
Figure 2. 6 : Réseau à connexions locales 24
Figure 2. 7 : Réseau à connexions récurrentes 25
Figure 2. 8 : Réseau à connexions complète 25
Figure 2. 9 : Schéma fonctionnel typique d'un CNNs [47] 26
Figure 2. 10 : Représentation de convolution sur un image de taille « 4*4 » pixels et un convolution filtre 27
Figure 2. 11 : Représentation graphique de « max pooling » Et « average pooling » [45] 28
Figure 2. 12 : Représentation graphique de la fonctionnalité ReLU [45] 28
Figure 2. 13 : Exemple de machines Boltzmann 29
Figure 2. 14 : Un réseau Hopfield de trois unités [61] [62] 31
Figure 2. 15 : Représentation compacte des RNN. Toutes les flèches représentent des connexions complètes. La flèche en pointillée représente les connexions ayant un décalage temporel « t – 1 » 32
Figure 2. 16 : Représentation dépliée des RNNs 32
Figure 2. 17 : Représentation dépliée hiérarchisée des RNNs 33
Figure 2. 18 : Architecture du modèle de mémoire à court terme et dépliée 34
Figure 2. 19 : Réseau de croyances profondes composé de plusieurs couches de RBM 35

Chapitre 04

Figure 4. 1 : L'interface graphique de l'environnement MATLAB (version R2018a) 48
Figure 4. 2 : L'environnement de travail de google colab 49
Figure 4. 3 : Exemples des chiffres de la base de données CVL. 50
Figure 4. 4 : L'architecture de (CNN1) proposée pour la classification des chiffres manuscrits. 51
Figure 4. 5 : L'architecture de « *CNN2* » proposée pour la classification des chiffres manuscrits. 53
Figure 4. 6 : Exemple de normalisation la taille avec la technique « Padding » 56

Introduction Générale

Introduction générale

I. Introduction générale

La reconnaissance des chiffres manuscrits est le premier problème de recherche de la communauté de l'analyse et de la reconnaissance de documents depuis plus de trois décennies. Les sous-problèmes de reconnaissance des chiffres manuscrits comprennent principalement la segmentation au niveau des lignes, des mots ou des caractères, la reconnaissance des caractères isolés, des mots ou des lignes / paragraphes complets et la reconnaissance des chaînes des chiffres et des chiffres isolés. Parmi ces différentes modalités de reconnaissance des chiffres manuscrits, ce projet de fin d'étude est consacré à la reconnaissance des chiffres isolés, le problème classique de reconnaissance des formes qui offrant des nombreuses d'applications. Contrairement à l'alphabet, les dix glyphes des chiffres arabes occidentaux les plus couramment utilisés sont partagés par de nombreux scripts et langues à travers le monde, ce qui les rend universellement acceptables. Les principaux défis de la reconnaissance des chiffres manuscrits proviennent des variations des taille, des forme, d'inclinaison et plus important encore des variations dans les styles d'écriture des individus.

Avec les récents progrès dans l'analyse des images et la classification des formes, des systèmes sophistiqués de reconnaissance des chiffres ont été proposés, qui visent à améliorer les performances de reconnaissance globales en améliorant la génération de caractéristiques et / ou les techniques de classification utilisées. Certaines des études visent à améliorer les performances de classification en utilisant une combinaison de plusieurs classificateurs, tandis que d'autres visent à combiner plusieurs caractéristiques et à sélectionner l'ensemble des caractéristiques le plus pertinent et optimal pour ce problème.

L'apprentissage profond est une partie des algorithmes d'apprentissage automatisés et a donc été classé dans une section plus large de l'intelligence artificielle (IA). L'apprentissage profond de diverses architectures concerne les réseaux de neurones profond « deep neural network », les réseaux de neurones récurrents « Recurrent Neural Network », les réseaux de croyances profond « Deep Belief Networks » et ceux-ci ont été appliqués à divers domaines du monde informatique tels que la reconnaissance vocale, la traduction automatique, le traitement du langage naturel, le filtrage de réseaux sociaux, la bio-informatique et la conception de médicaments. Le Réseau Neuronal Convolutif « *CNNs* » est l'une des formes les plus importantes d'apprentissage profond. Le « *CNNs* » traite plusieurs couches d'un réseau de neurones simple et constitue donc l'algorithme le plus important pour classifier des images ou des symboles manuscrits.

La reconnaissance des chiffres manuscrits isolés est un problème difficile ces dernières années, Bien que de nombreux algorithmes de classification fondés sur

l'apprentissage profond soient étudiés pour cela, le taux de reconnaissance et la durée d'exécution doivent encore être améliorées.

Dans ce travail, nous nous intéresserons à la reconnaissance des chiffres manuscrits isolés en utilisant l'apprentissage profond pour connaître le contenu d'une image de la base de données non normalisé CVL, on vise à classifier les images en dix classes de (*0 à 9*).

Nous avons proposé une approche, basé sur le modèle de « *Convolutional Neural Networks- CNNs* » pour la classification des chiffres manuscrits isolés. Notre CNNs à de trois couches de convolution de taille des filtres (*5 x 5*), avec trois couches de Max-pooling chacune de taille de noyau (*4 x 4*) et une couche entièrement connectée qui donne meilleurs taux finals.

La thèse est organisée de la façon suivante :

Dans le chapitre 1, nous présentons les différentes étapes pour la réalisation d'un système de reconnaissance des chiffres manuscrits à partir de la phase de prétraitement, d'extraction de descripteurs jusqu'aux approches de classification basé sur les techniques de Machine Learning.

Le deuxième chapitre se focalisera sur les méthodes et techniques neuronales de l`apprentissage profond, une description plus détaillée sur les réseaux de neurones convolutifs (*CNNs*) qui est la méthode choisie dans notre expérimentation.

Le troisième chapitre présente un état de l'art qui donne une vision générale sur les méthodes d'analyse utilisé dans le système de reconnaissance des chiffres manuscrits et les résultats obtenus dans la littérature.

Le dernier chapitre constitue notre contribution, il exposera les résultats obtenus pour la validation du système.

Le travail se termine par une conclusion sur les résultats obtenus par la méthode utilisée, et des perspectives qui montreront les évolutions possibles de ce travail.

Chapitre 01

Un système de reconnaissance des chiffres manuscrits isolés

Chapitre 01 : Un système de reconnaissance des chiffres manuscrits isolés

1 Introduction

Le système de reconnaissance des chiffres manuscrits isolés (SRCMI) est considéré comme un point initial dans le domaine de la reconnaissance des formes.

Dans ce chapitre, Nous présentons les différentes phases de reconnaissance des chiffres manuscrits isolés en mettant l'accent sur toutes les phases du système de reconnaissance des chiffres manuscrits.

2 Le système de reconnaissance des chiffres manuscrits

Un système de reconnaissance des chiffres manuscrits (SRCM) est devenu un domaine intéressant qui attire les chercheurs en décennie années précédentes. Le système de reconnaissance basé sur les techniques de Machine Learning est composé en cinq phase importantes.

2.1.1 Phase de prétraitement

Les objectifs visés par les étapes de prétraitement sont nombreux. Il est principalement utilisé pour réduire le bruit sur les données manuscrites, parfois pour corriger les défauts, et essayez de ne conserver que les informations importantes du formulaire de représentant.

Le prétraitement joue un rôle très important et peut influer directement sur les performances de la reconnaissance, Depuis il y a beaucoup d'étapes pour compléter la reconnaissance avec succès, on peut les citer ci-dessous :

2.1.1.1 Éliminer le bruit

Élimination du bruit comprent principalement le décrochage, le lissage et la correction du point sauvage, caractérisé par de grandes variations angulaires comme monter dans l'article de [1]. Il s'agit d'éliminer le maximum de bruit pour augmenter la discrimination. Voir « *la figure 1.1* » :

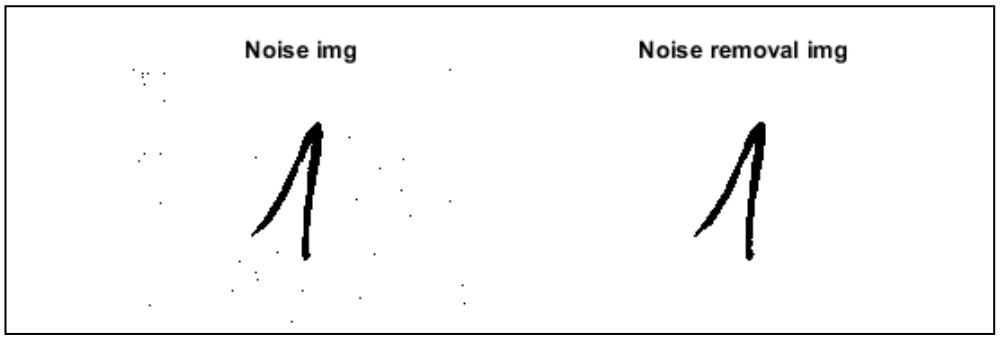

Figure 1.1 : Exemple de Suppression du bruit.

2.1.1.2 Binarisation

Le but de la binarisation d'une image est affecter le niveau 1 aux pixels dont la valeur est supérieure à un seuil S et le niveau 0 aux autres « *Inferieur* » en utilisant le seuillage.

Le document numérisé est une image qui peut être binarisé ou à niveaux de gris, à l'aide d'une méthode de seuillage. La détermination du seuil peut être globale, c'est-à-dire que le seuil a la même valeur pour tous les pixels du document [4], ou locale pour laquelle le seuil varie d'une position à une autre.

- Les méthodes de seuillage global ne sont pas trop consommatrices en termes de temps de calcul, par contre, elles ne donnent de bons résultats que si le document est uniformément éclairé.
- Les méthodes de seuillage local sont plus robustes à de telles dégradations mais leur temps de calcul est plus important [8].

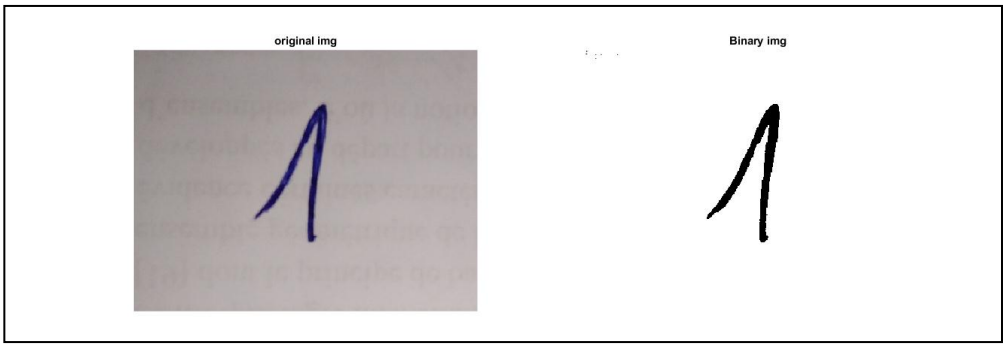

Figure 1.2 : Exemple de la binarisation.

2.1.1.3 Squelettisation

Il s'agit d'une représentation compacte intuitive formée de lignes d'épaisseur égale à un, centrée sur la figure, homotopique à la forme de départ « *même nombre de composantes connexes* » et ayant les mêmes formes et topologie que celle-ci [6].

Le processus de squelettisation permet de réduire et de compacter la taille de l'image, garder la forme générale de la particule et trouver un axe médian, définit comme l'ensemble de pixels S qui possèdent une égalité de distance par rapport aux pixels de frontière qui les entourent [7].

La représentation squelettique possède les avantages suivants [7] :

- Une bonne manière pour représenter au mieux les relations structurelles entre les composants de modèle.
- Très utilisée dans les systèmes de reconnaissance de caractères, mot, signature et empreinte.

Remarque : Selon la qualité du document à traiter, des variations des taille, des formes, la méthode d'analyse adoptée et plus important encore des différences dans les styles d'écriture des individus, une ou plusieurs techniques de prétraitement sont utilisées.

Figure 1.3 : Exemple de squelettisation

2.1.1.4 Normalisation de la taille

Pour faciliter la phase de reconnaissance, les chiffres nécessitent d'être standardisées à une taille fixée en utilisant des techniques des échantillonnages.

La contrainte primordiale réside dans le choix de cette taille. La taille exagérément petite a un risque d'une perte d'information. Trop grande, la phase de reconnaissance va opérer lentement [9].

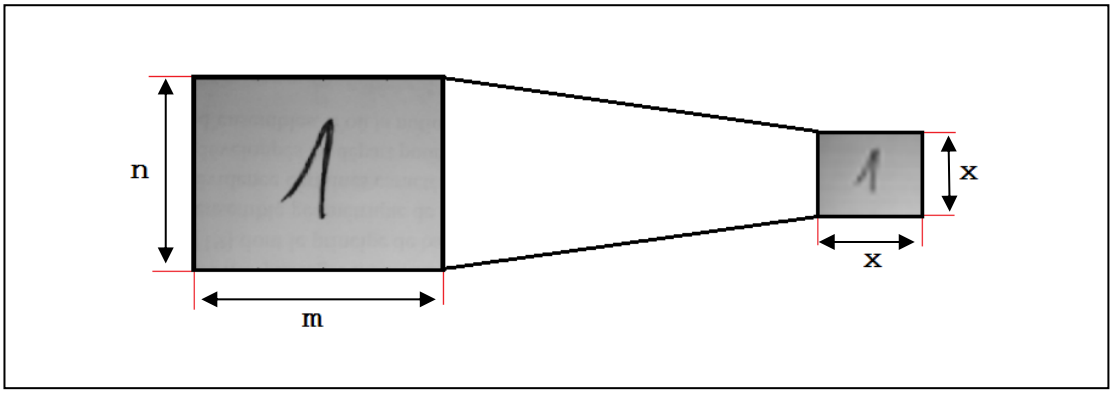

Figure 1.4 : Exemple d'un chiffre manuscrit normalisé

2.1.1.5 Zonage

Le zonage est l'une des fonctionnalités les plus populaires et la plus simple à mettre en œuvre. Voir « la figure 1.5)

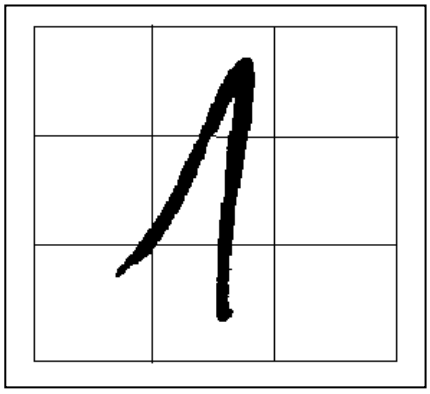

*Figure 1.5 : Exemple d'un chiffre zoné de taille (3 * 3)*

Lorsque redimensionnée une image individuelle de taille « *n* * *n* », les pixels sont divisés en (x) zones ou blocs égaux, chacun d'eux ayant la même taille. Les caractéristiques sont extraites en comptant le nombre de pixels noirs dans chaque zone. Cette procédure est répétée séquentiellement pour toutes les zones (x) qui sont stockées sous la forme d'un tableau de signatures pour chaque caractère. Ainsi, pour chaque caractère, nous obtenons un tableau de signatures de longueur (x) calculé à partir de chaque zone [21].

2.1.2 Phase de segmentation

La segmentation d'images ainsi définie est un domaine vaste où l'on retrouve de très nombreuses approches [10].

La segmentation permet de diviser le texte manuscrit en plusieurs entités telles que les mots et les chiffres. Pour la segmentation du chiffre numérique, cela consiste à diviser la chaine des chiffres en plusieurs images comportant chacune un chiffre isolé. De même, pour les images littérales.

Il existe deux grandes catégories de segmentations :

- ❖ La Segmentation explicite
- ❖ La Segmentation implicite

2.1.2.1 Segmentation explicite

Dans cette approche, la séparation des chiffres consiste à diviser la chaine des chiffres en plusieurs images parfaitement que possible, Voir « *la figure 1.6* ».
Les chemins de segmentation sont généralement obtenus par des points caractéristiques, tels que [5] :

- ✓ Des minima locaux du contour supérieur.
- ✓ Des espaces entre les caractères ou bien les sous-mots.
- ✓ Des points d'intersection les plus probables par une analyse des composantes dans le mot.

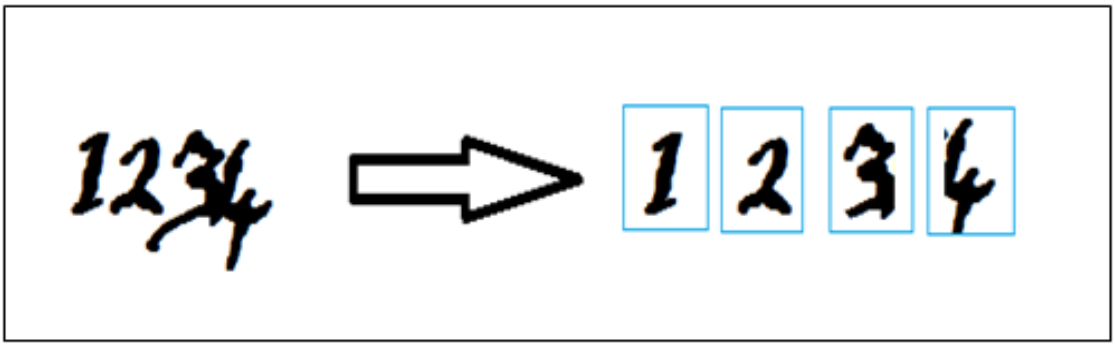

Figure 1.6 : Exemple de segmentions des chiffres manuscrits (explicite).

2.1.2.2 Segmentation implicite

La Segmentation implicite consiste à segmenter le mot en parties inférieures aux lettres appelées graphèmes et à retrouver les lettres puis les mots par combinaison de ces graphèmes [14], Voir « *la figure 1.7* ».
Cette segmentation présente des avantages et des inconvénients ce qui fait que cette approche est insuffisante pour une modélisation optimale de l'écriture [11] :

- L'avantage de cette segmentation c'est que l'information est localisée par les modèles des lettres et la validation se fait par ses modèles. Il n'y aura pas d'erreur de segmentation et enfin on contourne le dilemme de Sayre car en connaissant les lettres, on n'engendre pas d'erreur de segmentation.
- Les défauts de cette segmentation viennent du fait que l'espace de recherche des limites se trouve très augmenté et le problème est ramené à un problème de recherche de zones où se trouvent ces limites.

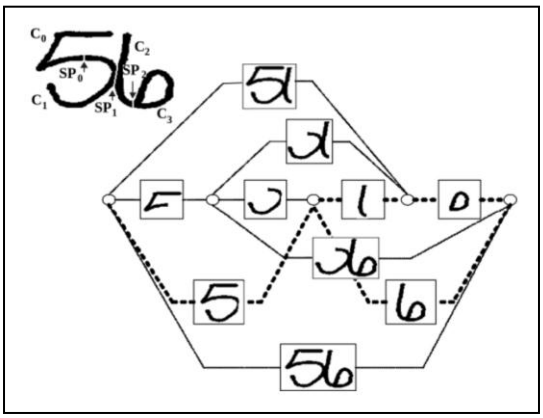

Figure 1.7 : Exemple de segmentions des chiffres manuscrits « implicite » [78]

2.1.3 Phase d'extraction des caractéristiques

L'objectif de l'extraction des caractéristiques dans le domaine de la reconnaissance consiste à exprimer les primitives sous une forme numérique ou symbolique appelée codage. Selon le cas, les valeurs de ces primitives peuvent être réelles, entières ou binaires [12].

La difficulté ici est de trouver de bonnes caractéristiques. De « *bonnes* » caractéristiques permettent aux classifieurs de reconnaître facilement les différentes classes d'objets ; on dit alors qu'elles sont discriminantes. Elles doivent aussi être invariantes à certaines transformations « *le chiffre 1 appartient à la même classe quelle que soit sa taille et la structure* » [13].

2.1.3.1 caractéristiques structurelles

Les caractéristiques structurelles de l'image qui peut considérer la forme de caractère exemple « *contour, point final, points de branchement, boucles* » directions caractéristiques et autres [15].

Une description structurale d'une texture implique la recherche des motifs élémentaires, leur description, puis, la détermination des règles conditionnant leur position [16]. En générale, Les caractéristiques structurelles utilisées dépendent de la forme à classifier.

2.1.3.2 caractéristiques texturale

L'analyse de la texture fait référence à la caractérisation des régions d'une image par leur contenu de texture. L'analyse de texture tente de quantifier les qualités intuitives décrites par des termes tels que rugueux, lisse, soyeux ou cahoteux, en fonction de la variation spatiale de l'intensité des pixels. En ce sens, la rugosité ou les bosses font référence aux variations des valeurs d'intensité, ou niveaux de gris.

2.1.3.3 caractéristiques globale

Les caractéristiques globales combinent les informations de modèle à longue portée en une valeur de caractéristique unique [17]. À représenter au mieux la forme générale d'un caractère [18], dépendre de la totalité des pixels d'une image ces primitives sont donc dérivées de la distribution des pixels [9].

Les primitives globales cherchent à représenter au mieux la forme générale d'un caractère et sont donc calculées sur des images relativement grandes « *ex : transformé de Fourier et transformé de Hough* » [7].

2.1.3.4 caractéristiques morphologique

La morphologie mathématique permettent d'extraire un squelette pixellique qui garantit la connexité de l'image initiale [19].

La morphologie mathématique constitue une Technique d'analyse d'images à part entière et peut être utilisée pour résoudre un grand nombre de problèmes de traitement d'images tels que [20] :

- **Le filtrage non linéaire d'images :** Pour conserver ou supprimer des structures d'une image possédant certaines caractéristiques, notamment de forme « *morphologiques* ».
- **Mesure :** Pour obtenir des valeurs numériques caractérisant certaines propriétés des objets de l'image « *exemple : granulométrie, analyse de textures, ...* ».
- **Segmentation :** Pour obtenir une partition de l'image en ses différentes régions d'intérêt. Généralement, on cherche à séparer les objets de l'image du fond. Le paradigme de segmentation morphologique s'appuie sur l'opérateur de ligne de partage des eaux.

2.1.4 Phase de classification

La classification est l'élaboration d'une règle de décision qui transforme les attributs caractérisant les formes en appartenance à une classe « *passage de l'espace de codage vers l'espace de décision* » [22].

Selon [23], La processus de classification est réalisé en deux phases :

2.1.4.1 l'apprentissage

Cette étape permet de construire un dictionnaire de prototype. Il s'agit de regrouper en classes plusieurs prototypes dont les caractéristiques se rapprochent.
D'après [24], On distingue ainsi trois types d'apprentissage :

2.1.4.1.1 Apprentissage supervisé « *Classification* »

L'apprentissage supervisé « *ou classification* » consiste à construire un modèle basé sur un base d'apprentissage et des Étiquettes « *labels nom des catégories ou des classes* » et à l'utiliser pour classer des données nouvelles [25] [26].

Il existe plusieurs algorithmes et techniques utilisés pour la classification supervisée telles que :

- K-Nearest Neighbors (*KNN*)
- Classification Bayésienne.
- Machine à vecteurs de support (*SVM*).
- Réseau neuronal artificiel.
- Arbres de décision.
- …

2.1.4.1.2 Apprentissage non supervisé

L'apprentissage non supervisé ou sans professeur consiste à doter le système d'un mécanisme automatique qui s'appuie sur des règles précises de regroupement pour trouver les classes de référence avec une assistance minimale. Dans ce cas les échantillons sont introduits en un grand nombre par l'utilisateur sans indiquer leur classe [28].

On distingue plusieurs algorithmes de clustering, exemple :

- Artificial Neural Networks « *ANN* ».
- K-moyennes « *KMeans* ».
- Fuzzy KMeans
- Espérance-Maximisation (*EM*)
- Regroupement hiérarchique
- …

2.1.4.1.3 Apprentissage semi-supervisé

L'apprentissage semi-supervisé utilise un ensemble de données étiquetées et non-étiquetés. Il se situe ainsi entre l'apprentissage supervisé qui n'utilise que des données étiquetées et l'apprentissage non-supervisé qui n'utilise que des données non-étiquetées. L'utilisation de données non-étiquetées, en combinaison avec des données étiquetées, permet d'améliorer de façon significative la qualité de l'apprentissage. Un autre avantage vient du fait que l'étiquette de données nécessite l'intervention d'un utilisateur humain. Lorsque les jeux de données deviennent très grands, cette opération peut s'avérer fastidieuse. Dans ce cas, l'apprentissage semi-supervisé, qui ne nécessite que quelques étiquettes, revêt un intérêt pratique évident et indiscutable [29]. Dans le cas au la reconnaissance des chiffres manuscrits, la classification supervisée a été adoptée car elle simplifie considérablement le problème [27].

2.1.4.2 la reconnaissance et décision

La reconnaissance peut conduire à un succès si la réponse est unique « un seul modèle répond à la description de la forme du caractère ». Elle peut conduire à une confusion si la réponse est multiple « plusieurs modèles correspondent à la description ». Enfin elle peut conduire à un rejet de la forme si aucun modèle ne correspond à sa description. Dans les deux premiers cas, la décision peut être accompagnée d'une mesure de vraisemblance, appelée aussi score ou taux de reconnaissance [28]. Selon [30], Il existe trois approches principales et une approche hybride :

2.1.4.2.1 Approche statistique

Elle est fondée sur l'étude statistique des mesures que l'on effectue sur les formes à reconnaître. L'étude de leur répartition dans un espace métrique et la caractérisation statistique des classes, permettent de prendre une décision de reconnaissance du type « plus forte probabilité d'appartenance à une classe » [28] [31]. Mais elle a besoin d'un nombre élevé d'exemples pour effectue un apprentissage correct des lois de probabilité des différentes classes. L'étude de leur répartition dans un espace métrique et la caractérisation statistique des classes permet de prendre une décision du type : plus forte probabilité d'appartenance à une classe. Cette approche bénéficie des méthodes d'apprentissage automatique qui s'appuient sur des bases théoriques connues telles que :

- Les méthodes non paramétriques « *méthode des k plus proches voisins, …* » [34 [32].

Parmi ces nombreuses théories et méthodes on peut citer à titre d'exemple :

- L'approche bayésienne.
- La méthode du plus proche voisin.
- Les réseaux de neurones artificiels.
- Méthode du plus proche voisin.
- …

2.1.4.2.2 Approche structurelle ou syntaxique

Les méthodes structurelles reposent sur l'extraction de primitives en prenant compte de l'information structurelle [35], et sur la structure physique des caractères. Elles cherchent à trouver des éléments simples ou primitives, et à décrire leurs relations. Les primitives sont de type topologique telles que : une boucle, un arc… et une relation peut être la position relative d'une primitive par rapport à une autre [35] [36].

La différence principale entre ces méthodes et les méthodes statistiques est que ces caractéristiques sont des formes élémentaires « *les primitives sont de type topologiques* » et non pas des mesures [31].

Cette approche Contient plusieurs méthodes telles que :

- Les structures de graphes.
- Les structures syntaxiques.
- Le calcul de distance d'édition entre deux chaînes.
- La programmation dynamique.
- Les méthodes de tests.
- La comparaison de chaînes.
- …

2.1.4.2.3 Approche stochastique

L'approche stochastique utilise un modèle pour la reconnaissance, prenant en compte la grande variabilité de la forme. Dans ce type d'approche, les modèles sont souvent discrets et de nombreux travaux reposent sur la théorie des chaines de Markov et l'estimation bayésienne [28].

L'approche stochastique considère la forme comme un signal continu dans le temps, observable à différents endroits constituant les *états d'observation* « *elle n'est donc utilisable que sur des objets comportant une information temporelle* ». Le modèle décrit ces états à l'aide de probabilités, de transition d'état à état, et d'observation d'états [37] [38].

2.1.4.2.4 Approche hybride

Pour améliorer les performances de reconnaissance, la tendance aujourd'hui est de construire des systèmes hybrides qui utilisent différents types de caractéristiques, et qui combinent plusieurs classifieurs en couches [37] [40].

L'Approche hybride, Est une approche intègre deux ou plusieurs des approches qui ont été précédemment rapportés pour surmonter les faiblesses de chaque approche et obtenir des résultats plus précis, plus meilleure que les résultats qui auraient été obtenus si l'application de chaque approche séparément, comme les approche qui ont été fusionnés pour former celui intégré [39].

Un exemple de cette approche une approche hybride qui combine les approches statistiques et structurelles voir (Figure 1.8)0

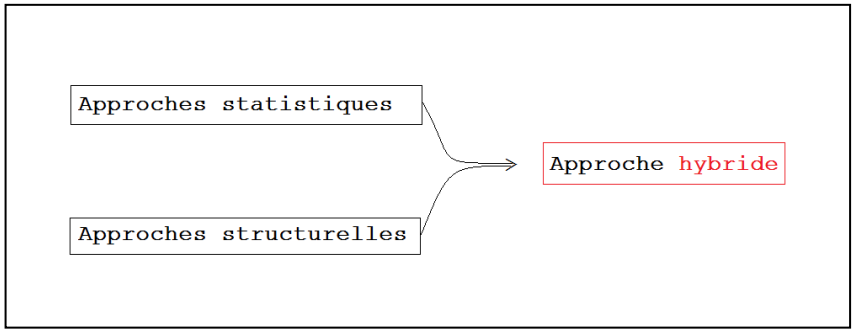

Figure 1.8 : Exemple d'hybridation entre deux approches.

3 Conclusion

Dans ce chapitre, nous avons présenté d'une manière générale les différentes phases de système de reconnaissance des chiffres manuscrits (SRCM), à partir de la phase de prétraitement jusqu'aux approches de classification basé sur les techniques de Machine Learning.

Dans le deuxième chapitre, nous aurons abordé les principaux méthodes et techniques de l`apprentissage profond « *Deep Learning* ».

Chapitre 02

L'apprentissage profond

(Deep Learning)

Chapitre 02 : L'apprentissage profond (Deep Learning)

1 Introduction

L'apprentissage profond est l'un des domaines d'apprentissage les plus difficiles et les plus novateurs, car il permet un apprentissage automatique à plusieurs niveaux pour représenter la distribution de base des données à concevoir.

Dans ce chapitre, nous nous concentrons principalement sur les modèles de réseau de neurones et les types d'apprentissage profond.

2 Système de reconnaissance basé sur l'apprentissage profond « Deep Learning »

L'apprentissage profond est un sous-ensemble d'apprentissage automatique « *Machine Learning* » qui prend les données en entrée et prend des décisions intuitives et intelligentes à l'aide d'un réseau de neurones artificiels « *RNA* » superposés et contenant des couches cachées. D'autre part, est nécessité le module de prétraitement « *a été expliqué dans le chapitre 1* » pour améliorer et réglé l'image entrée.

2.1 Réseaux de neurones artificiels

Les réseaux de neurones artificiels « *RNA* » sont des réseaux fortement connectés de processeurs élémentaires fonctionnant en parallèle. Chaque processeur élémentaire calcule une sortie unique sur la base des informations qu'il reçoit. Toute structure hiérarchique de réseaux est évidemment un réseau [34]. D'autre part, chaque processeur utilise des perceptrons ou bien des neurones pour faire une seule sortie.

Le principe est de regarder la sortie par rapport à ce qui était attendu et de mettre à jour les liaisons entre les neurones « *les renforcer ou les inhiber* » pour améliorer le résultat final, qui sera une prédiction de la part du réseau.

2.1.1 C'est quoi un perceptron ?

Le perceptron « *un seul neurone* » est un modèle de réseau de neurones artificiels avec algorithme d'apprentissage [8].

2.1.1.1 perceptron linéaire à seuil

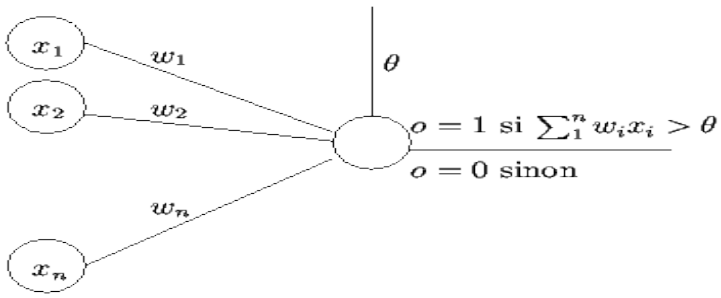

Figure 2. 1 : Le perceptron avec seuil

Un perceptron linéaire à seuil « *voire figure 2.1* » prend en entrée n valeurs « $x_1, \dots, x_n$ » et calcule une sortie o Un perceptron est défini par la donnée de « $n +$

1 » constantes : les coefficients synaptiques « $w_1, \dots, w_n$ » et le seuil « *ou le biais* » (θ) [8] [34].

2.1.1.2 Perceptron avec entrée supplémentaire

On doit remplacer le seuil par une entrée supplémentaire x_0 qui prend toujours comme valeur d'entrée la valeur « $x_0 = 1$ ». À cette entrée est associée un coefficient synaptique w_0. Le modèle correspondant est décrit dans « *la figure 2.2* ». On peut décomposer le calcul de la sortie *o* en un premier calcul de la quantité « $\sum_0^i w_i$ » appelée « *potentiel post-synaptique* ou l'*entrée totale* » suivi d'une application d'une *fonction d'activation* (H) sur cette entrée totale. La fonction d'activation est la fonction de Heaviside définie par [8] :

$$f(x) = \begin{cases} 1 \; x > 0 \\ 0 \; Sinon \end{cases} \text{..... (1)}$$

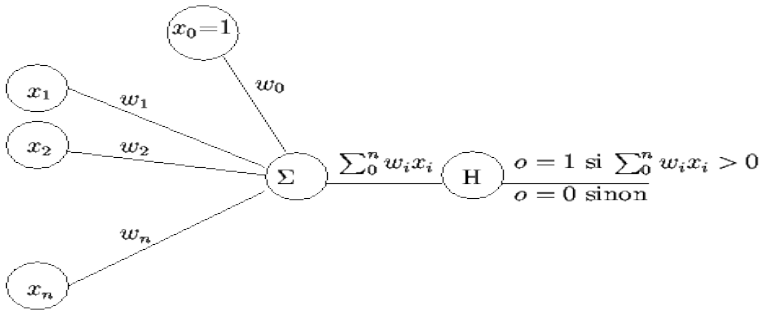

Figure 2. 2 : Le perceptron avec entrée supplémentaire

2.1.1.3 Perceptrons qui calculent le OÙ

Un perceptron qui calcule le OU logique avec les deux versions : seuil ou entrée supplémentaire est présenté dans « la figure 2.3 » [8][34].

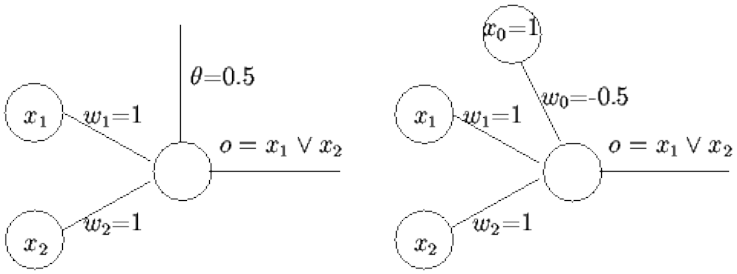

Figure 2. 3 : perceptrons qui calculent le OÙ

2.1.1.4 Fonction d'activations

La fonction d'activation est une fonction qui appliquée à un signal en sortie d'un neurone artificiel. L'activation a pour but de transformer le signal de manière à obtenir une valeur de sortie à partir de transformations complexes entre les entrées.

A titre illustratif voici quelques fonctions couramment utilisées dans le domaine des réseaux de neurones :

- ✓ La fonction de Heaviside (1), qui renvoie 1 si le signal en entrée est positif, ou 0 s'il est négatif.

- ✓ La fonction de Signe doux (2), qui renvoie 1 si le nombre est strictement positif, 0 si le nombre est nul, et -1 si le nombre est strictement négatif [75].

$$\forall x \in R, sgn(x) = \begin{cases} -1 \; si \; x < 0 \\ 0 \; six = 0 \\ 1 \; si \; x > 0 \end{cases} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (2)$$

- ✓ La fonction sigmoïde (3) est utilisée pour le calcul de la valeur de sortie (x) d'un neurone artificiel. Celle-ci est de forme [76] :

$$f(x) = \frac{1}{1+e^{-x}} \quad \ldots\ldots\ldots (3)$$

- ✓ La fonction linéaire c'est une fonction simple de la forme [77] :

$$\begin{cases} f(x) = ax \\ \quad \text{ou} \\ f(x) = x \end{cases} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots (4)$$

En gros, l'entrée passe à la sortie sans une très grande modification ou alors sans aucune modification.

2.1.2 perceptrons multicouche

Le perceptron multicouche « *Multilayer perceptron-MLP en anglais* » est un type de réseau organisé en plusieurs couches au sein desquelles une information circule. Un MLP comprend au moins trois couches de nœuds voire la « *Figure : 2.4* » [72] [73] :

- Une couche d’entrée (*n*) entrées « *pas neurones* ».
- Une couche de sortie de (*m*) neurones.
- Plusieurs couches intermédiaires « *cachées* » avec plusieurs neurones.

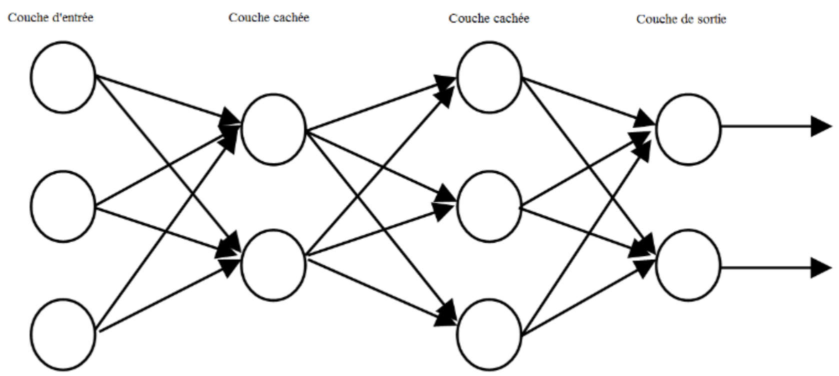

Figure 2. 4 : Example d’un perceptron multicouche

À l'exception des nœuds d'entrée, chaque nœud est un neurone utilisant une fonction d'activation non linéaire. MLP utilise une technique d'apprentissage supervisée appelée rétropropagation pour la formation. Ses multiples couches et son activation non linéaire distinguent MLP d'un perceptron linéaire. Il peut distinguer des données qui ne sont pas séparables linéairement [74].

2.1.3 Structure d'interconnexion

Les connexions entre les neurones qui composent le réseau décrivent la topologie du modèle. Elle peut être quelconque, mais le plus souvent il est possible de distinguer une certaine régularité [34].

Dons les Réseau multicouche « *RMC* », les neurones sont arragés par couche. Il n'y a pas de connexion entre neurones d'une même couche et les connexions ne se font qu'avec les neurones des couches avales « *figure 2.5* ». Habituellement, chaque neurone d'une couche est connecté à tous les neurones de la couche suivante et celle-ci seulement. Ceci nous permet de soumission la notion de sens de parcours de l'information « *de l'activation* » au sein d'un réseau et donc définir les concepts de neurone d'entrée, neurone de sortie.

Par extension, on appelle couche d'entrée l'ensemble des « *neurones d'entrée* », couche de sortie l'ensemble des « *neurones de sortie* » et les couches intermédiaires sont appelés « *les couches cachée* » [8] [34].

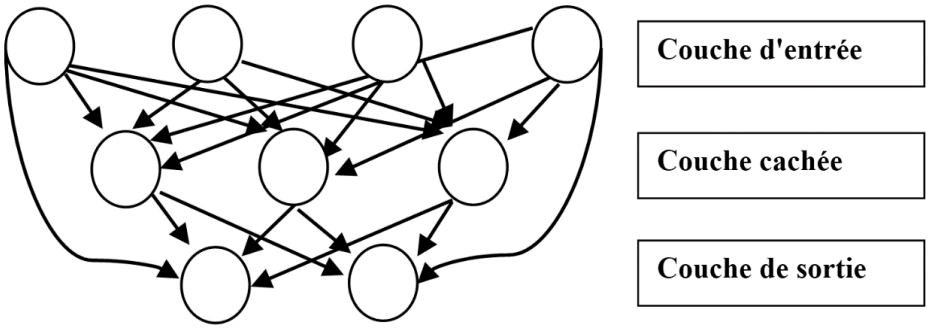

Figure 2. 5 : Définition des couches d'un réseau multicouche

2.1.3.1 Réseau à connexions locales :

Il s'agit d'une structure multicouche, mais qui à l'image de la rétine, conserve une certaine topologie. Chaque neurone entretien des relations avec un nombre réduit et localisé de neurones de la couche avale « *figure 2.6* ». Les connexions sont donc moins nombreuses que dans le cas d'un réseau multicouche classique [34].

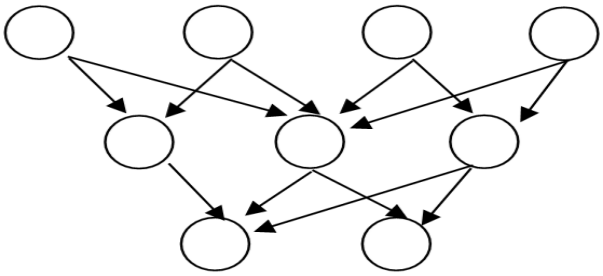

Figure 2. 6 : Réseau à connexions locales

2.1.3.2 Réseau à connexions récurrentes :

Les connexions récurrentes rebroussent l'information en arrière par rapport au sens de propagation défini dans un réseau multicouche. Ces connexions sont le plus souvent locales « *figure 2.7* » [34].

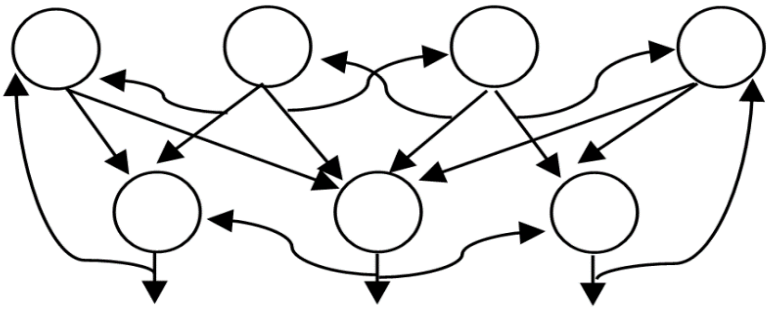

Figure 2. 7 : Réseau à connexions récurrentes

2.1.3.3 Réseau à connexion complète :

C'est la structure d'interconnexion la plus générale « *figure 2.8* ». Chaque neurone est connecté à tous les neurones du réseau « *et à lui-même* » [34].

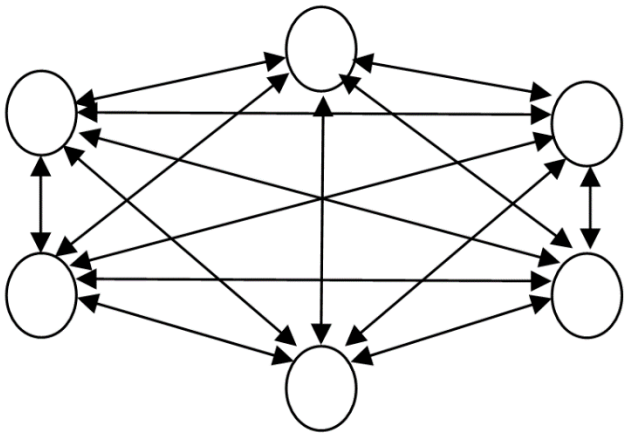

Figure 2. 8 : Réseau à connexions complète

Il existe de nombreuses autres topologies possibles, mais elles n'ont pas eu à ce jour la notoriété des quelques-unes que nous avons décrites ici.

3 Réseaux de neurone profond

Les réseaux de neurones dits *« profond »* « *Deep Neural Networks en anglais* » sont des perceptrons multicouches-PMC « *Multilayer perceptrons-MLP en anglais* » avec un nombre de couches supérieur à trois. Pendant longtemps les méthodes d'apprentissage pour ce type de réseaux de neurones acycliques ne permettaient pas de converger vers un réseau de neurones performant. Des avancées majeures sur les méthodes d'entraînement et le choix de la fonction de transfert Rectified Linear Unit « *ReLU* » qui minimise l'impact de la dilution du gradient dans les couches basses du réseau ont permis d'utiliser des réseaux de neurones de plus en plus gros [63].

4 Les types d'apprentissage profond « *Deep Learning* »

Il existe différentes méthodes conçues pour appliquer l'apprentissage profond. Chaque méthode proposée à un cas d'utilisation spécifique, comme le type de données.

Donc, Sur la base de ces facteurs, voici quelques-unes des méthodes d'apprentissage profond :

4.1 Réseau de neurones convolutifs « CNNs »

Le terme réseau convolutifs « *CNNs-Convolutional neural networks en anglais* » est utilisé pour décrire une architecture permettant d'appliquer des réseaux de neurones à des réseaux bidimensionnels « *généralement des images* », basée sur une entrée neurale localisée dans l'espace [71]. Cette architecture a également été décrite comme la technique des poids partagés ou des champs récepteurs locaux [41] [42]. Le but de « *CNNs* » est d'utiliser des informations spatiales entre les pixels d'une image [43]. Les « *CNNs* » sont particulièrement utiles pour extraire des caractéristiques dans les images afin de reconnaître des chiffres manuscrits. Ils apprennent directement à partir de données d'images, en utilisant des modèles pour classer les images et en éliminant le besoin de module d'extraction de caractéristique ordiner, voir la « *figure 2.9* ».

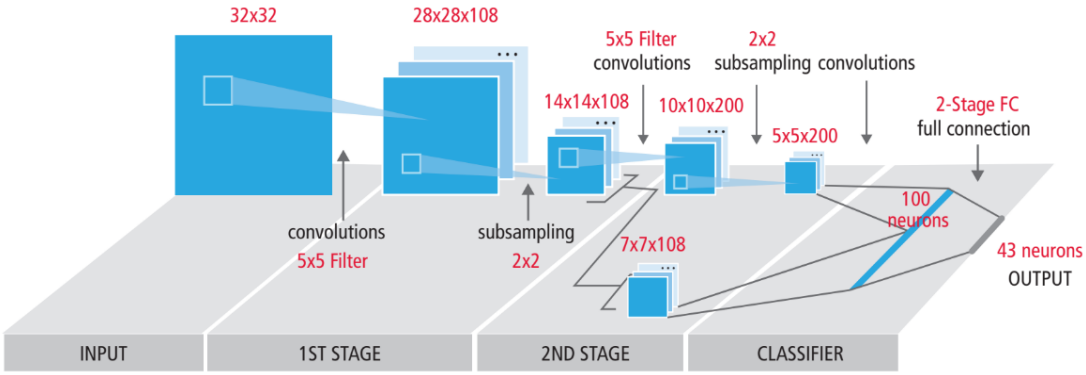

Figure 2. 9 : Schéma fonctionnel typique d'un CNNs [47]

En empilant plusieurs couches différentes dans un « CNNs », des architectures complexes sont construites pour résoudre les problèmes de classification. Les quatre types de couches les plus courants sont les suivantes :

4.1.1 Couche de convolution « CONV » :

L'opération de convolution extrait différentes caractéristiques de l'entrée. La première couche de convolution extrait les caractéristiques de bas niveau telles que les arêtes, les lignes et les angles. Les couches de niveau supérieur extraient des caractéristiques de niveau supérieur. « *La figure 2.10* » illustre le processus de convolution utilisé dans les « *CNNs* » [43] [44] [45].

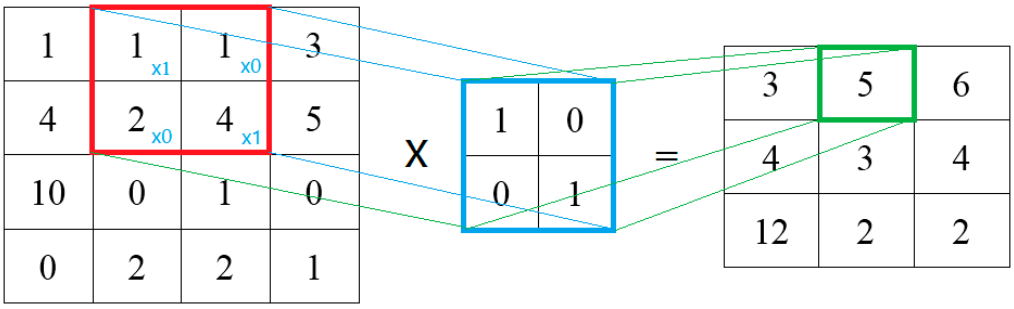

*Figure 2. 10 : Représentation de convolution sur un image de taille « 4*4 » pixels et un convolution filtre*

Au niveau d'une couche de convolution, les mappes d'entités de la couche précédente sont convolutionnées avec des noyaux pouvant être appris et transmises à la fonction d'activation pour former la mappe d'entités en sortie. Chaque carte en sortie peut combiner des convolutions avec plusieurs cartes en entrée [44].

4.1.2 Couches de regroupement (Pooling)

Un autre outil très puissant utilisé par les « *CNNs* » s'appelle le Pooling « *Subsampling en anglais ou sous-échantillonnage en français* ». Le Pooling est une méthode permettant de prendre une large image et d'en réduire la taille tout en préservant les informations les plus importantes qu'elle contient.

La couche de regroupement réduit la résolution des fonctionnalités. Il rend les fonctionnalités robustes contre le bruit et la distorsion. Il y a deux façons de mettre en commun [43] [45] :

- Max pooling.
- Mise en commun moyenne.

Dans les deux cas, l'entrée est divisée en espaces bidimensionnels ne se chevauchant pas. Par exemple, sur la « *figure 2.8* », la couche « *2* » est la couche de regroupement. Chaque entité en entrée mesure « *28 x 28* » et est divisée en régions « *14 x 14* » de taille « *2 x 2* ». Pour le regroupement moyen, la moyenne des quatre valeurs de la région est calculée. Pour la mise en pool maximale, la valeur maximale des quatre valeurs est sélectionnée.

« *La figure 2.11* » explique plus en détail le processus de mise en commun. L'entrée est de taille « *4 x 4* ». Pour le sous-échantillonnage « *2 x 2* », une image « *4 x 4* » est divisée en quatre matrices non superposées de taille « *2 x 2* ». Dans le cas d'une mise en pool maximale, la valeur maximale des quatre valeurs de la matrice « *2 x 2* » est la sortie. En cas de regroupement moyen, la moyenne des quatre valeurs est la sortie.

Remarque : pour la sortie avec le filtre « *2, 2* », le résultat de la moyenne est une fraction qui a été arrondie à l'entier le plus proche.

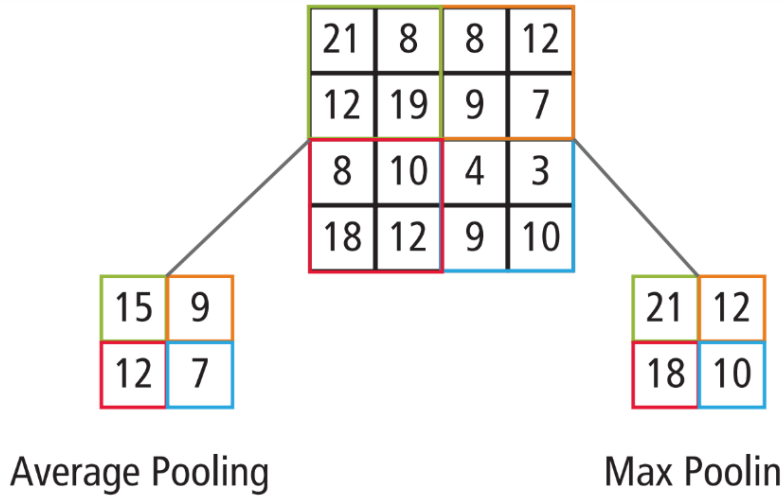

Figure 2. 11 : Représentation graphique de « max pooling » Et « average pooling » [45]

4.1.3 Couches non linéaires « Non-linear layers »

Les réseaux de neurones en général et les « *CNNs* » en particulier s'appuient sur une fonction de « *déclenchement* » non linéaire pour signaler une identification distincte des caractéristiques probables sur chaque couche masquée. Les « *CNNs* » peuvent utiliser diverses fonctions spécifiques, telles que des unités linéaires rectifiées « *ReLU* » et des fonctions de déclenchement continues « *non linéaires* », pour mettre en œuvre efficacement ce déclenchement non linéaire [43] [44] [45].

Un ReLU implémente la fonction « *y = max (x, 0)* », de sorte que les tailles d'entrée et de sortie de cette couche sont identiques. Il augmente les propriétés non linéaires de la fonction de décision et du réseau global sans affecter les champs récepteurs de la couche de convolution. Par rapport aux autres fonctions non linéaires utilisées dans les CNNs « *par exemple, tangente hyperbolique, absolue de la tangente hyperbolique et sigmoïde* », l'avantage d'une ReLU est que le réseau s'entraîne beaucoup plus rapidement. La fonctionnalité ReLU est illustrée à « *la figure 2.11* », avec sa fonction de transfert tracée au-dessus de la flèche [45].

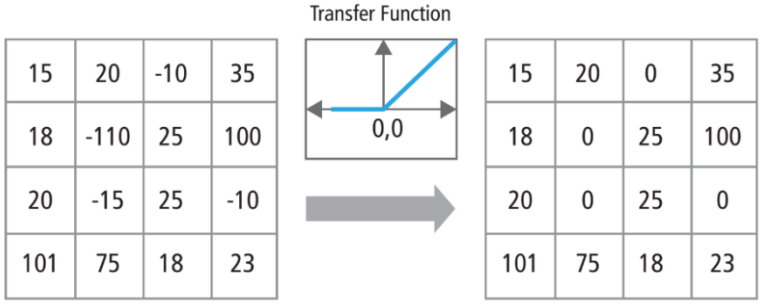

Figure 2. 12 : Représentation graphique de la fonctionnalité ReLU [45]

Chaque fois qu'il y a une valeur négative dans un pixel, on la remplace par un *0*. Le résultat d'une couche « *ReLU* » est de la même taille que ce qui lui est passé en entrée, avec simplement toutes les valeurs négatives éliminées.

4.1.5 Couches entièrement connectées « *Fully connected layers (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 [43] [48]. Les couches entièrement connectées sont souvent utilisées en tant que couches finales d'un « *CNNs* ». Ces couches additionnent mathématiquement une pondération de la couche précédente d'entités, indiquant le mélange précis d'ingrédients pour déterminer un résultat de sortie cible spécifique. Dans le cas d'une couche entièrement connectée, tous les éléments de toutes les entités de la couche précédente sont utilisés dans le calcul de chaque élément de chaque entité en sortie.

4.2 Boltzmann Machines « *BMs* »

Une machine Boltzmann est un réseau d'unités neuronales connectées symétriquement qui prennent des décisions stochastiques quant à l'activation ou non. Les machines Boltzmann disposent d'un algorithme d'apprentissage simple qui leur permet de découvrir des caractéristiques intéressantes dans des jeux de données composés de vecteurs binaires. L'algorithme d'apprentissage est très lent dans les réseaux comportant de nombreuses couches de détecteurs de caractéristiques, mais il peut être rendu beaucoup plus rapide en apprenant une couche de détecteurs de caractéristiques à la fois [8].

Les machines Boltzmann sont utilisées pour résoudre deux problèmes de calcul très différents. Pour un problème de recherche, les poids sur les connexions sont fixes et servent à représenter la fonction de coût d'un problème d'optimisation. La dynamique stochastique d'une machine de Boltzmann lui permet ensuite d'échantillonner des vecteurs d'états binaires qui représentent de bonnes solutions au problème d'optimisation [3] [8].

Pour un problème d'apprentissage, un ensemble de vecteurs de données binaires est présenté à la machine Boltzmann. Elle doit trouver des pondérations sur les connexions afin que les vecteurs de données apportent de bonnes solutions au problème d'optimisation défini par ces pondérations. Pour résoudre un problème d'apprentissage, les machines Boltzmann apportent de nombreuses modifications mineures à leurs poids, et chaque mise à jour les oblige à résoudre de nombreux problèmes de recherche différents [2] [3] [8].

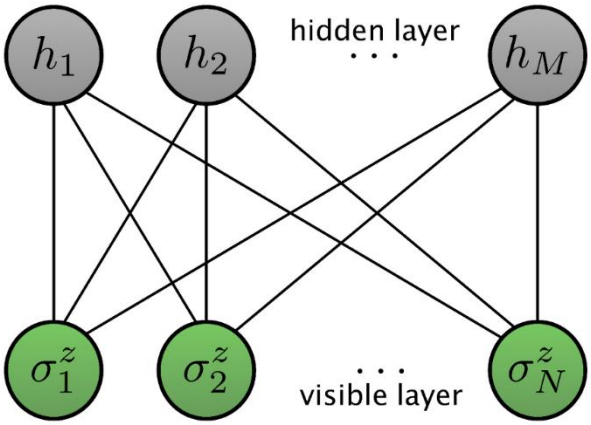

Figure 2. 13 : Exemple de machines Boltzmann

4.2.1 Apprentissage dans les machines Boltzmann

Pour un ensemble de vecteurs d'état d'apprentissage « *les données* », l'apprentissage consiste à rechercher des poids et des biais « *les paramètres* » qui rendent ces vecteurs d'état bons. Plus spécifiquement, l'objectif est de trouver les « *poids / biais* » qui définissent une distribution de Boltzmann dans laquelle les vecteurs d'apprentissage ont une probabilité élevée [8] [60].

4.2.2 La vitesse d'apprentissage

L'apprentissage est généralement très lent dans les machines Boltzmann comportant de nombreuses couches cachées, car les grands réseaux peuvent mettre longtemps à se rapprocher de leur distribution d'équilibre, en particulier lorsque les poids sont importants et que la distribution d'équilibre est hautement multimodale, comme c'est généralement le cas lorsque les unités visibles ne sont pas attachées. Même si des échantillons de la distribution d'équilibre peuvent être obtenus, le signal d'apprentissage est très bruyant, car il s'agit de la différence entre deux attentes échantillonnées. Ces difficultés peuvent être surmontées en limitant la connectivité, en simplifiant l'algorithme d'apprentissage et en apprenant une couche cachée à la fois [8].

4.3 Réseau Hopfield

Le réseau Hopfield consiste en un ensemble de neurones interconnectés qui mettent à jour leurs valeurs d'activation de manière asynchrone. Les valeurs d'activation sont binaires, généralement *{-1,1}*. La mise à jour d'une unité dépend des autres unités du réseau et d'elle-même. Une unité i sera influencée par une autre unité j avec un certain poids et aura une valeur seuil.

Il existe donc une contrainte À cause aux autres neurones et À cause au seuil spécifique de l'unité [61].

4.3.1 Réseaux asynchrones

Dans un réseau asynchrone, chaque unité calcule son excitation à des moments aléatoires et change son état en 1 ou -1 indépendamment des autres et en fonction du signe de son excitation totale. La probabilité que deux unités tirent simultanément est égale à zéro. Par conséquent, la même dynamique peut être obtenue en sélectionnant une unité de manière aléatoire, en calculant son excitation et en mettant à jour son état en conséquence. Il n'y aura pas de délai entre le calcul de l'excitation et la mise à jour de l'état [62].

4.3.2 Mise à jour et paramètres

La nouvelle valeur d'activation « *état* » d'un neurone est calculée, en temps discret, par la fonction [61] :

$$\begin{cases} x_i(t+1) = sign(\sum_{j=1}^{n} x_j(t) w_{ij} - \theta_i) \\ Ou\ X = sign(XW - T) \end{cases} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots (4)$$

- Où X, W, T et la fonction de signe sont : $X = \begin{cases} x_1 \\ x_1 \\ \vdots \\ x_n \end{cases}$ …… (5)

- W est la matrice de poids : $W = \begin{pmatrix} w_{11} & w_{12} & \cdots & w_{1n} \\ w_{21} & w_{22} & \cdots & w_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ w_{n1} & \cdots & \cdots & w_{nn} \end{pmatrix}$….. (6) où w_{ij} peut être interprété comme l'influence du neurone i sur le neurone j « et réciproquement ».

T est le seuil de chaque unité : $T = \begin{pmatrix} \theta_1 \\ \theta_2 \\ \vdots \\ \theta_n \end{pmatrix}$ …… (7)

- La fonction de signe est définie comme : $\begin{cases} 1 \; si \; x \geq 0 \\ -1 \; sinon \end{cases}$ ……. (8)

Nous pouvons facilement représenter un réseau Hopfield par un graphe non orienté pondéré [61] :

- ✓ Chaque unité est un sommet.
- ✓ L'arête pondérée entre chaque sommet est le poids, W est alors utile en tant que matrice adjacente.

Par exemple, « *La figure 2.14* » correspondra à :

$$X = \begin{pmatrix} x_1 \\ x_2 \\ x^3 \end{pmatrix}, W = \begin{pmatrix} 0 & -1 & 1 \\ -1 & 0 & -1 \\ 1 & -1 & 0 \end{pmatrix}, T = \begin{pmatrix} -0.5 \\ -0.5 \\ -0.5 \end{pmatrix}$$

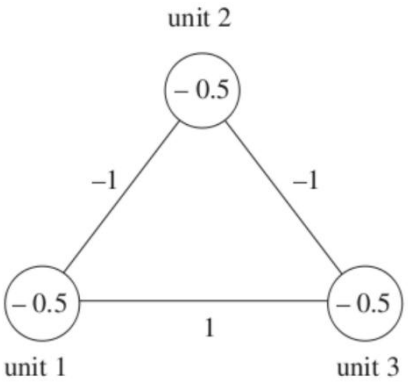

Figure 2. 14 : Un réseau Hopfield de trois unités [61] [62]

Généralement, un réseau Hopfield a une matrice de pondération symétrique, zéro diagonale « *pas de boucle, une unité n'influence pas sur elle-même* ».

4.4 Réseau de neurones récurrent « *RNR* »

Un réseau de neurones récurrent « *Recurrent Neural Network – RNN* » est un réseau de neurones dont le graphe de connexion contient au moins un cycle [63]. Contrairement aux « *MLP* », les réseaux de neurones récurrents « *Recurrent neural*

networks ou RNN », présentés dans « *la figure 2.15* », comportent des cycles au sein du graphe de neurones [64].

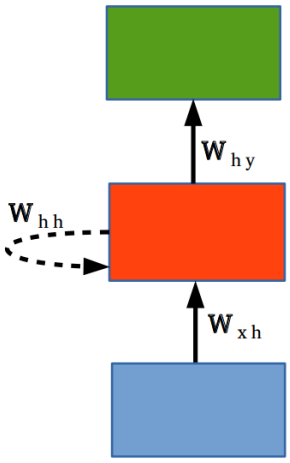

Figure 2. 15 : Représentation compacte des RNN. Toutes les flèches représentent des connexions complètes. La flèche en pointillée représente les connexions ayant un décalage temporel « t – 1 »

La motivation principale derrière ce type d'architectures est de pouvoir manipuler des séquences de vecteurs d'entrée, représentant chacun un événement temporel, et non pas seulement des données isolées n'ayant pas de signification temporelle. En déroulant, par rapport au temps, la modélisation compacte d'un « *RNN* » « *voir la figure 2.16* », ce type de réseaux peut ainsi être considéré comme une suite temporelle de réseaux « *MLP* » reliés entre eux à travers leurs couches cachées respectives. Cette liaison permet aux « *RNN* » d'encoder des dépendances latentes entre les événements d'une séquence de vecteurs d'entrée [65].

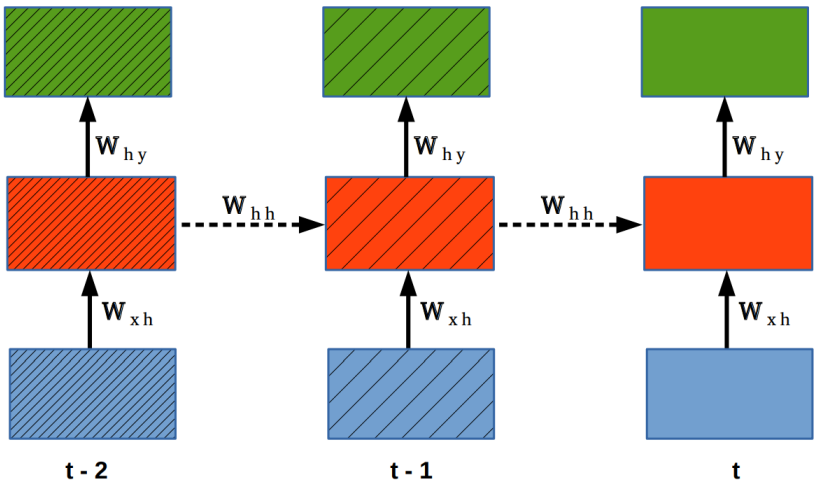

Figure 2. 16 : Représentation dépliée des RNNs

Suivant cette modélisation, un « *RNN* » prend en entrée une séquence d'événements $x = (x_1, x_2, \dots, x_t)$ et définit la séquence d'états cachés $h = (h_1, h_2, \dots, h_t)$ pour produire la séquence de vecteurs de sortie $y = (y_1, y_2, \dots, y_t)$en itérant de $t = 1$ à T [65] :

$$ht = H(W_{xh}x_t + W_{hh}h_{t-1}) \dots\dots (9)$$

$$yt = W_{hy}h_t + b_y \dots\dots (10)$$

Où T est le nombre total de vecteurs d'entrée, $W_{\alpha\beta}$ est la matrice de poids entre les couches α et β , et b_β est le vecteur de biais de la couche β . La fonction H utilisée dans le cas des « *RNNs* » est généralement la tangente hyperbolique « *tanh* ».

Étant conçu pour les réseaux non bouclés, l'algorithme de rétropropagation du gradient n'est pas suffisant pour la prise en compte des liaisons temporelles exprimées à travers les formules (9) et (10). Une solution à ce problème consiste à considérer une représentation dépliée « hiérarchisée » des « *RNNs* ». Dans la schématisation offerte dans « *la figure 2.17* », l'échelle temporelle, représentée à travers les arcs diagonaux, porte maintenant une signification hiérarchique dans le sens où les couches cibles sont de plus haut niveau. C'est à travers cette représentation hiérarchisée que nous pouvons utiliser l'algorithme de rétropropagation du gradient généralisé aux réseaux de neurones non bouclés. Cette version est appelée rétropropagation du gradient à travers le temps « *Backpropagation Through Time ou BPTT* » [65] [66].

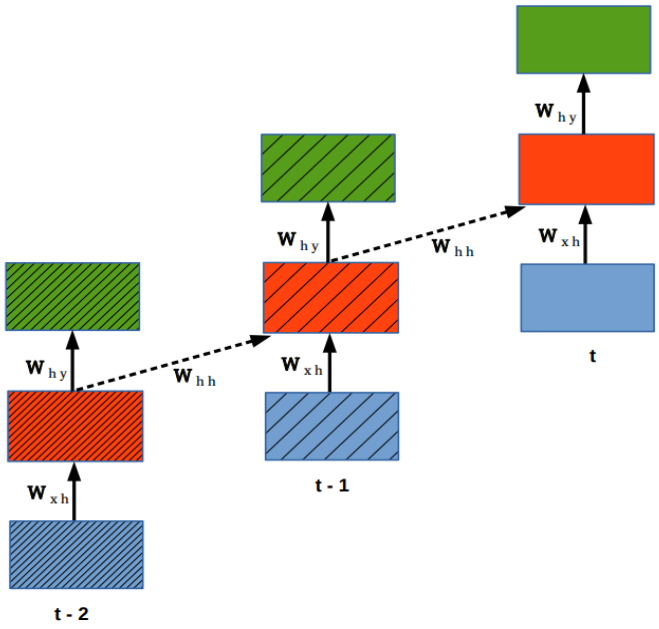

Figure 2. 17 : Représentation dépliée hiérarchisée des RNNs

4.5 Longue mémoire à court terme (LMCT)

LMCT « *Long Short-Term Memory – LSTM en Anglais* » est une architecture de réseau de neurones récurrents « *RNN*s » spécifique conçue pour modéliser des séquences temporelles [63] [67].

Le LMCT a des dépendances à longue portée qui le rendent plus précis que les RNN conventionnels. L'algorithme de rétro-propagation dans l'architecture RNN provoque un problème d'erreur de retour [65] [67]. Contrairement à « *RNNs*, LSTMs » contient des unités spéciales appelées blocs de mémoire dans la couche cachée récurrente. Les blocs de mémoire contiennent des cellules de mémoire avec des connexions automatiques, stockant l'état temporel du réseau, ainsi que des unités multiplicatives spéciales appelées portes, destinées à contrôler le flux d'informations. Chaque bloc de mémoire de l'architecture d'origine contenait trois types de portes, à savoir [67] :

- Porte d'entrée : la porte d'entrée contrôle le flux d'activations d'entrée dans la cellule de mémoire.
- Porte de sortie : la porte de sortie contrôle le flux de sortie des activations de cellules dans le reste du réseau.
- Oublier la porte : met à l'échelle l'état interne de la cellule avant de l'ajouter en tant qu'entrée à la cellule via la connexion auto-récurrente de la cellule, ce qui permet d'oublier ou de réinitialiser de manière adaptative la mémoire de la cellule.

En outre, l'architecture « *LMCT* » moderne contient des connexions de judas de ses cellules internes aux portes de la même cellule pour apprendre le minutage précis de la sortie.

Afin de faciliter l'analyse, l'architecture « *LSTM* » est souvent déployée sur une dimension time « *temps* » qui peut être représentée par le diagramme suivant « *voir la figure 2.17* » [67].

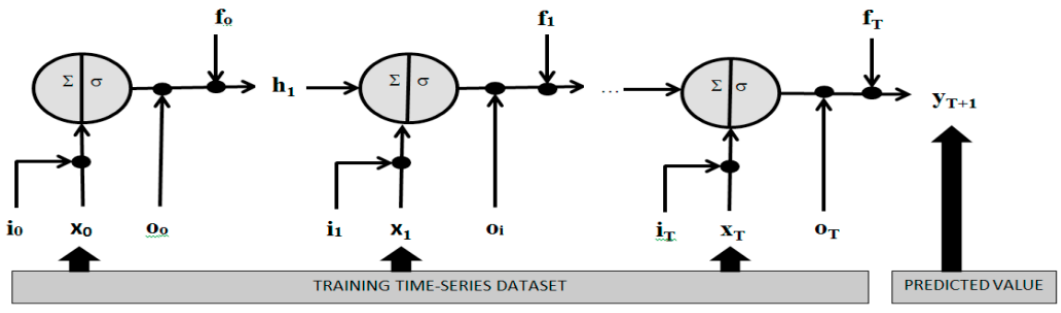

Figure 2. 18 : Architecture du modèle de mémoire à court terme et dépliée

Un réseau « *LMCT* » calcule un mappage d'une séquence d'entrée $x = (x_1, x_2, \dots, x_t)$ à une séquence de sortie $y = (y_1, y_2, \dots, y_t)$ en calculant les activations d'unité du réseau à l'aide des équations suivantes de manière itérative à partir de $t = 1$ à T comme suit [65] [67] :

$$i_t = \sigma(W_{xi}x_t + W_{hi}x_{t-1} + W_{ci}c_{t-1} + b_i) \ldots\ldots\ldots\ldots (11)$$

$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f) \ldots\ldots\ldots\ldots (12)$$

$$c_t = f_t c_{t-1} + i_t \tanh(W_{xc}x_t + W_{hc}x_{t-1} + b_c \ldots\ldots\ldots (13)$$

$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_{t-1} + b_o) \ldots\ldots\ldots\ldots (14)$$

$$h_t = o_t \tanh(c_t) \ldots\ldots\ldots\ldots\ldots\ldots (15)$$

Les « *LMCTs* » ont montré leur efficacité dans divers domaines d'application. Ils sont considérés actuellement comme l'approche état de l'art dans plusieurs tâches traitant des données séquentielles [65].

4.6 réseaux de croyance profond

Les réseaux de croyances profond « *Deep Belief Networks* » sont un type particulier de réseaux de neurones artificiels et comprennent plusieurs couches de machines de Boltzmann « *RBM* » restreintes « *voir la figure 2.19* » [69]. Les deux couches supérieures servent généralement de mémoire associative de l'entrée, ce qui permet de récupérer l'entrée dans la mémoire du réseau [68].

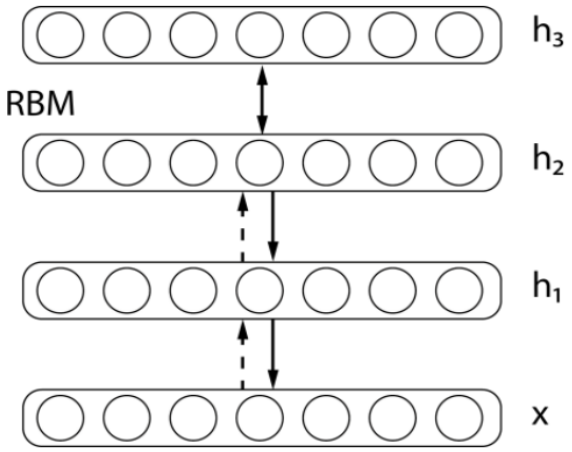

Figure 2. 19 : Réseau de croyances profondes composé de plusieurs couches de RBM

Le processus d'apprentissage de « *DBN* » diffère du processus d'apprentissage non supervisé utilisé pour la « *RBM* ». Le processus d'apprentissage comprend deux étapes :

- La première étape : le réseau extrait les entités couche par couche de manière non supervisée et la sortie d'une couche sert d'entrée à la couche suivante [70]. Ainsi, le réseau apprend à extraire des entités de celles extraites de la couche précédente. De ce fait, il est capable de détecter des corrélations d'ordre élevé dans les données en extrayant des caractéristiques plus profondes couche par couche [68].
- La deuxième étape : une fois que toutes les couches ont été pré-entraînées, les poids entre les couches simples sont ajustés de manière supervisée. Pour

l’apprentissage supervisé, un algorithme d’apprentissage par rétro-propagation est appliqué [70]. La rétro-propagation est connue pour ses faiblesses en matière de recherche de solutions globalement optimales et pour son inefficacité informatique. Cependant, son application à un réseau pré-formé accélère le processus d'apprentissage car les poids ne sont ajustés que dans l'apprentissage par rétro-propagation. Par conséquent, les inconvénients typiques ne s'appliquent pas à cette application [70].

5 Conclusion

Dans ce chapitre, nous avons présenté en particulier les concepts de base du système d'apprentissage profond, ainsi une variété que divers les systèmes de reconnaissances utilisés dans ce domaine. Nous sommes également intéressés par la mise en œuvre de « *CNNs* » dans notre expérience pour fournir une décision de classification.

Dans le chapitre suivant, nous présenterons un état de l’art sur les différentes techniques et méthodes utilisées dans le système de reconnaissance des chiffres manuscrits.

Chapitre 03

Etat de l'art

Chapitre 03 : Etat de l'art

3.1 Introduction

La reconnaissance des chiffres manuscrits fait l'objet d'un nombre important de travaux de recherche, Elle a connu ces dernières années plusieurs travaux ont récemment vu le jour.

Dans ce chapitre, nous présenterons les différents travaux de reconnaissance des chiffres manuscrits en fonction de la méthode d'extraction des caractéristiques proposé, la base de données utilisé et du résultat obtenu.

3.2 Description de différents travaux ultérieurs

Plusieurs méthodes de classification ont été développées en utilisant les différentes techniques de reconnaissances. Dans ce qui suit, nous allons présenter des travaux de différents auteurs sur la reconnaissance des chiffres manuscrits.

3.2.1 Reconnaissance des chiffres manuscrits isolés :

3.2.1.1 F. Menasri et N. Vincent. (2008) [50] :

F. Menasri et N. Vincent proposent un système de reconnaissance de chiffres farsis isolés en utilisant une autre topologie de réseaux de neurones à convolution (*CNNs*) et similaire à celle d'un LeNet5, mais pour lequel le dernier étage (*classifieur à base de combinaison de gaussiennes*) est remplacé par un perceptron multicouche à sorties softmax, qui s'appliquent également bien à la reconnaissance de chiffres indiens et ne nécessite pas de phase d'extraction de caractéristiques. Le système prend en entrées des images en niveaux de gris de taille 32x32. Le réseau est entraîné par rétropropagation du gradient d'erreur. Il apprend sa propre extraction de caractéristiques conjointement à l'apprentissage de la tâche de classification. La technique de déformations élastiques propose pour augmenter la taille de l'ensemble d'apprentissage à l'aide de données pseudo-aléatoires obtenues en déformant les données d'entraînement. L'objectif de cette méthode consiste à entraîner le modèle sur davantage de données, de telle sorte que le reconnaisseur dispose d'une couverture d'exemples plus large, de manière à mieux prendre en compte la variabilité de l'écriture. Le système proposé a été testé avec succès sur la base de données MNIST qui constitue 60.000 images d'apprentissage et 20.000 images de test, les résultats expérimentaux montrent un taux de reconnaissance de 99.02 %.

3.2.1.2 N. V. Rao et al. (2011) [51] :

Ce travail a proposé une technique d'extraction des caractéristiques pour améliorer les résultats de reconnaissance de deux chiffres de forme similaire. Cette technique repose sur trois Phases, la nouvelle approche en matière de prétraitement, d'extraction de caractéristiques et de classification. Les images d'entrées sont en niveaux de gris manuscrits isolés. Après le prétraitement (*élimination du bruit, normalisation de la taille et squelettisation*), une méthode d'extraction de

caractéristiques est appliquée pour capturer les caractéristiques les plus pertinentes du caractère en appliquant une grille à l'image et en calculant les valeurs en diagonale (*Diagonal feature extraction scheme*). En utilisant seulement ces caractéristiques, le classifieur basé sur le Réseaux de Neurones Artificiels (*RNA*) décide finalement l'appartenance de chiffre à une classe. Le taux de reconnaissance obtenus est 95.6% pour 20 et 30 nœuds cachés, sur la base de données MNIST qui contient un ensemble de 60.000 échantillons d'apprentissage et 10.000 échantillons de test.

3.2.1.3 A. Gattal et al. (2014) [52]

Ce travail a été proposé une nouvelle méthodologie basée sur la combinaison de caractéristiques pour atteindre des taux de reconnaissance élevés sur des chiffres manuscrits isolés non normalisés. Sont utilisé dix caractéristiques, sept de ces caractéristiques sont extraites directement de l'image du chiffre (*Hu's Moment Invariants, Skew Angle, Zernike Moments, Projection Histograms, Profile based features, Background features, Skeleton based features*), tandis que trois caractéristiques (*Global features, Histogram of contour chain code, Ridgelet transform*) sont extraites de différentes régions de l'image en appliquant la méthode de grille uniforme (*uniform grid*). L'idée de cette approche est de combiner ces différentes caractéristiques pour représenter au mieux les chiffres sans aucune normalisation de taille.

Le classifieur SVM proposé est basé sur une approche multiclass un-contre-tous (*one-against-all*) et évaluée sur la base de données CVL des chiffres isolés, qui contient un ensemble d'apprentissage de 21.780 chiffres et de 7.000 chiffres pour le test. Cette approche a permis d'atteindre un taux de reconnaissance estimé à 96,62%.

3.2.1.4 L. B. Saldanha et C. Bobda. (2015) [53]

Ce travail a été proposé une approche pour concevoir et implémenter un système embarqué sur FPGA (*Field Programmable Gate Arrays*), « en référence à l'importance des solutions mixtes (*matériel / logiciel*) » pour la reconnaissance des chiffres manuscrits, basé sur un réseau profond et un vaste perceptron. Les accélérateurs matériels sont utilisés pour le pipeline de traitement d'images, ce qui libère le logiciel de la lourde tâche liée au traitement séquentiel des pixels de l'image.

Afin d'augmenter les performances de l'application, une méthode de régularisation a été utilisée lors de la phase d'apprentissage du réseau de neurones artificiel (*RNA*) pour réduire considérablement les opérations en virgule flottante, grâce à une combinaison de réorganisation du code et d'accélérateurs matériels dédiés.

Ce système appliqué à la base de données MNIST qui contient 60.000 chiffres pour l'apprentissage et 10.000 chiffres pour le test (*chaque chiffre a 28 x 28 pixels en niveaux de gris et chaque pixel est représenté par 8 bits*). Les résultats de ce travail montrent que les systèmes embarqués sont déjà capables d'atteindre des performances acceptables lors de l'exécution de tâches complexes (*voir la table 3.1*).

Weight threshold	Number of weights	Misses (out of 10,000)	Prediction time (s)
0.000	266,200	335	2.938
0.001	11,565	335	1.179
0.021	2102	365	1.118
0.067	1218	497	1.099

Table 3.1 : La performance du logiciel

3.2.1.5 A. Gattal et al. (2016) [54]

Ce travail a été proposé pour améliorer l'étape d'extraction des caractéristiques dans les systèmes de reconnaissance de chiffres isolés sans aucune normalisation de la taille de l'image, en utilisant les caractéristiques de texture basée sur oBIFs (*oriented Basic Image Features*). L'auteur utilise une combinaison de (*oBIFs*) et de caractéristique de fond. Les oBIFs pour capturer les informations de texture sous forme un vecteur, tandis que les caractéristiques de fond exploitent les propriétés géométriques et topologiques des chiffres pour une représentation discriminatoire. La classification est effectuée à l'aide de la (*SVM*) multi-classes, basée sur l'approche un-contre-tous (*one-against-all*).

Les expériences ont été effectuées sur la base de données CVL (*Computer Vision Lab*). La base de données comprend 7.000 chiffres pour l'apprentissage, 21.780 pour le test et 7.000 pour la validation. Toutes les images sont binarisées à l'aide de la méthode de binarisation KittlerMet avant l'extraction de caractéristiques. La méthode proposée réalise un taux de reconnaissance de 95,21%.

3.2.1.6 H. Zhan et al. (2017) [55]

Ce travail, proposé un nouveau réseau basé sur "Recurrent Neural Network & Connectionist Temporal Classification" (*RNN-CTC*) pour la reconnaissance des chiffres manuscrits. Ce système utilisé un réseau résiduel plus efficace pour extraire des séquences de caractéristiques plus discriminantes et prédire les résultats de la reconnaissance et modifier le (*LSTM*) bidirectionnel standard en ajoutant une couche entièrement connectée (*fully connected layer*) avant de combiner les deux directions pour la convergence. Au sommet de ce réseau, une CTC standard est appliquée pour calculer la perte et obtenir les résultats finaux. Pour évaluer l'efficacité de ce modèle, l'auteur a conçu deux expériences avec deux bases de données : L'une consiste à montrer les performances de reconnaissance sur un base de données des chiffres isolé et l'autre à vérifier le potentiel du modèle pour la reconnaissance de chaînes de chiffre.

L'expérimentation a été effectué sur la base de données CVL HDS (*Computer Vision Lab Handwritten Digit String*), qui contient un ensemble de 1.262 échantillons pour l'apprentissage et 6.698 échantillons pour le test. L'autre base de données appelé ORAND-CAR, est constitué de 11.719 échantillons divisés en deux sous-ensembles, ORAND-CAR-A et ORAND-CAR-B.

L'expérimentation est effectué un taux de reconnaissance plus élevées sur ORAND-CAR-A et ORAND-CAR-B, mais cela fonctionne très mal sur la base de donne CVL HDS. (*Voir la table 3.2*)

Methods	CAR-A	CAR-B	CVL HDS	G-Captcha
Proposed	**0.8975**	**0.9114**	0.2707	**0.9515**

Table 3.2 : Taux de reconnaissance de différents modèles sur les ensembles de données.

3.2.1.7 S. R. Kulkarni et B. Rajendran. (2018) [56]

Dans ce travail, l'auteur se concentre sur l'application d'un algorithme d'apprentissage supervisé précis basé sur le problème de la reconnaissance des chiffres manuscrits en utilisant un réseau optimisé en termes de nombre de paramètres d'apprentissage selon les contraintes d'énergie et de mémoire. Dans le but de démontrer un réseau de neurone d'apprentissage SNN (*Spiking Neural Networks*) capable d'obtenir une précision et une efficacité élevées, en utilisant l'algorithme de descente normalisée approximative (*Normalized Approximate Descent-NormAD*) récemment proposé. A également former un réseau de neurones artificiels (*RNN*) équivalent avec la même architecture, pour les comparer avec le SNN proposé. Après avoir optimisé les hyper-paramètres du réseau, SNN et ANN ont été formés à l'ensemble des données MNIST (*60.000 images*) pour 20 (*epochs*). Le SNN proposé dépasse un réseau de neurones artificiels (*RNA*) non convergents équivalent, où elle a atteint un taux de reconnaissance de 98,17%, contrairement à (*ANN*) qui a atteint un taux de 98%.

3.2.1.8 J. Qiao et al (2018) [57]

Dans cet article, une stratégie (*Deep Q-learning*) adaptative est proposée pour améliorer encore la précision et réduire la durée d'exécution de la reconnaissance des chiffres manuscrits. En combinant l'apprentissage profond et Q-learning (L'apprentissage par renforcement) pour former un réseau Q-Learning adaptatif (*Q-learning Deep Belief network-Q-ADBN*). L'auteur utilise cette stratégie (*Q-ADBN*) d'une part, pour extraire les principales caractéristiques des images de chiffres manuscrites originales en utilisant la méthode auto-encodeur profond adaptatif - AEPA (*adaptive Deep auto-encoder-ADAE*), cette méthode permettre d'ajuster de manière adaptative le taux d'apprentissage, ce qui réduit le temps d'exécution. D'autre part, l'algorithme Q-learning est utilisé comme un classifieur et les caractéristiques extraites sont considérées comme les états actuels de l'algorithme Q-learning, généralement utilisé pour renforcer la phase d'apprentissage.

Le système appliqué sur la base de données MNIST, qui contient 60.000 images pour l'apprentissage et 10.000 images pour le test. Chaque image de chiffre manuscrit de taille 28 × 28 pixels et la plage de chaque point de pixel est comprise entre 0 et 1. Les résultats expérimentaux montrent que Le taux de reconnaissance atteint 99.18%.

3.2.1.9 S. M. Shamim et al. (2018) [58] :

Ce travail compare plusieurs modèles de classifieur au niveau de la décision pour la reconnaissance des chiffres manuscrites basée sur la plateforme WEKA (*Waikato Environment for Knowledge Analysis)*, qui contient un ensemble d'algorithmes et d'outils de visualisation pour la modélisation prédictive, l'analyse de données, ainsi que des interfaces utilisateur graphiques facilitant l'accès à cette fonctionnalité. Il prend en charge diverses tâches d'exploration de données standard, notamment le prétraitement, la classification, la sélection de caractéristiques. L'auteur utilise une base de données fourni par l'Institut autrichien de recherche sur l'intelligence artificielle (*Austrian Research Institute for Artificial Intelligence*).

Le système est évalué sur la base de 3.789 échantillons, 1.893 échantillons pour l'apprentissage et 1.796 échantillons pour le test. La précision globale la plus élevée de 90,37% est obtenue dans le processus de reconnaissance par Multilayer Perceptron (*neural network based classifier*).

3.2.2 Reconnaissance une chaine de chiffres manuscrits :

3.2.2.1 A. Gattal et al. (2017) [59]

Ce travail décrit un système de reconnaissance des chiffres manuscrites sans contrainte par un longueur connue, avoir la possibilité de segmenter correctement les chiffres espacés, en pente, en chevauchement et connectés sans aucune information sur la longueur de la chaîne. Le système proposé, basé sur les méthodes de segmentation explicites en combinant trois méthodes : l'histogramme de la projection verticale dédiée pour les chiffres séparés, l'analyse de contour dédiée aux chiffres en chevauchement et la transformation de Radon effectuée sur la fenêtre glissante dédiée pour les chiffres connectés, en fonction du lien de configuration entre les chiffres.

Les résultats rapportés a travers le classifier SVM sur un base de données synthétiques construit en concaténant des chiffres isolés du NIST SD19.

L'expérimentation a effectué un taux de reconnaissance de (*voir la table 3.3*).

String length	#Strings	Rec. rate when training SVMs (%)			Rec. rate when retraining SVMs (%)		
		Primary	Secondary	Overall	Primary	Secondary	Overall
2-digit	2370	99.73	79.55	93.65	99.88	80.46	94.03
3-digit	2385	99.49	85.67	93.22	99.87	86.94	94.00
4-digit	2345	98.69	86.92	93.04	99.20	88.45	94.04
5-digit	2316	97.69	88.38	92.90	98.31	89.76	93.91
6-digit	2169	98.13	87.81	93.53	98.68	90.06	94.84
10-digit	1217	97.81	86.36	92.74	98.38	88.64	94.08
Overall	–	98.59	85.78	93.18	99.05	87.38	94.15

Table 3.3 : Taux de reconnaissance obtenus.

3.2.2.2 G. Andre et al. (2018) [78] :

Ce travail décrit un système de reconnaissance une chaine des chiffres manuscrits sans contrainte par un longueur connue, de taille 2, 3 et 4 pour former des solutions d'end-to-end, en utilisant la stratégie sans segmentation (*segmentation-free*) basée sur le classifieur Convolutional Neural Networks (*CNNs*) pour prendre la décision. La technique a été basée sur la méthode bien connu le LeNet 5 pour effectuer une décision plus efficace.

Ce système appliqué sur un base de données synthétiques de tailles variables à partir de NIST SD19 (*voir la table 3.4*), Les résultats rapportés sur (*la table 3.5*).

Length/Classes	Samples	Authors	Purpose
1 (Isolated digits)	197,784	0000-2099	Training
10 classes	23,384	3850-4099	Validation
	23,621	3600-3849	Testing
2-Digit String	161,563	1000-1599	Training
100 classes	53,907	1600-1799	Validation
	55,091	1800-1999	Testing
3-Digit String	1,448,680	1000-1599	Training
1000 classes	484,346	1600-1799	Validation
	491,749	1800-1999	Testing
4-Digit String	100,000	1000-1599	Training
*	20,000	1600-1799	Validation
	20,000	1800-1999	Testing

*Data used to train the Length classifier.

Table 3.4 : Réparation des données utilisées pour l'apprentissage et le test de classifieur. Les échantillons sont uniformément repartis entre les classes.

Method	Single digit	2-digit	3-digit
End-to-end	97.68	94.09	96.05
End-to-end+$\mathcal{L}$	98.73	96.82	95.50
Hochuli et al. (2018)	99.56	99.00	94.88

Table 3.5 : Performance des approches sans segmentation sur les données synthétiques

3.2.2.3 A. G. Hochuli et al. (2018) [79] :

Dans ce travail, l'auteur postule que la segmentation manuscrite des chiffres peut être remplacée avec succès par un ensemble de classificateurs formés pour prédire la taille de la chaîne et les classer sans aucune segmentation, à partir d'une deux tâches

spécifiques, la prédiction de longueur de chaîne et la classification de chiffres. Pour ce là, l'auteur a utilisé quatre réseaux de neurones de convolution (*CNN*) pour prédire la taille de la chaîne et classer une chaine de chiffres de taille 1,2,3 chiffres. Le système est validé dans deux bases de données bien connues, La base de données synthétique et le NIST SD19.

Les résultats expérimentaux montrent que les classificateurs CNN peut traiter les cas complexes de chiffres en contact plus efficacement que tous les algorithmes de segmentation disponibles dans la littérature.

3.2.3 Compétition HDR :

Dans cette partie, nous avons présenté une compétition entre plusieurs membres dans la recherche sur le domaine de reconnaissance des chiffres manuscrits (*HDR*) :

3.2.3.1 Markus Diem et al. (2013) [80]

Ce travail présente les résultats du concurrence HDRC 2013 pour la reconnaissance des chiffres manuscrits organisé conjointement avec l'ICDAR 2013. L'objectif de cette compétition est de reconnaître les chiffres manuscrits isolés (*HDR*) et de présenter une nouvelle base de données stimulant pour la Comparaison.

Dans ce qui suit, neuf méthodes de reconnaissance des chiffres manuscrits (*HDR*) proposées par les sept participants. Deux groupes ont proposé deux systèmes différents.

3.2.3.1.1 Système de François Rabelais :

Ce système présente une technique pour l'extraction de vecteur de caractéristique basé sur la DT (*Delaunay Triangulation*) et le zonage. Le système a été appliqué par le SVM en utilisant un noyau RBF, pour la tache de classification.

3.2.3.1.2 Système de Hannover :

Après la normalisation, binarisation et la correction de l'inclinaison (*redressement*) de l'image. Trois méthodes ont été appliquées pour extrait les caractéristiques de l'image :

- Nombre de pixels noirs dans chaque ligne et colonne.
- Longueurs de 12 sondes dans différentes directions à partir de différentes positions.
- Moments centraux normalisés.

Ce système exécuté à l'aide d'un classifieur K-NN (*rnearest-k-neighbor*).

3.2.3.1.3 Système de Jadavpur :

Ce système combiner les deux techniques (*Quad-Tree-Based Longest-Run*) [81] et (*Convex-Hull-Based*) [82] comme méthode d'extraction des caractéristiques, basé sur la stratégie de (Fuzzy-Entropy-Based) avec le classifieur SVM.

3.2.3.1.4 Système de Orand :

Après le prétraitement, trois caractéristiques différentes des chiffres sont exploitées :

Premier, les orientations du trait, qui se base sur l'approche de (*HOG*) [83] ; deuxième, la relation entre le fond et la forme, qui se base sur la concavities [84] ; et finalement, le contour, qui se base sur l'approche de Zhang et Suen [85]. D'autre part, un SVM multi-classes en utilisant un noyau RBF est utilisé pour la classification.

3.2.3.1.5 Système de Orand :

Ce système présente une technique pour extrait les caractéristiques d'une image manuscrits prétraité, basé sur l'algorithme de AdaBoost.MH. Le classifieur de base étaient des (*Hamming trees*) sur des filtres de Haar [86].

3.2.3.1.6 Système de Salzburg I :

Ce système propose une méthode de classifieur (*RNA*) de FIR MLP (*Finite Impulse Response Multilayer Perceptron*) partiellement connecté avec quatre couches.

3.2.3.1.7 Système de Salzburg II :

Cette méthode est similaire à la précédente (*Salzburg I*), mais utilise un ensemble de quatre FIR MLP partiellement et entièrement connecté à quatre couches.

3.2.3.1.8 Système de Tébessa I :

L'objectif de ce système est d'améliorer les performances de reconnaissance en combinant les caractéristiques fond et les caractéristiques de la forme du squelette. La classification est effectuée à l'aide de la (*SVM*) multi-classes, basée sur l'approche un-conter-tous (*One-Against-All*).

3.2.3.1.9 Système de Tébessa II :

Ce système a été proposé une méthode qui combiner les caractéristiques classiques, transformation de l'ondelette (*wavelet*), les caractéristiques fond, les caractéristiques de forme (Contour, Squelette) et Multi-Scale Run Length Features.

Le classifieur de ce système est un multi-classes SVM basé sur l'approche un-conter-tous (*One-Against-All*).

3.2.3.2 Résultats

La table au-dessous présent les différentes méthodes qui sont participées à cette compétition (*Voir la table 3.6*).

Méthode	Normalisation	Précision %	Précision % 2éme
Salzburg II	Oui	97.74 %	99.33 %
Salzburg I	Oui	96.72 %	98.82 %
Orand	Non	95.44 %	98.58 %
Jadavpur	Non	94.75 %	-
Paris Sud	Oui	94.24 %	94.63 %
François Rabelais	Oui	91.66 %	-
Hannover	Oui	89.58 %	95.80 %
Tébessa II	Non	78.43 %	-
Tébessa I	Non	77.53 %	-

Table 3.6 : Taux de reconnaissance de diffèrent méthodes de la compétition

La compétition est appliquée à la base de données CVL qui contient 27.780 chiffres pour l'apprentissage et 7.000 chiffres pour le test (*longueur variable*). Le taux de reconnaissance de la compétition était basé sur la précision du premier et du second essai de chaque méthode. La meilleure performance pour les deux mesures de précision (*97,74% et 99,33%, respectivement*) a été réalisé par Salzburg II.

3.3 Comparaison entre les travaux ultérieurs

La reconnaissance des chiffres manuscrits (*RCM*) apparait comme un sujet de recherche toujours vivace, qui fait l'objet d'un nombre important de travaux, grâce à ses méthodes diverses et potentielles de classification qui permette de facilité la classification des formes.

Dans ce qui suit nous allons citons les principaux systèmes de classification utilisés, les différant bases de données et le taux obtenus par chaque méthode de classification comme illustré à la table suivante :

Auteur	Année	Extraction de caractéristique	Classifieur utilisée	Base de données utilisée	Taux Obtenu (%)
F. Menasri et N. Vincent	2008	/	CNNs	MNIST	99.02%
N. V. Rao et al	2011	Diagonal feature extraction scheme	RNA	MNIST	95.60%
A. Gattal et al	2014	Grille uniforme	SVM	CVL	96.62 %
L. B. Saldanha et C. Bobda	2015	/	RNA	MNIST	/
A. Gattal et al	2016	oBIFs	SVM	CVL	95.21%
H. Zhan et al	2017	/	RNN-CTC	CVL	95.15 %
S. R. Kulkarni et B. Rajendran	2018	/	SNN	MNIST	98,17%
Qiao et al	2018	AEPA	Q-learning	MNIST	99.18%
S. M. Shamim et al	2018	/	CNNs	Fourni par ARI-AI	99.37%

A. Gattal et al	2017	/	SVM	NIST SD19	/
G. Andre et al	2018	/	CNNs	NIST SD19	/
A. G. Hochuli et al	2018	/	CNNs	NIST SD19	/

Table 3.7 : Classifieurs utilisés dans les systèmes de reconnaissance des chiffres manuscrits.

3.4 Conclusion

Dans ce chapitre, nous avons fait une étude comparative de plusieurs travaux antérieurs dans le domaine de reconnaissance des chiffres manuscrites. Le but principal de cette étude est de nous aider à proposer une nouvelle approche en appliquant de nouvelles méthodes et techniques permettant un taux de reconnaissance plus élevé.

Chapitre 04

Résultats expérimentaux

Chapitre 04 : Résultats expérimentaux

4.1 Introduction

La reconnaissance des chiffres manuscrits n'est pas un sujet nouveau. Il remonte à plus d'une trentaine d'années. Ce dernier chapitre est consacré aux résultats expérimentaux pour la validation du système de reconnaissance des chiffres manuscrits isolé en utilisant l'apprentissage profond (CNNs). En commençant tout d'abord par les outils de développement adoptés ainsi que l'environnement utilisé pour l'implémentation de système et la base de données utilisé. Nous verrons ensuite montrons les protocoles effectuons pour présenter les résultats obtenus. Et enfin, nous avons comparé les résultats obtenus avec les travaux antérieurs dans le chapitre trois (section 3, table 3.6).

4.2 Outils de développement

4.2.1 Matériel

Le matériel réalisé est deux PC personnels :

- **PC 1 :**

  - Processeur Intel® Core ™ i7-4500U CPU @ 1.80GHs 2.40 GHz.
  - Mémoire (RAM) : 4.00 GB.
  - Type de système : Système d'exploitation Windows 10 Professionnel, 64 bits, processeur x64.
  - Stylet et fonction tactile : Prise en charge de la fonction tactile avec 10 points de contact.

- **PC 2 :**

  - Processeur Intel® Core ™ i3-2120U CPU @ 3.30GHs 3.30 GHz.
  - Mémoire (RAM) : 4.00 GB.
  - Type de système : Système d'exploitation Windows 7 édition intégral, service pack 1, 32 bits.

4.2.2 Environnement de développement

4.2.2.1 Matlab

Le nom MATLAB est la contraction du terme anglais « matrix laboratory », développé par la société « The MathWorks (http://www.mathworks.com) », est devenu un langage de référence pour l'analyse et la résolution de problème scientifique [87] (voir la figure 4.1). Il intégré à la fois des solutions de calcul, de visualisation et un environnement de développement [88]. Ce logiciel est spécialement conçu pour le calcul scientifique et la manipulation de vecteurs, et l'objet le plus commun dans Matlab est la matrice.

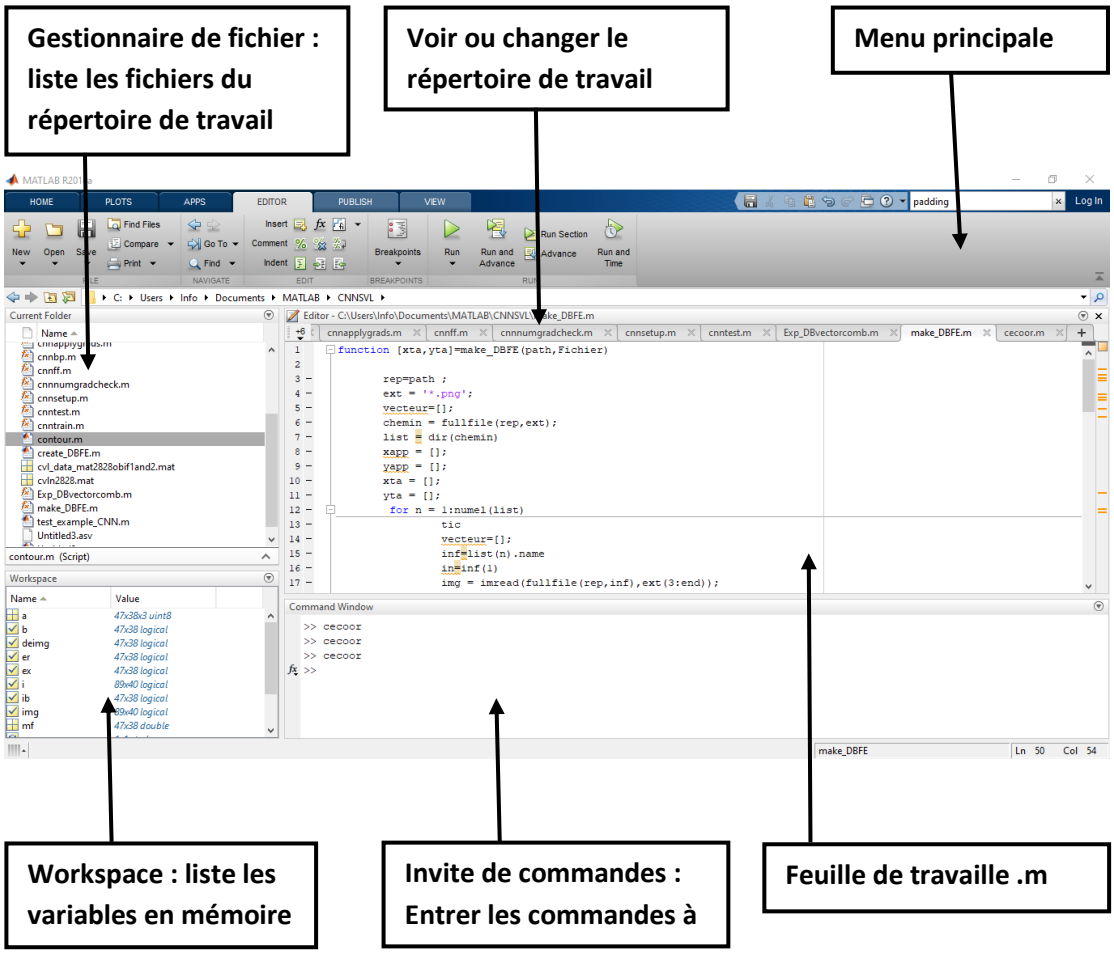

Figure 4. 1 : L'interface graphique de l'environnement MATLAB (version R2018a)

4.2.2.2 Python

Python est un langage de programmation open source et puissant, à la fois facile à apprendre et riche en possibilités, dont la première version est sortie en 1991 [89]. Il est un langage de programmation interprété (on ne passe pas par une étape de compilation avant d'exécuter son programme). Il est, en outre, très facile d'étendre les fonctionnalités existantes. Ainsi, il existe ce qu'on appelle des bibliothèques qui aident le développeur à travailler sur des projets particuliers. Plusieurs bibliothèques peuvent ainsi être installées pour, par exemple, développer des interfaces graphiques.

4.2.2.3 Google colab (Colaboratory)

Google Colaboratory ou Colab, un outil Google simple et gratuit qui ne nécessite aucune configuration et qui s'exécute entièrement dans le cloud, pour initier le Deep Learning ou collaborer sur des projets en science des données. Il permet d'écrire et d'exécuter du code, de sauvegarder et partager les analyses, et d'accéder à de puissantes ressources informatiques (voir la figure 4.2) [90].

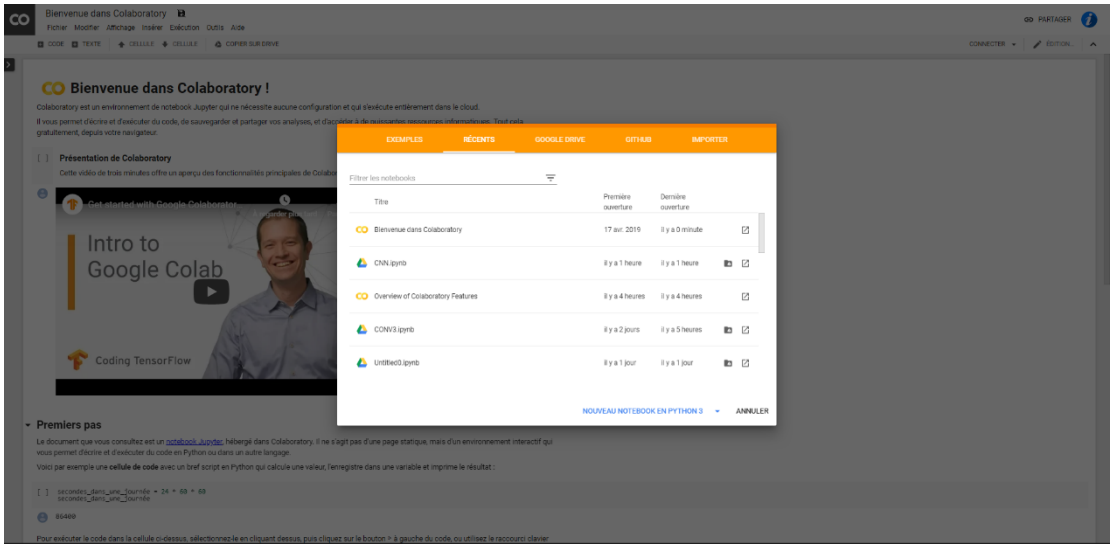

Figure 4. 2 : L'environnement de travail de google colab

- Les avantages de google colab :
  - Améliorer les compétences de codage en langage de programmation Python.
  - Développer des applications en Deep Learning en utilisant des bibliothèques Python populaires telles que Keras, TensorFlow, PyTorch et OpenCV.
  - Ne nécessite aucune configuration.
  - L'accès à un processeur graphique GPU.
  - Il fonctionne sur les serveurs Google.
  - Les documents Colab sont enregistrés directement dans le compte Google Drive.
- Les inconvénients de google colab :
  - Le taux d'exécution limité.
  - Chaque exécution prendre un résultat défirent à l'autre.

4.2.3 La bases de données utilisées

La base de données CVL (*Computer Vision Lab*) fait partie de la base de données CVL Handwritten Digit (*CVL HDdb*), qui a été collectée principalement auprès d'étudiants de l'Université de technologie de Vienne et d'une école secondaire autrichienne [80].

La base de données CVL est divisé en trois groupes : une base d'apprentissage, une base de validation et une base de test. Les images de chaque groupe ont été sélectionnées d'une manière aléatoire à partir d'un sous-ensemble de scripteurs de base de données CVL. La base de données CVL comprend 10 classes (0 - 9) avec 3 578 échantillons par classe, chacune de ces classes au même nombre d'images avec une écriture différente, y compris les variations de taille ainsi que le style d'écriture (voir la figure 1).

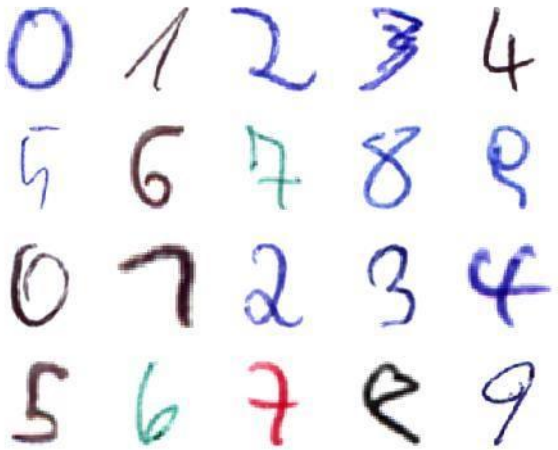

Figure 4. 3 : Exemples des chiffres de la base de données CVL.

La base de données CVL consiste en 7.000 chiffres uniques (*700 chiffres par classe*) pour la base d'apprentissage, une base de validation de taille égale (la base de validation peut être utilisé pour l'estimation et la validation de paramètres mais pas pour l'apprentissage supervisée) et la base de test qui contient 21.780 chiffres isolés (*2.178 chiffres par classes*) voir (*la Table 4.1*).

Classes	Test	Apprentissage	Validation	Totale
0	2178	700	700	3.578
1	2178	700	700	3.578
2	2178	700	700	3.578
3	2178	700	700	3.578
4	2178	700	700	3.578
5	2178	700	700	3.578
6	2178	700	700	3.578
7	2178	700	700	3.578
8	2178	700	700	3.578
9	2178	700	700	3.578
Totale	21780	7000	7000	35780

Table 4. 1 : Répartition des chiffres dans la base de données CVL.

4.3 Expérimentation

Dans cette expérimentation, les expériences sont réalisées sur la base de données CVL (*voir la section 4.2.3*). Les images de chiffres isolées sont prétraitées en trois étapes :

Premièrement, la normalisation de la taille des images effectue selon plusieurs scénarios (*taille de 20 × 20 à 48 × 48 pixels*). Deuxièmement, conversion image en (*RVB*) ou niveau de grain (Gray Scale) des pixels de gray scale en image binaire et finalement nous convertissons la matrice d'images se forme des vecteurs de taille (*n*m*1*) tel que n = m. Les vecteurs sont enregistrés dans un fichier (*.mat*) avec les étiquettes, ensuit nous utilisons ce fichier pour l'expérimentation sur Google colab.

Dans tous les protocoles nous avons utilisé la technique de gradient désigne (*Gradient Descent Optimizer*).

4.3.1 Protocole d'expérimentation 1

Comme l'illustre dans la (*figure 4.4*), nous avons conçu un réseau CNNs à deux couches de convolution pour la classification des chiffres manuscrits isolés de la base de données CVL.

Les images d'entrée étant de taille ($x = n \times m$) tel que n = m, la couche d'entrée du réseau comporte (x) neurones et la couche de sortie comporte 10 neurones (*0-9*). La couche de convolution a des filtres de taille (5×5) et de stride égal à 1. Il existe aussi deux couches de Max-pooling utilise des noyaux de taille (2×2) et des différences dans le nombre de Feature Maps (*FM*) pour chaque couche de convolution.

Pour la phase d'apprentissage, nous formons le réseau avec 100 étirassions (*epoch*) et taille de batch fixé à 7 échantillons (*batch size*).

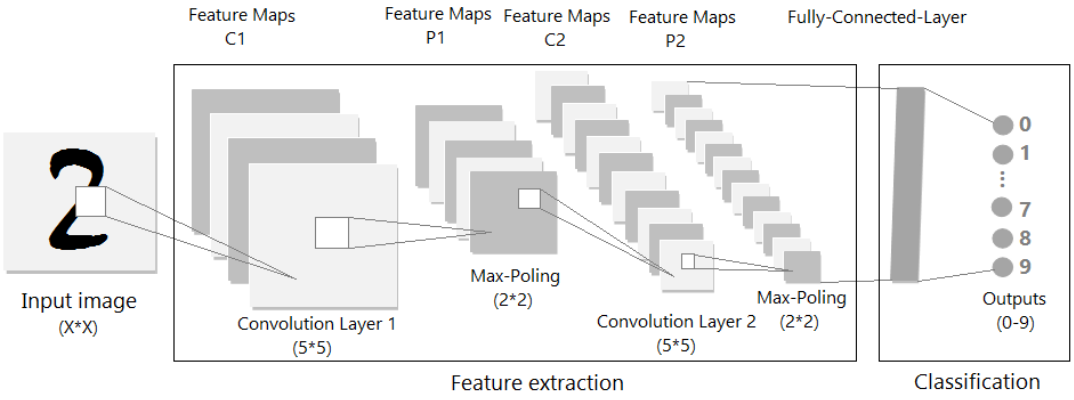

Figure 4. 4 : L'architecture de (CNNs1) proposée pour la classification des chiffres manuscrits.

4.3.1.1 les résultats expérimentaux

Dans cette expérimentation, nous avons appliqué le protocole ci-dessus à plusieurs paramètres comme nombre des filters de la premier couche (*FM1*), deuxième coche de convolution (*FM2*) avec différent normalisation des images.
Les résultats obtenus dans cette expérience illustrent dans le tableau suivant :

Taille de l'image	Feature Maps 1	Feature Maps 2	Epoch	Taux de reconnaissance
[20*20]	4	8	44	93.00%
	6	12	45	93.57%
	8	16	42	94.43%
	10	20	52	94.60%
	12	24	58	95.01%
	14	28	45	94.55%
	16	32	49	83.54%

[24*24]	4	8	43	93.78%
	6	12	31	94.42%
	8	16	20	94.25%
	10	20	5	94.43%
	12	24	43	92.53%
	14	**28**	**26**	**95.16%**
	16	32	43	92.98%
[28*28]	4	8	48	93.58%
	6	12	64	94.35%
	8	16	44	94.79%
	10	20	31	94.61%
	12	24	49	92.53%
	14	28	28	94.94%
	16	32	37	94.79%
[32*32]	4	8	45	93.83%
	6	12	49	92.00%
	8	16	25	94.66%
	10	20	26	92.12%
	12	24	42	85.63%
	14	28	26	94.61%
	16	32	30	94.68%
[36*36]	4	8	47	93.52%
	6	12	46	92.25%
	8	16	42	94.20%
	10	20	43	85.59%
	12	24	41	94.52%
	14	28	36	94.47%
	16	32	31	94.82%
[40*40]	4	8	82	91.66%
	6	12	34	94.42%
	8	16	38	94.41%
	10	20	32	94.38%
	12	24	32	94.42%
	14	28	34	94.57%
	16	32	33	94.77%
[44*44]	4	8	49	93.48%
	6	12	25	94.17%
	8	16	28	85.22%
	10	20	35	94.10%
	12	24	32	94.19%
	14	28	39	94.28%
	16	32	26	84.94%

[48*48]	4	8	32	93.20%
	6	12	29	93.95%
	8	16	40	94.24%
	10	20	32	93.92%
	12	24	21	94.12%
	14	28	21	94.45%
	16	32	33	94.21%

Table 4. 2 : Résultats obtenus dans le protocole un.

4.3.1.2 Discussion

D'après le tableau 4.2 ci-dessus, nous notons que le taux de reconnaissance le plus élevée est avec la normalisation de taille fixé à [24 x 24], fournis un meilleur taux de reconnaissance de 95.16 % avec le FM1 = 14, FM2 = 28 et nombre d'étirassions égal à 26 epoch. D'autre part le plus faible taux de reconnaissance atteint un taux de reconnaissance de 83.54% avec la normalisation [20 x 20] à FM1 = 16, FM2 = 32 et 49 étirassions (epoch). Car l'image perdit des informations intéressent avec cette taille limite.

Les autres normalisations donnent un taux de reconnaissance variable d'environ 84.94% et 94.82%.

4.3.2 Protocole d'expérimentation 2

Comme l'illustre dans « *la figure 4.5* », nous avons conçu un réseau « *CNNs* » similaire à l'expérience précédente, la différence ici est avec les paramètres de (*FM*), le nombre de couche de convolution et le Max-pooling. Nous avons utilisé trois couches de convolution pour la classification, trois couches de Max-pooling en choisissant la taille de noyau égal à « *4 x 4* ».

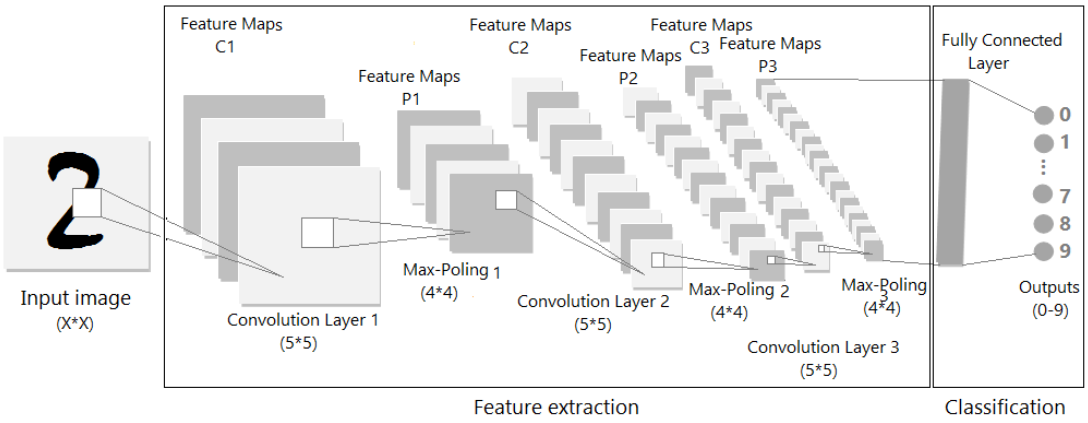

Figure 4. 5 : L'architecture de « CNN2 » proposée pour la classification des chiffres manuscrits.

4.3.2.1 les résultats expérimentaux

Dans cette expérimentation, nous avons appliqué le protocole ci-dessus à plusieurs paramètres comme les autres protocoles.

Les résultats obtenus dans cette expérience présenté dans le tableau suivant :

Taille de l'image	Feature Maps 1	Feature Maps 2	Feature Maps 3	Epoch	Taux de reconnaissance
[20-20]	4	8	16	28	93.61
	6	12	24	34	95.05
	8	16	32	33	95.28
	10	20	40	26	95.57
	12	24	48	36	95.61
	14	28	56	23	95.42
	16	32	64	44	86.69
[24*24]	4	8	16	49	95.07
	6	12	24	12	95.53
	8	16	32	47	86.19
	10	20	40	24	95.94
	12	24	48	34	96.01
	14	28	56	41	96.03
	16	32	64	23	95.59
[28*28]	4	8	16	49	95.05
	6	12	24	18	95.39
	8	16	32	18	95.88
	10	20	40	12	95.78
	12	24	48	87	95.85
	14	28	56	79	96.20
	16	**32**	**64**	**85**	**96.57**
[32*32]	4	8	16	40	95.16
	6	12	24	41	95.67
	8	16	32	49	95.60
	10	20	40	41	95.44
	12	24	48	33	95.63
	14	28	56	24	95.63
	16	32	64	30	95.95
[36*36]	4	8	16	32	95.18
	6	12	24	45	95.60
	8	16	32	24	86.19
	10	20	40	30	95.39
	12	24	48	24	95.67
	14	28	56	93	95.97
	16	32	64	47	95.95
[40*40]	4	8	16	20	94.84
	6	12	24	33	95.76
	8	16	32	30	95.55
	10	20	40	18	86.63
	12	24	48	38	95.76
	14	28	56	46	95.68
	16	32	64	17	95.81

[44*44]	4	8	16	24	85.54
	6	12	24	33	95.24
	8	16	32	24	95.26
	10	20	40	25	95.67
	12	24	48	30	95.59
	14	28	56	21	95.68
	16	32	64	21	95.44
[48*48]	4	8	16	34	95.41
	6	12	24	29	95.14
	8	16	32	26	95.31
	10	20	40	27	95.22
	12	24	48	24	95.18
	14	28	56	32	95.64
	16	32	64	39	95.72

Table 4. 3 : Les résultats obtenus dans le protocole deux.

4.3.2.2 Discussion

Sur le tableau 4.3 ci-dessus, nous notons que meilleure normalisation de taille est fixée à [*28 x 28*]. Fournis un meilleur taux de reconnaissance de 96.57% avec le FM1 = 16, FM2 = 32, FM3 = 64 avec 19 étirassions *« epoch »*. D'autre part le plus faible précision atteint un taux de reconnaissance de 85.54% avec la normalisation [*44 x 44*] à FM1 = 4, FM2 = 8, FM3 = 16 et (*24,47*) epoch.

Les autres normalisations donnent un taux de reconnaissance variable d'environ 86.19% et 96.20%.

4.3.3 Protocole d'expérimentation 3

Dans cette expérimentation, nous avons utilisé une autre technique pour normaliser la taille de l'image en gardant même structure et même contenu de chiffre par utilisation la technique de rembourrage « *Padding* ». Pour cela nous pouvant la taille le plus grande de la largeur (L) et de l'hauteur (H) sur la base de données pour spécifie l'espace de remplissage par ajoute les uns à l'écart entre l'image real et l'image de la taille « *Max (H, L) x Max (H, L)* ».

Dans cette expérimentation, nous utilisant des images rembourrées de chiffre manuscrit isolé de taille 132 x 132 pixels voir (*la figure 4.6*) et la plage de chaque point de pixel est comprise entre 0 et 1.

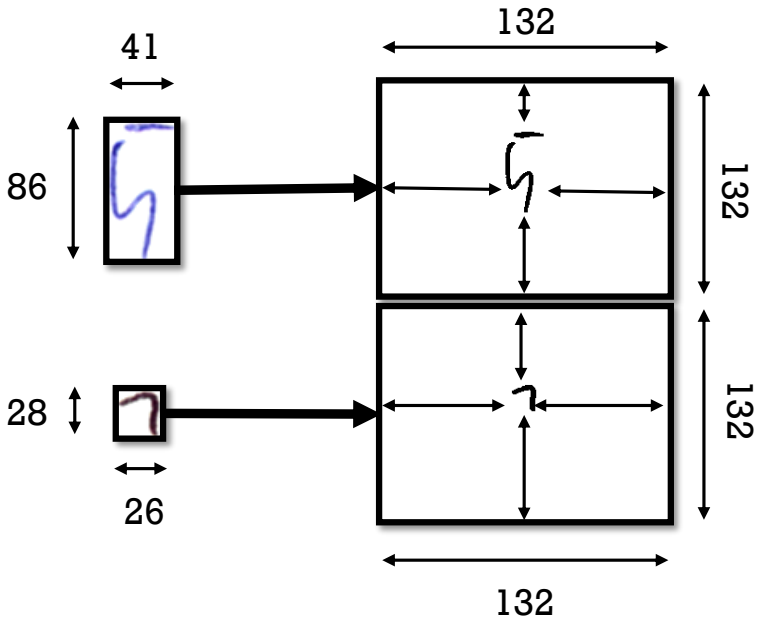

Figure 4. 6 : Exemple de normalisation la taille avec la technique « Padding »

Les résultats obtenus en utilisant les deux protocoles précédents (*voir la section 4.3.1 et 4.3.2)* sont suivant :

4.3.3.1 les résultats expérimentaux avec le protocole 1

Les résultats obtenus dans cette expérience présenté dans le tableau suivant :

Feature Maps 1	Feature Maps 2	Epoch / 3630		Taux de reconnaissance % / 3630	
4	8	37	36	92.12	92.20%
		37		91.81	
		31		93.03	
		36		91.29	
		44		92.89	
		32		92.06	
6	12	21	25	92.83	92.43%
		28		91.73	
		18		93.22	
		22		90.88	
		25		93.58	
		37		92.34	
8	16	19	25	93.14	92.63%
		36		92.47	
		33		93.19	
		24		91.29	
		19		93.33	
		20		92.36	

10	20	15	21	93.47	92.80%
		35		92.23	
		20		93.88	
		16		91.10	
		14		93.60	
		25		92.50	
12	**24**	**32**	**25**	**92.97**	**92.84%**
		27		**92.86**	
		21		**93.38**	
		22		**91.21**	
		27		**93.88**	
		23		**92.75**	
14	28	15	21	93.58	92.73%
		33		92.17	
		22		93.36	
		20		91.62	
		22		93.49	
		15		92.17	
16	32	17	20	93.14	89.47%
		36		92.25	
		10		84.35	
		16		91.07	
		20		93.11	
		19		82.89	

Table 4. 4 : Résultats obtenus avec le protocole un.

4.3.3.2 les résultats expérimentaux avec le protocole 2

Les résultats obtenus dans cette expérience peuvent être trouvés dans le tableau suivant :

Feature Maps 1	Feature Maps 2	Feature Maps 3	Epoch / 3630		Taux de reconnaissance % / 3630	
4	8	16	20	20	94.24	94.73
			28		94.76	
			14		94.43	
			19		95.06	
			29		94.95	
			15		94.98	
6	12	24	37	31	93.58	94.63
			27		94.21	
			29		94.57	
			27		94.82	
			50		95.29	
			17		95.31	

8	**16**	**32**	**28**	**22**	**95.78**	**95.50**
			35		**95.15**	
			24		**95.15**	
			18		**95.92**	
			17		**95.34**	
			14		**95.67**	
10	20	40	34	26	93.49	93.66
			35		93.30	
			25		93.55	
			13		93.66	
			36		94.07	
			15		93.91	
12	24	48	28	24	94.79	95.13
			23		95.09	
			34		95.37	
			17		95.09	
			27		95.20	
			15		95.23	
14	28	56	29	22	93.38	94.11
			18		94.24	
			16		93.82	
			24		94.10	
			28		94.40	
			21		94.76	

Table 4. 5 : Résultats obtenus avec le protocole deux.

4.3.3.3 Discussion

D'après cette étude dans les tableaux (*4.4*) et (*4.5*) ci-dessus, nous notons que les résultats obtenus par le protocole 1 est un peu plus faible par rapport aux résultats obtenus par le protocole 2.

Selon l'architecture 1 (*protocole 1*), la précision la plus élevée est de 92.84% avec les paramètres FM1 = 12, FM2 = 24 et 25 étirassions, et la plus faible de 89.47% avec FM1 = 16, FM2 = 32 et 20 étirassions. L'autre architecture (*protocole 2*) donne un meilleur taux de reconnaissance moyenne d'environ 95.50% avec FM1 = 8, FM2 = 16, FM2 = 32 et 22 étirassions.

On remarque que le taux de reconnaissance de chaque paramétrage dans le protocole 2 sont proches et les moyennes de chaque paramétrage pour le protocole 1 est très proches. Cette technique montre que le protocole 2 fournit un taux intéressant malgré les images non vraiment normalisé (*même structure et même contenu*).

4.4 Résultats & Discussion

Après avoir présenté les résultats obtenus dans les expérimentations précédentes, montré que le protocole numéro deux a été donné meilleur résultat par rapport les autres protocoles avec un taux de reconnaissance de 96.57% (*Voir la table 4.6*).

Méthode	Normalisation	Taux de reconnaissance
Protocole 1	Oui	95.16 %
Protocole 2	Oui	96.57 %
Protocole 3	Non	95.50 %

Table 4. 6 : Les résultats expérimentaux

4.5 Comparaison des résultats :

Nous avons comparé notre travail aux travaux existants dans la littérature, comme nous le montre dons (*la table 4.7*).

Notre système de classification classé la troisième position par rapport l'état de l'art.

Méthode	Normalisation	Précision %	Précision % 2éme
Salzburg II	Oui	97.74 %	99.33 %
Salzburg I	Oui	96.72 %	98.82 %
Protocole 2	**Oui**	**96.57 %**	-
Protocole 3	**Non**	**95.50 %**	-
Orand	Non	95.44 %	98.58 %
Protocole 1	**Oui**	**95.16 %**	-
Jadavpur	Non	94.75 %	-
Paris Sud	Oui	94.24 %	94.63 %
François Rabelais	Oui	91.66 %	-
Hannover	Oui	89.58 %	95.80 %
Tébessa II	Non	78.43 %	-
Tébessa I	Non	77.53 %	-

Table 4. 7 : Comparaison de notre travaille aux travaux existants dans la littérature

4.6 Conclusion

Dans ce chapitre, nous avons présentés les outils, l'environnement de développement et la base de données utilisé.

Ce chapitre a été consacré principalement à l'évaluation des protocoles proposé en utilisant la technique de l'apprentissage profond (*CNNs*) pour la reconnaissance des chiffres manuscrits isolés. Le but principal de cette évaluation est de nous aider à choisissez meilleur schéma de (*CNNs*) qui prendre le meilleur résultat.

Conclusion générale

Conclusion générale

II. Conclusion générale

Le domaine de reconnaissance des chiffres manuscrits isolés est un domaine de recherche exploré depuis plusieurs décennies et débouche sur un nombre important de travaux de recherche mais jusqu'à présent aucune solution perfective au problème de la reconnaissance des chiffres qui achève un taux de reconnaissance de 100% d'un manieur général. Pour cela le domaine de reconnaissance des chiffres isolés reste un sujet de recherche ouvert.

L'objectif de ce travail consacré à l'étude du choix de paramètres et la réalisation d'un système de reconnaissance des chiffres manuscrits isolés basé sur l'apprentissage profond qui a montré ses performances ces dernières années et nous avons choisi la méthode Convolutional Neural Network (*CNNs*) comme méthode de classification en utilisant la base de données Computer Vision Lab (*CVL*), ce choix est justifié par la simplicité et l'efficacité de la méthode.

Durant cette recherche, Nous avons traversé aux trois étapes classiques d'un système de reconnaissance, en obtenant des échantillons de la base de données (*CVL*), en passant par la phase de binarisation pour remédier aux défauts causés et améliorer leurs qualités et en finissant par la reconnaissance à partir le classifieur (*CNNs*).

Les résultats obtenus lors de la phase de test confirment l`efficacité de notre approche, car la qualité de la base de données est très réaliste avec des erreurs et des différences d'écriture d'un scripteur à un autre.

En perspective, le travail que nous avons réalisé durant ce mémoire, constitue une version initiale et un premier stade d'expérimentation et l'expérience pour la reconnaissance des chiffres manuscrits isolés basé sur l'apprentissage profond.

Malgré les résultats obtenus, un certain nombre d'amélioration de performance et des extensions peut être envisagé :

- Le prétraitement : tel que le lissage, élimination de bruit, ...etc.
- Améliorer le system par utiliser d'autre méthode de classification ou par combiner les CNNs avec d'autres ou plusieurs classifieurs.
- Tester avec d'autre bases de données en introduisant un plus grand nombre des scripteurs, pour inclure plus de variations dans les styles d'écritures, ainsi la généralisation pourra être plus performante.
- Utiliser autre critère, que la validation d'erreur, pour le choix des valeurs des hyper-paramètres.

Références Bibliographiques

Références Bibliographiques

III. Références Bibliographiques

[1] H. Swethalakshmi, Online handwritting character recognition for devanagari and tamil scripts using support vecteur machines, master of science, department of computer science and angineering indian institute of technology madras, pp. 123.124, 11-2009.

[2] Hinton, G. E., Sejnowski, T. J & Ackley, D. H, Boltzmann Machines : Constraint Satisfaction networks That learn (Tech. Rep. No. CMU-cs-84-119). Pittsburgh, PA : Carnegie-Mellon University, May-1984.

[3] Winston, P. H, Artificiul intelligence. (2nd ed) Reading, MA : Addison-Wesley, 1984.

[4] Y.B.A. Belaid, Reconnaissance des formes : Méthodes et applications, InterEditions, Paris, 1992.

[5] L. Chergui, Combinaison de classifieurs pour la reconnaissance de mots arabes manuscrits, Thèse de doctorat, Université de Mentouri, pages.24-25, 2012.

[6] É. Gabarra, la binarisation de documents vers la reconnaissance de symboles dans l'analyse de schémas électriques, Thèse de doctorat, Ecole de technologie supérieure, L'université de Pau et des pays de l'Adour côte basque école doctorale des sciences exactes et de leurs applications, pp.15-115, 4 décembre 2008.

[7] M. Zaiz Faouzi, Les Supports Vecteurs Machines (SVM) pour la reconnaissance des caractères manuscrits arabes, Université Mohamed Khider-BISKRA, pp.15 ,15/07/2010.

[8] G. E. Hinton, Boltzmann Machines, March 25, 2007.

[9] D. Abdelhakim, La reconnaissance des chiffres manuscrits par les machines à vecteurs de support (SVMs), pp. 16-17, Juin 2011.

[10] B. Moustafa, Reconnaissance Automatique des Chiffres Manuscrits, Master en Informatique, Université Abou Bakr Belkaid, pp. 30, 2 Mars 2017.

[11] A. Belaïd, Analyse et reconnaissance de documents, Cours INRIA : le Traitement électronique de Documents, Collection ADBS, 3-7 octobre, Aix-en-Provence, 2002.

[12] N. Benahmed, Optimisation de Réseaux de Neurones Pour la Reconnaissance des Chiffres Manuscrits Isolés, Sélection et Pondération des Primitives par Algorithmes Génétiques, Thèse pour l'obtention de la Maîtrise en Génie de la Pro duction Automatisée, Montréal, Mars 2002.

[13] G. Tremblay, Optimisation d'ensemble de classifieurs non paramétriques avec apprentissage par représentation partielle de l'information, Thèse de doctorat, Ecole de technologie supérieure, Université du Québec, 2004

[14] M. Zahra, Reconnaissance de l'écriture arabe manuscrite a base des machines a vecteurs de supports, diplôme de magister, Université Badji-Mokhtar-Annaba, page. 26, 2006.

[15] T. Thanh, Thai Hoang, Le choix de paramètres pour la reconnaissance des chiffres manuscrits, Diplôme de Magistère, University of Natural Sciences, HCMC, Vietnam.

[16] M. Soua, Extraction hybride et description structurelle de caractères pour une reconnaissance efficace de texte dans les documents hétérogènes scannés : Méthodes et Algorithmes parallèles, Université Paris-Est-Français, pp. 37-38,2016.

[17] C. Bahlmann, Advanced Sequence Classification Techniques Applied to Online Handwriting Recognition. Institut for Informatic : Shaker Verlag, 2005.

[18] D. Nasreddine, Combinaison de classifieurs pour la reconnaissance hors ligne des chiffres manuscrits isolés, Université de Tébessa, pages 12–13, 2012.

[19] R. Youssef, Squelettisation d'images en niveaux de gris et applications, Université paris Descartes, École doctorale ED386 de Sciences Mathématiques de Paris Centre, 26 novembre 2015.

[20] J.P. Gastellu-Etchegorry, Acquisition et Traitement D'image Numérique, Avril 2008.

[21] E. Hussain, A. Hannan, K. Kashyap. A Zoning based Feature Extraction method for Recognition of Handwritten Assamese Characters. Department of information Technology, Gauhati University, Guwahati, Assam, India, IJCST Vol. 6, Issue 2, April - June 2015

[22] S. Nemouchi, Reconnaissance de l'écriture arabe par systèmes flous, Mémoire magister, Université Badji Mokhtar, Annaba, 2010.

[23] A.K. Jain, R.P.W. Duin, J. Mao, Statistical pattern recognition : A review. IEEE Trans, on PAMI, vol. 22, no. 1, pages 4–37, 2000.

[24] B. belainine, Classification supervisée de textes courts et bruités, Université de Québec à Montréal, pages.7-10, 2017.

[25] Silva, Ribeiro, Inductive inference for large scale text classification : kernel approaches and techniques, volume 255. Springer,2009.

[26] Joachims, T. Leaming ta classify text Using suppot vector machines : Methods, theory and algorithms. Kluwer Academie Publishers, 2002.

[27] Abdelaali, Z. Saddam, Reconnaissance hors ligne des chiffres manuscrite isolé, Tros-Rivifieres, Université de Tébessa, pages. 12-15, 2015.

[28] N. Benamara, A. Belaid : « Une méthode stochastique pour la reconnaissance de l'écriture arabe imprimée », Forum de la recherche en informatique, Tunis, Tunisie, 1996.

[29] Zhu, Goldberg, Introduction to Semi-Supervised Learning. Synthesis Lectures on Artificial Intelligence and Machine Learning, 3(1), 1- 130, 2009.

[30] C.L. Liu and H. Fujisawa, Classification and Learning Methods for character recognition : advances and remaining problems, Studies in Computational Intelligence (SCI), Springer, vol. 90, pp. 139-161, 2008.

[31] Dargenton, Contribution à la segmentation et la reconnaissance de l'écriture manuscrite, Thèse de Doctorat. Institut National des Sciences Appliquées de Lyon, 174 pages. 1994

[32] C. Chatelain, Extraction de séquences numériques dans des documents manuscrits quelconques, Thèse de doctorat, Université de Rouen, France, 2006 p. 196.

[33] Y. Lei, C. S. Liu, X.Q. Ding & Q. Fu. A, Recognition Based System for Segmentation of Touching Handwritten Numeral Strings. IWFHR, pages 294–299, 2004.

[34] C. touzet, Les réseaux de neurones artificiels, Introduction au connexionnisme, Marseille, Juillet 1992.

[35] J. Anigbogu, Reconnaissance de textes imprimés mutifontes à l'aide de modèles stochastiques et métriques, thèse de doctorat, Université de Nancy I, 1992.

[36] T.M. Ha, G. Kaufmann, H. Bunke, Text localization and Handwritten recognition, Technical report, university of Berne, 1996.

[37] Y. Ramel :" Lecture automatique des partitions musicales ". Mémoire de DEA ingénierie informatique, LISPI - Equipe de Reconnaissance des Formes et Diagnostics, Université Lyon 1, France, 1993.

[38] M. Volker, E. Haikal, M. Pechwitz, Offline Handwritten Arabic Word Recognition Using HMM - a Character Based Approach without Explicit Segmentation ". Technical University of Braunschweig Institut for Communications Technology (IfN), Germany, Mars 2008.

[39] G. Abdeljalil, 2011, Segmentation automatique pour la reconnaissance numérique des chèques bancaires Algériens, Thèse de Magister, centre Universitaire de Khanchela.

[40] L. Najman, Morphologie mathématique, systèmes dynamiques et applications au traitement des images, Mémoire d'Habilitation à Diriger des Recherches en Informatique, Université de Marne-la-Vallée France, Mai 2006.

[41] R. DE, Hinton GE, Williams RJ, Learning internal representation by error propagation, In Rumelhart DE, McClelland JL (Eds) Parallel Distributed Processing : Explorations in the Microstructure of Cognition, 1 : 318–362. MIT, Cambridge, MA, (1986).

[42] L. KJ, Hinton GE, Dimensionality reduction and prior knowledge in e-set recognition, In Touretzky, DS (Ed.) Advances in Neural Information Processing Systems, 178–185. Morgan Kauffman, San Marteo, CA, 1990.

[43] B. Leibe, Understanding Convolutional Neural Networks, Springer-Verlag, Faculté de Mathématiques, Informatique et Sciences Naturelles Département d'informatique - alemania, 30-2014.

[44] J. Bouvrie, Notes on Convolutional Neural Networks, Center for Biological and Computational Learning Department of Brain and Cognitive Sciences Massachusetts Institute of Technology Cambridge, November 22, 2006.

[45] S. Hijazi, R. Kumar, C. Rowen, Using Convolutional Neural Networks for Image Recognition, Annals of Mathematical Statistics, Cadence.

[46] Ovtcharov, Kalin, Olatunji Ruwarse, Joo-Young Kim et al. Accelerating Deep Convolutional Networks Using Specialized Hardware., Microsoft Research, Feb 22, 2015.

[47] Sermanet, Pierre, Y. LeCun, Traffic Sign Recognition with Multi Scale Networks, Courant Institute of Mathematical Sciences, New York University,2011.

[48] B. Mohammed, B. Brahim, L`apprentissage profond (Deep Learning) pour la classification et la recherche d'images par le contenu, Université kasdi merbah ouargla, 2016.

[49] C. touzet, Les réseaux de neurones artificiels, Introduction au connexionnisme, Marseille, Juillet 1992.

[50] F. Menasri, Nicole Vincent, Emmanuel Augustin, Reconnaissance de chiffres farsi isolés par réseau de neurones à convolutions, October 2008.

[51] N. VenkateswaraRao, Dr. A. Srikrishna, Dr. B. RaveendraBabu, G. Rama Mohan Babu, An efficient Feature extraction and classification of Handwritten digits using neural networks, International Journal of Computer Science, Engineering and Applications (IJCSEA) Vol.1, No.5, October 2011.

[52] A. Gattal, C. Djeddi, Y. Chibani, I. Siddiqi, Improving Isolated Digit Recognition using a Combination of Multiple Features.

[53] Luca B. Saldanha, C. Bobda, An embedded system for handwritten digit recognition, Journal of Systems Architecture 61, Elsevier, Pp. 693–699, 2015.

[54] A. Gattal, C. Djeddi, Y. Chibani, I. Siddiqi, Isolated Handwritten Digit Recognition Using oBIFs and Background Features, 2016.

[55] H. Zhan, Q. Wang and Y. Lu, Handwritten Digit String Recognition by Combination of Residual Network and RNN-CTC, Springer International Publishing AG, Part VI, LNCS 10639, pp. 583–591, October 2017.

[56] S.R. Kulkarni, B. Rajendran, Spiking neural networks for handwritten digit recognition—Supervised learning and network optimization, Neural Networks 103, Elsevier Ltd,Pp. 118–127, 2018 .

[57] J. Qiao, Gongming Wang a, Wenjing Li, Min Chen c, An adaptive deep Q-learning strategy for handwritten digit recognition, Neural Networks 107, Elsevier Ltd, Pp. 61–71, 2018.

[58] S. M. Shamim, M. B. A. Miah, Angona Sarker, Masud Rana & Abdullah Al Jobair, Handwritten Digit Recognition using Machine Learning Algorithms, Global Journal of Computer Science and Technology, Volume XVIII Issue I Version I, 2018.

[59] A. Gattal, Y. Chibani, B. Hadjadji, Segmentation and recognition system for unknown-length handwritten digit strings, Springer-Verlag London, 2017.

[60] D. H. ACKLEY GEOFFREY E. HINTON, A Learning Algorithm for Boltzmann Machines, Computer Science Department Carnegie-Mellon University, pp. 147-150, 1985.

[61] E. Thierry, Hopfield Network, Th´eorie des graphes avancée, pp. 04-06, ENS Lyon.

[62] R. Rojas, Neural Networks, Springer-Verlag, pp. 141-345, Berlin 1996.

[63] M. Gregory Gelly, Réseaux de neurones récurrents pour le traitement automatique de la parole, Thèse de doctorat de l'Université Paris-Saclay préparée à l'Université Paris-Sud, pp. 16-00, 22 septembre 2017.

[64] J. L. Elman, Finding structure in time. Cognitive science 14(2), PP. 179-211, 1990.

[65] M. Bouaziz, Réseaux de neurones récurrents pour la classification de séquences dans des flux audiovisuels parallèles, Réseau de neurones [cs.NE]. Université d'Avignon, PP. 64-66, Français, 2017.

[66] D. E. Rumelhart, G. E. Hinton, & R. J. Williams, Learning internal representations by error propagation, Rapport technique, DTIC Document, 1985.

[67] A. G. Salmana, Y. Heryadib , E. Abdurahmanb , W. Supartac, Single Layer & Multi-layer Long Short-Term Memory (LSTM) Model with Intermediate Variables for Weather Forecasting, 3rd International Conference on Computer Science and Computational Intelligence, PP.91-93, 2018.

[68] G. E. Hinton, S. Osindero, Y. W. Teh, A Fast Learning Algorithm for Deep Belief Nets. Neural Computation. 2006 ; 18(7):1527–1554.

[69] O. Fink, E. Zio, U. Weidmann. Development and Application of Deep Belief Networks for Predicting Railway Operation Disruptions, International Journal of Performability Engineering, 11 (2), pp.122-123, 2015.

[70] Ackley, D. H., G. E. Hinton, and T. J. Sejnowski, A Learning Algorithm for Boltzmann Machines, Cognitive Science, 1985; 9(1):147–169.

[71] M. Browne, Saeed Shiry Ghidary, Norbert Michael Mayer, Convolutional Neural Networks for Image Processing with Applications in Mobile Robotics, December 2007.

[72] Rosenblatt, Frank. x. Principles of Neurodynamics : Perceptrons and the Theory of Brain Mechanisms. Spartan Books, Washington DC, 1961

[73] Rumelhart, David E., Geoffrey E. Hinton, and R. J. Williams. "Learning Internal Representations by Error Propagation". David E. Rumelhart, James L. McClelland, and the PDP research group. (editors), Parallel distributed processing: Explorations in the microstructure of cognition, Volume 1: Foundation. MIT Press, 1986.

[74] Cybenko, G. 1989. Approximation by superpositions of a sigmoidal function Mathematics of Control, Signals, and Systems, 2(4), 303–314.

[75] Glorot, Xavier and Bengio, Yoshua."Understanding the difficulty of training deep feedforward neural networks". In Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS'10). Society for Artificial Intelligence and Statistics. 2010.

[76] pierre-luc, Dérivation fonction sigmoïde, https://urlz.fr/9EQz, 15-01-2019.

[77] F. Simon, Deep Learning, les fonctions d'activation , https://urlz.fr/9EQw, 15-01-2019.

[78] A. G. Hochuli, Luiz S. Oliveira, Alceu, Robert Sabourin, Segmentation-Free Approaches for Handwritten Numeral String Recognition, arXiv :1804.09279v3 [cs.CV], 28 April 2018.

[79] A. G. Hochuli, L. S. Oliveira, A. S. Britto Jr and R. Sabourin, Handwritten Digit Segmentation : Is it still necessary ?, Article in Pattern Recognition, June 2018.

[80] M. Diem, S. Fiel, A. Garz, M. Keglevic, F. Kleber and R. Sablatnig, ICDAR 2013 Competition on Handwritten Digit Recognition (HDRC 2013), In Proc. of the 12th Int. Conference on Document Analysis and Recognition (ICDAR), pp. 1454-1459, 2013.

[81] N. Das, J. M. Reddy, R. Sarkar, S. Basu, M. Kundu, M. Nasipuri, and D. K. Basu, "A Statistical–Topological Feature Combination for Recognition of Handwritten Numerals," Applied Soft Computing, vol. 12, no. 8, pp. 2486–2495, 2012.

[82] N. Das, S. Pramanik, S. Basu, P. Saha, R. Sarkar, M. Kundu, and M. Nasipuri, "Recognition of Handwritten Bangla Basic Characters and Digits using Convex Hull based Feature Set," in International Conference on Artificial Intelligence and Pattern Recognition, 2009, pp. 380–386.

[83] N. Dalal and B. Triggs, "Histograms of Oriented Gradients for Human Detection," in Conference on Computer Vision and Pattern Recognition, vol. 1, 2005, pp. 886–893.

[84] L. Oliveira, R. Sabourin, F. Bortolozzi, and C. Suen, "Automatic Recognition of Handwritten Numerical Strings: A Recognition and Verification Strategy," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 24, no. 11, pp. 1438–1454, 2002.

[85] T. Y. Zhang and C. Y. Suen, "A Fast Parallel Algorithm for Thinning Digital Patterns," Communications of the ACM, vol. 27, no. 3, pp. 236–239, 1984.

[86] M. Jones and P. Viola, "Fast Multi-View Face Detection," Mitsubishi Electric Research Lab TR-20003-96, vol. 3, 2003.

[87] L. Ott, Initiation MATLAB 1, SCALab.

[88] D. Jean-Luc, Introduction à MATLAB, Université de Picardie.

[89] V. le Goff, Apprenez à programmer en Python, Open Class rools, Vol 2, 24/06/2014.

[90] Google Colab, Page Officiel de google colab, https://colab.research.google.com, 02/05/2019.

ANNEXES

IV. Annexes

IV.1 Code sources :

```
[10] from google.colab import drive
     drive.mount('/content/gdrive')
     import scipy.io as scio

     input_data= scio.loadmat('/content/gdrive/My Drive/Colab Notebooks/cvl/cnn2et3couche/cvln2828b1.mat')
```

```
[12] import tensorflow as tf
     import numpy as np
     import time
     sess = tf.InteractiveSession()
     # test
     x_train = input_data['test_x'].astype('float32')/255
     y_train = input_data['test_y'].astype('float32')
     x_test = input_data['train_x'].astype('float32')/255
     y_test = input_data['train_y'].astype('float32')
     #Analyze the Data
     #test the shape of the train and test
     print('x_train shape:', x_train.shape)
     print('x_test shape: ', x_test.shape)
     print('ytrain samples:', y_train.shape)
     print('ytest samples: ', y_test.shape)
```

```
# declaration
def weight_variable(shape):
  #This specifies the weights for network.
  #randomaize
    initial = tf.truncated_normal(shape, stddev=0.03, seed=1)
    return tf.Variable(initial)

def bias_variable(shape):
  #This defines the bias elements for network.
    initial = tf.constant(0.03, shape=shape)
    return tf.Variable(initial)

def conv2d(x, w):
#The first and last stride must always be 1,
#because the first is for the image-number and the last is for the input-channel
#input x, weights W, bias b and strides.
    return tf.nn.conv2d(x, w, strides=[1, 1, 1, 1], padding='SAME')

def max_pool_2x2(x):
#SAME PADDING to access the boundary pixels by adding zeros at the boundaries of img
    return tf.nn.max_pool(x, ksize=[1, 2, 2, 1], strides=[1, 2, 2, 1], padding='SAME')
```

```
[14] imgsize=28;

     # input&output
     xs = tf.placeholder(tf.float32, [None, imgsize*imgsize])
     ys = tf.placeholder(tf.float32, [None, 10])
     # receive dimension as float at the time when you will feed in the data to them.
     keep_prob = tf.placeholder(tf.float32)
     #-1 means that it will infer = Tastantj the first dimension on its own.
     x_image = tf.reshape(xs, [-1, imgsize, imgsize, 1])
```

```
[15]  # CNN structure
      Featuremaps1=14;
      Featuremaps2=28;
      kk = 7; #filter size
      # CNN structure over-fitting
      # Feature maps =16
      W_conv1 = weight_variable([5, 5, 1, Featuremaps1])
      b_conv1 = bias_variable([Featuremaps1])
      #activation function Rectified Linear Unit
      h_conv1 = tf.nn.relu(conv2d(x_image, W_conv1) + b_conv1)
      h_pool1 = max_pool_2x2(h_conv1)
      W_conv2 = weight_variable([5, 5, Featuremaps1, Featuremaps2])
      b_conv2 = bias_variable([Featuremaps2])
      h_conv2 = tf.nn.relu(conv2d(h_pool1, W_conv2) + b_conv2)
      #7 * 7 to feed this as input to the fully connected layer
      h_pool2 = max_pool_2x2(h_conv2)
      # Fully connected layer
      # Reshape conv2 output to fit fully connected layer input
      h_pool2_flat = tf.reshape(h_pool2, [-1, kk*kk*Featuremaps2])
      #two dictionaries,  weight and bias parameter.
      W_fc1 = weight_variable([kk*kk*Featuremaps2, 10])
      b_fc1 = bias_variable([10])
      # Output, class prediction
      # finally we multiply the fully connected layer with the weights and add a bias term.
      #connect all the neurons of fc1 in the output layer with 10 neurons
      y_conv = tf.nn.relu(tf.matmul(h_pool2_flat, W_fc1) + b_fc1)
```

```
# train on minibatches
batch_size=7 #will be divided in a fixed batch size
numbatches=(7000//batch_size)
num_epochs=100 #the number of times you train your network.

imgsize = 28;
tab = [];
di = dict();

kk=np.random.RandomState(seed=1).permutation(7000)

# Loop over number of epochs
for epoch in range(num_epochs):
  start_time = time.time()
  jj=0
  for i in range(numbatches):
    jj=jj+batch_size;
    ii= kk[batch_size*i:jj]
    if i % 100 == 0:
       #input the images based on the batch size x and y
      train_accuracy = accuracy.eval(feed_dict={xs: x_train[ii].reshape(batch_size, imgsize*imgsize), ys: y_train[ii].reshape(batch_size, 10)})
      print("step %d, training accuracy %g" % (i, train_accuracy))
      # Calculate batch loss and accuracy
    sess.run(train_step, feed_dict={xs: x_train[ii].reshape(batch_size, imgsize*imgsize), ys: y_train[ii].reshape(batch_size, 10)})
  taux = sess.run(accuracy, feed_dict={xs: x_test, ys: y_test})
  tab.append(taux)
  di[taux] = epoch+1;
  print(taux)
  end_time = time.time()
  print("Epoch "+str(epoch+1)+" completed : Time usage "+str(int(end_time-start_time))+" seconds")
print("MAX :")
mmmm=max(zip(di.keys(), di.values()))
print(mmmm)
```

```
[16]  # loss function
      #a factor that is multiplied with the weights
      cross_entropy = tf.nn.l2_loss(ys - y_conv)
      train_step = tf.train.GradientDescentOptimizer(0.002).minimize(cross_entropy)

      #to keep track of the performance of the model.
      #check the index of img with labelled image.
      correct_prediction = tf.equal(tf.argmax(y_conv, 1), tf.argmax(ys, 1))
      #calculate accuracy
      accuracy = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))
      # Initializing the variables the w and bias
      sess.run(tf.initialize_all_variables())
```