MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE KASDI MERBAH OUARGLA

Faculté des nouvelles technologies de l'information et de la communication

Département d'électronique et communications

MEMOIRE DE FIN D'ETUDE

En Vue De L'obtention Du Diplôme master académique en Sciences et technologie
Spécialité : Electronique
Option : Contrôle industriel

THEME

La reconnaissance des caractères arabes manuscrits par les réseaux des neurones convolutionnels

Présenté et soutenu publiquement par :

LAKHDARI Ahmed Toufik et ABBACI Salih

Le :23/05/2017

Composition du jury :

Président : DR. DJALAL Adel	MAA	UKM Ouargla
Examinateurs : LAKHAL brahim	MAA	UKM Ouargla
Encadreur : M. BENCHABANE Abderrazak	MC(B)	UKM Ouargla

Année universitaire : 2016/2017

Remerciements

Nous remercions dieu le tout puissant de nous avoir aidé et éclairé le chemin pour la réalisation de ce mémoire.

Nous voudrions adresser toute nos gratitudes à l'encadreur de ce mémoire, Dr Abderrazak BENCHABANE, pour sa patience, sa disponibilité et surtout ses judicieux conseils, qui ont contribué à alimenter notre réflexion et ses encouragements lors de la réalisation de notre mémoire.

Nos sincères remerciements s'adressent à Mesdames et Messieurs les Membres du Jury qui me font l'honneur de juger ce mémoire.

J'adresse toute ma gratitude à tous mes ami(e)s et à toutes les personnes qui m'ont aidé dans la réalisation de ce travail.

Table des matières

Chapitre 2 : Les réseaux des neurones convolutionnels

Chapitre 3 : Application des réseaux convolutionnels a la reconnaissance

Conclusion générale

Liste des figures

Liste des tableaux

Liste des abréviations

1D	: Une dimension.
2D	: Deux dimensions.
MCR	: Reconnaissance des caractères magnétiques.
MNIST	: La base des données de l'Institut national de normes et de technologie modifiée.
OCR	: Reconnaissance optique des caractères.
PC	: Ordinateur personnel.
PMC	: Le perceptron multi couche.
ReLU	: Rectified Linear Units.
RN	: Le réseau de neurone.
RNA	: Le réseau de neurone artificiel.
RNC	: Le réseau de neurones convolutionnel.
XOR	: La fonction OU exclusif.
RAM	: random acces memory.

Introduction générale

Nous vivons dans un monde numérique, où les informations sont stockées, traitées, indexées et recherchées par des systèmes informatiques, ce qui rend leur récupération une tâche rapide et pas cher. Au cours des dix dernières années, des progrès considérables ont été réalisés dans le domaine de la reconnaissance de l'écriture manuscrite. Ce progrès est dû aux nombreux travaux dans ce domaine et à la disponibilité des bases des données internationales sur les normes pour l'écriture manuscrite qui ont permis aux chercheurs de signaler de manière crédible l'exécution de leurs approches dans ce domaine, avec la possibilité de les comparer à d'autres approches qui Ils utilisent les mêmes bases. La langue arabe n'était pas si chanceuse, contrairement au latin, elle est encore au niveau de la recherche et de l'expérimentation, c'est-à-dire que le problème demeure un défi ouvert pour les chercheurs

Donc la reconnaissance automatique des caractères arabes est une très jeune discipline de recherche avec des problèmes très difficiles et importants. En effet, avec l'air de l'Internet, du multimédia, la reconnaissance de l'arabe est utile de contribuer comme ses disciplines proches, la reconnaissance de l'écriture latine, la reconnaissance vocale et de traitement de vision, dans les applications actuelles autour des bibliothèques numériques, la sécurité des documents et dans le traitement des données numériques en général.

Le script arabe est cursif par nature, il pose des nombreux problèmes pour les systèmes automatiques de reconnaissance de l'écriture manuscrite. Le problème le plus difficile dans la conception d'un système de reconnaissance de l'écriture manuscrite est la segmentation des mots manuscrits pour leur reconnaissance, ce qui nécessite beaucoup de temps et de calcul. D'autre part, les informations locales sont quelque peu négligées dans les systèmes basés sur une analyse globale qui peut réduire considérablement les performances.

Pour remédier à ces problèmes, des approches de réseau neuronal ont été proposées pour la reconnaissance des caractères arabes manuscrits. Un tel système nécessite la prise en compte d'un grand nombre des variabilités des caractères, est récemment a été utilisé les réseaux des neurones convolutionnels comme un extracteur de vecteur de caractéristique efficace. Bien qu'un tel réseau puisse être utilisé comme un cadre unifié pour l'extraction et la classification des caractéristiques, il est plus efficace en tant qu'extracteur des caractéristiques

que comme classificateur. Et maintenant Les réseaux des neurones convolutionnels sont parmi les architectures les plus appropriées pour le domaine de reconnaissance.

Ce mémoire se compose de trois chapitres. Le premier chapitre fournit une généralité sur la reconnaissance de l'écriture manuscrite. Le deuxième chapitre fournit l'arrière-plan théorique pour les réseaux des neurones artificiels étudiés. Le dernier chapitre présente une application des réseaux convolutionnels a la reconnaissance des chiffres et caractères arabes manuscrits.

Dans le premier chapitre nous présenterons rapidement les grandes étapes qui composent une chaine de reconnaissance de l'écriture manuscrite. Ici nous intéresserons plus particulièrement à l'écriture arabe et aux travaux effectués dans le domaine de la reconnaissance automatique.

Dans le deuxième chapitre nous commencerons par la théorie derrière les réseaux des neurones artificiels explique la structure d'un neurone comme élément fondamental du réseau, puis nous discuterons les algorithmes des réseaux multi couche et les réseaux convolutionnels pour la classification des caractéristiques.

Dans le troisième chapitre nous utiliserons une application de réseau convolutionnel pour la reconnaissance des chiffres et des caractères arabes manuscrits puis nous discuterons les résultats obtenus après chaque simulation.

Enfin nous terminerons ce mémoire par une conclusion et des perspectives sur les travaux futurs dans ce domaine de recherche.

Chapitre 1

Généralités sur la reconnaissance de l'écriture manuscrite

1.1 Introduction :

La reconnaissance de l'écriture manuscrite est une voie traditionnelle qui a attiré l'attention des chercheurs depuis plusieurs décennies et reste un domaine de recherche très ouvert dû à son grand nombre d'applications pratiques. Aujourd'hui, les avancées réalisées dans ce domaine se concrétisent par un certain nombre d'applications comme la lecture automatique des chèques bancaires, d'adresses postales ou les adresses des formulaires. C'est un domaine très actif, d'autant plus que les interfaces homme-machine s'orientent de plus en plus vers la communication manuscrite comme les tablettes, PC, les applications orientées stylo …etc.

La reconnaissance de l'écriture manuscrite est la transcription des données manuscrites au format numérique et puis de retrouver sa signification. Cette tâche requiert toujours une étape de base dite la segmentation, c-à-d la séparation des graphèmes, des mots ou des chaînes numériques. Le problème de la segmentation des caractères manuscrits constitue le défi principal de la reconnaissance de l'écriture manuscrite ainsi plus que la segmentation est correcte plus que le système de reconnaissance est meilleur.

L'une des applications particulière de la reconnaissance est la reconnaissance du montant chiffre sur chèques bancaires du fait de la présence des bruits, le chevauchement des chiffres et même la fragmentation d'un chiffre en plusieurs parties difficilement identifiables.

1.2 Reconnaissance en ligne et hors ligne :

1.2.1 Systèmes en ligne :

La reconnaissance des caractères en ligne, dits aussi dynamiques, se réfère au processus de reconnaissance de l'écriture manuscrite enregistrée avec un numériseur en tant que séquence temporelle des coordonnées du stylo. En cas de reconnaissance en ligne des caractères manuscrits, l'écriture manuscrite est saisie et stockée sous forme numérique via différents moyens comme les montrent la Figure 1-01, Habituellement, un stylo spécial est utilisé en conjonction avec une surface électronique. Lorsque la plume se déplace sur la surface, les coordonnées bidimensionnelles des points successifs sont représentées en fonction du temps et sont stockés dans l'ordre.

Il est généralement admis que la méthode en ligne de reconnaissance du texte manuscrit a obtenu des meilleurs résultats que son homologue hors ligne. Cela peut être attribué au fait que plus d'informations peuvent être saisies dans le cas en ligne tels que la direction, la vitesse et l'ordre des coups de l'écriture manuscrite.

Figure 1-01 : Différentes moyens des systèmes en ligne.

Les outils de la Figure 1-01 ci-dessus sont des micro-ordinateurs munis des logiciels commercialisés de reconnaissance de l'écriture en ligne et un matériel d'acquisition beaucoup plus efficace permettant la saisie des mots [1].

Aujourd'hui d'autres applications se sont développées à partir de ces assistants personnels liées au domaine de la médecine, management, marketing, l'éducation, l'industrie et la gestion.

1.2.2 Systèmes hors ligne :

La reconnaissance d'écriture manuscrite hors ligne se rapporte au processus de reconnaissance des mots qui ont été numérisés à partir d'une surface (telle qu'une feuille de papier) et sont stockés numériquement au format d'échelle de gris. Après avoir été stocké, il est classique d'effectuer un traitement ultérieur pour permettre une reconnaissance supérieure.

La reconnaissance des caractères hors ligne peut être regroupée en deux types:

- Reconnaissance des caractères magnétiques (MCR)
- Reconnaissance optique des caractères (OCR)

Dans MCR, les caractères sont imprimés avec de l'encre magnétique. Le dispositif de lecture peut reconnaître les caractères selon le champ magnétique unique de chaque caractère. MCR est principalement utilisé dans les banques pour l'authentification par chèque. OCR traite de la reconnaissance des caractères d'acquisition par des moyens optiques, généralement un scanner ou une caméra. Les caractères sont sous forme d'images pixellisées et peuvent être imprimés ou manuscrits, de n'importe quelle taille, forme ou orientation.

L'OCR peut être subdivisé en reconnaissance des caractères manuscrite et en reconnaissance des caractères imprimés. La reconnaissance des caractères manuscrits est plus difficile à mettre en œuvre que la reconnaissance des caractères imprimée en raison des divers styles d'écriture humaine et les coutumes. Dans la reconnaissance des caractères imprimés, les

images à traiter sont dans les formes des polices standards comme Times New Roman, Arial, Courier, etc.

1.2.3 Comparaison :

s.n	Comparaison	caractères en ligne	caractères hors ligne
1	Disponibilité ou pas du coups de stylo	Oui	Non
2	exigences des données brutes	Échantillons	points
3	Mode d'écriture	Utilisation de stylo numérique	Document papier
4	Taux de reconnaissance	Plus élevé	plus bas
5	précision	Plus élevé	plus bas

Tableau 1-1 : Comparaison de la reconnaissance des caractères en ligne et hors ligne.

1.3 Formes du texte :

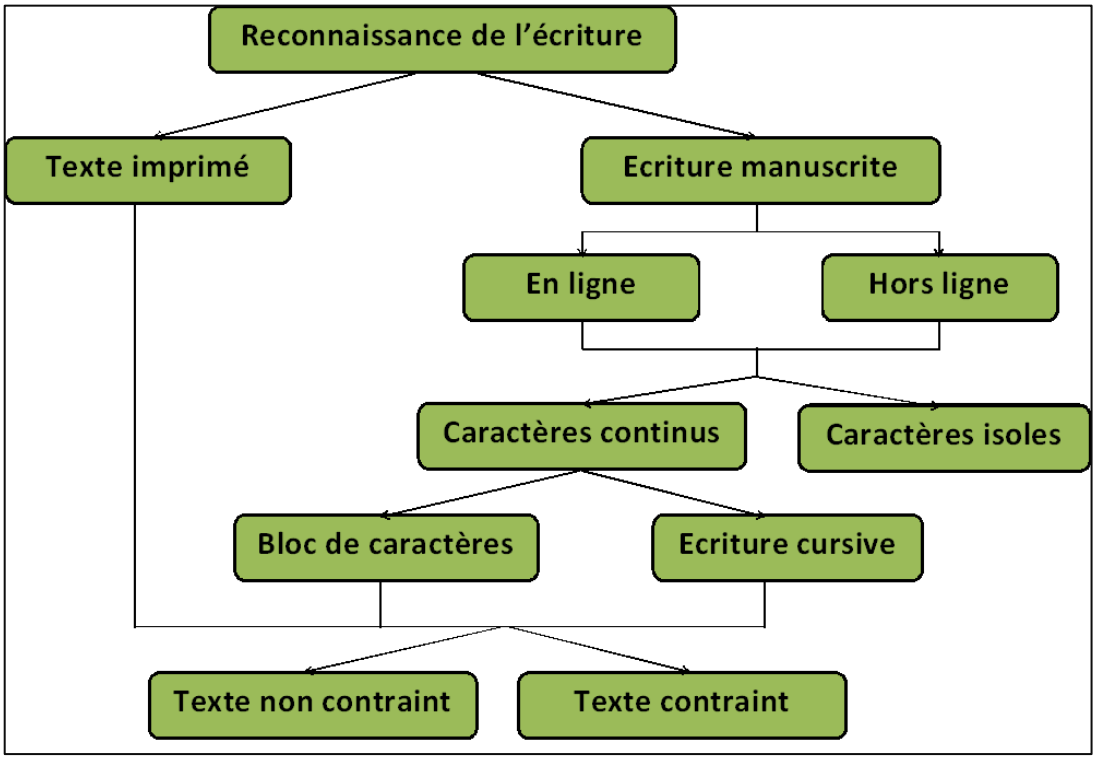


Figure 1-02 : Les différentes formes du texte manuscrit [2].

Il existe plusieurs formes du texte manuscrit : caractères isolés ou continus, texte cursif ou non cursif et texte contraint ou non contraint que nous allons présenter. La reconnaissance des caractères isolés est simple ; elle procède par la classification des caractères isolés sans avoir besoin d'une étape de segmentation alors que le texte continu nécessite de segmenter l'image d'entrée avant la phase de la classification. L'entrée du système de reconnaissance d'un texte imprimé est toujours considérée isolée. Un cas

particulier du texte continu est la reconnaissance de l'écriture cursive. La segmentation dans ce cas est beaucoup plus difficile. L'écriture Non cursive est également connue en tant que caractères isolés [2].

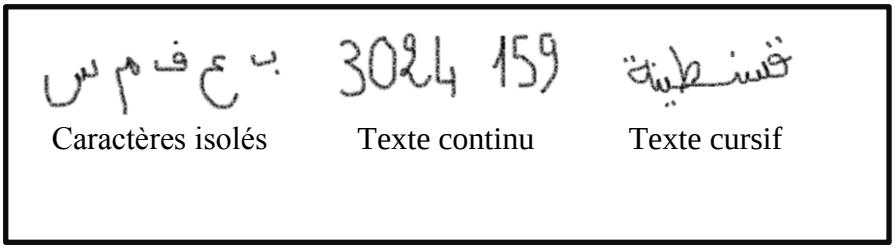


Figure 1-03 : Les différents types d'écritures.

Un texte imprimé ou manuscrit peut être également contraint ou non contraint. Le texte contraint inclut des positions où l'utilisateur est conseillé d'écrire en remplissant des emplacements prédéfinis. Ainsi, le système a des informations a priori de ce qu'il va reconnaître et où se trouve le texte. Dans le cas du texte non contraint, l'utilisateur a une page blanche qu'elle peut être complétée par n'importe quelle information et dans n'importe quelle direction. Dans la pratique, un texte constitué de plus d'une ligne sans contrainte d'entrée ni le nombre des mots dans chaque ligne est considéré non contraint [2]. Des exemples des textes contraints et non contraint sont montrés dans la figure suivante :

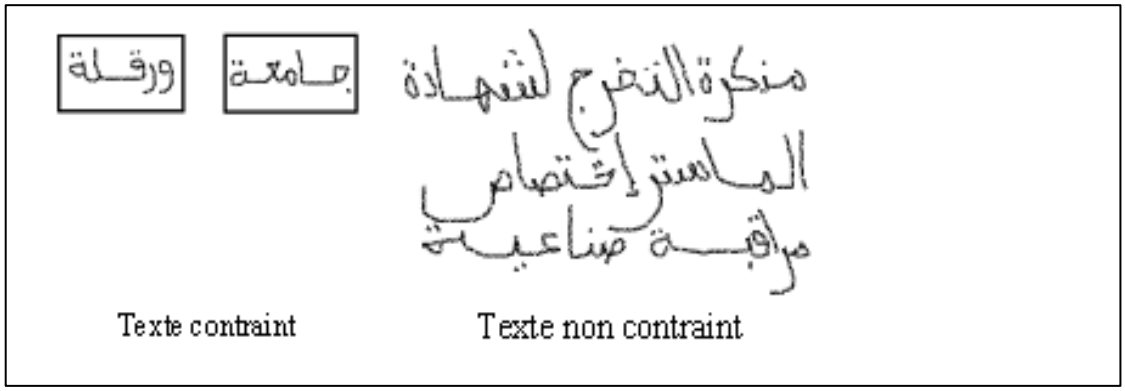


Figure 1-04 : Exemple du texte contraint et non contraint.

De plus, chaque langue possède ses propres caractéristiques telles que le nombre de lettres et l'écriture de ces lettres dans un mot.

Les travaux sur la reconnaissance de l'écriture manuscrite peuvent être distingués en quatre catégories, selon si l'unité de reconnaissance est un caractère, un mot, une phrase, ou encore une partie d'un texte [3], comme nous allons montrer par la suite.

1.4 Difficultés spécifiques au texte arabe :

Le texte en arabe est écrit de droite à gauche et se compose de 29 caractères (vingt-huit si l'on exclut la hamza, qui se comporte, soit comme une lettre à part entière, soit comme un diacritique), sans majuscules ou minuscules d'une manière cursive. Une lettre arabe peut avoir jusqu'à quatre formes différentes de base selon la position de la lettre dans le mot, comme le montre le tableau 1-2. La première colonne donne le numéro du caractère, la deuxième représente le signe d'un caractère isolé, le troisième étant Son apparition au début du mot, le quatrième et cinquième colonne représentent leur apparition au milieu et à la fin du mot, respectivement.

N	Isole	Début	Milieu	Fin	N	Isole	Début	Milieu	Fin
1	ا	-	-	ـا	15	ض	ضـ	ـضـ	ـض
2	ب	بـ	ـبـ	ـب	16	ط	طـ	ـطـ	ـط
3	ت	تـ	ـتـ	ـت	17	ظ	ظـ	ـظـ	ـظ
4	ث	ثـ	ـثـ	ـث	18	ع	عـ	ـعـ	ـع
5	ج	جـ	ـجـ	ـج	19	غ	غـ	ـغـ	ـغ
6	ح	حـ	ـحـ	ـح	20	ف	فـ	ـفـ	ـف
7	خ	خـ	ـخـ	ـخ	21	ق	قـ	ـقـ	ـق
8	د	-	-	ـد	22	ك	كـ	ـكـ	ـك
9	ذ	-	-	ـذ	23	ل	لـ	ـلـ	ـل
10	ر	-	-	ـر	24	م	مـ	ـمـ	ـم
11	ز	-	-	ـز	25	ن	نـ	ـنـ	ـن
12	س	سـ	ـسـ	ـس	26	ه	هـ	ـهـ	ـه
13	ش	شـ	ـشـ	ـش	27	و	-	-	ـو
14	ص	صـ	ـصـ	ـص	28	ي	يـ	ـيـ	ـي

Tableau 1-2 : Formes des caractères arabes.

Les points jouent un rôle important dans les caractères arabes par ce que la forme des certains caractères est similaire, mais la différence se pose avec la position et le nombre des points, cela peut se produire soit au-dessus ou en dessous des caractères. Par exemple, trois caractères tels que (ب, ت, ث) ont une forme similaire. Les points peuvent apparaître sous la forme de deux points distincts ou peuvent être connectés en ligne dans des textes manuscrits. En outre, des marques courtes comme un «hamza», peuvent être placées au-dessus ou au-dessous de cinq caractères particuliers ou peuvent apparaître comme des caractères isolés. Certains caractères arabes ont une boucle, comme (ص, ف, و). De plus, le texte arabe est cursif; Ce qui signifie que les caractères d'un mot sont reliés par une ligne horizontale imaginaire appelée ligne de base. En outre, il y a des lignes qui apparaissent au-dessus et au-dessous de la ligne de base, appelé ascendants et descendants, comme le montre la figure 1-05 [4].

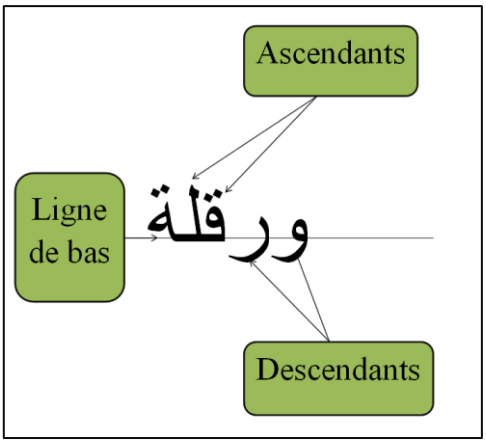


Figure 1-05 : La ligne de base et les caractères ascendants et descendants.

En outre, il existe six caractères (ا, د, ذ, ر, ز, و) qui n'ont pas de forme au début et au milieu du mot, comme le montre le tableau 1-2.

Par conséquent, ces caractères ne se connectent pas au caractère suivant dans un mot et cela provoque une séparation du mot en parties, comme montré à la figure 1-06. Ces parties sont appelées sous-mots. Les espaces séparent les mots et les espaces courts séparent les sous-mots.

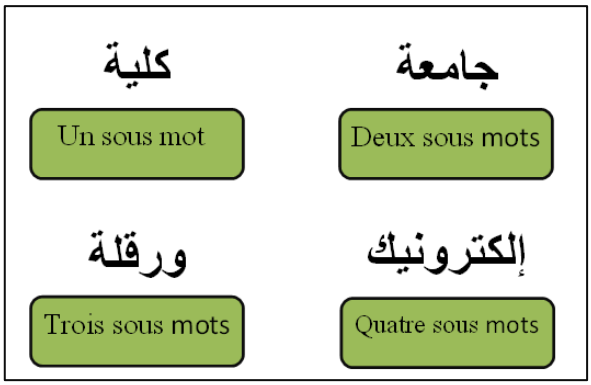


Figure 1-06 : Mots avec différents sous-mots.

Le même mot dans un texte écrit à la main en arabe pourrait avoir des styles et des tailles différents pour le même auteur ainsi que pour différents auteurs, comme le montre la figure1-07.

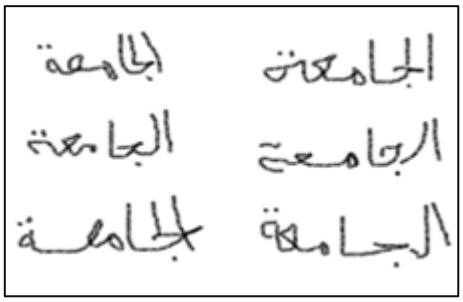


Figure 1-07 : Des styles et des tailles différents pour le même mot.

En outre, deux ou plusieurs caractères dans la langue arabe manuscrite peuvent être combinés verticalement et représentés par différentes formes. Ce chevauchement entre les caractères voisins est appelé une ligature et signifie que le second caractère peut apparaître avant le premier dans certains cas [5]. Une ligature peut se produire lorsque des caractères tels que (ل ,م ,ج ,خ ,ح) apparaissent après certains autres caractères.

Dans le 1er cas de la Figure 1-08 en vois que le chevauchement des trois premiers caractères (ا ,ل ,ج). Dans certains cas, deux caractères peuvent se toucher par erreur, comme les caractères (ر ,و) comme le montre le 2éme cas de la Figure 1-08. De plus, deux caractères peuvent être chevauchés verticalement sans toucher l'un l'autre, tels que les caractères (د, ا, و), dans le 3émé cas de la Figure 1-08.

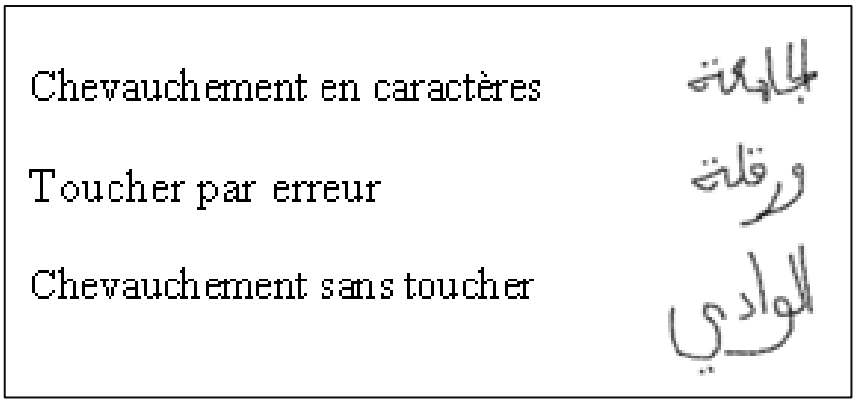


Figure 1-08 : Différents ligatures des mots.

Dans les textes arabes manuscrits, certains caractères semblent similaires, bien qu'ils soient différents, et il est difficile même pour l'œil humain de trouver la différence [6]. Il existe des différences entre la longueur et la largeur des caractères arabes, par exemple (ا, ب). En outre, le même caractère peut sembler différent dans ses diverses formes, telles que (غ،ف) [7]. En outre, la grande similitude entre certains des caractères manuscrits rend la classification de ces caractères un autre défi, comme le montre la figure 1-09.

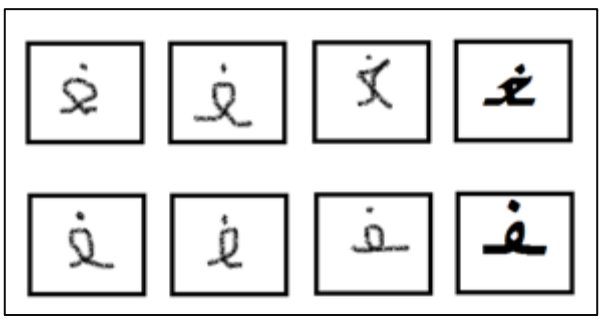


Figure 1-09 : Similitude entre certains caractères.

1.5 Architecture d'un système de reconnaissance des caractères arabes :

La reconnaissance des caractères manuscrits hors ligne comporte des nombreux défis en raison de la complexité et de l'ambiguïté des styles d'écriture. Les systèmes de reconnaissance des caractères peuvent être classés en différentes catégories, comme le montre la figure 1-10.

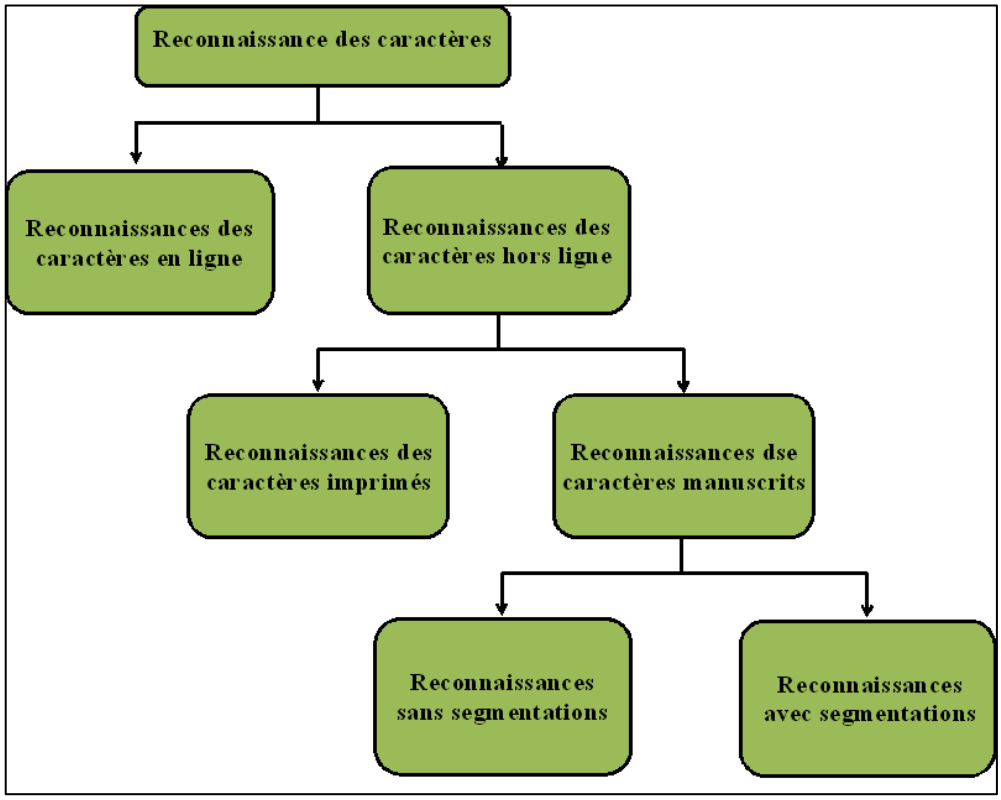


Figure 1-10 : Classification des systèmes de reconnaissance des caractères.

La figure indique différentes voies pour la reconnaissance des caractères arabes avec ou sans segmentation. La reconnaissance des caractères peut être divisée en deux types : reconnaissance des caractères en ligne et hors ligne [8]. La reconnaissance en ligne des caractères est classifiée alors que l'utilisateur écrit. Cette technique utilise un équipement tel qu'un crayon spécial et une tablette, la trace numérisée du stylo étant utilisée pour reconnaître le caractère. Par conséquent, il ne pouvait pas être appliqué pour la reconnaissance des documents pré-écrits. Pour plus de détail sur la différence entre la reconnaissance des caractères en ligne et hors ligne, les lecteurs sont référés à l'article [9]. D'autre part, les systèmes de reconnaissance hors ligne traitent des images numérisées des documents précédemment écrits. La reconnaissance hors ligne des textes peut être divisée en deux catégories: la reconnaissance des caractères imprimés et des caractères manuscrits. Les caractères imprimés ont un style et une taille pour n'importe quelle police donnée. Cependant,

les caractères manuscrits ont des styles et des tailles qui varient, à la fois pour le même auteur et entre les différents écrivains. Les mots manuscrits peuvent être reconnus de deux façons: la reconnaissance d'un mot entier sans segmentation, ou la reconnaissance basée sur la segmentation. En raison de la présence de la ligature et de la nature cursive de l'écriture arabe, plusieurs chercheurs ont présenté des techniques basées sur la reconnaissance du mot entier sans segmentation [10, 11]. La reconnaissance manuscrite des mots implique plusieurs étapes pour obtenir la classification en tant que fichier texte. La figure 1-11 illustre les étapes générales de reconnaissance de texte manuscrit. Puisque le processus de segmentation est la principale source d'erreurs dans la reconnaissance, la plupart des systèmes évitent cette étape et ne reconnaissent que le mot entier.

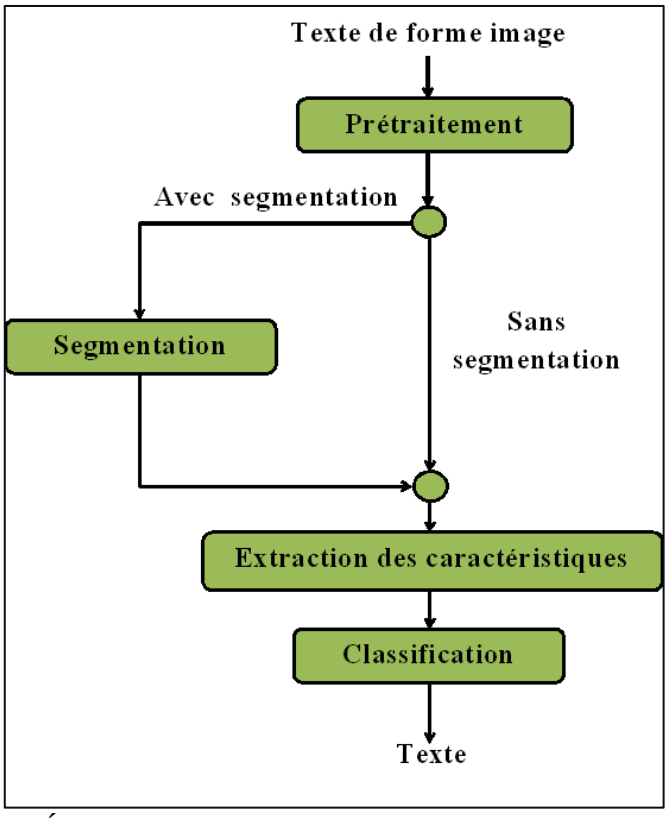


Figure 1-11 : Étapes générales pour la reconnaissance du texte manuscrit.

1.5.1 Prétraitement :

La tâche de prétraitement est une étape importante dans tout système de reconnaissance [12]. Le but de cette étape dans la reconnaissance de texte manuscrit est d'améliorer la lisibilité de l'image de texte et de supprimer les détails qui n'ont pas de pouvoir discriminatif dans le processus de reconnaissance [13, 14].

L'étape de prétraitement comprend habituellement plusieurs tâches: binarisation, suppression du bruit, détection de référence et normalisation.

a. Binarisation :

La binarisation est un processus qui permet de convertir une image de texte en format binaire: les valeurs des pixels d'arrière-plan comme 1 (blanc) et les valeurs des pixels d'avant-plan comme 0 (noir). Ce processus est réalisé en choisissant une valeur de méthode de seuillage efficace. Un des avantages de ce procédé est qu'il augmente la vitesse de traitement [15].La distorsion de l'image se produit généralement pendant le processus de numérisation. Les petits objets qui ne font pas partie de l'écriture peuvent être considérés comme du bruit et doivent être supprimés.

b. Suppression de bruit :

Le bruit est introduit lors de l'acquisition d'images d'entrée via des scanners optiques ou des dispositifs d'écriture et provoque une distorsion dans l'image d'entrée. Avant la reconnaissance du texte, il est essentiel de supprimer ces variantes. Bien qu'il existe des nombreuses techniques utilisées pour éliminer le bruit, deux approches principales sont largement utilisées [16]:

b.1 Filtrage :

Il existe plusieurs filtres de domaine spatial et de fréquence qui peuvent être conçus pour supprimer le bruit et réduire le nombre des points parasites. Les filtres peuvent être conçus pour le seuillage, l'affûtage, le lissage et pour le réglage du contraste. La mise en œuvre d'un filtre spatial linéaire implique la conception d'un masque et l'exécution d'une convolution de l'image et du masque comme suit : $Out(x,y) = \sum_i \sum_j w_{i,j} * inp(x-i, y-j)$

Où $w_{i,j}$ est les poids des niveaux de gris des pixels de masque à l'emplacement (i,j).

b.2 Opérations morphologiques mathématiques :

Des nombreux filtres spatiaux et de domaine fréquentiel peuvent être conçus pour différentes applications, comme le lissage [17] et l'élimination du bruit [18]. Le lissage réduit le bruit en utilisant des opérations de morphologie mathématique. Il est généralement réalisé de deux façons: en ouvrant les espaces entre les objets en contact ou en comblant des petits espaces [19]. Les deux méthodes, l'ouverture et la fermeture, appliquent respectivement les opérations morphologiques d'érosion et de dilatation. Mathématiquement [20], l'érosion de A par B est l'ensemble des tous les points z et peut être définie comme :

Erosion de A par B : $A \ominus B = \{Z | (\mathrm{B})_Z \cap A \neq \emptyset\}$

Dilatation de A par B : $A \oplus B = \left\{ Z \middle| \left(\widehat{\mathrm{B}}\right)_Z \cap A \neq \emptyset \right\}$

c. Détection de la ligne de base :

La ligne de base est une ligne horizontale imaginaire qui relie les caractères d'un mot [4]. La détection de la ligne de base est l'une des tâches importantes dans l'étape de prétraitement [14]. Il pourrait être utile dans la correction oblique ainsi que la segmentation des caractères [21]. Une méthode commune utilisée pour extraire la ligne de base utilise une projection horizontale et cela fonctionne bien, en particulier avec le texte imprimé [21, 22]. La détection de la ligne de base permet de déterminer certaines caractéristiques structurelles, telles que les points et leurs positions, les ascendants et les descendants, ainsi que la correction d'inclinaison.

d. Seuillage :

Est le processus de conversion d'une image en noir et blanc, image binaire. Les principaux avantages du processus de seuillage sont qu'il réduit les exigences de stockage et augmente la vitesse de traitement. Par conséquent, il est recommandé de convertir des images en couleur ou en gris en images binaires en choisissant une valeur de seuil correcte.

L'opération de seuillage sépare le premier plan (encre) de son arrière-plan (papier) [09], en utilisant la fonction de mappage suivante : Supposons que l'image d'entrée est $inp(x, y)$, et de sortie est $Out(x, y)$ donc :

$$Out(x,y) = \begin{cases} 0 \; si \;\; inp(x,y) < seil \\ 1 \; si \;\; inp(x,y) > seil \end{cases}$$

e. Squelettisation :

Joue un rôle important dans des nombreux systèmes de reconnaissance. La plupart des systèmes de reconnaissance mettent en œuvre l'éclaircissage en tant que phase de prétraitement avant les phases de segmentation, d'extraction des caractéristiques et de reconnaissance. Grâce à l'opération consistant à réduire la largeur du trait du texte à un seul pixel, des approches d'amincissement sont utilisées pour obtenir des squelettes de texte imprimé et manuscrit [23]. Les chercheurs en écriture arabe font face à un problème après avoir appliqué des algorithmes d'amincissement [24].

Figure 1-12 : Résultat d'une opération d'éclaircie.

Cela est dû à la forme similaire des lettres arabes avec différents points tels que, ي, ش, ذ. Pour surmonter un tel problème, les points sont extraits avant éclaircissage et sont reconnus séparément. La figure 1-12 montre les résultats d'une opération d'éclaircie.

f. Normalisation :

L'écriture manuscrite a des styles et des tailles différents. Par conséquent, le processus de normalisation est l'une des tâches les plus importantes dans le processus de reconnaissance du texte. Il permet de réduire la variation entre les images du texte et d'ajuster la taille du caractère ou du mot. La normalisation des tailles est couramment utilisée pour réduire la variation de taille et ajuster les tailles des caractères ou des mots à une taille identique [25]. Certains chercheurs [26] ont proposé de diviser le caractère en un certain nombre des zones, puis chaque zone de mise à l'échelle séparément. En raison d'inexactitudes dans le processus de numérisation, le document numérisé peut être légèrement tourné. La correction d'inclinaison est utilisée pour corriger l'angle d'orientation. La ligne de base du texte est habituellement utilisée pour détecter l'obliquité dans les textes arabes. En 2005, d'autres chercheurs ont proposé un algorithme pour corriger l'asymétrie et incliner le mot, basé sur la transformation du Radon [27].

1.5.2 Segmentation :

L'étape de segmentation consiste à segmenter le texte en sous-unités, telles que des lignes, des mots ou des caractères. La segmentation est une étape importante car elle a un effet sur le taux de reconnaissance [28, 29].

a. Types de segmentation :

a.1 Segmentation d'une page en lignes :

En général, le paragraphe ou la page se compose de plusieurs lignes. En reconnaissance de texte, le texte comporte plusieurs lignes se segmentant en lignes séparées. Il existe certaines méthodes utilisées à cet effet, telles que la projection horizontale [09, 30].

a.2 Segmentation d'une ligne en mots :

Après la segmentation des lignes, la ligne est segmentée en mots. Segmenter la ligne en mots dépend de l'espace entre les mots. Cependant, certains caractères ne se connectent pas à un caractère ultérieur d'un mot, ce qui entraîne une séparation du mot en sous-mots. Les espaces plus longs séparent les mots tandis que les espaces courts séparent les sous-mots. Par conséquent, la plupart des chercheurs supposent dans leur technique que l'espace entre les

mots est plus grand que l'espace entre les sous-mots [31]. Certaines des méthodes analysent les distances entre les composants connectés pour segmenter les mots [32].

a.3 Segmentation d'un mot en caractères :

Cette partie de segmentation implique la segmentation du mot en caractères individuels. Comme décrit dans [33], les points de segmentation sont identifiés à la fin d'un caractère et au début de la suivante. La nature cursive de la langue arabe rend la segmentation des mots en caractères individuels une tâche difficile.

b. Différentes techniques appliquées pour la segmentation :

b.1 Technique de segmentation basée sur la projection verticale :

La technique de projection verticale réduit les informations bidimensionnelles en une seule dimension. Cette technique est basée sur le fait que l'épaisseur de la course de connexion entre les caractères dans le mot est toujours inférieure aux autres parties. Des nombreux algorithmes ont été proposés dans cette technique pour la segmentation des caractères tels que [33, 34, 35, 36]. La projection verticale est utile pour segmenter les mots, les sous-mots et les caractères, tandis que la projection horizontale est habituellement utilisée pour la segmentation de ligne et l'extraction de ligne de base.

b.2 Technique de segmentation basée sur l'éclaircissement :

Lors de la reconnaissance des caractères, le squelette d'un caractère fournit les informations essentielles sur la forme du caractère. Un certain nombre d'algorithmes ont été proposés pour extraire les squelettes [37, 38, 39].

b.3 Technique de segmentation basée sur le traçage des contours :

Une autre technique basée sur le traçage des contours d'un mot est également utilisée pour la segmentation. Cette technique est obtenue en traçant le contour extérieur du corps principal du mot. Dans [40] ont proposé un algorithme de segmentation des caractères basé sur l'analyse de contour et les règles topologiques. Il commence par trouver le point minimum local du contour inférieur. Ensuite, il utilise les règles topologiques pour déterminer si le point de minimum local est un point de segmentation.

b.4 Technique de Segmentation basée sur les réseaux neuronaux artificiels :

Les RNAs sont utilisés pour vérifier les points de segmentation valides. Dans [41], Hamid et Haraty ont proposé une technique pour segmenter des textes arabes manuscrits basés sur RNAs. Ils ont identifié des points de pré-segmentation en utilisant les

caractéristiques topographiques de chaque bloc des caractères connecté, comme la densité des pixels noirs et les trous. Les RNA sont ensuite utilisées pour vérifier ces points de segmentation. Les points potentiels sont classés manuellement en points de segmentation valides et non valides, et ces points sont alimentés avec leurs caractéristiques, dans les RNA.

b.5 Technique de segmentation basée sur des opérateurs morphologiques :

Des opérations morphologiques telles que la fermeture suivie de l'ouverture sont utilisées pour la segmentation des caractères. Dans [42] ont proposé un algorithme pour segmenter des mots arabes manuscrits. Ils supposent que les traits verticaux qui peuvent représenter le début et la fin du caractère sont trouvés par singularités. Les régularités qui ont des informations sont nécessaires pour connecter un caractère à un caractère ultérieur. Par conséquent, l'image originale est balayée de droite à gauche afin d'identifier les régularités. Les singularités sont identifiées en appliquant une ouverture au mot. Les régularités sont extraites en soustrayant les singularités de l'image du mot. Ces régularités sont utilisées pour identifier les candidats des points de segmentation.

Cependant, la plupart des algorithmes de segmentation actuellement proposés ne résolvent pas le problème du chevauchement des caractères en écriture arabe. L'étape de segmentation est l'étape la plus difficile et la principale source d'erreurs dans la reconnaissance. Les techniques de segmentation représentent donc encore un défi dans la reconnaissance du texte et doivent être améliorées.

1.5.3 Extraction des caractéristiques :

Dans les textes imprimés et manuscrits, les caractéristiques saisissent les informations extraites de l'image textuelle. Ces informations doivent avoir les caractéristiques essentielles du caractère ou du mot qui le différencient; En d'autres termes, filtrer tous les attributs et préserver les propriétés qui rendent un caractère ou un mot différent d'un autre [19]. Ces informations sont transmises au classificateur pour faciliter le processus de classification. Les techniques d'extraction des caractéristiques diffèrent d'une application à l'autre. Les techniques qui réussissent dans une application, peuvent ne pas être couronnées de succès pour d'autres applications [43]. Par conséquent, la sélection de la méthode d'extraction des caractéristiques reste l'étape la plus importante pour obtenir une précision de reconnaissance élevée. Les caractéristiques des textes manuscrits peuvent être classées dans les catégories suivantes :

a. Caractéristiques structurelles :

Les caractéristiques structurelles sont les caractéristiques les plus courantes utilisées par les chercheurs [44]. Ils illustrent les caractéristiques géométriques et topologiques de l'image textuelle en décrivant leurs propriétés locales et globales [19]. Les caractéristiques structurelles dépendent de la catégorie du modèle à classer. Pour un texte en arabe, par exemple, les caractéristiques incluent les points et leur position, les traits, la largeur et la hauteur du trait, les directions, l'intersection des segments de ligne et des boucles [45, 46].

b. Caractéristiques statistiques :

Les caractéristiques statistiques analysent la répartition spatiale des pixels en comptant les caractéristiques locales à chaque pixel et en dérivant un ensemble des statistiques à partir des distributions des caractéristiques locales [19, 47]. Les caractéristiques statistiques majeures du caractère incluent le zonage, où le caractère se divise en zone se chevauchant ou non se chevauchant et la distribution de densité des pixels de caractère dans différentes régions est analysée. Mohiuddin et Mao dans [48] ont mesuré la direction du contour du caractère en divisant le caractère d'image en zones. Ensuite, des histogrammes des codes de chaîne sont utilisés pour calculer la direction du contour dans ces régions.

c. Transformation globale :

Les techniques de transformation globale ont la capacité de convertir la représentation des pixels en une forme plus compacte. Ces techniques peuvent représenter le signal par une combinaison linéaire d'une série des fonctions plus simples et bien définies.

L'expansion en série fournit un codage compact par les coefficients de la combinaison linéaire [49, 50]. Les techniques courantes de transformation utilisées dans la reconnaissance des caractères sont:

- Transformée de Fourier.
- Transformée de cosinus discret.
- Ondelettes.
- Transformée de Hough.
- Moments.

1.5.4 Classification :

La classification est une tâche qui tente d'identifier un objet en comparant ses caractéristiques à l'un d'un ensemble donné des classes. Il suppose que les cours d'apprentissage et le modèle d'apprentissage ont été fournis. Le modèle d'apprentissage se

compose d'un ensemble d'instances où les bonnes réponses (sorties) pour chaque entrée sont fournies. Par conséquent, un classificateur est utilisé pour identifier un objet en utilisant ses caractéristiques, ceux-ci sont ensuite comparés et sauvegardés comme modèles pour les classes formées. Les caractéristiques d'un objet dans la phase de test seront extraites et comparées avec les caractéristiques des modèles d'apprentissage pour identifier l'objet inconnu. Les méthodes courantes dans la phase de classification sont basées sur:

- Réseaux des neurones artificiels.
- Modèle de Markov caché.
- k-voisin le plus proche.

1.6 Conclusion :

Nous avons présenté dans ce chapitre les différents problèmes à résoudre dans un système de reconnaissance de l'écriture, ainsi que les techniques pouvant être utilisées. Il s'agit premièrement d'obtenir une base des données comportant des exemples de chaque caractère, puis de passer par une phase de prétraitement qui comporte une normalisation de chaque caractère, puis une extraction des attributs. La troisième phase est celle de la classification. Dans notre projet, nous avons utilisé les réseaux des neurones pour classifier les caractères. Dans les chapitres suivants nous présentons les réseaux des neurones et nous détaillons encore plus les différents algorithmes de prétraitement que nous avons utilisés.

Chapitre 2

Les réseaux des neurones convolutionnels

2.1. Introduction :

La reconnaissance des caractères arabes manuscrite a attiré une attention considérable au cours des dernières décennies. Les chercheurs ont réalisé des percées significatives dans ce domaine avec le développement rapide d'algorithmes d'apprentissage en profondeur est les applications les plus réussies dans ce domaine Jusqu'à présent sont eux qui utilisent les méthodes basées sur les réseaux des neurones. Ce chapitre fournit un bref aperçu des réseaux neuronaux ainsi que du réseau neuronal convolutionnel, et ce dernier est largement utilisé car il présente un certain nombre d'avantages par rapport à d'autre technique.

2.2. Réseaux des neurones artificiels :

Les réseaux des neurones (RN) sont des systèmes populaires pour la reconnaissance des formes en général. Ils sont fabriqués à partir d'unités de traitement de base, reliés entre eux avec des connexions pondérées et dirigées, de sorte que la sortie de certaines unités est une entrée pour les autres. L'appellation «Réseau de neurone artificiel (RNA) » provient de la similitude entre les unités de ces modèles et les neurones biologiques ou les intrants dans un neurone artificiel correspondent aux dendrites dans un neurone biologique, tandis qu'une seule sortie d'un neurone artificiel correspond à l'axone dans un neurone biologique, qui est représenté à la Figure 2-01.

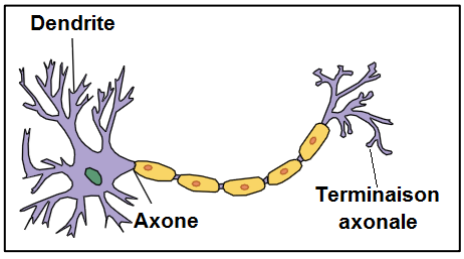


Figure 2-01 : Modèle d'un neurone biologique.

La première description formelle des réseaux des neurones artificiels a été proposée par McCulloch & Pitts [51]. Les algorithmes pour ajuster les poids des connexions conduisent au perceptron [52] et aux Le réseau des neurones multicouches [53]. Parmi les différents types des réseaux neuronaux, on peut remarquer des réseaux neuronaux convolutionnels (RNC) [54], adaptés aux entrées d'images.

Généralement, les RNAs sont constitués d'un ensemble des neurones artificiels. Fortement, un neurone artificiel a n entrées représentées comme un vecteur $x \in \boldsymbol{R^n}$. Chaque entrée $i, 1 \leq i \leq n$, a un poid assigné $w_1, \ldots, w_n$. Les valeurs des entrée pondérées sont combinées et exécutées à l'aide d'une fonction d'activation produisant une sortie y, comme le montre la Figure 2-02. Le réseau est formé en reliant la sortie du neurone à l'entrée d'un neurone différent.

Un RNA fonctionne en introduisant les données dans les neurones d'entrée. Les données s'écoulent dans la direction des bords orientés et se terminent lorsque les neurones de sortie sont atteints. Le résultat est interprété à partir des valeurs obtenues dans les neurones de sortie.

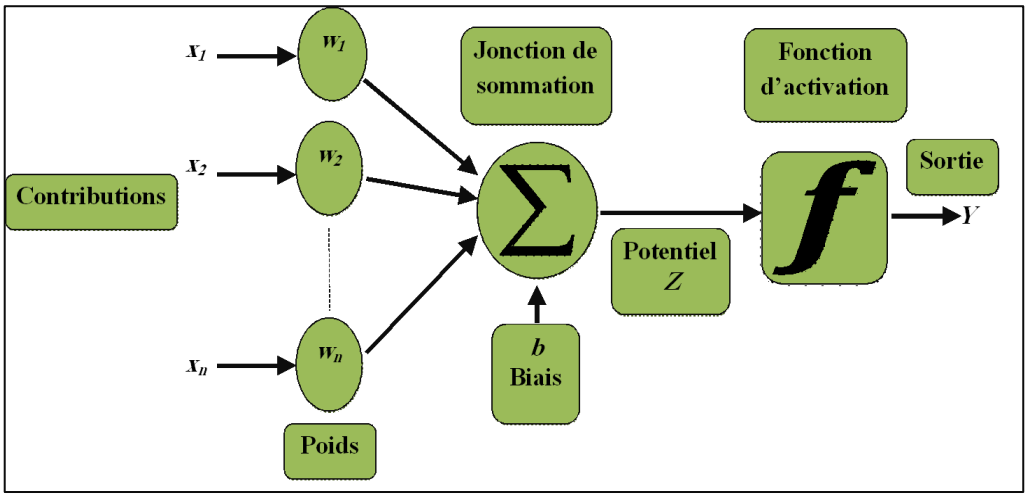


Figure 2-02 : Modèle d'un neurone artificiel.

Considérons le neurone j avec sa contribution $x_j = (x_{1j}, \ldots, x_{nj})$, les poids $w_{1j}, \ldots, w_{nj}$ et le biais b_j. Ensuite, le potentiel du neurone est calculé:

$$Z = \sum_{i=1}^{n} w_{ij} * x_{ij} + b_j \tag{2.1}$$

Considérons la fonction d'activation (section 2.3) suivante:

$$f(Z_j) = \frac{1}{1 + e^{-Z_j}} \tag{2.2}$$

Ensuite, la sortie Y_j du neurone j est calculée:

$$Y_j = f(Z_j) = f\left(\sum_{i=1}^{n} w_{ij} * x_{ij} + b_j\right) \tag{2.3}$$

2.3. Perceptron :

Le réseau neuronal le plus simple est le perceptron qui se compose d'un seul neurone entièrement fonctionnel. Sa sortie est calculée directement par l'équation 2.3.

2.4. Fonctions d'activation :

On dit qu'une fonction d'activation est activée si sa sortie n'est pas nulle. Il est également dit avoir une forte activation si la sortie est relativement élevée et une activation faible si sa sortie est relativement faible. Une fonction d'activation doit être non linéaire, continuellement différentiable et monotone, il est en outre souhaité que la fonction $f(x) \approx x$ lorsque x approche de 0. Les fonctions d'activation doivent être non linéaires car il s'agit d'une caractéristique nécessaire pour que le réseau neural soit une approximation universelle [55]. Des fonctions d'activation continuellement différentiables sont nécessaires pour les méthodes d'optimisation basées sur les gradients. Les fonctions d'activation monotone garantissent une surface d'erreur convexe d'un modèle à une seule couche [56]. Et si $f(x) \approx x$ lorsque x s'approche de 0, les réseaux peuvent s'entraîner plus efficacement. Il existe plusieurs fonctions d'activation à choisir. La fonction sigmoïde, la tangente hyperbolique et la fonction d'activation unité de rectification linéaire (ReLU) sont introduites et définies dans cette section. Certaines fonctions de classification et de régression utilisées dans la couche finale des réseaux de neurones sont également introduites.

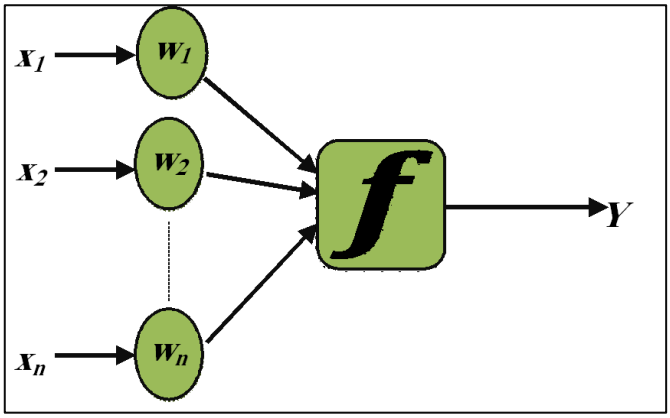


Figure 2-03 : Placement de la fonction d'activation dans le modèle de réseau neuronal.

2.4.1. Tangente hyperbolique :

L'une des fonctions d'activation les plus populaires est la fonction tangente hyperbolique (Equation 2.4). L'entrée x est une combinaison linéaire pondérée des entrées du nœud. Cette fonction fonctionne le plus efficacement sur les entrées dans la plage (0; 1), produisant des sorties en intervalle (-1; 1).

$$\tanh(x) = \frac{\sinh(\mathrm{x})}{\cosh(\mathrm{x})} = \frac{\mathrm{e}^{x} - \mathrm{e}^{-x}}{\mathrm{e}^{x} + \mathrm{e}^{-x}} = \frac{\mathrm{e}^{2x} - 1}{\mathrm{e}^{2x} + 1} \qquad (2.4)$$

2.4.2. Sigmoïde :

La fonction sigmoïde logistique (Equation 2.5) est une fonction d'activation largement utilisée biologiquement plus plausible que la tangente hyperbolique [57]. L'une des raisons pour lesquelles la fonction sigmoïde est largement utilisée est le fait, la fonction sigmoïde est différente à chaque point. La fonction sigmoïde (dite aussi courbe en S) est définie par :

$$f(\mathrm{x}) = \frac{1}{1 + e^{-\mathrm{x}}} \tag{2.5}$$

2.4.3. Unité de rectification linéaire (ReLU) :

La fonction de l'unité de Rectification Linéaire (Equation 2.6) est utilisée dans le but d'augmenter la non-linéarité du réseau. Les neurones de rectification sont considérés comme biologiquement plus plausibles que les neurones tangentiels sigmoïdes ou hyperboliques tangents [59]. Ils bénéficient de leur simplicité, entraînant une formation plus rapide et des améliorations de performance dans des cas particuliers [58], et donc souvent utilisés dans les RNC. ReLU est donné par l'équation:

$$f(\mathrm{x}) = \max(0, x) \tag{2.6}$$

2.4.4. Softmax :

La fonction d'activation de softmax (équation 2.7) est habituellement utilisée dans la dernière couche du réseau,

$$softmax(x_i) = \frac{e^{x_i}}{\sum_{k=1}^{N} e^{x_K}} \tag{2.7}$$

Lorsque le problème de classification n'est pas binaire mais contient plusieurs classes, la fonction softmax est généralement utilisée, mais elle est également viable pour le cas binaire. Il s'agit d'une généralisation de la fonction logistique et n'est qu'un autre nom pour un modèle de classification multinomiale quand on suppose qu'il n'existe aucune hiérarchie parmi les classes. La fonction softmax est agréable car elle donne une approximation de la probabilité qu'une classe soit correcte. L'approche la plus simple consiste à simplement choisir la classe avec la probabilité la plus élevée et ignorer le reste. Mais étant donné qu'il s'agit d'une fonction probabiliste, il peut également être utilisé pour un modèle générateur. Les scores de softmax (probabilités) sont calculés par la fonction de normalisation.

2.5. Perceptron multicouches (PMC) :

Un perceptron simple couche est capable de classer les données uniquement linéairement séparables. Par exemple, il n'est pas capable de résoudre le problème XOR. Ce fait a été remarqué par Minsky et Papert [59] dans leur célèbre livre "Perceptrons" en 1969. Le livre a contribué à la stagnation dans la recherche sur les réseaux des neurones pour un certain temps. On savait que le perceptron multicouche résoudrait des problèmes linéairement non séparables, même si des algorithmes efficaces pour l'apprentissage des PMC n'étaient pas connus à ce moment-là. Le premier algorithme réussi, appelé rétro-propagation, a été développé plusieurs années plus tard [60] [61] et depuis lors, le domaine des réseaux neuronaux s'est rapidement développé.

Un perceptron multicouche est un réseau qui consiste généralement en deux ou trois couches des neurones et d'une couche d'entrée supplémentaire. La couche d'entrée est comptée par certains auteurs comme une couche réseau distincte alors que par d'autres, elle ne l'est pas.

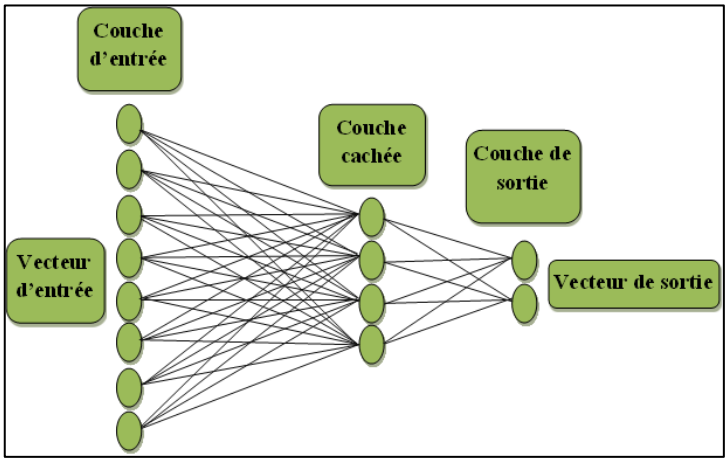


Figure 2-04 : Perceptron multicouches fortement connecté aux couches cachées.

Les perceptrons sont disposés en $k \geq 2$ couches. Considérons un réseau M avec des couches k. L'ensemble des neurones C est divisé en sous-ensembles disjoints mutuellement appelés couches $L_1, \ldots, L_\kappa$. Plus formellement, il tient $\forall\, i, j\colon 1 \leq i, j \leq \kappa\ (L_i \neq \phi \wedge L_i \cap L_j \neq \phi) \Rightarrow i = j$. Les couches de réseau sont empilées l'une sur l'autre, L_1 étant la couche d'entrée, $L_2, \ldots L_{k-1}$ étant les couches cachées et L_κ étant la couche de sortie. Comme le montre la figure 2-04, les bords sont tous orientés dans la direction de la couche d'entrée L_1 vers la couche de sortie L_κ. Chaque neurone dans la couche L_i est connecté à chaque neurone dans la couche L_{i+1}. En d'autres termes, toutes les couches voisines forment des graphiques bipartites complets.

La sortie du réseau est calculée séquentiellement, couche par couche. Nous commençons par la couche d'entrée en attribuant directement $\vec{y}^0 = \vec{x}$. Ensuite, le calcul se déroule en assignant l'entrée $\vec{x}^i = \vec{y}^{i-1}$ pour la couche L_i. Les poids et la fonction d'activation sont donnés par le réseau, donc la sortie de chaque couche dépend uniquement de la sortie de la couche précédente. La sortie finale du réseau est ensuite produite comme $\vec{y}^k$ dans la couche de sortie L_κ.

2.5.1. Le processus d'apprentissage :

La capacité d'apprentissage est le concept clé des réseaux des neurones. Le but du processus est de trouver les paramètres (et la structure) optimaux du réseau pour résoudre la tâche donnée. Avant le début du processus d'apprentissage, les paramètres du réseau doivent être initialisés. Les valeurs initiales sont souvent choisies au hasard, mais certaines heuristiques peuvent conduire à un réglage plus rapide des paramètres en fonction des valeurs optimales. L'apprentissage est ensuite effectué sur l'ensemble de formation en alimentant les données d'apprentissage via le réseau. Il s'agit d'un processus itératif, où les sorties produites sur chaque entrée de l'ensemble d'apprentissage sont analysées et le réseau est à plusieurs reprises ajusté pour produire des meilleurs résultats. Le réseau est considéré comme formé après avoir atteint la performance cible sur les données de formation. Il existe différents paramètres pour évaluer la performance du réseau Que nous définirons formellement dans le texte suivant.

Un problème courant rencontré dans le processus d'apprentissage est le sur-apprentissage. Il se produit généralement lorsque l'apprentissage se fait trop longtemps, et surtout lorsque l'ensemble d'apprentissage est trop faible pour représenter uniformément tous les types des modèles à partir du domaine des entrées possibles du réseau. Dans un tel cas, l'apprentissage peut ajuster le réseau aux fonctions aléatoires présentes dans les données d'apprentissage. L'accouplement est observé pendant le processus d'apprentissage, lorsque la performance prédictive du réseau s'améliore sur l'ensemble d'apprentissage, tout en aggravant les données de test précédemment non vues.

Pour lutter contre ce problème, les données marquées sont divisées en un ensemble d'apprentissage et un ensemble de validation. La principale raison pour laquelle utiliser l'ensemble de validation est qu'il montre les taux d'erreur sur les données indépendantes des données sur lesquelles nous nous entraînons. Une étude de Guyon suggère que le rapport optimal entre la taille de l'ensemble des données d'apprentissage et de validation dépend du nombre des classes reconnues et de la complexité des caractéristiques de classe [62]. Une

estimation de la complexité des caractéristiques est cependant assez lourde. Un bon point de départ pour déterminer ce ratio est de mettre 80% des données disponibles dans l'ensemble d'apprentissage et 20% dans l'ensemble de validation. Une expérimentation supplémentaire peut aider à se rapprocher du rapport optimal. Tout en apprenant, la performance de RNA est régulièrement examinée sur l'ensemble des données de validation. Lorsque les erreurs récupérées sur les données de validation atteignent un point d'arrêt, le processus d'apprentissage est arrêté (Figure 2-05) et le réseau est considéré comme formé.

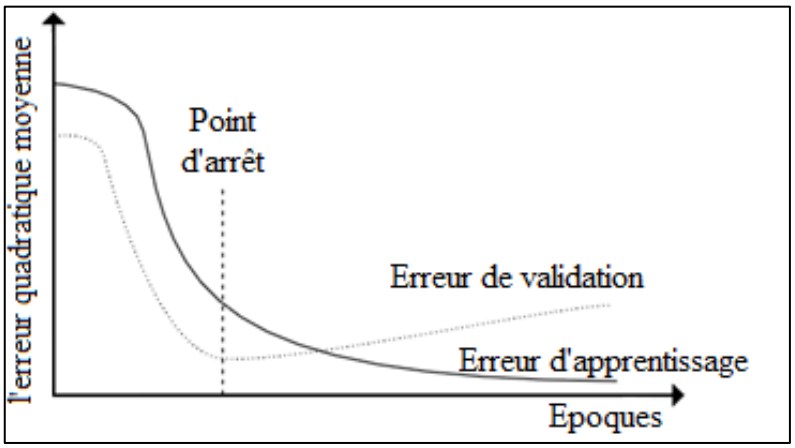


Figure 2-05 : L'évolution de l'erreur d'apprentissage par rapport à l'erreur de validation.

Pour définir le processus d'apprentissage de façon formelle et plus détaillée, considérons les modèles d'apprentissage P marquées $(\vec{x}^p, \vec{d}^p)$, où $\vec{x}^p$ est le vecteur d'entrée, $\vec{d}^p$ est le vecteur de sortie souhaité, et $1 \leq p \leq P$. Donné La configuration actuelle du réseau, l'entrée $\vec{x}^p$ produit la sortie $\vec{y}^p$. Ensuite, pour chaque modèle p, nous voulons que $\vec{y}^p$ soit aussi proche que possible de la sortie souhaitée $\vec{d}^p$. On peut définir l'erreur de chaque neurone j dans la couche de sortie comme suit:

$$e_j^p = y_j^p - d_j^p \tag{2.8}$$

Maintenant, nous pouvons définir l'erreur quadratique pour les modèles p comme suit:

$$E_P = \frac{1}{m_k}\sum_{j=1}^{m_k}\left(e_j^p\right)^2 = \frac{1}{m_k}\sum_{j=1}^{m_k}\left(y_j^p - d_j^p\right)^2 \tag{2.9}$$

Où m_k est le nombre des neurones dans la couche de sortie. Notez que si la sortie réelle est exactement la même que la sortie souhaitée, nous obtenons zéro pour l'erreur au carré. En d'autres termes, ce qui suit est vrai:

$$\forall j : E_P = 0 \Leftrightarrow e_j^p = 0 \Leftrightarrow y_j^p = d_j^p \tag{2.10}$$

Il peut être utile de résumer l'erreur moyenne pour tous les modèles d'entrée pour évaluer les performances du réseau sur l'ensemble des données, ce qui peut être réalisé simplement en calculant l'erreur quadratique moyenne [63].

$$E_{moy} = \frac{1}{P}\sum_{p=1}^{p}(E_p) \tag{2.11}$$

Lors de l'apprentissage, pour chaque paire des neurones interconnectés (i, j), où i est un neurone dans la couche l, j est un neurone dans la couche $l+1$ et $w_{i,j}$ le poid de leur connexion, nous voulons ajuster $w_{i,j}$ pour minimiser L'erreur quadratique moyenne E_{moy}. Pourvu que la fonction d'activation soit différenciable sur son domaine, E_{moy} est également différentiable. Lors de l'ajustement du poid $w_{i,j}$ du neurone j situé dans la couche de sortie κ, nous sommes donc intéressés par la dérivée partielle:

$$\frac{\partial E_{moy}}{\partial w_{i,j}} = \frac{1}{P}\frac{\partial}{\partial w_{i,j}}\sum_{p=1}^{p}(E_p) = \frac{1}{P}\sum_{p=1}^{p}\frac{\partial E_p}{\partial w_{i,j}} \tag{2.12}$$

Pour pouvoir ajuster le réseau après chaque modèle d'entrée présentée, nous sommes intéressés par la dérivée pour chaque modèle donné P et son E_p correspondant. Dans les équations suivantes, nous omettons donc l'indice de modèle P. En appliquant la règle de chaine à l'équation 2.12, nous obtenons:

$$\frac{\partial E}{\partial w_{i,j}} = \frac{\partial E}{\partial y_j}\frac{\partial y_j}{\partial w_{i,j}} \tag{2.13}$$

En utilisant l'équation 2.9, nous pouvons évaluer le premier facteur comme suit:

$$\frac{\partial E}{\partial y_j} = (y_j - d_j) \tag{2.14}$$

Ensuite, nous évaluons le deuxième facteur:

$$\frac{\partial y_j}{\partial w_{i,j}} = \frac{\partial y_j}{\partial Z_j}\frac{\partial Z_j}{\partial w_{i,j}} = f'(Z_j)\frac{\partial}{\partial w_{i,j}}\sum_{p=1}^{p} w_{k,j}.y_k = f'(Z_j)y_i \tag{2.15}$$

En combinant les deux facteurs évalués, nous obtenons:

$$\frac{\partial E}{\partial w_{i,j}} = (y_j - d_j)f'(Z_j)y_i \tag{2.16}$$

Pour plus de commodité, Haykin définit le prétendu gradient local δ_j pour le neurone j dans la couche de sortie [63].comme la relation suivante:

$$\delta_j = \frac{\partial E}{\partial Z_j} = \frac{\partial E}{\partial y_j}\frac{\partial y_j}{\partial Z_j} = (y_j - d_j)f'(Z_j) \tag{2.17}$$

Ensuite, nous pouvons reformuler l'équation 2.16 dans:

$$\frac{\partial E}{\partial w_{i,j}} = \delta_j y_i \tag{2.18}$$

En utilisant l'équation 2.18, nous pouvons calculer le gradient de la fonction d'erreur pour chacun des modèles donnés P. Nous devons ajuster les poids $w_{i,j}$ proportionnellement au gradient mais dans la direction opposée. Toutefois, pour chaque configuration d'entrée, cela produirait un système très instable. Pour lutter contre ce problème, nous pouvons utiliser un paramètre d'apprentissage $0 < \eta < 1$. L'ajustement des poids est ensuite calculé par:

$$\Delta w_{i,j} = -\eta \frac{\partial E}{\partial w_{i,j}} = -\eta \delta_j y_i \tag{2.19}$$

L'ajustement des poids $\Delta w_{i,j}$ dans l'équation 2.19 n'est applicable qu'aux neurones dans la couche de sortie. Le calcul de l'ajustement pour les neurones dans les couches cachées est plus compliqué. Par exemple, considérons 3 neurones i, j et k, tous se succédant sur le même chemin le long des couches $l-1$, l et $l+1$, respectivement, comme illustré à la figure 2-06. Ensuite, l'ajustement de $w_{i,j}$ doit être fait avec soin, car en plus d'influencer la sortie du neurone i lui-même, il affecte également toutes les sorties (et donc les erreurs) dans toutes les couches suite à l. Dans ce cas, nous allons attirer l'attention sur les couches $l < L$ dans le texte suivant.

Notez que l'équation 2.16 s'applique toujours aux couches cachées. Cependant, nous devons examiner à nouveau la définition du gradient local.

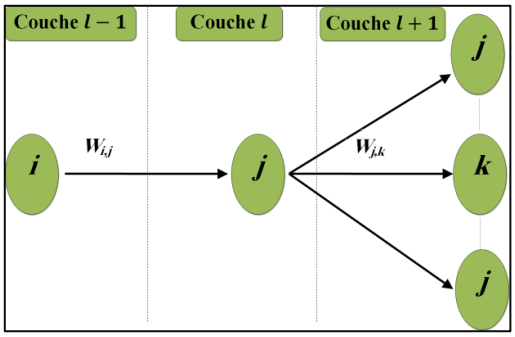


Figure 2-06 : La variation des poids du neurone dans la couche cachée.

Dans l'équation précédente 2.17, nous utilisons la sortie désirée d_j pour calculer $\partial E/\partial y_j$. Bien sûr, il n'y a pas des résultats souhaités connus dans les couches cachées. En fait, cela dépend de la conception du réseau. Pour cette raison, nous devons reculer et utiliser la définition suivante pour δ_j:

$$\delta_j = \frac{\partial E}{\partial Z_j} = \frac{\partial E}{\partial y_j}\frac{\partial y_j}{\partial Z_j} = \frac{\partial E}{\partial y_j} f'(Z_j) \tag{2.20}$$

Maintenant, nous devons redéfinir $\partial E/\partial y_j$ pour les couches cachées. Pour toute couche cachée l, la couche suivante $l+1$ doit exister (sinon l serait la couche de sortie). Compte tenu des neurones i, j et k chacun dans une couche différente, comme illustré à la figure 2-06, nous pouvons utiliser le potentiel Z_k.

$$\frac{\partial E}{\partial y_j} = \sum_{k=1}^{m_{l+1}} \frac{\partial E}{\partial Z_k} \frac{\partial Z_k}{\partial y_j} = \sum_{k=1}^{m_{l+1}} \frac{\partial E}{\partial Z_k} w_{j,k} = \sum_{k=1}^{m_{l+1}} \delta_k w_{j,k} \tag{2.21}$$

En combinant les équations 2.20 et 2.21, nous obtenons:

$$\delta_j = \left(\sum_{k=1}^{m_{l+1}} \delta_k w_{j,k} \right) f'(Z_j) \tag{2.22}$$

L'équation 2.22 nous dit que, en connaissant le gradient local du neurone k dans la couche $l+1$, on peut calculer le gradient local pour le neurone j dans la couche l. Ce fait nous permettra d'ajuster de manière récursive les poids du réseau en couche par couche dans la direction de la sortie à l'entrée (vers l'arrière). Il est utilisé dans l'algorithme d'apprentissage le plus courant décrit dans la section 2.5.2 Enfin, nous pouvons résumer l'ajustement des poids applicable à toutes les couches d'un réseau donné:

$$\Delta w_{i,j} = -\eta \frac{\partial E}{\partial w_{i,j}} = -\eta \delta_j y_i \tag{2.23}$$

Où

$$\forall\, j \text{ en couche } l < L \quad \delta_j = \left(\sum_{k=1}^{m_{l+1}} \delta_k w_{j,k} \right) f'(z_j) \tag{2.24}$$

$$\forall\, j \text{ en couche } L \qquad \delta_j = (y_j - d_j) f'(Z_j)$$

2.5.2. Algorithme de rétro-propagation :

Le but de l'algorithme de rétro-propagation est d'ajuster les poids synaptiques des neurones, de sorte que le réseau génère la sortie souhaitée. L'algorithme décrit le processus d'apprentissage. Le résultat de cet algorithme est un réseau neuronal configuré pour minimiser l'erreur lors de la résolution d'un problème.

L'apprentissage doit être effectué sur les données marquées et donc supervisée. Avant que l'algorithme ne commence, les poids doivent être initialisés à certaines valeurs. L'initialisation ne fait pas partie de la spécification de l'algorithme, car il peut y avoir des approches différentes, la plus courante étant l'initialisation la plus banale - aléatoire. Ensuite, l'algorithme d'apprentissage commence.

Chaque modèle d'entrée avec son étiquette de classe ($\vec{x}_p$,$\vec{d}_p$) est séquentiellement traité en deux phases. La première phase appelée phase avant met le modèle d'entrée comme entrée du réseau, en configurant $\vec{y}^0 = \vec{x}_p$. Le réseau calcule ensuite sa sortie $\vec{y}^L$. Le seul but de la phase avant est de calculer la sortie pour le modèle présenté, et le réseau n'est pas réglé du tout. À ce point, la phase vers l'arrière commence. Le but de cette phase est d'ajuster les poids du réseau afin d'obtenir une meilleure évaluation des données d'entrée.

L'apprentissage se fait à plusieurs époques. Dans chaque époque, toutes les données de l'ensemble de l'apprentissage est traitée. La durée d'une époque dépend directement de la taille du réseau et de la taille de l'ensemble d'apprentissage, car chaque model d'entrée de l'ensemble de l'apprentissage est traité exactement une fois. Cependant, le nombre d'époques n'est pas limité. Une décision importante du processus d'apprentissage est donc la détermination du critère d'arrêt. Après chaque époque nous validons l'erreur sur l'ensemble des données de validation. Une fois que cette erreur commence à augmenter, nous avons atteint un minimum (local ou peut-être global), et il est généralement judicieux d'arrêter le processus d'apprentissage à ce point (voir la figure 2-05).

Formellement, l'algorithme de rétro-propagation sera décrit ci-dessous:

Entrées :

• model d'entraînement($\vec{x}^p, \vec{d}^p$), $1 \leq p \leq P$

• model de validation($\vec{v}^q, \vec{d}^q$), $1 \leq q \leq Q$

• fonction d'activation f et sa dérivation $f'(Z)$

- par exemple. La fonction sigmoïde : $f(Z_j) = \frac{1}{1+e^{-Z_j}}$; $f'(Z) = f(Z)\,(1 - f(Z))$

• paramètre d'apprentissage $\eta \in (0, 1)$

• réseau neuronal vers l'avant M avec poids initialisés au hasard

Sortie:

• un réseau neuronal vers l'avant formé M'

Algorithme:

1. Définir $E_{moy} = \infty$.

2. Démarrez une nouvelle époque.

3. Pour chaque $p \in \{1, \ldots, P\}$ présente le model (x^p, d^p).

• Définir $y^0 = x^p$.

• phase vers avant

- Pour $l = 1, 2, \ldots, K$ calculer la sortie de M pour x^p de la manière suivante :

$\forall j \in L_l$ calcul $Z_j = \sum_{i=1}^{n} w_i * x_i + b$ et $y_j = f(Z)$

• phase vers l'arrière

$\forall i \in L_{k-1}, j \in L_k$ calcul $\delta_j = (y_j - d_j) f'(Z_j)$ et $\Delta w_{i,j} = -\eta \delta_j y_i$

- $Pour\ l = \kappa - 1, \ldots, 1$ faire :

$\forall i \in L_{l-1}, j \in L_k$ calculé $\delta_j = \left(\sum_{k=1}^{m_{l+1}} \delta_k w_{j,k}\right) f'(Z_j)$, $\Delta w_{i,j} = -\eta \delta_j y_i$.

- Pour $l = 1, 2, \ldots, k\ \forall\ (i,j)\ \in L_{l-1} * L_l$, ajuste $w_{i,j}$ par $\Delta w_{i,j}$.

- Calculer l'erreur pour le model p: $E_P = \frac{1}{m_k} \sum_{j=1}^{m_k} \left(y_j^p - d_j^p\right)^2$

4. Définissez $E_{prev} = E_{moy}$ et comptez le nouveau $E_{moy} = \frac{1}{P} \sum_{p=1}^{p} (E_p)$ Pour l'ensemble de données de validation

5. Si $E_{moy} < E_{prev}$ passe à l'étape 2.

6. Terminer.

2.6. Réseaux des neurones convolutionnels (RNC) :

Le réseau neuronal convolutionnel (RNC), proposé à l'origine par LeCun [64], est un modèle de réseau neuronal avec trois idées architecturales clés: champs réceptifs locaux, partage des poids et sous-échantillonnage dans le domaine spatial. Le réseau est conçu pour la reconnaissance des motifs visuels bidimensionnels. Le réseau neuronal convolutionnel a beaucoup des points forts. Tout d'abord, l'extraction et la classification des caractéristiques sont intégrées dans une seule structure et sont entièrement adaptables. Deuxièmement, le réseau extrait les images 2D à des échelles dyadiques croissantes. Troisièmement, il est relativement invariant aux distorsions géométriques locales dans l'image. RNC a été utilisée dans plusieurs applications, y compris la reconnaissance numérique des chiffres, la détection des visages et la reconnaissance faciale.

Dans cette section, nous décrivons d'abord l'architecture de RNC, puis nous présentons un modèle mathématique détaillé du réseau.

2.6.1. Architecture :

Les réseaux des neurones convolutionnels sont conçus pour traiter l'image bidimensionnelle (2D). Un RNC se compose de trois types principaux des couches: couches de convolution, couches de sous-échantillonnage, et une couche de sortie. Les couches de réseau sont disposées dans une structure de vers l'avant : chaque couche de convolution est suivie d'une couche de sous-échantillonnage, et la dernière couche de convolution est suivie de la couche de sortie (voir la figure 2-07).

Les couches de convolution et de sous-échantillonnage sont considérées comme des couches 2D, alors que la couche de sortie est considérée comme une couche 1D. Dans RNC, chaque couche 2D comporte plusieurs plans. Un plan consiste en des neurones qui sont disposés dans un réseau 2D. La sortie d'un plan s'appelle une carte des caractéristiques.

Dans une couche convolutionnelle, chaque plan est connecté à une ou plusieurs cartes des caractéristiques de la couche précédente. Une connexion est associée à un masque de convolution, qui est une matrice 2D d'entrées réglables appelées poid. Chaque plan calcule d'abord la convolution entre ses entrées 2D et ses masques de convolution. Les sorties de convolution sont additionnées et ensuite ajoutées avec un scalaire réglable, appelé terme biais. Enfin, une fonction d'activation est appliquée sur le résultat pour obtenir la sortie du plan. La sortie du plan est une matrice 2D appelée une carte des caractéristiques; Ce nom provient parce que chaque sortie de convolution indique la présence d'une caractéristique visuelle sur un emplacement de pixel donné. Une couche de convolution produit une ou plusieurs cartes des caractéristiques. Chaque carte des caractéristiques est ensuite connectée à un plan exactement dans le prochain sous-échantillonnage.

Une couche de sous-échantillonnage a le même nombre des plans que la couche de convolution précédente. Un plan de sous-échantillonnage divise son entrée 2D en blocs non superposés de taille $2 * 2$ pixels. Pour chaque bloc, on calcule la somme des quatre pixels; Cette somme est multipliée par un poids réglable avant d'être ajoutée à un terme de biais. Le résultat est passé par une fonction d'activation pour produire une sortie pour le bloc $2 * 2$. De toute évidence, chaque plan de sous-échantillonnage réduit sa taille d'entrée de la moitié, le long de chaque dimension. Une carte de caractéristique dans une couche de sous-échantillonnage est connectée à un ou plusieurs plans dans la prochaine couche de convolution.

Dans la dernière couche de convolution, chaque plan est connecté à une carte des caractéristiques précédente. Cette couche utilise des masques de convolution qui ont exactement la même taille que ses cartes des caractéristiques d'entrée.

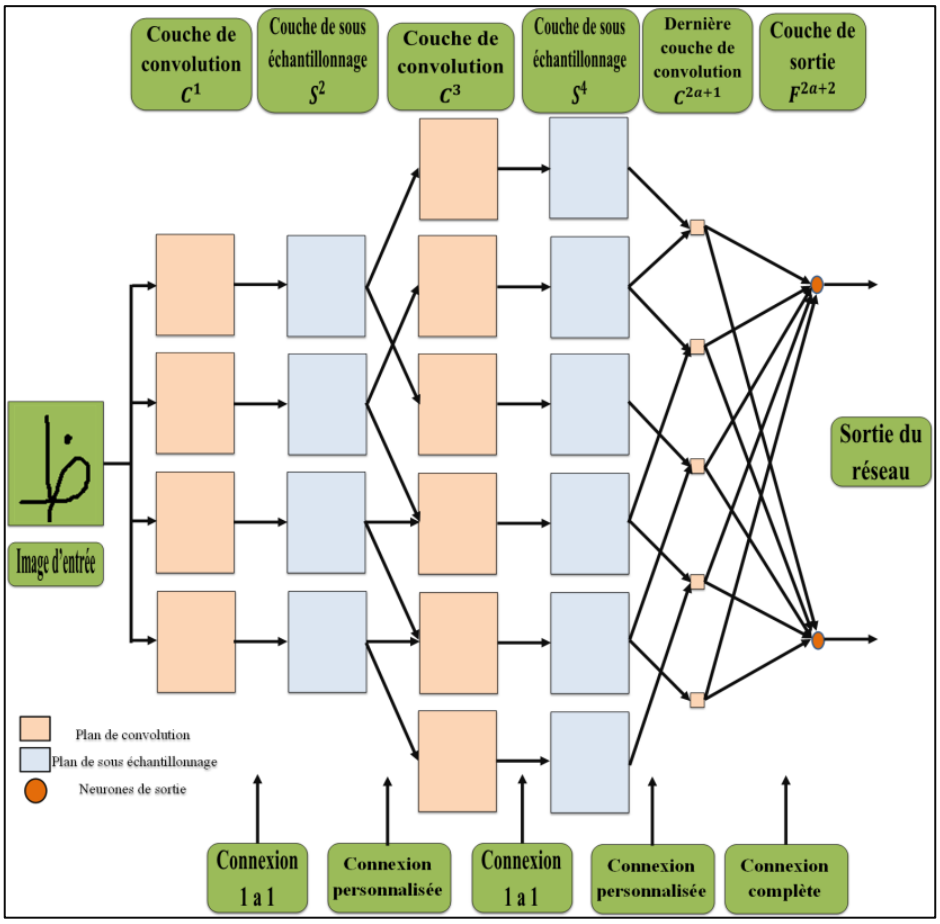


Figure 2-07 : Les couches d'un réseau de neurone convolutionnel.

Par conséquent, chaque plan dans la dernière couche de convolution produira une sortie scalaire. Les sorties de tous les plans de cette couche sont ensuite connectées à la couche de sortie.

La couche de sortie, en général, peut être construite à partir des neurones sigmoïdaux. Les sorties de cette couche sont considérées comme les sorties du réseau. Dans des applications telles que la classification des modèles, ces sorties indiquent la catégorie de l'image d'entrée.

2.6.2. Modèle mathématique :

Le tableau 2-1 résume les notations utilisées pour décrire les aspects fonctionnels de RNC. Le symbole l indique l'indice d'une couche de réseau. L'indice de couche $1 \leq l \leq L$, où L est le nombre de couches de réseau. Ici, on suppose que $L = 2a + 2$, où a est un nombre entier positif. Soit N^l le nombre de cartes de caractéristiques dans la couche l, et f^l soit la fonction d'activation de la couche l. Soit y_n^l la n-ième carte des caractéristiques (sortie) de la couche l.

La description	symbole
Taille de l'image d'entrée	$H^0 * W^0$
Pixel de l'image d'entrée	$x(i,j)$ ou $y_1^0(i,j)$
Indice de couche	l
Nombre des couches	$L = 2a + 2$
Couches de convolution	$C^1, C^3, \ldots\ldots, C^{2a+1}$
Couches de sous-échantillonnage	$S^2, S^4, \ldots\ldots, S^{2a}$
Couche de sortie	F^{2a+2}
Fonction d'activation de la couche l	f^l
Nombre des cartes des caractéristiques dans la couche l	N^l
Taille du masque de convolution pour la couche C^l	$r^l * c^l$
Masque de convolution de la carte des caractéristiques m dans la couche S^{l-1} à la carte des caractéristiques n dans la couche C^l	$\{w_{m,n}^l(i,j)\}$
Biais pour la carte des caractéristiques n dans la couche de convolution C^l	b_n^l
Poids pour la carte des caractéristiques n dans la couche S^l	w_n^l
Biais pour la carte de caractéristique n dans la couche de sous-échantillonnage S^l	b_n^l
Carte des caractéristiques n dans la couche l	$y_n^l(i,j)$
Taille d'une carte des caractéristiques dans la couche l	$H^l * W^l$

Tableau 2-1 : Notation d'architecture pour le RNC

a. Couche de convolution :

Considérons une couche de convolution l. Dans cette structure, l est un entier impair, $l = 1, 3, \ldots, 2a + 1$. Soit $H^l * W^l$ la taille du masque de convolution pour la couche l. Pour la carte des caractéristiques n, soit :

- $\mathrm{w}_{\mathrm{m,n}}^{\mathrm{l}} = \{\mathrm{w}_{\mathrm{m,n}}^{\mathrm{l}}(\mathrm{i},\mathrm{j})\}$ être le masque de convolution à partir de la carte des caractéristiques m dans la couche $(l-1)$ Pour représenter la carte n dans la couche l
- b_n^l le terme de biais associé à la carte des caractéristiques n,
- V_n^l désignent la liste de tous les plans en couche $(l-1)$ qui sont connectés à la carte des caractéristiques n. Par exemple, $V_4^l = \{2,3,5\}$ signifie que les cartes 2, 3 et 5 de la couche $(l-1)$ sont connectées à la carte 4 de la couche l.

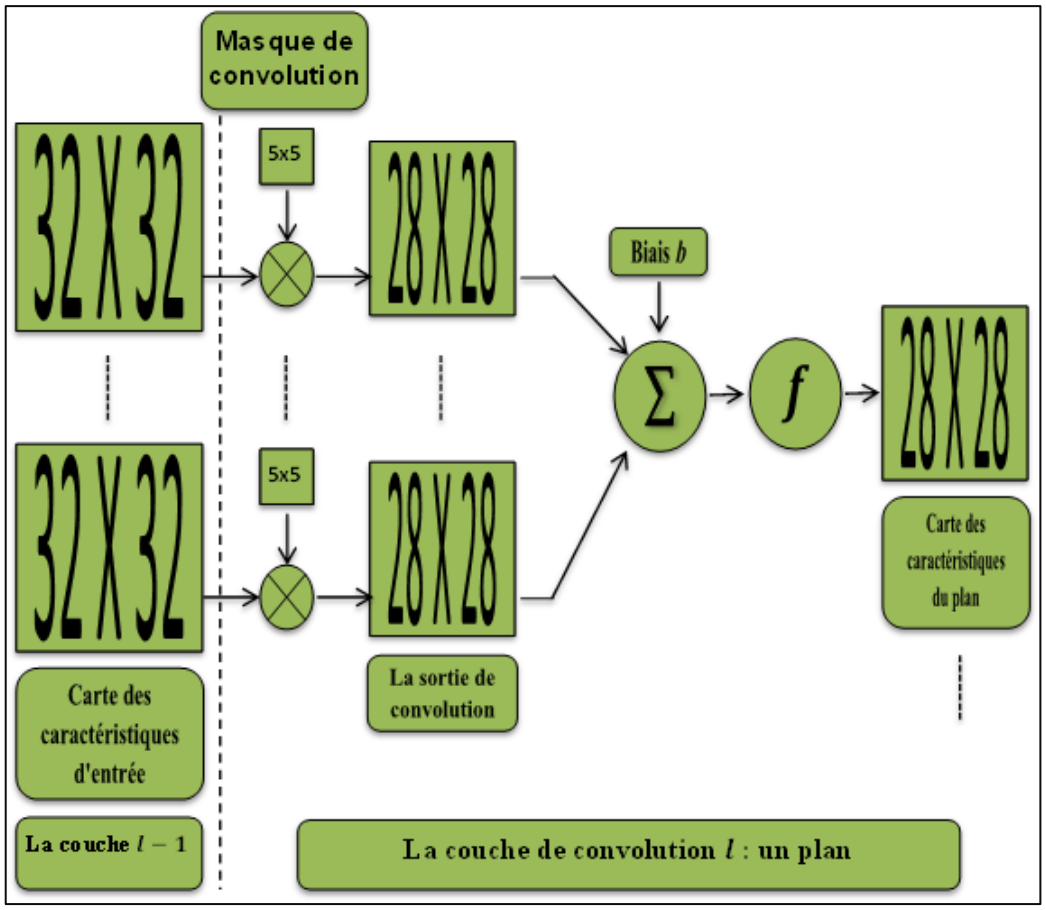


Figure 2-08 : Une couche de convolution dans le RNC.

Soit ⊗ l'opérateur de convolution 2D. La carte des caractéristiques n de la couche de convolution l est calculée par :

$$y_n^l = f^l \sum_{m \in V_n^l} y_n^{l-1} \otimes w_{m,n}^l + b_n^l \qquad (2.25)$$

Supposons que la taille des cartes des caractéristiques d'entrée y_n^{l-1} est $H^{l-1} * W^{l-1}$ pixels, et la taille des masques de convolution $w_{m,n}^l$ est $r^l * c^l$ pixels. La taille de la carte de sortie est $(H^{l-1} - r^l + 1) * (W^{l-1} - c^l + 1)$ pixels.

b. Couche de sous-échantillonnage :

Maintenant, nous considérons une couche de sous-échantillonnage l. Dans cette structure, l est un entier pair, $l = 2, 4, \ldots, 2a$. Pour la carte des caractéristiques n, soit le poid w_n^l et b_n^l soit le terme de biais. Nous divisons la carte des caractéristiques n de la couche de convolution $(l-1)$ en blocs non superposés de taille $2 * 2$ pixels. Soit z_n^{l-1} une matrice obtenue en additionnant les quatre pixels dans chaque bloc :

$$z_n^{l-1}(i,j) = y_n^{l-1}(2i-1,2j-1) + y_n^{l-1}(2i-1,2j) + y_n^{l-1}(2i,2j-1) + y_n^{l-1}(2i,2j) \quad (2.26)$$

Dans cette étape en peut obtenir la matrice z_n^{l-1} par le calcul du moyenne des quatre pixels dans chaque bloc ou bien en prend le pixel de la plus grande valeur parmi les quatre.

La carte des caractéristiques n de la couche de sous-échantillonnage l est maintenant calculée par :

$$y_n^l = f^l(z_n^{l-1} * w_n^l + b_n^l) \tag{2.27}$$

De toute évidence, une carte de caractéristique y_n^l dans la couche de sous-échantillonnage l aura une taille de $H^l * W^l$, où :

$$H^l = \frac{H^{l-1}}{2} \text{ et } W^l = \frac{W^{l-1}}{2}$$

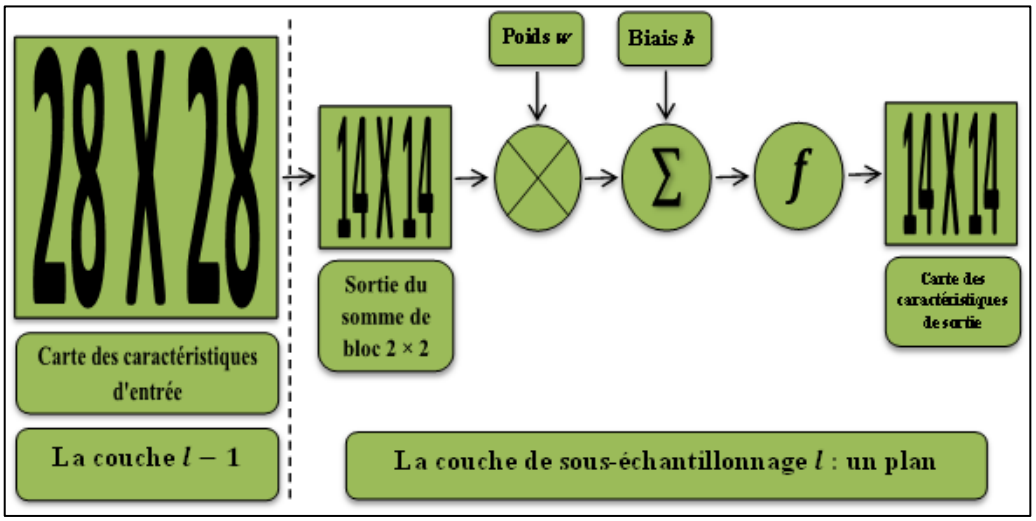


Figure 2-09 : Une couche de sous-échantillonnage dans le RNC

c. Couche de sortie :

Dans cette étude, nous considérons la couche de sortie L qui se compose des neurones sigmoïdaux. Soit N^L le nombre des neurones sigmoïdaux de sortie. Soit $w_{m,n}^L$ le poid de la carte m de la dernière couche convolutionnel, au neurone n de la couche de sortie. Soit b_n^l le terme de biais associé au neurone n de la couche L. La sortie du neurone sigmoïdal n est calculée par :

$$y_n^L = f^L\left(\sum_{m=1}^{N^{L-1}} y_m^{L-1} * w_{m,n}^L + b_n^l\right) \tag{2.28}$$

Les sorties de tous les neurones sigmoïdaux forment les sorties du réseau:

$$y = [y_1^L, y_2^L, \dots\dots, y_{N^L}^L]$$

2.6.3. L'apprentissage du RNC :

Pour que les RNC effectuent différentes tâches de reconnaissance visuelle, un algorithme d'apprentissage doit être conçu. Dans une application donnée, l'objectif d'apprentissage du réseau est de minimiser la fonction d'erreur, qui est définie en termes des sorties réelles du réseau et les sorties souhaitées.

Le tableau 2-2 résume les différentes définitions utilisées dans l'apprentissage du RNC. Supposons que l'ensemble d'apprentissage comporte K images d'entrée et K vecteurs de sortie souhaités. Soit x^k la k-ième image d'apprentissage, et d^k soit le vecteur de sortie souhaité correspondant. La fonction d'erreur est définie comme :

$$E(w) = \frac{1}{K * N^L} \sum_{k=1}^{K} \sum_{n=1}^{N^L} (y_n^k - d_n^k)^2 \tag{2.29}$$

Où y_n^k est la sortie réelle du réseau. Ceci est une fonction de tous les paramètres de réseau (poids et des biais).

La description	Symbole	Formule
Indice d'image de formation	k	$k = 1, 2, \ldots\ldots, K$
Image d'entrainement k	x^k	$x^k = \{x^k(i,j), i = 1,..,H^0; j = 1,..,W^0\}$
Exemple de sortie souhaité k	d^k	$d^k = (\, d_1^k,\, d_2^k, \ldots..,\, d_{N^L}^k)$
Entrée ou sortie réseau de la couche 0	$y^{0,k}(i,j)$	$y^{0,k}(i,j) = (x^k(i,j), i = 1,..,H^0; j = 1,..,W^0)$
Entrée de somme pondérée pour les neurones (u,v) dans la couche de convolution l, la carte des caractéristiques n	$s_n^{l,k}(u,v)$	$n = 1, \ldots..., N^l$ $u = 1, \ldots..., H^l; v = 1, \ldots..., W^l$
Sortie des neurones (u,v) dans la couche de convolution l, la carte des caractéristiques n pour l'image d'entrée k	$y_n^{l,k}(u,v)$	$y_n^{l,k}(u,v) = f^l(\, s_n^{l,k}(u,v))$
Entrée de somme pondérée pour les neurones (u, v) dans la couche de sous-échantillonnage l, la carte des caractéristiques n	$s_n^{l,k}(u,v)$	$n = 1, \ldots..., N^l$ $u = 1, \ldots..., H^l; v = 1, \ldots..., W^l$
Sortie des neurones (u,v) dans la couche de sous-échantillonnage l, carte des caractéristiques n	$y_n^{l,k}(u,v)$	$y_n^{l,k}(u,v) = f^l(\, s_n^{l,k}(u,v))$
La n-ième erreur pour l'image k	e_n^k	$e_n^k = \, y_n^{l,k} - \, d_n^k$
Fonction d'erreur	$E(w)$	$E(w) = \frac{1}{K * N^L} \sum_{k=1}^{K} \sum_{n=1}^{N^L} (y_n^k - d_n^k)^2$
Sensibilité d'erreur du pixel (u,v) dans la couche 2D	$\xi_{u,v}^{l,k}$	$\xi_{u,v}^{l,k} = \frac{\partial E}{\partial\, s_{u,v}^{l,k}}\ l \le L_p$
Erreur de sensibilité du neurone n dans la couche 1D l	$\xi_n^{l,k}$	$\xi_n^{l,k} = \frac{\partial E}{\partial\, s_n^{l,k}}\ l \ge L_p$

Tableau 2-2 : Notation pour l'algorithme d'apprentissage de RNC.

Il existe deux approches majeures pour l'apprentissage du RNC. La première approche, appelée l'apprentissage en ligne, met à jour les paramètres du réseau après chaque échantillon d'apprentissage. Cette approche nécessite moins de mémoire, mais elle est moins stable car chaque échantillon peut pousser les paramètres du réseau selon une nouvelle direction. La deuxième approche, connue sous le nom d'apprentissage par lot, met à jour les poids et les biais du réseau après l'introduction de tous les échantillons d'apprentissage. Cette approche nécessite un stockage de mémoire important car elle doit accumuler les modifications apportées aux paramètres du réseau. Puisque l'apprentissage en ligne est un cas particulier de formation par lots (lorsque $K = 1$), nous nous concentrerons sur l'apprentissage par lots. Notez que dans l'apprentissage par lot, une évaluation des résultats du réseau pour tous les échantillons et une mise à jour de tous les paramètres du réseau sont appelés collectivement comme une période d'apprentissage.

a. Dérivation du gradient d'erreur de RNC :

Cette section présente une méthode pour calculer le gradient de la fonction d'erreur définie en équation 2.29 . Nous discutons de la définition de la sensibilité aux erreurs et des méthodes pour calculer la sensibilité aux erreurs et le gradient de l'erreur.

a.1. Définition de la sensibilité d'erreur :

Le gradient d'erreur est calculé par des sensibilités d'erreur, qui sont définies comme les dérivées partielles de la fonction d'erreur par rapport à l'apport de somme pondéré à un neurone.

Pour le neurone (i, j) dans la carte des caractéristiques n de la couche de convolution l, sa sensibilité aux erreurs est définie par :

$$\xi_n^{l,k}(i,j) = \frac{\partial E}{\partial\, s_n^{l,k}(i,j)}, \text{pour } l = 1, 3, \dots\dots, 2a+1 \tag{2.30}$$

Pour le neurone (i, j) dans la carte des caractéristiques n de la couche de sous-échantillonnage l, sa sensibilité à l'erreur est donnée par :

$$\xi_n^{l,k}(i,j) = \frac{\partial E}{\partial\, s_n^{l,k}(i,j)}, \text{pour } l = 2, 4, \dots\dots, 2a \tag{2.31}$$

Pour le neurone n dans la couche de sortie L, sa sensibilité aux erreurs est :

$$\xi_n^{L,k} = \frac{\partial E}{\partial\, s_n^{L,k}} \tag{2.32}$$

a.2. Calcul de la sensibilité des erreurs :

En utilisant le tableau 2-2 et la règle de la chaîne de différenciation, nous pouvons exprimer les sensibilités d'erreur comme suit.

- Couche de sortie: $l = L$

$$\xi_n^{L,k} = \frac{2}{K * N^L} * e_n^k * f'^L\left(s_n^{L,k}\right), \text{pour } n = 1, 2, \dots, N^L \tag{2.33}$$

- Dernière couche de convolution: $l = L - 1$

$$\xi_n^{l,k} = f'^l\left(s_n^{l,k}\right) \sum_{m=1}^{N^{L+1}} \xi_n^{l+1,k} * w_{n,m}^{l+1}, \text{pour } n = 1, 2, \dots, N^l \tag{2.34}$$

Parce que les cartes des caractéristiques dans la couche de convolution $(L-1)$ ont une taille de $1 * 1$ pixel, l'écriture $\xi_n^{L-1,k}$ ou $\xi_n^{L-1,k}(1,1)$ est équivalente.

- Dernière couche de sous-échantillonnage : $l = L - 2$

$$\xi_n^{l,k}(i,j) = f^l\left[s_n^{l,k}(i,j)\right] * w_{n,m}^{l+1}(i,j) * \xi_n^{l+1,k}(1,1) \tag{2.35}$$

$$o\grave{u}\ n = 1, 2, \dots, N^{L-2}\ ; i = 1, 2, \dots, H^{L-2}; j = 1, 2, \dots, W^{L-2}$$

- Autre couche de convolution: $l = 2a + 1$

$$\xi_n^{l,k}(i,j) = f^l\left[s_n^{l,k}(i,j)\right] * w_n^{l+1}(i',j') * \xi_n^{l+1,k}, o\grave{u}\ i' = \frac{i}{2}; j' = \frac{j}{2} \tag{2.36}$$

- Autre couche de sous-échantillonnage: $l = 2a$

$$\xi_n^{l,k}(i,j) = f^l\left[s_n^{l,k}(i,j)\right] * \sum_{m \in U_n^l} \sum_{(i',j') \in R^l(i,j)} \xi_n^{l+1,k}(i',j') * w_{n,m}^{l+1}(i-i', j-j') \tag{2.37}$$

a.3. Calcul du gradient d'erreur :

- Couche de sortie: $l = L$
  - ✓ Poids $w_{m,n}^l$:

$$\frac{\partial E}{\partial w_{m,n}^l} = \sum_{k=1}^{K} \xi_n^{l,k} * y_m^{l-1,k} \quad o\grave{u} \quad m = 1, 2, \dots, N^l\ ; n = 1, 2, \dots, N^l \tag{2.38}$$

  - ✓ Biais b_n^l :

$$\frac{\partial E}{\partial b_n^l} = \sum_{k=1}^{K} \xi_n^{l,k}, o\grave{u}\ n = 1, 2, \dots, N^l \tag{2.39}$$

- Dernière couche de convolution: $l = L - 1$
  - ✓ Poids $w_{m,n}^l(i,j)$:

$$\frac{\partial E}{\partial w_{m,n}^l(i,j)} = \sum_{k=1}^{K} \xi_n^{l,k}(i,j) * y_{n,n}^{l-1,k}, o\grave{u}\ n = 1, 2, \dots, N^l \tag{2.40}$$

- ✓ Biais b_n^l :

$$\frac{\partial E}{\partial b_n^l} = \sum_{k=1}^{K} \sum_{(i,j)} \xi_n^{l,k}(i,j)\,, o\grave{u}\ n = 1, 2, \dots, N^l \tag{2.41}$$

- Couche de sous-échantillonnage: $l = 2a$
  - ✓ Poids w_n^l :

$$\frac{\partial E}{\partial w_n^l} = \sum_{k=1}^{K} \sum_{(i',j')} \xi_n^{l,k}(i',j') * y_{n,n}^{l-1,k}(i',j')\,, o\grave{u}\ n = 1, 2, \dots, N^l \tag{2.42}$$

  - ✓ Biais b_n^l :

$$\frac{\partial E}{\partial w_n^l} = \sum_{k=1}^{K} \sum_{(i',j')} \xi_n^{l,k}(i',j')\,, o\grave{u}\ n = 1, 2, \dots, N^l \tag{2.43}$$

- Autre couche de convolution: $l = 2a + 1$
  - ✓ Poids $w_{m,n}^l(i,j)$:

$$\frac{\partial E}{\partial w_{m,n}^l(i,j)} = \sum_{k=1}^{K} \xi_n^{l,k}(i,j) * y_{n,n}^{l-1,k}\,, o\grave{u}\ n = 1, 2, \dots, N^l \tag{2.44}$$

  - ✓ Biais b_n^l :

$$\frac{\partial E}{\partial b_n^l} = \sum_{k=1}^{K} \sum_{(i,j)} \xi_n^{l,k}(i,j)\,, o\grave{u}\ n = 1, 2, \dots, N^l \tag{2.45}$$

b. Algorithmes d'apprentissage du RNC :

Une fois que le gradient d'erreur $\nabla E(t)$ est dérivé, des nombreux algorithmes d'optimisation pour minimiser E peuvent être appliqués pour former le réseau. Ici, nous nous concentrons sur cinq algorithmes d'apprentissage représentatifs:

- descente de gradient [61],
- descente de gradient avec momentum et taux d'apprentissage variable [65],
- rétro-propagation élastique [66],
- gradient conjugué [67],
- Levenberg-Marquardt [68].

Les trois premiers algorithmes sont des méthodes d'optimisation de premier ordre. L'algorithme de gradient conjugué peut être considéré comme intermédiaire entre les méthodes de premier et second ordre, tandis que l'algorithme de Levenberg-Marquardt est une méthode de région de confiance qui utilise l'approximation de Gauss-Newton de la matrice de Hessian. Puisque les détails de ces algorithmes peuvent être trouvés dans les références données, nous résumons ici leurs principales caractéristiques (voir tableau 2-3).

Algorithme	**La description**
descente de gradient	Les poids sont mis à jour le long du gradient négatif $\Delta w(t) = -\eta * \nabla E(t)$, η est le taux d'apprentissage scalaire, $\eta > 0$
descente de gradient avec momentum et taux d'apprentissage variable	La mise à jour des poids est une combinaison linéaire de gradient et de mise à jour de poids précédente $\Delta w(t) = \lambda \Delta w(t-1) - (1-\lambda) * \eta(t) * \nabla E(t)$ λ Est le paramètre momentum, $0 < \lambda < 1$ $\eta(t)$ Est le taux d'apprentissage scalaire adaptatif
rétro-propagation élastique	La mise à jour du poids dépend uniquement du signe du gradient $\Delta w_i(t) = -\text{sign}\left\{\frac{\partial E}{\partial w_i}(t)\right\} * \Delta_i(t)$ $\Delta_i(t)$ Est une étape d'adaptation spécifique au poids w_i, définie par : $\Delta_i(t) = \begin{cases} \eta^+ * \Delta_i(t-1), \text{si } \frac{\partial E}{\partial w_i}(t) * \frac{\partial E}{\partial w_i}(t-1) > 0 \\ \eta^- * \Delta_i(t-1), \text{si } \frac{\partial E}{\partial w_i}(t) * \frac{\partial E}{\partial w_i}(t-1) < 0 \\ \Delta_i(t-1), \text{autrement} \end{cases}$ $\eta^+ > 1, 0 < \eta^- < 1$ Termes scalaires
gradient conjugué	Les poids sont mis à jour le long des directions mutuellement conjuguées Matrice hessienne $\Delta w(t) = \eta(t) * s(t)$ Où le sens de recherche défini par : $s(t) = \begin{cases} -\nabla E(t), \text{ si } t = 1 \\ -\nabla E(t) + \beta(t) * s(t-1), \text{ autrement} \end{cases}$ L'étape d'apprentissage $\eta(t)$ se trouve à travers une recherche par ligne[69] $\beta(t) = \frac{[\nabla E(t) - \nabla E(t-1)]^T * \nabla E(t)}{\|\nabla E(t-1)\|^2}$
Levenberg-Marquardt	L'expansion de Taylor de 2ème ordre et l'approximation de Gauss-Newton de la matrice hessienne $\Delta w(t) = -[\mathbf{J}^{\mathrm{T}}\mathbf{J} + \mathrm{u}\mathbf{I}]^{-1} * \nabla E$, **J** est la matrice jacobienne définie par : $\mathbf{J}_{(q-1)K+k,i} = \frac{\partial e_q^k}{\partial w_i}$, $q = 1, 2, \dots, N_L; k = 1, 2, \dots, K; i = 1, 2, \dots, P$ Le gradient ∇E est calculé à travers la matrice jacobienne **J**. u est un paramètre adaptatif contrôlant la taille de la région de confiance.

Tableau 2-3 : Algorithmes d'apprentissage du RNC.

Le calcul de la matrice jacobienne est similaire au calcul du gradient ∇E. Cependant, nous devons modifier les définitions des sensibilités aux erreurs. Pour la matrice jacobienne, les sensibilité aux erreurs sont définies pour chaque erreur de réseau e_q^k, où $q = 1, 2, \dots, N_L$, au lieu de la fonction d'erreur.

2.7. Conclusion :

Dans ce chapitre, nous avons présenté le réseau de neurone convolutionnel et divers types des réseaux des neurones (simple et multi couche). Nous avons expliqué comment ils ont été construits et formés, et ont montré des exemples de leur application individuelle, mais on a basée beaucoup plus sur le RNC parce que notre application de reconnaissance des chiffres et des caractères arabes manuscrits dans le chapitre suivants sera basée à ce réseau.

Chapitre 3

Application des réseaux convolutionnels a la reconnaissance

3.1. Introduction :

Les systèmes de reconnaissance des caractères et des chiffres arabes manuscrits font face à plusieurs défis, y compris la variation illimitée de l'écriture humaine et des grandes bases des données publiques. Dans ce chapitre, nous utilisons les réseaux des neurones convolutionnels qui peut être appliqué efficacement à la reconnaissance des caractères arabes manuscrits. Donc on va voir dans ce chapitre l'utilisation du RNC dans deux modes de reconnaissance (chiffres et caractères).

3.2. Préparation de l'apprentissage et le test des données :

Le réseau de neurone convolutionnel nécessite une grande donnée de formation des images à des caractères manuscrits pour obtenir un bon résultat. Les données disponibles pour la formation sont divisées en deux ensembles différents: Ensemble d'apprentissage et Ensemble de validation. Il ne devrait pas y avoir de chevauchement entre ces deux ensembles des données afin d'améliorer la capacité de généralisation d'un réseau neuronal. Cette technique s'appelle la validation croisée [71]. Les performances réelles d'un réseau ne sont révélées que lorsque le réseau est testé avec des données de test pour mesurer le rendement du réseau sur les données qui n'ont pas été vues pendant l'apprentissage. Le test est conçu pour accéder à la capacité de généralisation de réseau. Une bonne généralisation signifie que le réseau fonctionne correctement sur des données similaires, mais différentes des données d'apprentissage.

3.3. La reconnaissance des chiffres manuscrits :

3.3.1. La base des données MNIST :

La base des données MNIST (Base des données de l'Institut National de Normes et de Technologie Modifiée)[72], est une grande base de données des chiffres manuscrits couramment utilisés pour la formation des différents systèmes de traitement d'image surtout les réseaux des neurones convolutionnels. La base des données MNIST contient 60 000 images utilisées pour l'apprentissage et 10 000 images pour le test. Dans la base des données MNIST, les images sont en niveaux de gris en raison de technique anti crénelage utilisé par

l'algorithme de normalisation et les images ont été centrées dans une taille $28 * 28$ pixels. La figure 3-01 montre un exemple des images des chiffres de la base des données MNIST.

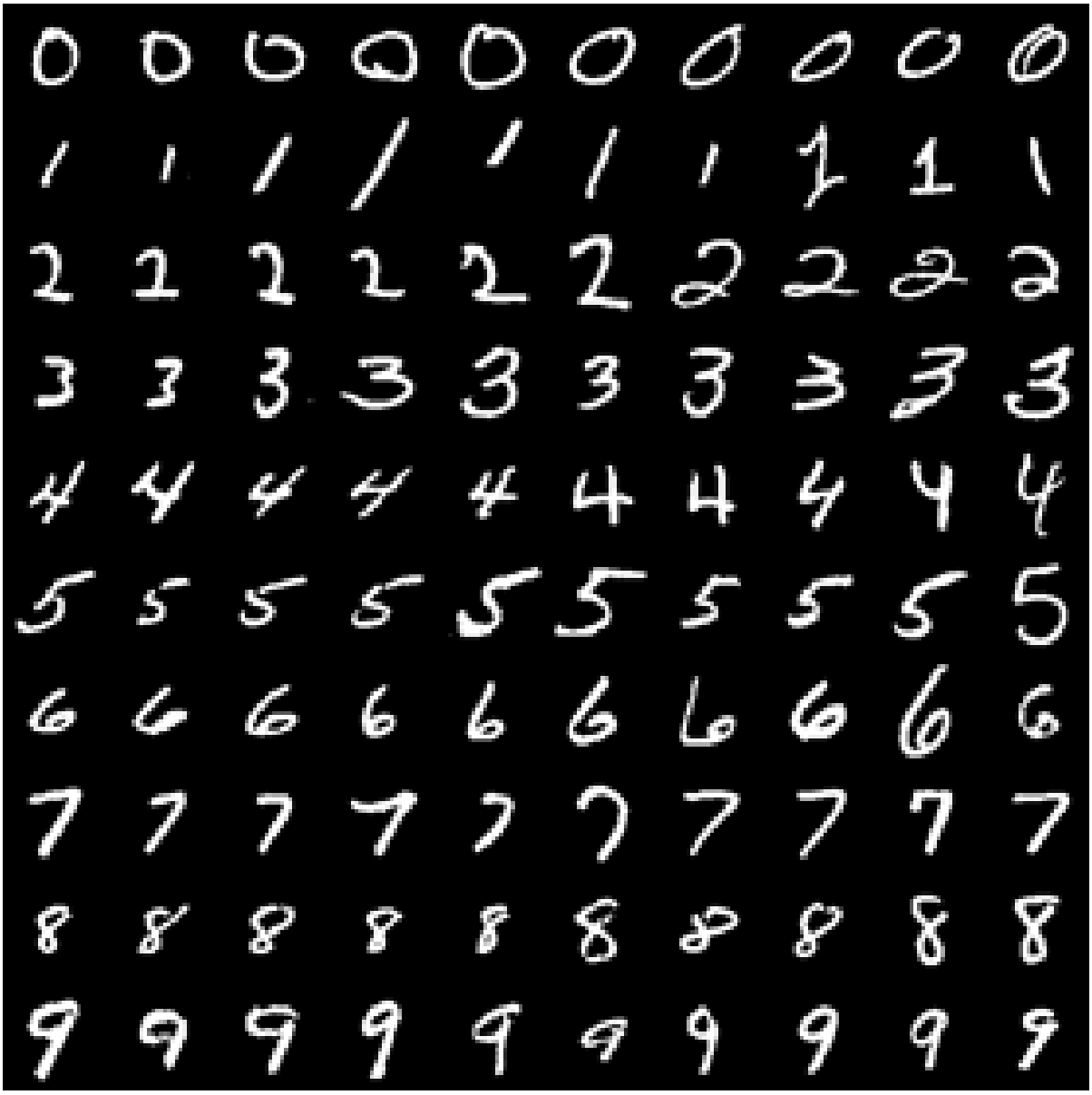


Figure 3-01 : Exemples de la base des données MNIST.

3.3.2. Architecture du réseau :

Pour la reconnaissance des chiffres manuscrite on va utiliser un réseau de neurone convolutionnel à une structure exactement pareil à la figure 3-02.

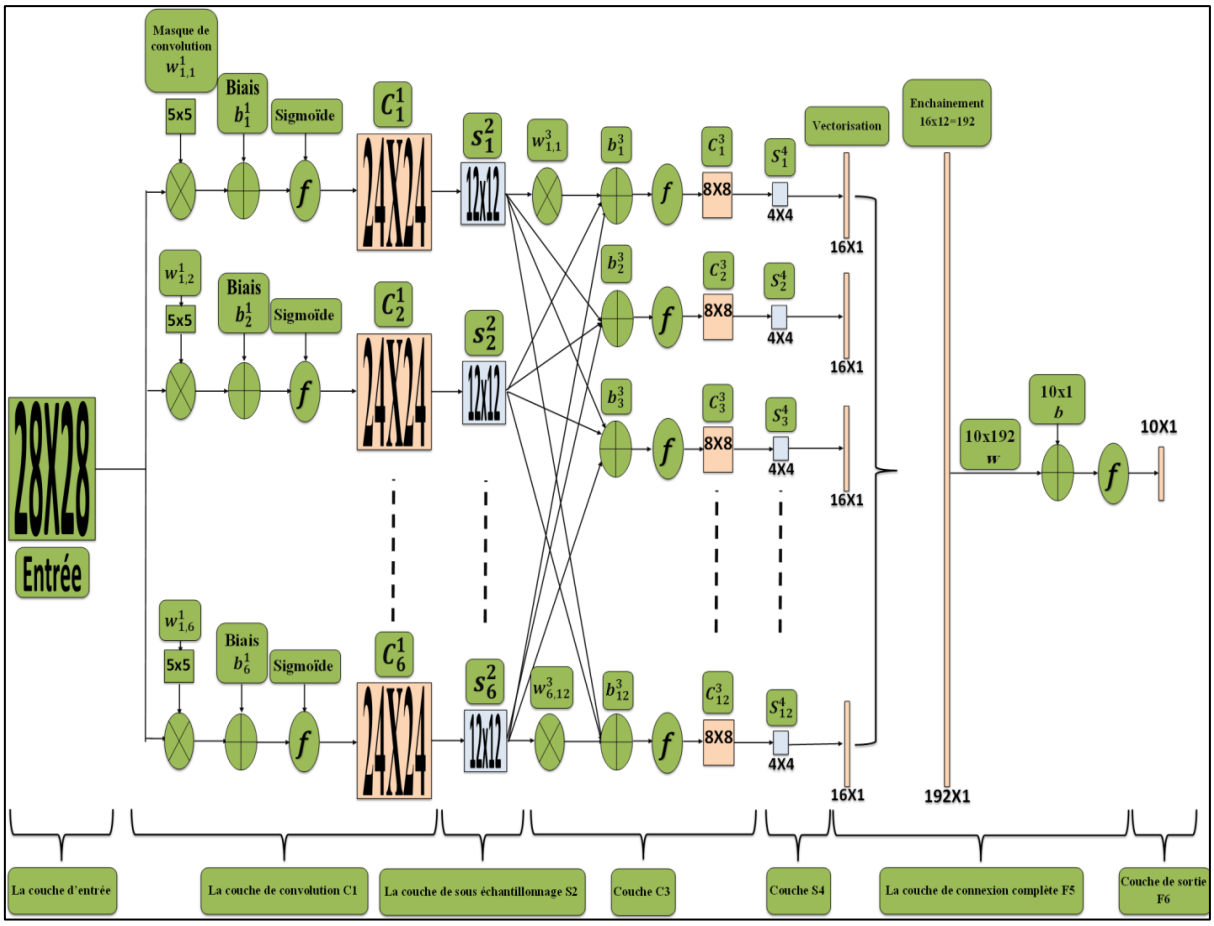


Figure 3-02 : Structure de RNC utilisé à la reconnaissance des chiffres manuscrits.

a. Initialisation des paramètres :

Les paramètres sont :

- **La couche $C1$:** $w_{1,n}^1$ (taille $5x5$) et b_n^1 (taille $1x1$), $n = 1, 2, \dots, 6$.
- **La couche $C3$:** $w_{m,n}^3$ (taille $5x5$) et b_n^3 (taille $1x1$), $n = 1, 2, \dots, 12$ et $m = 1,2 \dots ,6$.
- **La couche F :** w (taille $10x192$) et b (taille $10x1$).
- Toutes les biais, b_n^1, b_n^3 et b, sont initialisées à zéro.

b. Couche de convolution C1 :

$$C_n^1 = f\left(I * w_{1,n}^1 + b_n^1\right) ou\ f(x) = \frac{1}{1+e^{-x}} \tag{3.1}$$

$$C_n^1(i,j) = f\left(\sum_{u=0}^{4}\sum_{v=0}^{4} I(i+u, j+v) * w_{1,n}^1(u+1, v+1) + b_n^1\right) \tag{3.2}$$

Où $n = 1, 2, \cdots, 6$ car il existe 6 cartes des caracterstiques dans la couche C1, la taille de C_n^1 est $24 * 24$, plutôt que l'image d'entree I et de taille $28 * 28$.

c. Couche de sous-échantillonnage S2 :

$$S_n^2(i,j) = \frac{1}{4}\sum_{u=0}^{1}\sum_{v=0}^{1} C_n^1(2i-u, 2j-v) \text{ et } i,j = 1,2,\dots,12 \tag{3.3}$$

d. Couche de convolution C3 :

$$C_n^3 = f\left(\sum_{m=1}^{6} S_m^2 * w_{m,n}^3 + b_n^3\right) \tag{3.4}$$

$$C_n^3(i,j) = f\left(\sum_{m=1}^{6}\sum_{u=-2}^{2}\sum_{v=-2}^{2} S_m^2(i-u, j-v)\, w_{m,n}^3 + \ b_n^3\right) \tag{3.5}$$

Où $n = 1,2,\cdots,12$ car il existe 12 cartes des caractéristiques dans la couche $C3$, la taille de C_n^3 est $8 * 8$, plutôt que la taille de S_m^2 et $12 * 12$.

e. Couche de sous-échantillonnage S4 :

$$S_n^4(i,j) = \frac{1}{4}\sum_{u=0}^{1}\sum_{v=0}^{1} C_n^3(2i-u, 2j-v) \text{ et } i,j = 1,2,\dots,4 \tag{3.6}$$

f. Dernier couche de convolution (Vectorisation et enchaînement) :

Chaque S_n^4 est une matrice $4 * 4$, et il existe 12 de telles matrices sur la couche $S4$. Tout d'abord, chaque S_n^4 est vectorisé par balayage de colonne, alors tous les 12 vecteurs sont concaténés pour former un vecteur long avec une longueur de $4 * 4 * 12 = 192$. Nous désignons ce processus par :

$$g = G\left(\{S_n^4\}_{n=1,2,\dots,12}\right) \tag{3.7}$$

Et le processus inverse est :

$$\{S_n^4\}_{n=1,2,\dots,12} = G^{-1}(g) \tag{3.8}$$

g. Couche de connexion complète F :

$$y = f(w * g + b) \tag{3.9}$$

3.3.3. Résultats des Simulations :

Pour connaître les différents paramètres qui contrôlent la qualité de notre programme On va discuter les différentes résultats obtenus à chaque changements des paramètres .les résultats obtenu va ne conduire donc à choisir les paramètres les plus efficace. Nous avons mené des nombreux essais, dans le but de savoir comment obtenir des résultats efficaces. Ces résultats peuvent être connus par la valeur de l'erreur obtenue dans le processus de

reconnaissance. Et comme le réseau de neurone convolutionnel est basé sur l'apprentissage, donc en va jouer seulement sur le nombre d'époque ou bien le nombre d'itération.

Le test a était fait avec toutes la base des données qui contiens 10 000 chiffre. Les valeurs de l'erreur obtenue pour des différentes époques sont montrées dans le tableau 3-1 :

Nombre d'époque	Taux d'erreur
1	11.22%
10	2.62%
20	2.02%
30	1.62%
40	1.61%
50	1.41%

Tableau 3-1 : Le taux d'erreur avec différentes époques.

D'après le tableau on conclut que le taux d'erreur démunie lorsque on augmente le nombre d'époque. Donc pour obtenir un minimum de taux d'erreur ou bien un maximum de taux de reconnaissance notre choix doit respecter un nombre d'époque élevée. La figure 3-03 montre la diminution de l'erreur quadratique moyenne par rapport au nombre d'époque.

L'exécution du programme pendant 50 époque nous a prenait un délai de 8598 seconds (172sec/époque). Cette réalisation a été faite avec matlab 2016a dans un micro portable avec les caractéristiques suivantes : windows7 64 bits, microprocesseur duel-core 2.0Ghz, RAM 4Ghz.

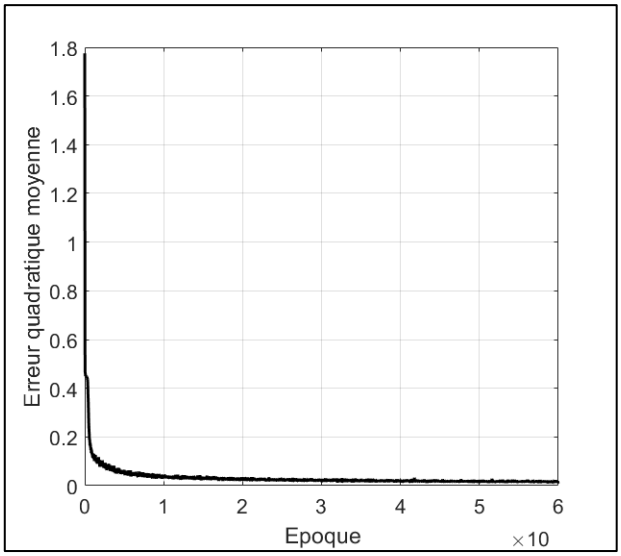


Figure 3-03 : L'erreur quadratique moyenne pendant l'apprentissage.

On peut constater que nous atteignons des taux de mal classification très passants de 0.0%, 0.0%, pour les chiffres (1, 0) comme le montre la figure 3-05. La totalité de la mauvaise

classification est de 141 chiffres (figure 3-04). Cela montre l'efficacité de reconnaissance de notre RNC à classer les chiffres manuscrits.

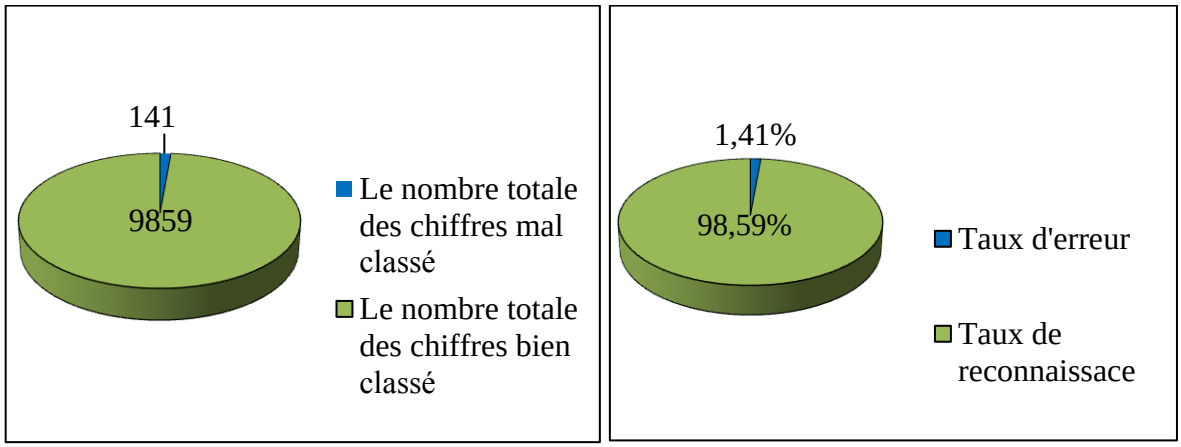


Figure 3-04 : Taux d'erreur (mal classé) et taux de reconnaissance (bien classé).

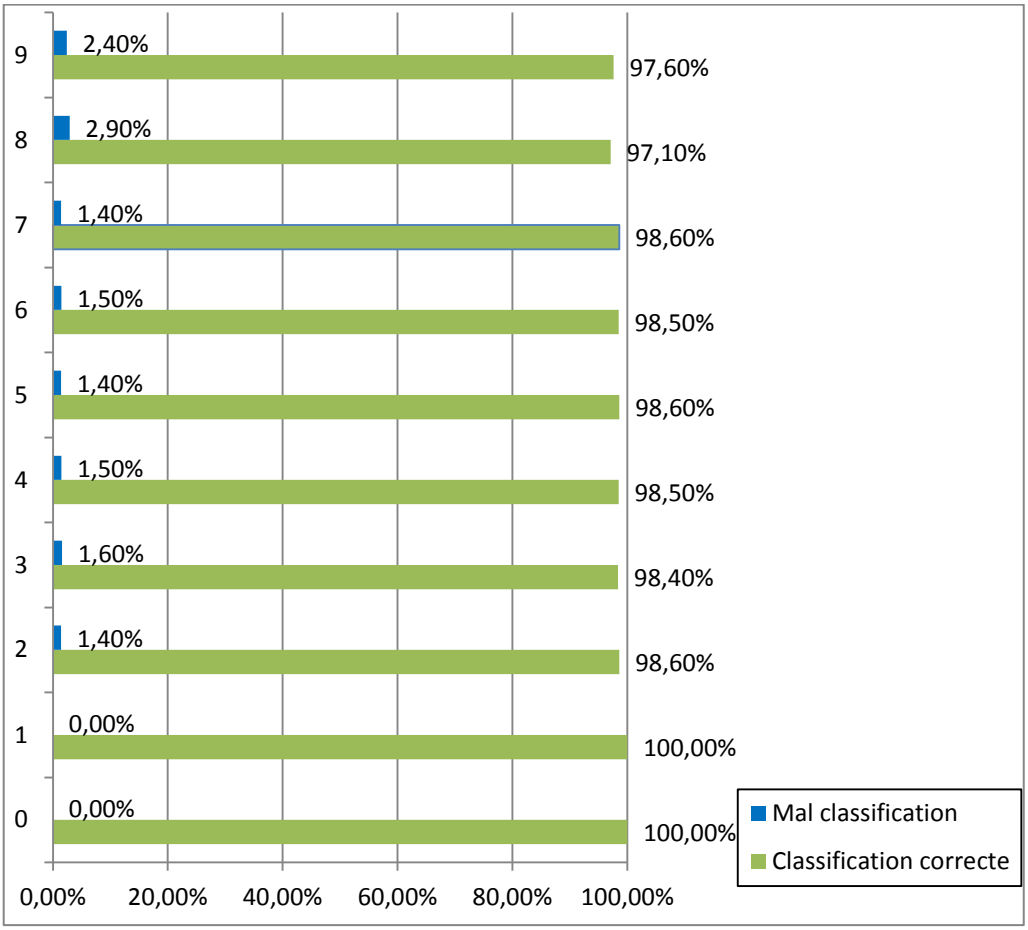


Figure 3-05 : Précision du classement des caractères par classe.

D'après la figure 3-05 nous remarquons qu'il n y a pas des grandes erreurs a la classification sauf pour les chiffres 8 et 9 à cause de similitude avec des autres chiffres mais cette erreur reste toujours un peu négligeable.

La figure 3-06 nos montre l'architecture du réseau RNC et le résultat de classification pour le chiffre 5 cette simulation a était réalisé a 50 époques d'apprentissage.

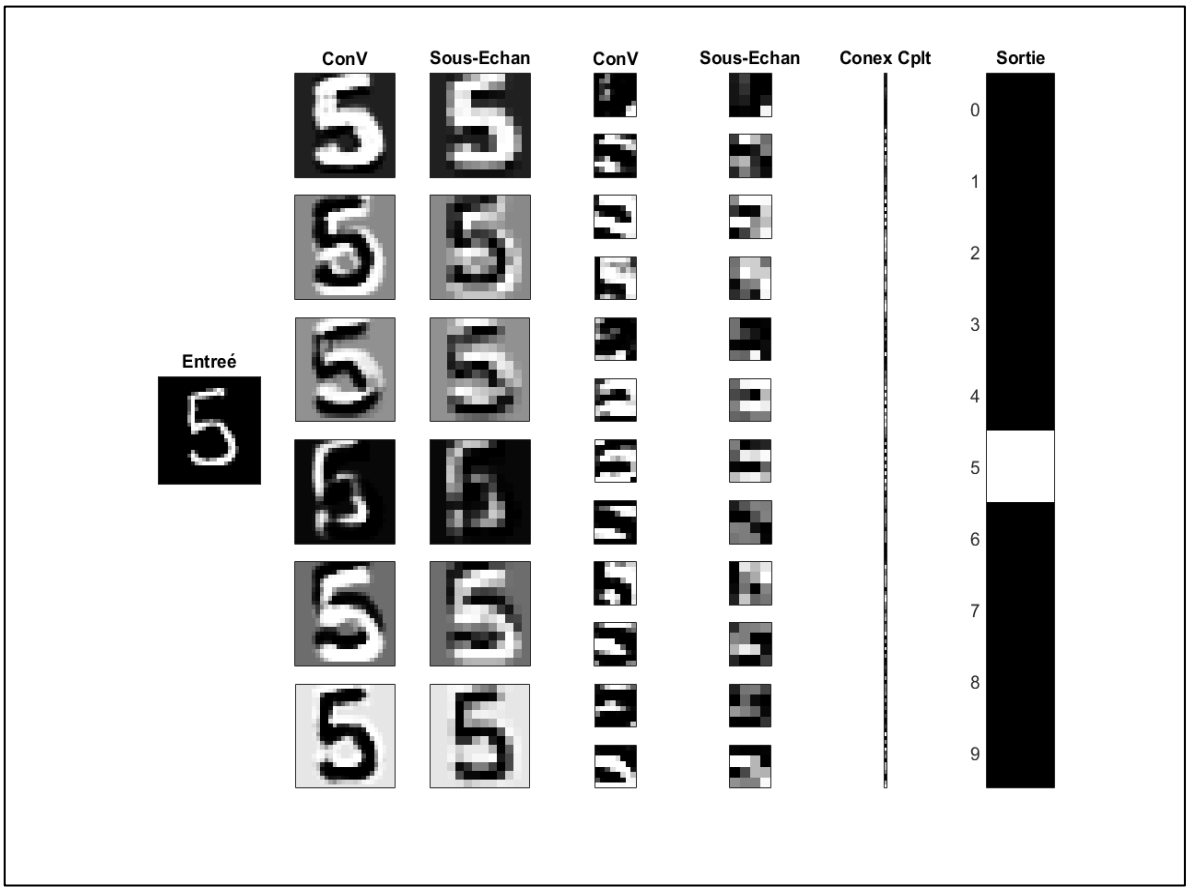
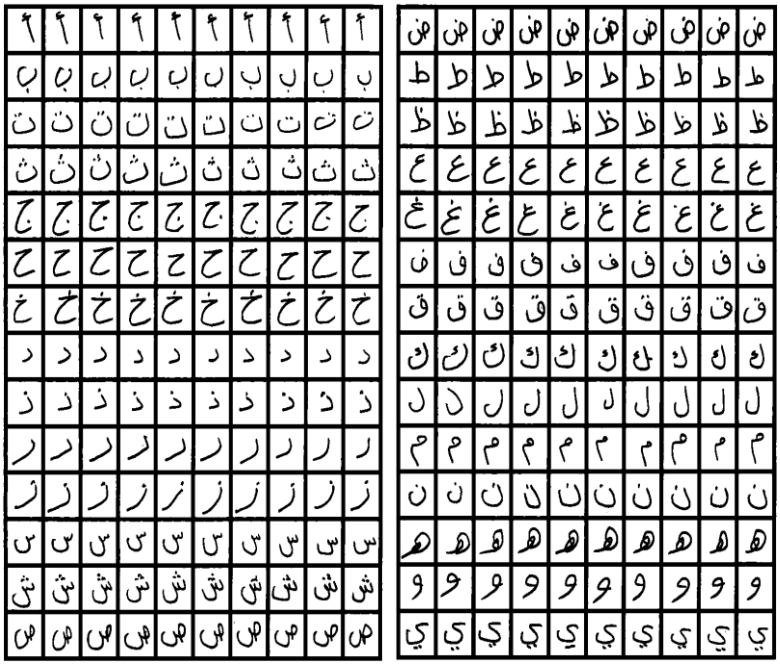


Figure 3-06 : Un exemple de classification.

3.4. La reconnaissance des caractères arabes manuscrits :

3.4.1. La base des données pour les caractères arabes manuscrits :

Cette base des données a était écrites par 60 participants leurs âge est comprise entre 19 et 40 ans, et 90% des participants utilise la main droite. Chaque participant a écrit chaque caractère (de 'ا' à 'ي') dix fois sur deux formes comme le montre la Figure 3-07 (a) et (b). Les formulaires ont été scannés à la résolution de 300 dpi. Chaque bloc est segmenté automatiquement à l'aide de Matlab 2016a pour déterminer les coordonnées de chaque bloc. L'ensemble des données est composé de 16 800 caractères. Ces caractères est partitionnée en deux ensembles: un ensemble pour l'apprentissage contiens 13,440 caractères à 480 images par classe, et un ensemble pour le test contiens 3 360 caractères à 120 images par classe [70].



(a) (b)

Figure 3-07 : Exemples de la base des données des caractères arabes

3.4.2. Architecture du réseau :

Pour la reconnaissance des caractères manuscrite, la même architecture est utilisé comme dans la reconnaissance des chiffres, mais avec des changements comme il est indiquée dans le tableau 3-2 :

Architecture / Type de reconnaissance	N^bre des cartes						Fonction d'activation			N^bre des neurones à chaque carte						
	C1	S2	C3	S4	F5	F6	C1	C3	F6	E	C1	S2	C3	S4	F5	F6
Reconnaissance des chiffres	6	6	12	12	1	1	sigmoïde			28*28	24*24	12*12	8*8	4*4	192*1	10
Reconnaissance des caractères	32	32	64	64	1	1	Relu		soft max	32*32	28*28	14*14	10*10	5*5	1600*1	28

Tableau 3-2 : Comparaison entre le RNC des chiffres et le RNC des caractères.

Donc le réseau utilise pour la reconnaissance des caractères arabes manuscrits sera pareil a la figure 3.08.

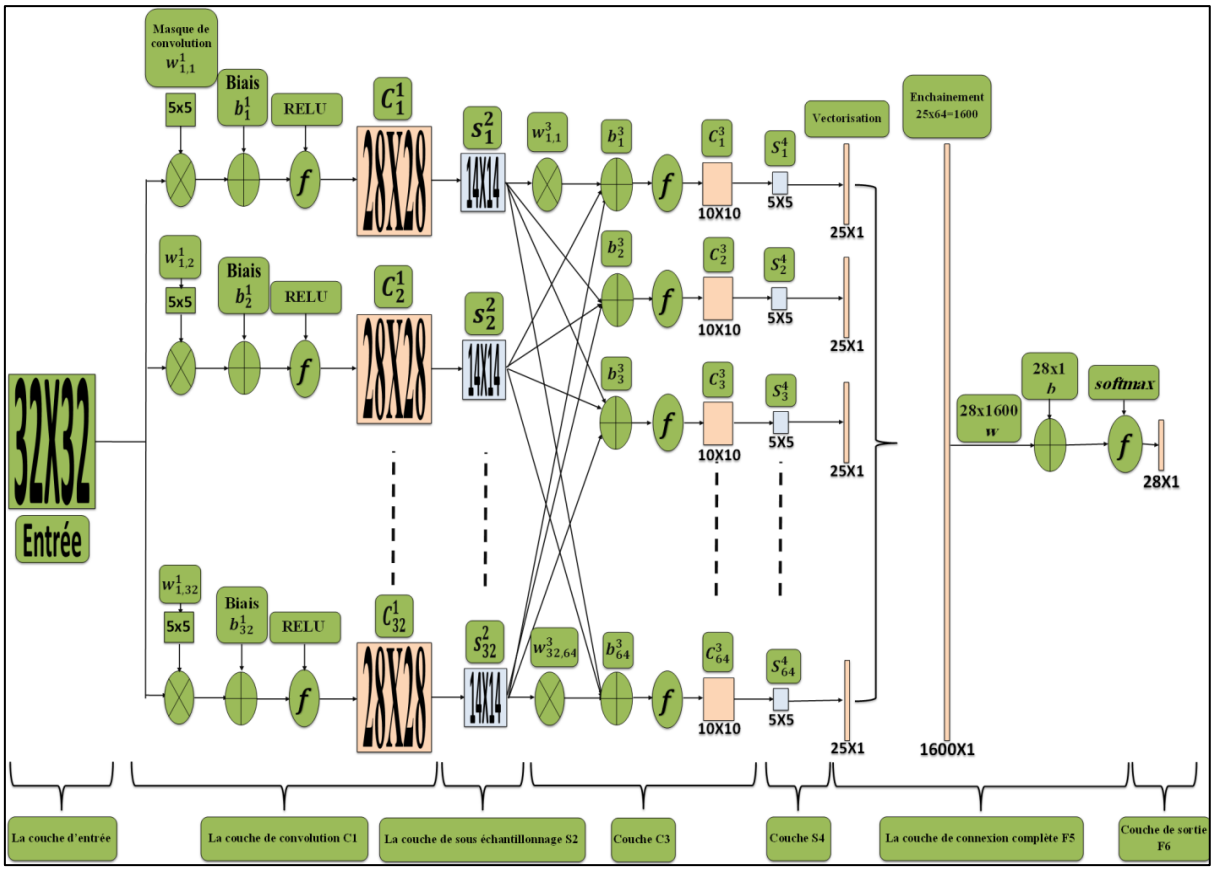


Figure 3-08 : Structure de RNC utilisé à la reconnaissance des caractères manuscrits.

a. Initialisation des paramètres :

Les paramètres sont :

- **La couche $C1$:** $w_{1,n}^1$ (taille $5x5$) et b_n^1 (taille $1x1$), $n = 1, 2, \dots, 32$.
- **La couche $C3$:** $w_{m,n}^3$ (taille $5x5$) et b_n^3 (taille $1x1$), $n = 1, 2, \dots, 64$ et $m = 1,2 \dots ,32$.
- **La couche F :** w (taille $28x1600$) et b (taille $28x1$).
- Toutes les biais, b_n^1, b_n^3 et b, sont initialisées à zéro.

b. Couche de convolution C1 :

$$C_n^1 = f\left(I * w_{1,n}^1 + b_n^1\right) \; ou \; f(\mathrm{x}) = \max(0, x) \; (\mathrm{RELU}) \tag{3.10}$$

$$C_n^1(i,j) = f\left(\sum_{u=0}^{4}\sum_{v=0}^{4} I(i+u, j+v) * w_{1,n}^1(u+1, v+1) + b_n^1\right) \tag{3.11}$$

Où $n = 1, 2, \cdots, 32$ car il existe 32 cartes des caracterstiques dans la couche C1, la taille de C_n^1 est $28 * 28$, plutôt que l'image d'entree I et de taille $32 * 32$.

c. Couche de sous-échantillonnage S2 :

$$S_n^2(i,j) = \frac{1}{4}\sum_{u=0}^{1}\sum_{v=0}^{1} C_n^1(2i-u, 2j-v) \text{ et } i,j = 1,2,\dots,14 \quad (3.12)$$

d. Couche de convolution C3 :

$$C_n^3 = f\left(\sum_{m=1}^{32} S_m^2 * w_{m,n}^3 + b_n^3\right) \quad (3.13)$$

$$C_n^3(i,j) = f\left(\sum_{m=1}^{32}\sum_{u=0}^{4}\sum_{v=0}^{4} S_m^2(i+u, j+v)\, w_{m,n}^3(u+1, v+1) + b_n^3\right) \quad (3.14)$$

Où $n = 1,2,\cdots,64$ car il existe 64 cartes des caractéristiques dans la couche $C3$, la taille de C_n^3 est $10 * 10$, plutôt que la taille de S_m^2 et $14 * 14$.

e. Couche de sous-échantillonnage S4 :

$$S_n^4(i,j) = \frac{1}{4}\sum_{u=0}^{1}\sum_{v=0}^{1} C_n^3(2i-u, 2j-v) \text{ et } i,j = 1,2,\dots,5 \quad (3.15)$$

f. Dernier couche de convolution (Vectorisation et enchaînement) :

Chaque S_n^4 est une matrice $5 * 5$, et il existe 64 de telles matrices sur la couche $S4$. Tout d'abord, chaque S_n^4 est vectorisé par balayage de colonne, alors tous les 64 vecteurs sont concaténés pour former un vecteur long avec une longueur de $5 * 5 * 64 = 1600$. Nous désignons ce processus par :

$$g = G\left(\{S_n^4\}_{n=1,2,\dots,12}\right) \quad (3.16)$$

Et le processus inverse est :

$$\{S_n^4\}_{n=1,2,\dots,12} = G^{-1}(g) \quad (3.17)$$

g. Couche de connexion complète F :

$$y = \mathrm{f}(w * g + b), \text{ou } \mathrm{f}(x_i) = softmax(x_i) = \frac{e^{x_i}}{\sum_{k=1}^{N} e^{x_K}} \quad (3.18)$$

3.4.3. Résultats des Simulations :

Au début, pour évaluer les performances de RNC sur les caractères arabes, une approche d'apprentissage supplémentaire a été utilisée sur l'approche proposée avec le mode mini-lot. Comme le montre la figure 3-10 [70], le taux de classification des erreurs pour les données d'entraînement atteint 0% aux époques de 25 à 30.

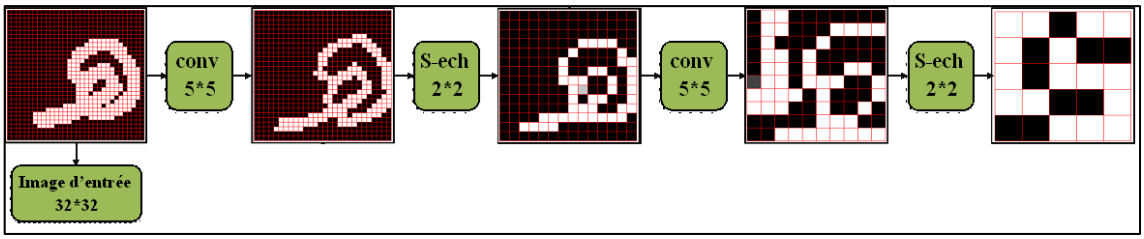


Figure 3-09 : Un exemple sur les couches de convolution et de sous-échantillonnage.

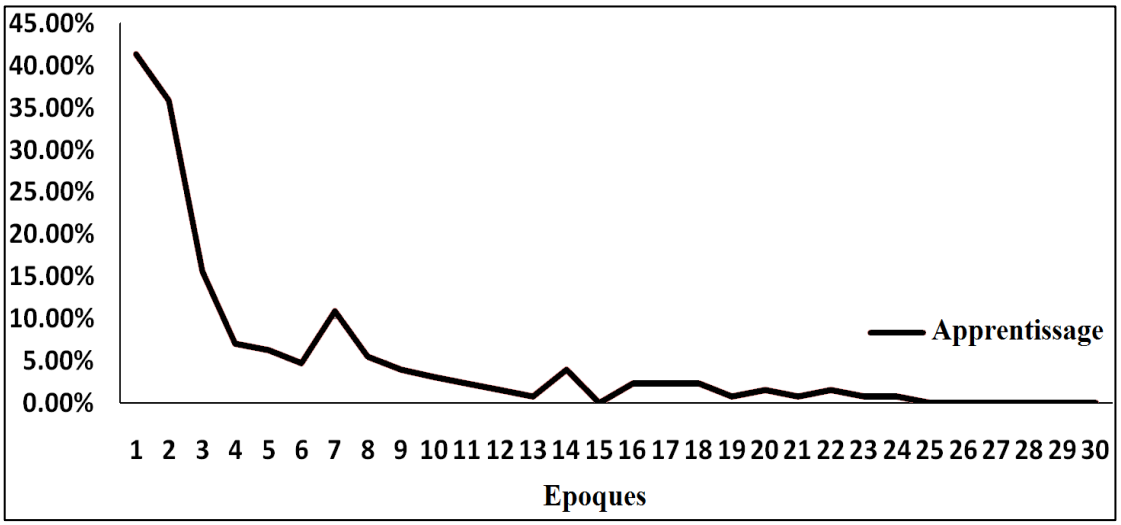


Figure 3-10 : Taux de mauvaise classification pour les données d'entraînement [70].

La base des données utilisé dans [70] comporte 3360 caractères de test est comme il y a 28 caractères dans l'écriture arabe, donc le test a était faites avec 120 modèles par caractères. Le taux de mal classification est le nombre des caractères mal classé sur l'ensemble des données de test. On peut constater que le taux de mal classification atteigne 0.0%, 0.0%, 0.8%, 0.8% pour les caractères (ا, ر, ل, م) comme le montre la figure 3-12. La totalité de la mauvaise classification est de 173 comme le montre la figure 3-11. Cela montre le grand potentiel et la capacité de reconnaissance de ce RNC à classer les caractères manuscrits arabes.

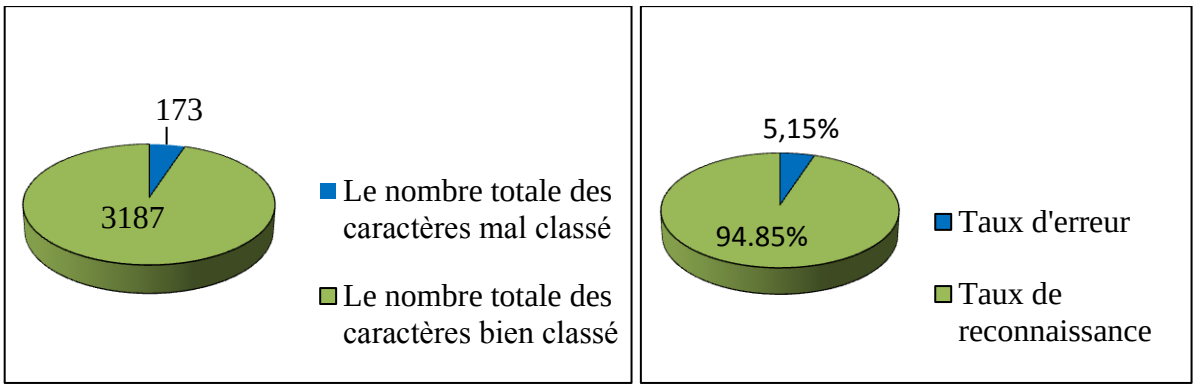


Figure 3-11 : Taux d'erreur (mal classé) et taux de reconnaissance (bien classé).

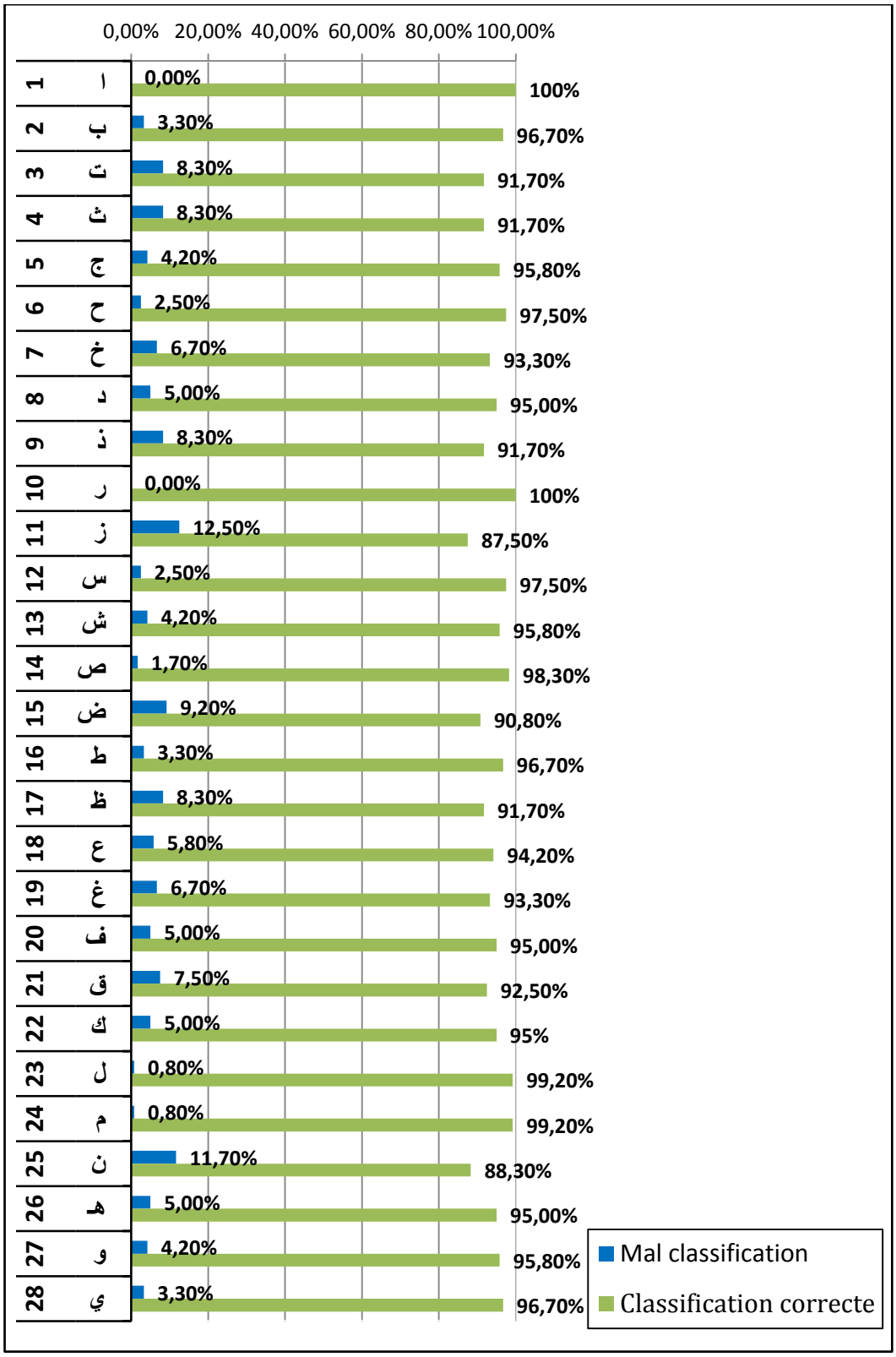


Figure 3-12 : Le taux de classification par caractère.

D'après les résultats dans [70], on peut conclue que :

- Certains caractères sont très confus avec des caractères similaires qui se trouvent dans le tableau 3-3. La petite distinction de la structure de l'accident entraîne des défis pour certaines

paires des caractères similaires, comme ص et ض. La différence de ص et de ض est le point qui caractérise le caractère de ض.

Caractères similaires	Traits de caractères
ب، ت، ث	ب
ج، ح، خ	ح
د، ذ	د
ر، ز	ر
س، ش	س
ط، ظ	ط
ص، ض	ص
ع، غ	ع
ف، ق	ف
ل، ك	ل

Tableau 3-3 : Similitude des traits des caractères arabes

- Certains caractères sont très confus avec la structure d'autres caractères. Le caractère arabe ز a un nombre plus élevé de mauvaise classification. Le caractère est facilement confondu par quatre caractères arabes د, ذ et ر.
- Certains traits des caractères manquent avec la structure du caractère. D'autres caractères peuvent avoir une touche supplémentaire de course qui s'affiche à la figure 3-13 [70].

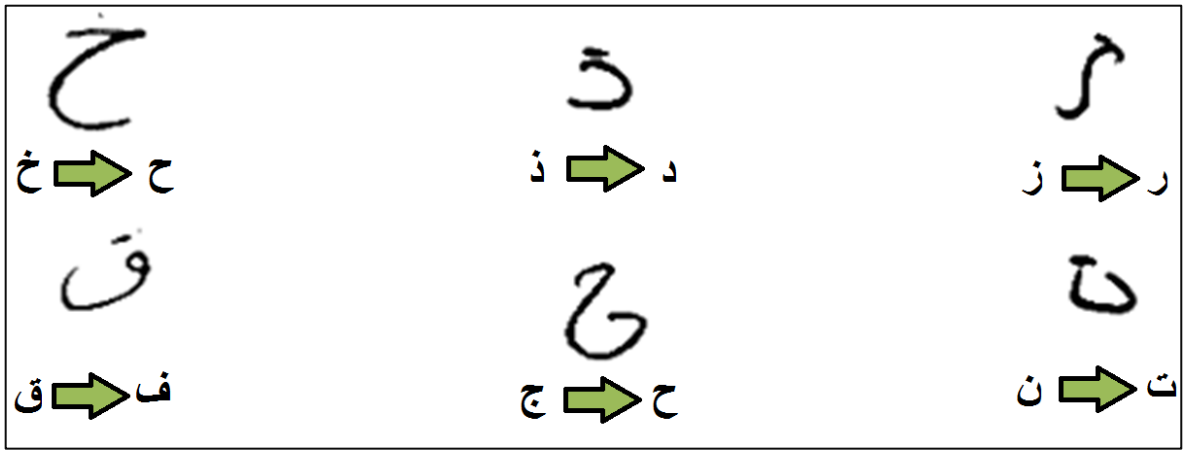


Figure 3-13 : Traits des caractères manquent avec la structure du caractère.

Les trois phénomènes ci-dessus peuvent être la raison principale pour laquelle les caractères de ces types des caractères arabes sont très difficiles à reconnaître. En outre, nous notons également que la mauvaise classification moyenne de ces classes est généralement plus élevée, la figure 3-14 nos montre le totale des caractères mal classés.

Figure 3-14 : Le total de la mauvaise classification des caractères.

3.5. Conclusion :

L'objectif de ce chapitre était d'implémenter et tester l'efficacité des réseaux des neurones convolutionnels pour la reconnaissance des chiffres arabes manuscrits en comparant les résultats obtenus avec ceux de l'articule [70] pour la reconnaissance des caractères, nous avons montré que les résultats étaient prometteurs avec un taux d'exactitude de classification de 98.59% pour les chiffres et de 94.85% pour les caractères. Avec ses résultats nous remarquons que le RNC est très efficace pour la reconnaissance des chiffres par rapport aux caractères.

Cette différence d'efficacité est à cause de plusieurs facteurs, y compris : le nombre des caractères, la complexité de chaque caractère, la similitude entre les caractères, en plus de la qualité de la base des données et les moyens de l'exercer.

Conclusion générale

1. Conclusion :

Comme mentionné précédemment au cours des différentes parties de ce mémoire est que la langue arabe a encore connue une grande exclusion par les chercheurs malgré sa grande portée dans le monde. donc notre objectif principal de ce mémoire est de poursuivre la série de recherches déjà faite pour atteindre les techniques optimales d'un système fiable est très efficace pour la reconnaissance de l'écriture arabe manuscrite, au début nous avons abordé les caractéristiques des caractères arabes qui se distinguent a des autres caractères aux nombreuses difficultés, et afin de faciliter le processus de reconnaissance nous avons discuté les différentes méthodes de traitement et de classification qui doivent être respectées. Parmi ces différents processus existants on a conclu que le plus efficace actuellement est les réseaux des neurones artificiels, qui a fait l'objet de nos recherches dans le deuxième chapitre de ce mémoire.

Pour en savoir plus sur les manières d'utilisations des réseaux des neurones pour la reconnaissance de l'écriture arabe manuscrite on a parlé dans le deuxième chapitre des avantages de ce réseau et leur capacité d'apprendre et de test, puis nous nous sommes concentrés sur les réseaux des neurones convolutionnels, qui sont considérés en raison de leurs caractéristiques parmi les plus importantes réseaux dans le domaine de reconnaissance. C'est pour ça nous avons expliqué leurs architectures en détail dans le but de l'utiliser dans notre application au troisième chapitre.

Nous avons expérimenté le réseau convolutionnel pour la reconnaissance des chiffres, où nous avons conclu que ce réseau est très efficace mais il est toujours insuffisant pour les caractères par rapport aux chiffres à cause des facteurs suivants :

- ✓ Le nombre des chiffres est limité à 10 chiffres de 0 à 9, tandis que les caractères est de 28 caractères de ا à ي.
- ✓ La base des données pour les chiffres (60 000+10 000) et très grand par rapport à celle des caractères (13 440+3 360).
- ✓ La forme simple des chiffres, qui sont toujours au-dessus de la ligne d'écriture par rapport aux lettres qui prennent des positions différentes.
- ✓ Les chiffres n'ont pas des contrecoups alors que les caractères ont des nombreux disciples dans différents endroits tels que les points.

- ✓ Une différence claire entre les caractères peuvent ne pas être en mesure de différencier l'œil humain entre deux caractères différents.

A travers ses résultats obtenus comme le taux d'erreur et de 1.41% pour les chiffres et de 5.15% pour les caractères le réseau de neurone convolutionnel reste toujours incomplète et susceptible à l'erreur.

Le réseau de neurone convolutionnel est basé sur une grande base des données pour l'apprentissage et l'expérimentation est ce qui implique nécessairement un dispositif très efficaces pour s'y conformer, et cela signifie que dans l'avenir avec le développement scientifique il sera possible d'atteindre un niveau élevé de reconnaissance.

2. Perspectives :

Les travaux accomplis, dans ce mémoire, ouvrent plusieurs perspectives de futurs travaux suivant deux voies d'amélioration de nos approches :

- ➢ Actuellement, nous avons réalisé un système de la reconnaissance des chiffres et des caractères arabes manuscrits. Or, comme les applications réelles sont basées sur la reconnaissance des chaines des caractères, nous pensons de choisir un domaine d'application réel tel que la reconnaissance de la somme numérique des chèques bancaires. Dans un deuxième temps, nous allons travailler sur l'automatisation de la reconnaissance du montant littéral des chèques bancaires en se basant sur les approches élaborées. Pour cela nous proposons de développer des techniques de segmentation automatiques des chiffres connectés. Les réseaux de neurones récurrents mériteraient d'être explorés.

- ➢ Le deuxième aspect à approfondir concerne les méthodes de classification. Les approches développées sont basées sur les réseaux de neurones convolutionnels, mais il nous semble après lecture de la littérature qu'il serait intéressant d'utiliser d'autres types comme les réseaux de neurones récurrents.

Bibliographie

[1] Poisson E., "Architecture et apprentissage d'un système hybride neuro markovien pour la reconnaissance de l'écriture manuscrite en-ligne", Thèse de Doctorat, Université de Nantes, 2005.

[2] Salima Nebti, "Reconnaissance de Caractères Manuscrits par Intelligence Collective", Thèse de doctorat Université Ferhat Abass-Sétif, Algeria, 2013.

[3] J. Park," Hierarchical Character Recognition and its use in handwritten word/phrase recognition". Ph.D thesis. 1999.

[4] L. M. Lorigo and V. Govindaraju, "Offline Arabic Handwriting Recognition", A Survey, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 28, no. 5, (2006), pp. 712-724.

[5] M. S. Jelodar, M. J. Fadaeieslam, N. Mozayani and M. Fazeli, "A Persian OCR System Using Morphological Operators, In World Academy of Science, Engineering and Technology, (2005), pp. 137–140.

[6] M. M. Altuwaijri and M. A. Bayoumi, "Arabic Text Recognition Using Neural Networks, In IEEE international Symposium on Circuits and Systems, vol. 6, (1994), pp. 415–418.

[7] H. A. Al-Hamad and R. Abu Zitar, "Development of an Efficient Neural-Based Segmentation Technique for Arabic Handwriting Recognition", Pattern Recognition, vol. 43, no. 8, (2010), pp. 2773–2798.

[8] K. Jumari and M. Ali, "A Survey and Comparative Evaluation of Selected Off-Line Arabic Handwritten Character Recognition Systems", Jurnal Teknology, Malaysian University of Technology, (2002), pp. 1-18.

[9] R. Plamodon and S. N. Srihari, "Online and Offline Handwriting Recognition", A Comprehensive Survey, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 22, no. 1, (2000), pp. 63-84.

[10] J. H. AlKhateeb, R. Jinchang, J. Jianmin, S. S. Ipson and H. El-Abed, "Word-based Handwritten Arabic Scripts Recognition using DCT Features and Neural network Classifier", In 5th International Multi-Conference on Systems, Signals and Devices, (2008), pp. 1–5.

[11] R. El-Hajj, L. Likforman-Sulem and C. Mokbel, "Arabic Handwriting Recognition Using Baseline Dependent Features and Hidden Markov Modeling", In Proceedings of The Eighth International Conference on Document Analysis and Recognition, (2005), pp. 893–897, Washington, USA.

[12] H. El-Abed and V. Margner, "Comparison of Different Preprocessing and Feature Extraction Methods for Offline Recognition of Handwritten Arabic Words", In Ninth International Conference on Document Analysis and Recognition, vol. 2, (2007), pp. 974-978.

[13] H. Al-Rashaideh, "Preprocessing Phase for Arabic Word Handwritten Recognition", Russian Academy of Sciences, Russian Federation, vol. 6, no. 1, (2006), pp. 11-19.

[14] F. Farooq, V. Govindaraju and M. Perrone, "Pre-processing Methods for Handwritten Arabic Documents", In Eighth International Conference on Document Analysis and Recognition, vol. 1, (2005), pp. 267–271.

[15] J. Alkhateeb, "Word Based Off-line Handwritten Arabic Classification and Recognition, PhD Thesis, School of Computing, Informatics and Media, University of Bradford, (2010).

[16] GONZALEZ, R. C. & WOODS, R. E. (2002) Digital Image Processing.

[17] R. Legault and C. Y. Suen, "Optimal Local Weighted Averaging Methods in Contour Smoothing", IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 19, (1997), no. 8, pp. 801 –817.

[18] W. L. Lee and K. C. Fan, "Document Image Pre-processing Based on Optimal Boolean Filter", Signal Processing, vol. 8, no. 1, (2000), pp. 45–55.

[19] M. Khorsheed, "Off-line Arabic Character Recognition - A Review, Pattern Analysis Applications, vol. 5, (2002), pp. 31–45.

[20] R. C. Gonzalez and R. E. Woods, "Digital Image Processing", Dorling Kindersley Pvt. Ltd, (2009).

[21] A. AL-Shatnawi and K. Omar, “A Comparative Study Between Methods of Arabic Baseline Detection”, In International Conference on Electrical Engineering and Informatics, vol. 1, (2009), pp. 73–77.

[22] M. Pechwitz and V. Margner, “Baseline Estimation for Arabic Handwritten Words, In Eighth International Workshop on Frontiers in Handwriting Recognition, (2002), pp. 479–484.

[23] MELHI, M. H. (2001) Off-Line Arabic Cursive Handwriting Recognition Using Artificial Neural Networks; PhD thesis. Department of Cybernetics, Internet and Virtual Systems. Bradford, University Bradford.

[24] AL-BADR, B. & MAHMOUD, S. A. (1995) Survey and bibliography of Arabic optical text recognition. Signal Processing, 41, 49-77.

[25] Z. Q. Liu, J. Cai and R. Buse, “Handwriting Recognition Soft Computing and Probabilistic Approaches”, Springer, (2010).

[26] B. A. Yanikoglu and P. A. Sandon, “Recognizing Off-line Cursive Handwriting”, In IEEE Computer Society Conference on Computer Vision and Pattern Recognition, (1994), pp. 397 –403.

[27] J. Dong, P. Dominique, A. Krzyyzak and C. V. Suen, “Cursive Word Skew/Slant Corrections Based on Radon Transform”, In Eighth International Conference on Document Analysis and Recognition, (2005), pp. 478–483.

[28] Y. M. Alginahi, “A Survey on Arabic Character Segmentation”, International Journal on Document Analysis and Recognition, (2002), pp. 1-22.

[29] R. G. Casey and E. Lecolinet, “Strategies in Character Segmentation”, A Survey, In Proceedings of The Third International Conference on Document Analysis and Recognition, vol. 2, (1995), pp. 1028–1033.

[30] Z. Al-Aghbari and S. Brook, “Hah Manuscripts: A Holistic Paradigm for Classifying and Retrieving Historical Arabic Handwritten Documents”, Expert Systems with Applications, vol. 36, no. 8, (2009), pp. 10942-10951.

[31] G. Kim, V. Govindaraju and S. N. Srihari, “An Architecture for Handwritten Text Recognition Systems, International Journal on Document Analysis and Recognition, vol. 2, no. 1, (1999), pp. 37–44.

[32] J. Alkhateeb, J. Ren, S. Ipson and J. Jiang, “Component-based Segmentation of Words from Handwritten Arabic Text”, International Journal of Computer Systems Science and Engineering, vol. 5, no. 1, (2009), pp. 54–58.

[33] L. Lorigo and V. Govindaraju, “Segmentation and Pre-recognition of Arabic Handwriting”, In Proceedings of the Eighth International Conference on Document Analysis and Recognition, vol. 2, (2002), pp. 605-609.

[34] L. Zheng, A. H. Hassin and X. Tang, “A New Algorithm for Machine Printed Arabic Character Segmentation”, Pattern Recognition Letters, vol. 25, (2004), pp. 1723–1729.

[35] B. Parhami and A. Taraghi, “Automatic Recognition of Printed Farsi Texts, In Pattern Recognition, vol. 14, (1981), pp. 395–403.

[36] S. N. Nawaz, M. Sarfraz, A. Zidouri and W. G. Al-Khatib, “An Approach to Offline Arabic Character Recognition Using Neural Networks”, In 10th IEEE International Conference on Electronics, Circuits and Systems, vol. 3, (2003), pp. 1328– 1331.

[37] M. Altuwaijri and M. Bayoumi, “A New Thinning Algorithm for Arabic Characters Using Self-organizing Neural Network”, In IEEE International Symposium on Circuits and Systems, vol. 3, (1995), pp. 1824–1827.

[38] M. Tellache, M. Sid-Ahmed and B. Abaza, “Thinning Algorithms for Arabic OCR”, In IEEE Pacific Rim Conference on Communications, Computers and Signal Processing, vol. 1, (1993), pp. 248 –251.

[39] J. Cowell and F. Hussain, “Thinning Arabic Characters for Feature Extraction”, In Fifth International Conference on Information Visualisation, (2001), pp. 181–185.

[40] T.Sari, L. Souici and M. Sellami, “Off-line Handwritten Arabic Character Segmentation Algorithm: ACSA”, In Proceedings of the Eighth International Workshop on Frontiers in Handwriting Recognition, (2002), pp. 452–457.

[41] A. Hamid and R. Haraty, “A Neuro-heuristic Approach for Segmenting Handwritten Arabic Text, In International Conference on Computer Systems and Applications, (2001), pp. 110-113.

[42] D. Motawa, A. Amin and R. Sabourin, ‘Segmentation of Arabic Cursive Script”, In Proceedings of The 4th International Conference on Document Analysis and Recognition, (1997), pp. 625–628.

[43] S. Al-Ma’adeed, “Recognition of Off-line Handwritten Arabic Words”, PhD Thesis, University of Nottingham, (2004).

[44] V. K. Govindan and A. P. Shivaprasad, “Character Recognition A Review”, Pattern Recognition, vol. 23, no. 7, (1990), pp. 671–683.

[45] A. Amin and J. Mari, “Machine Recognition and Correction of Printed Arabic Text”, IEEE Transactions on Systems, Man and Cybernetics, vol. 19, no. 5, (1989), pp. 1300 –1306.

[46] M. S. Khorsheed, “W. F. Structural Features of Cursive Arabic Script”, In Proceedings of The Tenth British Machine Vision Conference, (1999), pp. 442 - 431.

[47] G. N. Srinivasan and G. Shobha, “Statistical Texture Analysis”, In Proceedings of World Academy of Science, Engineering and Technology, vol. 36, (2008), pp. 1264–1269.

[48] K. M. Mohiuddin and J. Mao, “A Comparative Study of Different Classifiers for Handprinted Character Recognition, In Pattern Recognition in Practice IV, (1994), pp. 437–448.

[49] M. S. Khorsheed and W. F. Clocksin, “Multi-font Arabic Word Recognition Using Spectral Features”, In 15th International Conference on Pattern Recognition, vol. 4, (2000), pp. 543–546.

[50] N. Arica and F. T. Yarman-Vural, “An Overview of Character Recognition Focused on Off-line Handwriting”, IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews, vol. 31, no. 2, (2001), pp. 216 –233.

[51] McCulloch, W. S., & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. The bulletin of mathematical biophysics, 5 (4), 115‑133.

[52] Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review, 65* (6), 386.

[53] Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1988). Learning representations by back-propagating errors. *Cognitive modeling*.

[54] LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., & Jackel, L. D. (1989). Backpropagation applied to handwritten zip code recognition. *Neural computation*, *1* (4), 541‑551.

[55] Chen, T. and H. Chen (1995). “Universal approximation to nonlinear operators by neural networks with arbitrary activation functions and its application to dynamical systems”. In: Neural Networks, IEEE Transactions on 6.4, pp. 911–917.

[56] Wu, H. (2009). “Global stability analysis of a general class of discontinuous neural networks with linear growth activation functions”. In: *Information Sciences* 179.19, pp. 3432–3441

[57] X. Glorot, A. Bordes, and Y. Bengio, \Deep sparse rectifier neural networks," in JMLR W&CP: Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics (AISTATS 2011), Apr. 2011.

[58] A. L. Maas, A. Y. Hannun, and A. Y. Ng, \Rectifier nonlinearities improve neural network acoustic models," Proc. ICML, vol. 30, 2013.

[59] Minsky, Marvin and Papert, Seymour. Perceptrons. 1969, MIT Press, Cambridge.

[60] P. Werbos, “Beyond regression: new tools for prediction and analysis in the behavioral science”, Doctoral Dissertation, Harward, Cambridge, MA, 1974

[61] D. E. Rumelhart, G. E. Hinton, and R. J.Williams, “Learning internal representations by error propagation,” in Parallel distributed processing: explorations in the microstructure of cognition. Cambridge, MA: Bradford Books, 1986, vol. I, p. 318 362.

[62] Guyon, Isabelle. A Scaling Law for the Validation-Set Training-Set Size Ratio. In: AT&T Bell Laboratories. 1997.

[63] Haykin, Simon. Neural networks : a comprehensive foundation. Second edition.Delhi: Pearson Education, 1999. ISBN 81-7808-300-0.

[64] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” Proceedings of the IEEE, vol. 86, no. 11, pp. 2278–2324, 1998.

[65] M. T. Hagan, H. B. Demuth, and M. H. Beale, Neural network design. Boston, MA: PWS Publishing,1996.

[66] M. Riedmiller and H. Braun, “A direct adaptive method of faster backpropagation learning: The rprop algorithm,” in IEEE International Conference on Neural Networks, San Francisco, 1993, pp. 586–591.

[67] E. K. P. Chong and S. H. Zak, An introduction to optimization. New York: John Wiley and Sons, Inc., 1996.

[68] M. T. Hagan and M. B. Menhaj, “Training feedforward networks with the marquardt algorithm,” IEEE Transactions on Neural Networks, vol. 5, no. 6, pp. 989–993, 1994.

[69] C. Charalambous, “A conjugate gradient algorithm for the efficient training of artificial neural networks,” IEE Proceedings Part G, vol. 139, no. 3, pp. 301–310, 1992.

[70] Ahmed El-Sawy, Mohamed Loey, Hazem EL-Bakry Arabic Handwritten Characters Recognition using Convolutional Neural Network, E-ISSN: 2415-1513. Volume 5, 2017

[71]. Duffner, S.: Face Image Analysis with Convolutional Neural Networks. [Dissertation], [cited Doctoral Thesis / Dissertation; 192] (2007)

[72] Y. LeCun, C. Cortes and C. J. Burges, "Yann LeCun's Home Page," [Online]. Available: http://yann.lecun.com/exdb/mnist/. [Accessed 29 Oct. 2014].

Résumé

Dans ce mémoire, nous modélisons une architecture d'apprentissage en profondeur qui peut s'appliquer efficacement à la reconnaissance des caractères et des chiffres manuscrits arabes. Cette architecture est basée sur l'utilisation du réseau de neurone convolutionnel. Ce réseaux a formé et testé nos bases des données destiné pour les chiffres et pour les caractères.

Dans ce mémoire, nous situons les méthodes d'optimisation mises en œuvre pour augmenter les performances du réseau de neurone convolutionnel. L'utilisation de ce RNC entraîne des améliorations importantes par rapport à différents algorithmes de classification par apprentissage machine. Notre RNC proposé donne une erreur moyenne de mauvaise classification de 1.41% pour les chiffres et de 5.15% pour les caractères.

Mots clés : réseau de neurone convolutionnel, reconnaissance, chiffres et caractères arabes manuscrits.

ملخص

في هذه المذكرة وضعنا نموذج للتعلم العميق يستطيع أن يطبق بفعالية في عملية التعرف على الأرقام و الحروف العربية المكتوبة بخط اليد. هذا النمودج يستند على الشبكة العصبونية الإلتفافية التي تقوم بتدريب و تجربة قواعد البيانات الخاصة بالأرقام و الحروف.

في هذه المذكرة وضعنا الطرق المثلى الواجب تنفيذها لزيادة آداء الشبكة العصبونية الإلتفافية. حيث إن إستخدام هذه الشبكة يعطينا ميزات عديدة مقارنة بخوارزميات أخرى تعتمد على التصنيف عن طريق التدريب . كما أن الشيكة التي إقترحناها تعطينا متوسط خطأ بنسبة 1.41% بالنسبة لسوء تصنيف الأرقام ،و 5.15% بالنسبة لسوء تصنيف الحروف.

كلمات دلالية: الشبكة العصبوانية الإلتفافية ، التعرف،إنتقاء ،الأرقام و الحروف العربية المكتوبة باليد.

Abstract

In this memory, we model a learning architecture in depth which can effectively be applied to the recognition of digits and characters Arabic manuscripts. This architecture is based on the use of convolutional neural network. This network has trained and tested our databases intended for both digits and characters.

In this memory, we situate the optimization methods implemented to increase the performance of the convolutional neural network. The use of this RNC leads to significant improvements over different machine learning classification algorithms. Our proposed RNC gives an average misclassification error of 1.41% for the digits and 5.15% for the characters.

Key-words : convolutional neural network, recognition, digits and characters Arabic manuscripts.