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

Université Saad Dahlab Blida (USDB)
Faculté des sciences
Département d'informatique

MEMOIRE DE FIN D'ETUDE
En vue de l'obtention du diplôme de Master
Filière : Informatique
Option : Traitement Automatique de la Langue (TAL)

Par : ABDELKADER Amrani

La reconnaissance d'écriture Arabe Manuscrite

Soutenue publiquement devant le jury :

Mme.	CHERIGUENE Soraya	MCB	USDB 1	Président
Mme.	BERRAMDANE Djamila	MAA	USDB 1	Examinateur
Mlle.	DJEDDAR Afrah	MCB	USDB 1	Encadreur
Mr.	AMROUCHE Aissa	MRA	CRSTDLA	Co-encadreur

PROMOTION 2022/2023

Remerciements

*On tient à remercier avant tout **Allah** le tout puissant de nous avoir donné la foi, la volonté et le courage pour mener à bien ce modeste travail**, El Hamdou Li Allah.***

Nous adressons le grand remerciement à notre encadreur Nous le remercions pour la documentation mise à notre disposition, son aide précieuse et ses conseils tout au long de ce projet.

Nous tenons également à remercier notre promoteur pour la confiance qu'il nous a fait en acceptant de diriger ce travail pour son assistance ininterrompue et ses conseils judicieux qui nous ont aidés à mener à bout ce travail.

Nos remerciements vont particulièrement à Messieurs les membres du jury pour avoir accepté d'évaluer et de juger notre modeste travail.

Nos remerciements s'adressent aussi à tous les enseignants et à tous ceux qui nous ont accompagnés tout au long des deux années de Master.

Nous ne pouvons terminer sans remercier nos familles pour leurs soutiens indéfectibles, nos amis et camarades de promotion pour leurs présences et conseils ainsi que tous ceux qui ont contribué de près ou de loin à la bonne réalisation de ce travail.

Dédicace

Je dédie ce modeste travail :

A mes chers parents,

Pour tous leurs sacrifices, leurs amours, leurs tendresses, leurs soutiens et leurs prières tout au long de mes études,

A mes chers sœurs et frères, pour leurs appuis et leurs encouragements,

A toute ma famille pour leur soutien tout au long de mon parcours universitaire.

Que ce travail soit l'accomplissement de vos vœux tant allégués, et le fuit de votre soutien infaillible,

Merci d'être toujours là pour moi.

Chapitre 1 : Reconnaissance de l'écriture arabe manuscrite

Chapitre 2 : Etat de l'art de la segmentation

Chapitre 3 : Les réseaux de neurones convolutionnels (CNN)

Chapitre 4 : Conception de l'architecture RNN pour la reconnaissance des caractères arabes

Chapitre 1

Chapitre 2

Chapitre 3

Chapitre 4

Chapitre 1

Introduction générale

La reconnaissance automatique de la langue arabe écrite en caractères arabes est un domaine crucial dans le domaine de l'intelligence artificielle et du traitement du langage naturel. La langue arabe est l'une des langues principales dans le monde et est utilisée dans de nombreux pays et cultures, ce qui en fait un défi pour la reconnaissance automatique des langues naturelles en raison de ses complexités grammaticales et morphologiques ainsi que ses différences notables dans l'écriture et la prononciation par rapport aux autres langues.

Cette étude vise à explorer le model CNN a base de Deep Learning utilisées dans la reconnaissance automatique de la langue arabe écrite en caractères arabes, à identifier les difficultés rencontrées dans ce domaine et les moyens de les surmonter. L'accent sera mis sur les méthodes de reconnaissance automatique modernes telles que les réseaux de neurones profonds et l'apprentissage profond, ainsi que leurs applications dans la reconnaissance automatique des textes arabes écrits en caractères arabes. Les principaux défis dans ce domaine seront également discutés.

Cette recherche comprendra également une étude de cas pour évaluer les performances des model bases à Deep Learning et réseaux de neurones convolutionnel pour la reconnaissance automatique de la langue arabe écrite en caractères arabes .

Cette étude vise à mettre en lumière l'importance de la reconnaissance automatique de la langue arabe écrite en caractères arabes et ses défis, ainsi qu'à fournir des solutions efficaces pour surmonter ces défis et améliorer les performances des technologies utilisées dans ce domaine.

Chapitre 1
Reconnaissance de l’écriture arabe manuscrite

Introduction

Au cours des récentes années, des avancées significatives ont été réalisées dans le déploiement des systèmes de reconnaissance de l'écriture manuscrite, attribuables en partie à l'ampleur des recherches menées dans ce domaine et en partie à la production de tels systèmes.

Des micro-ordinateurs à coût abordable ainsi que divers dispositifs d'acquisition tels que les tablettes numériques et les scanners sont soumis à une évaluation. Bien que le nombre de projets de recherche concernant l'écriture arabe soit moindre par rapport à d'autres systèmes d'écriture, tels que le latin et le japonais, le domaine de la reconnaissance par ordinateur de l'écriture manuscrite est étendu.

Le présent chapitre se propose d'examiner les caractéristiques de l'OCR (Reconnaissance Optique de Caractères), ses diverses caractérisations ainsi que les méthodes de reconnaissance de l'écriture manuscrite. En conclusion, quelques sujets pertinents dans ce domaine seront abordés.

.

1. Caractéristiques de l'écriture arabe

Plus de 420 millions de personnes ont l'arabe comme langue maternelle, ce qui en fait l'une des langues les plus parlées dans le monde. En plus de la langue arabe [6], il existe plusieurs langues qui utilisent l'alphabet arabe, telles que l'ourdou, le farsi (persan), le pashto, le jawi et le kurde.

- L'écriture arabe s'écrit de droite à gauche et est composée de 28 caractères, sans majuscules ni minuscules. Chaque personnage a deux ou quatre formes ; la forme du caractère dépend de sa position dans le mot, comme le montre le tableau (1). La première colonne donne le numéro du caractère, la deuxième colonne est son nom, la troisième représente le signe d'un caractère isolé, la quatrième étant son apparition au début du mot [1].
- La langue arabe est écrite en utilisant une écriture cursive, ce qui signifie que les lettres sont généralement liées les unes aux autres, qu'elles soient écrites à la main ou imprimées. Contrairement à l'écriture latine, qui se lit de gauche à droite, l'écriture arabe se lit de droite à gauche [2].
- L'écriture arabe est semi-cursive par nature, ce qui rend la segmentation des caractères en une opération cruciale dans les systèmes de reconnaissance optique de caractères (OCR) arabes. Toutefois, certains caractères de l'alphabet arabe ne peuvent pas être liés aux lettres qui les suivent dans un mot. Par conséquent, la présence de l'une de ces lettres peut diviser le mot en deux composantes connexes, ce qui complique la tâche de segmentation pour les systèmes OCR arabes. La figure suivante montre trois mots arabes : la premier (a) avec une seule composante connexe de six lettres, le deuxième

(b) avec deux composantes connexes, une de trois lettres et l'autre avec une seule lettre, le troisième mot est composé de cinq composantes connexes.

	Isolé	Au début	Au milieu	Fin
Alef	أ	أ		ـأ
Ba	ب	بـ	ـبـ	ـب
Ta	ت	تـ	ـتـ	ـت
Tha	ث	ثـ	ـثـ	ـث
Jeem	ج	جـ	ـجـ	ـج
Ha	ح	حـ	ـحـ	ـح
Kha	خ	خـ	ـخـ	ـخ
Dal	د	د		ـد
Thal	ذ	ذ		ـذ
Ra	ر	ر		ـر
Zy	ز	ر		ـز
Seen	س	سـ	ـسـ	ـس
Sheen	ش	شـ	ـشـ	ـش
Sad	ص	صـ	ـصـ	ـص
Dhad	ض	ضـ	ـضـ	ـض
Tal	ط	ط	ـطـ	ـط
Dha	ظ	ظ	ـظـ	ـظ
Ain	ع	عـ	ـعـ	ـع
Gain	غ	غـ	ـغـ	ـغ
Fa	ف	فـ	ـفـ	ـف
Qaf	ق	قـ	ـقـ	ـق
Kaf	ك	كـ	ـكـ	ـك
Lam	ل	لـ	ـلـ	ـل
Meem	م	مـ	ـمـ	ـم
Noon	ن	نـ	ـنـ	ـن
Ha	ه	هـ	ـهـ	ـه
Waw	و	و		ـو
Ya	ي	يـ	ـيـ	ـي

Tableau 1-1. L'alphabet arabe dans ses différentes formes [5]

- Mot en deux composantes connexes, ce qui complique la tâche de segmentation pour les systèmes OCR arabes. La figure suivante montre trois mots arabes : la premier (a) avec une seule composante connexe de six lettres, le deuxième (b) avec deux composantes connexes, une de trois lettres et l'autre avec une seule lettre, le troisième mot est composé de cinq composantes connexes Figure (1.1).

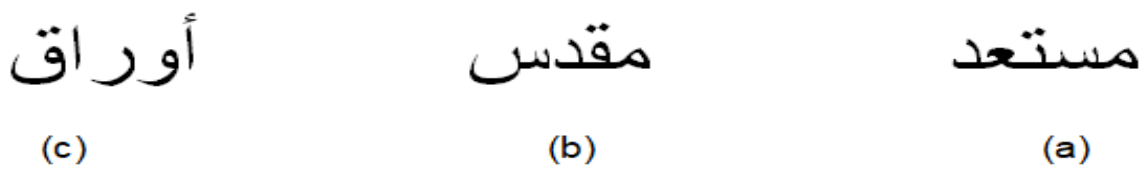


Figure 1.1. Des mots arabes avec leurs composantes connexes [2]

- Il existe de nombreux styles d'écriture arabe différents, mais seuls quelques-uns sont couramment utilisés dans le monde arabo-musulman. Parmi les styles les plus populaires, on trouve notamment le Thuluth, le Naskh, le Ruq'ah, le Dewani, le Farsi et le Kufique. La Figure (1.2) ci-dessous illustre certains exemples de ces différents styles d'écriture arabe.

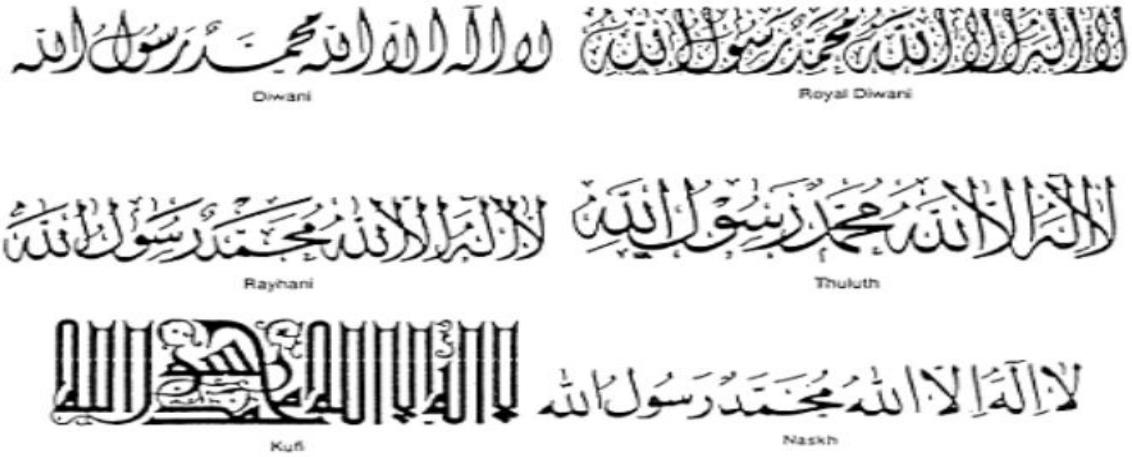


Figure 1.2. Différents styles et fontes pour l'écriture arabe [6]

- L'utilisation des signes diacritiques est une autre caractéristique distinctive de l'arabe qui peut modifier la prononciation et le sens du mot. (Figure 1.3) [5] [6] [2]

Fathah	Kasrah	Dammah	Maddah	Sukun	Tanwin Fathah	Tanwin Kasrah	Tanwin Dammah	Shaddah	Hamza
ـَ	ـِ	ـُ	ـّ	ـْ	ـً	ـٍ	ـٌ	ـّ	ء

Figure 1.3. Marques en arabe écrites

2. Type de reconnaissance d'écriture en ligne et hors ligne

La reconnaissance d'écriture peut être effectuée en ligne ou hors ligne, selon le contexte d'utilisation et le type de technologie utilisée [3]. (Figure1.4)

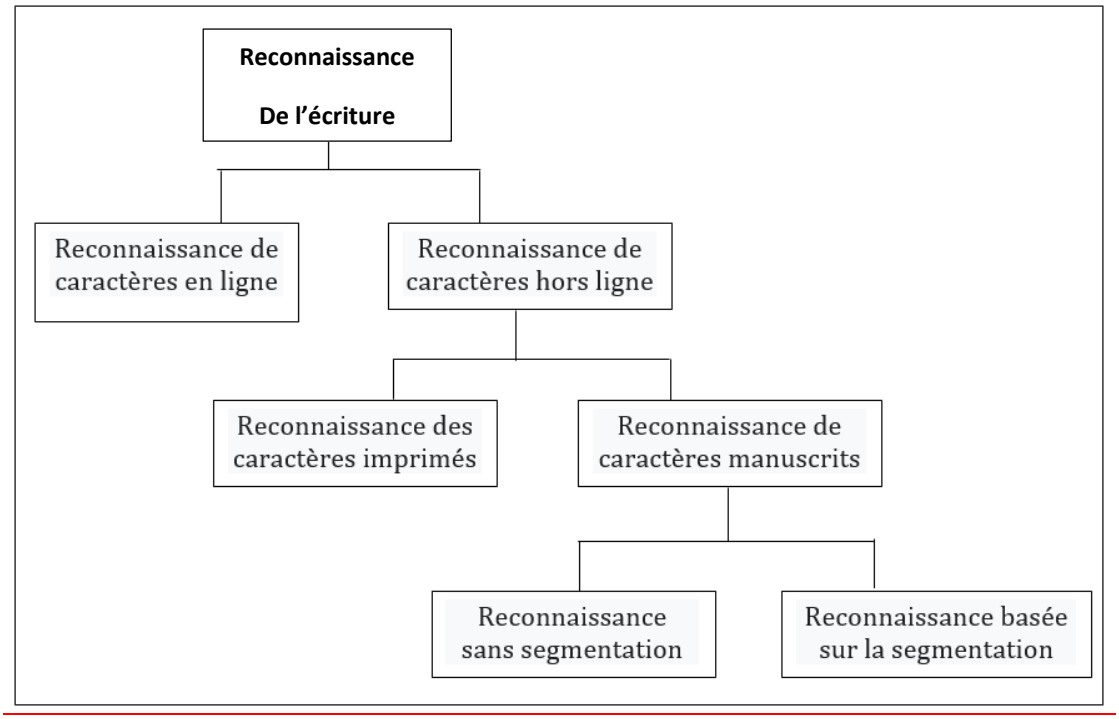


Figure 1.4. Classification des systèmes de reconnaissance de caractères

Dans les deux cas, la reconnaissance d'écriture peut être basée sur des techniques telles que la reconnaissance optique de caractères (OCR) ou la reconnaissance de formes, qui utilisent des algorithmes de traitement d'images pour identifier les caractères écrits à la main et les convertir en texte numérique.

2.1. La reconnaissance d'écriture en ligne

La reconnaissance d'écriture en ligne est effectuée en temps réel alors que l'utilisateur écrit. Elle est souvent utilisée pour la saisie de texte à la main sur des écrans tactiles, des tablettes ou des smartphones. Cette technologie utilise des algorithmes pour interpréter les mouvements de l'utilisateur et convertir leur écriture manuscrite en texte numérique.

Les avantages majeurs de reconnaissance d'écriture manuscrite en ligne

- Moins de bruit car l'utilisateur écrit sur une table spéciale
- La possibilité de correction et de modification de l'écriture de manière interactive vu la réponse en continu du système [5] [7]

Figure 1.5. Outille d'accusation [5]

2.2. La reconnaissance d'écriture hors ligne

La reconnaissance d'écriture hors ligne est effectuée sur des documents manuscrits préexistants, tels que des formulaires papier ou des notes écrites à la main. Cette technologie utilise des scanners ou des appareils photo pour capturer l'image du document manuscrit, puis des algorithmes pour reconnaître les caractères écrits à la main et les convertir en texte numérique.

2.3. Type de mode hors ligne

2.3.1. Reconnaissance de texte ou analyse de documents

Dans le premier cas il s'agit de reconnaître un texte de structure limitée à quelques lignes ou mots. La recherche consiste en un simple repérage des mots dans les lignes, puis à un découpage de chaque mot en caractères. Dans le second cas (analyse de document), il s'agit de données bien structurées dont la lecture nécessite la connaissance de la typographie et de la mise en page du document. Ici la démarche n'est plus un simple prétraitement, mais une démarche experte d'analyse de document il y'a localisation des régions, séparation des régions graphiques et photographique, étiquetage sémantique des zones textuelles à partir de modèles, détermination de l'ordre de lecture et de la structure du document [7].

2.3.2. Reconnaissance de l'imprimé ou du manuscrit

Les approches diffèrent selon qu'il s'agisse de reconnaissance de caractères imprimés ou manuscrits. Les caractères imprimés sont dans le cas général alignés horizontalement et séparés verticalement, ce qui simplifie la phase de lecture. La forme des caractères est définie par un style calligraphique (fonte) qui constitue un modèle pour l'identification. Dans le cas du manuscrit, les caractères sont souvent ligaturés et leur graphisme est inégalement proportionné provenant de la variabilité intra et inter scripteurs. Cela nécessite généralement l'emploi de techniques de délimitation spécifiques et souvent des connaissances contextuelles pour guider la lecture [7].

2.4. La différence entre les modes en ligne et hors ligne

Les modes en ligne et hors-ligne se sont deux modes totalement différents le premier s'opère en temps réel (pendant l'écriture) mais l'autre démarre après l'acquisition, il convient aux documents imprimés et les manuscrits déjà rédigés et aussi La reconnaissance en ligne

présente un avantage majeur c'est la possibilité de correction et de modification de l'écriture de manière interactive vu la réponse en continu du système. On a aussi le tableau 1 présente quelque différence entre les deux modes [2].

Caractère en ligne	Caractère hors ligne
Utilisation de stylos numériques	Utilisation de papier
Disponibilité de coup de stylo	Non disponible de stylo
Taux de reconnaissance élevé	Taux de reconnaissance bas
Grande précision	Faible précision

Tableau 1-2. La de différence entre les deux modes en ligne et hors-ligne

Le mode hors-ligne l'information se présente sous formes d'un ensemble de pixels qui représente l'image de texte acquis par une caméra C.C.D ou matricielle ou par un scanner ; cette image contient une quantité d'informations ; et l'épaisseur de tracé peut avoir plusieurs pixels comme le montre la figure 4.

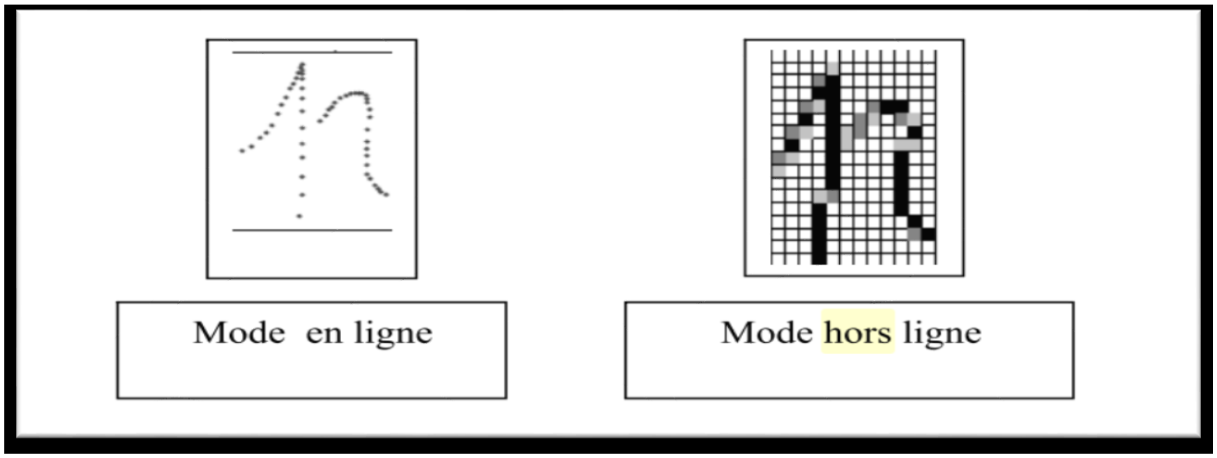


Figure 1.6. Représentation de l'information dans les deux modes [2]

2.5. L'objectif de la reconnaissance des caractères arabes

- Transformer un texte écrit en une représentation compréhensible par une machine et facilement reproductible par un traitement de texte [1].
- Prendre une décision sur le contenu sémantique du message transmis à partir de sa représentation physique
- Doter les ordinateurs de capacités de l'être Humain

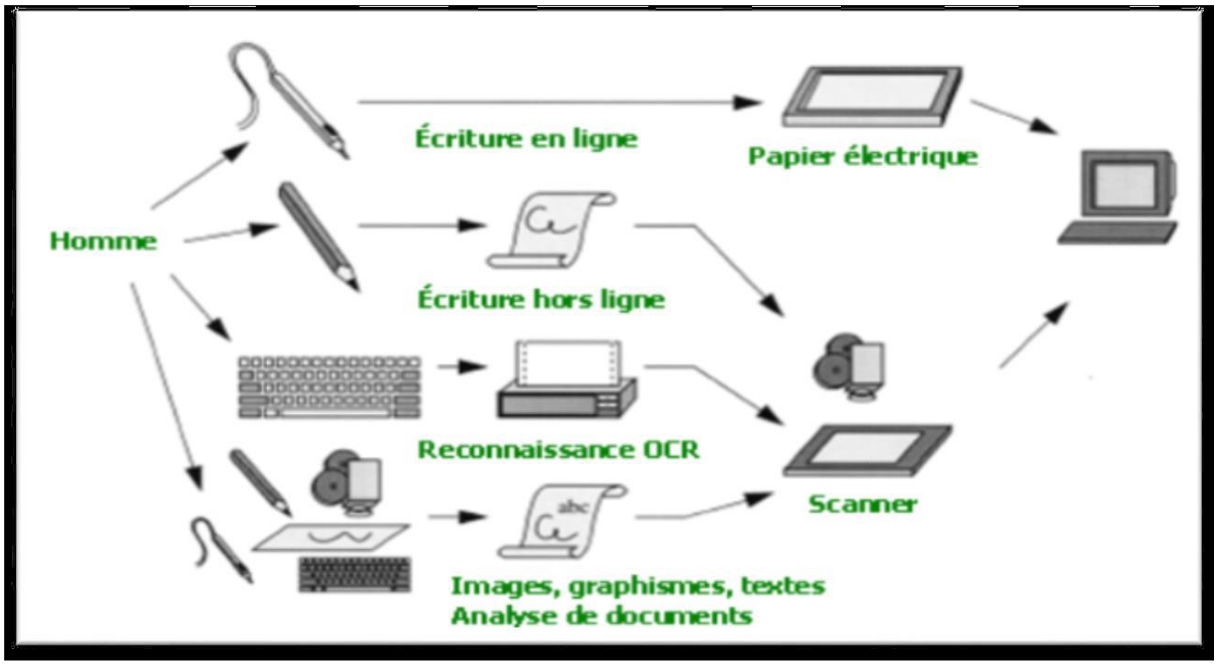


Figure 1.7. Type de communication homme machine [5]

3. Les approche de la reconnaissance d'écriture manuscrite

Il existe plusieurs algorithmes et méthodes proposées par la communauté scientifique pour le problème de la reconnaissance de l'écriture manuscrite. Ces méthodes appréhendent la reconnaissance différemment selon l'approche adoptée classique ou bien l'approche récente deep Learning.

3.1. La déférence entre les deux approches

Le processus de l'approche classique commence par des caractéristiques pertinentes qui sont extraites manuellement des images. Les caractéristiques sont ensuite utilisées pour créer un modèle qui classe les objets dans l'image.

Mais pour le deep Learning, les fonctionnalités pertinentes sont automatiquement extraites des images [9].

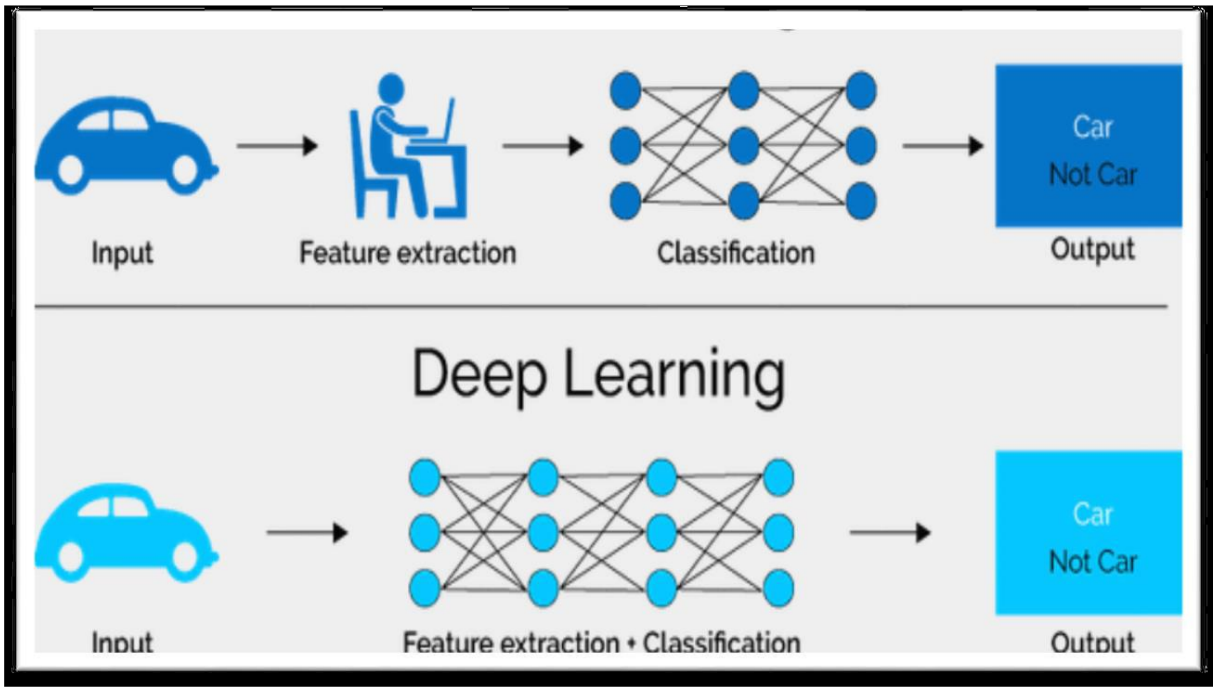


Figure 1.8. L'approche classique et l'approche deep Learning [8]

3.2. L'approche classique

Les approches initiales de résolution de la reconnaissance de l'écriture manuscrite, impliquaient des méthodes d'apprentissage automatique (Machine Learning) telles que le modèle SVM (Support Vector Machine), etc. Une fois le texte initial prétraité, une extraction de caractéristiques est effectuée pour identifier les informations clés telles que les boucles, les points d'inflexion, le rapport hauteur / largeur, etc. d'un caractère individuel. Ces fonctionnalités générées sont maintenant transmises à un classificateur, pour obtenir les résultats. Les performances des modèles d'apprentissage automatique sont assez limitées en raison de la phase d'extraction manuelle des fonctionnalités et de leur capacité d'apprentissage limitée. L'étape d'extraction des fonctionnalités varie pour chaque langue et n'est donc pas évolutive.

On peut citer quelques travaux avec cette approche :

- Le travail de [Baka Abdeladim et fillali Hicham] [10] sur traitement et reconnaissance des caractères

L'OCR est évidemment une technique utile, mais il faut en connaître les limites et en tenir compte, en prévoyant une ou plusieurs lectures personnelles du document. Parfois les documents à traiter peuvent être dégradé physiquement ou lors de leur acquisition pour cela l'étape de prétraitement est donc indispensable an de rendre able l'étape de conversion de l'image vers un texte, Les filtres linéaires pour le traitement du bruit, tel que le filtre gaussien, moyen ...etc. permettent de lisser l'image et ainsi diminuer le bruit qui pourrait impacter négativement sur le résultat de reconnaissance sans garantir la conversion des contours. Pour améliorer les résultats de l'OCR, nous allons utiliser quelques prétraitements pour améliorer la performance de l'OCR.

- Le travail de [Benkhessirate Walid et khenniche Oussama] [11] sur L'élaboration d'un OCR basé sur les modèles de Markov cachés : application au texte arabe imprimé non voyelle

Le but de leur projet est l'implémentation d'un AOCR. La segmentation et la classification sont les opérations cœur des OCR en générale la nature cursive des caractères arabes biais les résultats finaux de la reconnaissance. Le caractère non segmenté ou sur segmenter conduisent à des mauvais résultats. C'est pour cela que la segmentation et la classification dans les AOCR sont un sérieux problème de recherche. La segmentation d'un texte arabe comprendre trois niveaux à savoir la segmentation en ligne en pseudo mot et en caractère. Au cœur de leur projet ils ont choisi les techniques de contour, la projection vertical et templte Matching pour faire la segmentation des trois niveaux respectivement. D'autre part, la classification comprend deux niveaux, à savoir la classification des caractères et la classification des mots. Au cours de leirs projet ils ont choisi des modèles de classification basé sur les modèles de Markov cachés. Ils ont aussi discuté quelques problèmes liés à la langue arabe et étudié d'autres module une concernant les OCR, à savoir l'acquisition des données, le prétraitement et l'extraction des caractéristiques. Les résultats d'implémentation sont prometteurs.

- Le travail de [Zerdani Amina] [5]. sur Reconnaissance de caractères arabes manuscrits par réseau de Hop Field

Leur travail porte sur une étude concernant le domaine de reconnaissance hors-ligne de caractères arabes manuscrits basé sur un nouveau classifieur inspiré de réseau de neurones qui est le réseau de Hop Field. A cet effet, ils s'intéressons à construire leur propre base de caractères arabes manuscrits à l'aide des scripteurs pour l'évaluation des performances du système de reconnaissance de caractères arabes manuscrits conçu. Leur système inclut les étapes de : prétraitement comprenant les opérations de redressement, de normalisation ; d'une étape d'extraction de primitives en se servant de la méthode de zonage et finalement l'étape de reconnaissance est réalisée à travers l'utilisation du classifieur Hop Field. Leurs tests ont été effectués sur leur base déjà construite où ils ont obtenu un taux de reconnaissance intéressant.

3.3. Approche a base de technique deep Learning

Après l'approche classique les chercheurs ont développé une nouvelle méthode avec des nouveaux algorithmes elle est appelée le deep Learning elle utilise les réseaux de neurones on peut citer quelque algorithme comme CNN et RNN nous allons les voir ensemble dans le prochain chapitre.

Quelque travail par l'approche deep Learning

- Le travail de [Mujtaba Husnain Malik Muhammad Saad Missen, Shahzad Mumtaz , Muhammad Zeeshan Jhanidr , Mickaël Coustaty , Muhammad Muzzamil Luqman , Jean-Marc Ogier and Gyu Sang Choi] [13] sur Reconnaissance des caractères manuscrits en ourdou à l'aide Réseau de neurones convolutifs Dans leurs article, ils proposent l'utilisation du réseau de neurones convolutifs pour reconnaître le multi font

hors ligne Caractères écrits à la main en ourdou dans un environnement sans contrainte. Ils proposent également un nouvel ensemble de données de caractères manuscrits en ourdou, car il n'existe aucun ensemble de données de ce type accessible au public. Une série de des expériences sont effectuées sur leurs ensembles de données proposées. La précision atteinte pour la reconnaissance de caractères est parmi les meilleurs en comparaison avec ceux rapportés dans la littérature pour la même tâche .

4. Les difficultés de la reconnaissance d'écriture arabe

La reconnaissance d'écriture arabe présente plusieurs difficultés en raison de la nature complexe de la langue arabe et de son système d'écriture. Voici quelques-unes des difficultés courantes de la reconnaissance d'écriture arabe [14][15] :

Formes des lettres : Les lettres arabes ont différentes formes en fonction de leur position dans un mot. En outre, certaines lettres peuvent être liées entre elles, tandis que d'autres ne le sont pas. Cela rend la segmentation des mots arabes difficile.

1. Connexion des lettres : Les lettres arabes sont souvent connectées les unes aux autres, ce qui rend difficile la reconnaissance des caractères individuels. Les algorithmes de reconnaissance d'écriture doivent donc être capables de reconnaître non seulement les lettres individuelles, mais aussi les connexions entre les lettres.
2. Diacritiques : Les diacritiques sont des signes qui indiquent la prononciation correcte des mots arabes. Ils sont souvent omis dans l'écriture manuscrite, ce qui rend la reconnaissance d'écriture arabe plus difficile.
3. Différentes formes d'écriture : L'arabe est écrit dans plusieurs styles différents, tels que le style calligraphique, le style cursif et le style imprimé. Chacun de ces styles a des caractéristiques différentes, ce qui rend la reconnaissance d'écriture plus complexe.
4. Variations régionales : La langue arabe est parlée dans de nombreux pays différents, et il existe des variations régionales dans la façon dont elle est écrite. Cela peut rendre la reconnaissance d'écriture plus difficile pour les algorithmes qui ont été entraînés sur un ensemble de données spécifique.

En raison de ces difficultés, la reconnaissance d'écriture arabe reste un défi pour les développeurs de logiciels, mais de nouvelles techniques et approches sont constamment développées pour améliorer la précision de la reconnaissance d'écriture arabe.

Conclusion

En conclusion, la reconnaissance des manuscrits arabes est un domaine de recherche en constante évolution, qui présente de nombreux défis en raison de la complexité de l'écriture arabe et de la variabilité des styles d'écriture manuscrite.
Dans ce chapitre, nous avons examiné les différentes caractéristiques de l'écriture arabe, telles que la direction de l'écriture, la connectivité des lettres, la variation des formes en fonction de la position des lettres et la présence de signes diacritiques.

Chapitre 2
Etat de l’art à base de la segmentation

Introduction

La segmentation est définie, comme étant l'opération qui cherche à décomposer une image de texte en pseudo-images de symboles individuels. Le résultat de cette opération, est une forme isolée à partir d'une image et qui pourrait être un caractère ou un symbole. C'est une étape critique dans plusieurs systèmes de reconnaissance d'écriture, elle est dans la plupart des cas sujette aux erreurs et très pénalisante en temps de calcul. On peut distinguer plusieurs niveaux de segmentation. A titre d'exemple, dans le traitement automatique de document, consiste à isoler les différentes parties logiques (graphiques, textes, photos, etc.) constituant une page de document. Aussi, dans les applications où les informations traitées sont des pages de texte, la séparation des lignes de paragraphes, l'extraction des mots ou pseudo-mots d'une ligne, et enfin, la décomposition des mots ou pseudo-mots en caractères. Dans ce chapitre nous allons exposer les différentes techniques de segmentation, et en particulier la segmentation du mot en caractères.

1. Structure d'un système de reconnaissance d'écriture

Un système de reconnaissance d'écriture typique se compose de plusieurs composants clés qui travaillent ensemble pour convertir l'écriture manuscrite en texte numérique. Voici les composants communs d'un système de reconnaissance d'écriture (figure 2.1) :

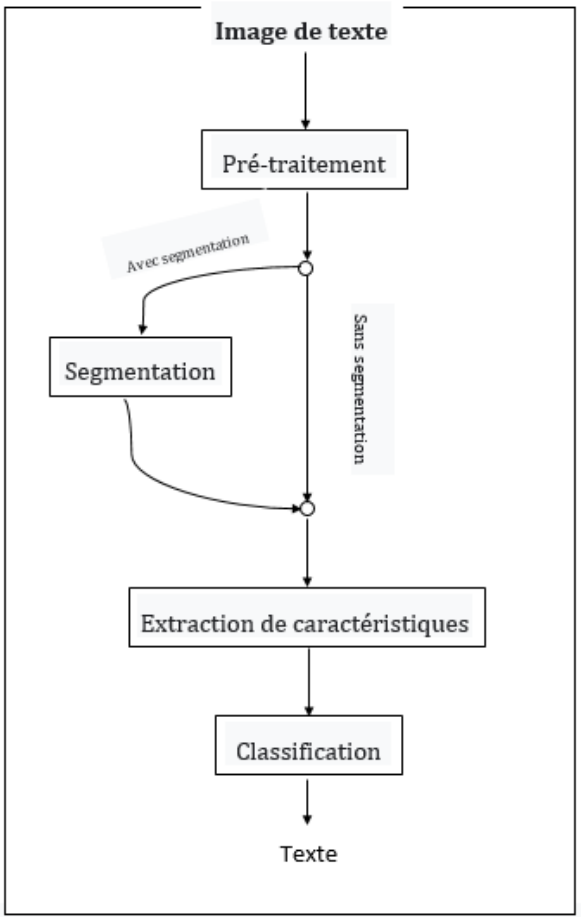
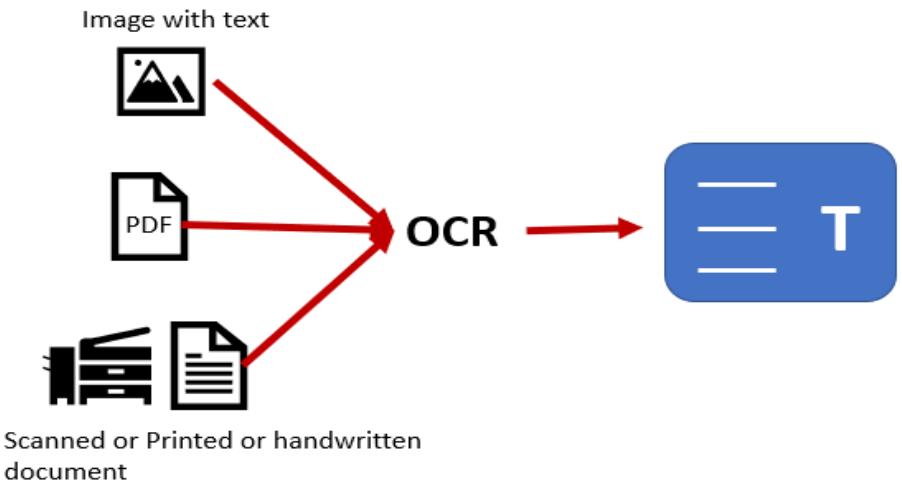


Figure 2.1. Étapes générales pour la reconnaissance de texte manuscrit

1.1. Acquisition de données

Cette étape consiste à capturer l'image de l'écriture manuscrite, soit en utilisant un scanner ou une caméra pour la reconnaissance hors ligne, soit en utilisant un écran tactile pour la reconnaissance en ligne (figue 2.2).



é

Figure 2.2. Étapes acquisition de données

1.2. Prétraitement d'image

Cette étape consiste à nettoyer et à améliorer l'image de l'écriture manuscrite pour faciliter sa reconnaissance ultérieure. Cela peut inclure des techniques telles que la correction de la distorsion, le filtrage de bruit, la normalisation de l'échelle et l'ajustement du contraste (figure 2.3) [19].

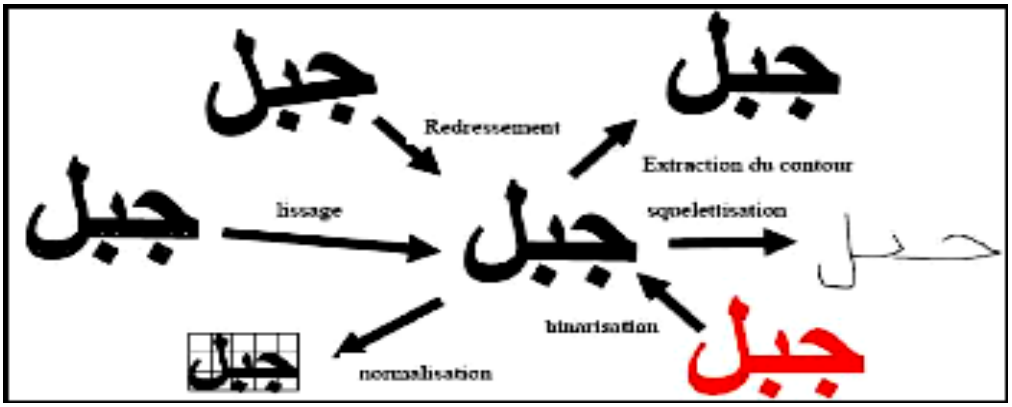


Figure 2.3. Effets de certaines opérations de prétraitement [2]

1.2.1. La Binarisation

Le document numérisé est une image qui peut être binarisé ou à niveaux de gris, à l'aide d'une méthode de seuillage. La détermination du seuil peut être globale, c'est-à-dire que le seuil a la même valeur pour tous les pixels du document [20], ou locale pour laquelle le seuil varie d'une position à une autre. Les méthodes de seuillage global ne sont pas trop

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

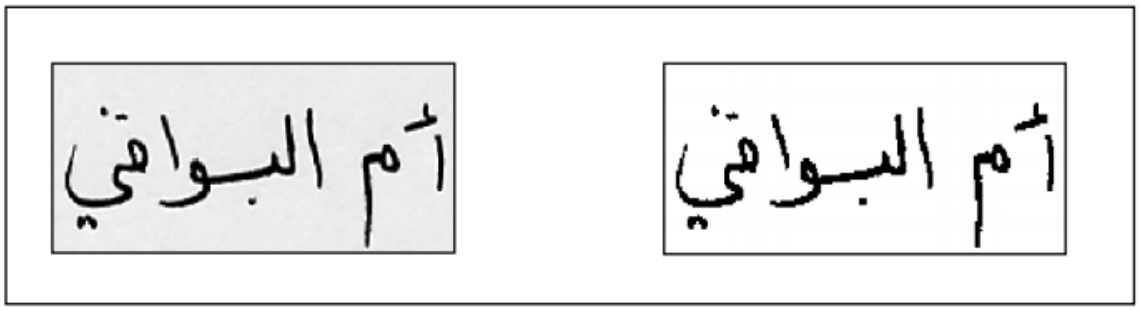


Figure 2.4. Exemple de la binarisation [21]

1.2.2. Suppression du bruit

Consiste à détecter et à éliminer les pixels qui représentent des bruits. Il s'agit d'éliminer plus de bruit que d'information signifiante afin d'augmenter la discrimination Figure (2.5**)**.

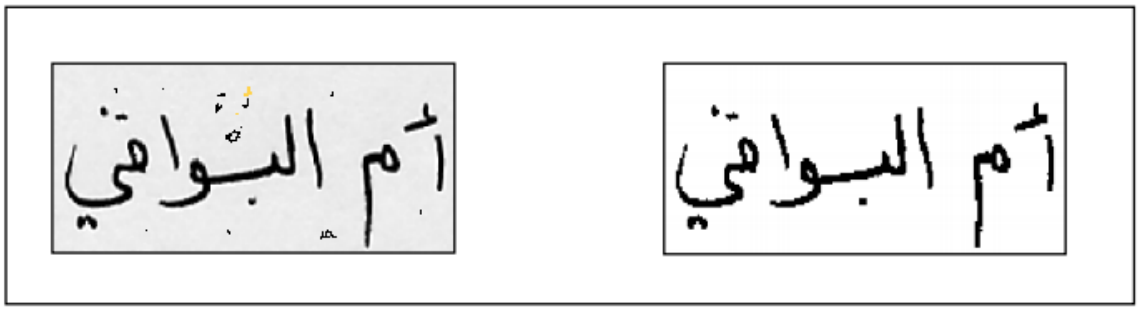


Figure 2.5. Exemple de suppression du bruit [5]

1.2.3. Redressement de l'écriture

Cette opération consiste à corriger la pente d'un mot ou à redresser l'inclinaison des lettres dans un mot afin de faciliter la segmentation [38]. L'opération est effectuée à l'aide d'une transformation ligne par ligne où chaque pixel noir de coordonnées (x, y) est remplacé par les coordonnées (x', y') données par l'équation 2.1 :

(1.2) Deux sortes d'inclinaisons sont distinguées dans les images de mots. La première est l'inclinaison globale du mot par rapport au cadre de l'image, que l'on représente par l'axe moyen du mot. La seconde est l'inclinaison des lettres par rapport à cet axe (figure 2.6).

$$\begin{bmatrix} \acute{x} \\ \acute{y} \end{bmatrix} = \begin{bmatrix} 1 & -\tan\theta \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \qquad (1.2)$$

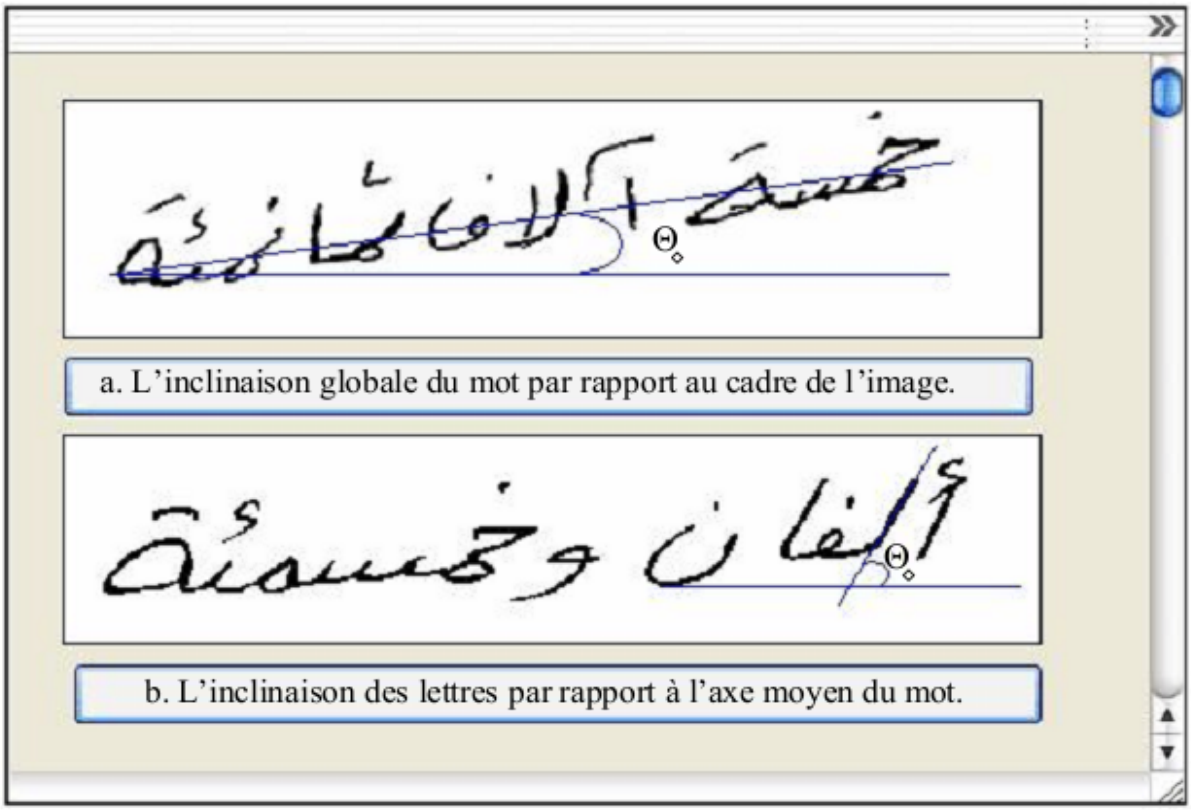


Figure 2.6. Détection de l'inclinaison dans une phrase arabe

1.2.4. Lissage

Il permet de réduire au maximum les discontinuités introduites dans l'image au cours des différentes transformations et ainsi de rétablir la régularité et la continuité du contour du mot.

Le lissage consiste à examiner le voisinage d'un pixel et de lui attribuer la valeur 1 si le nombre de pixel noir dans cette zone est supérieur à un seuil [16]. L'image des caractères peut être entachée de bruits dus aux artefacts de l'acquisition et à la qualité du document, conduisant soit à une absence de points ou à une surcharge de points. Les techniques de lissage permettent de résoudre ces problèmes par des opérations locales qu'on appelle opérations de bouchage et de nettoyage [18]. L'opération de nettoyage permet de supprimer les petites tâches et les excroissances de la forme. Pour le bouchage il s'agit d'égaliser les contours et de boucher les trous internes à la forme du caractère en lui ajoutant des points noirs [19].

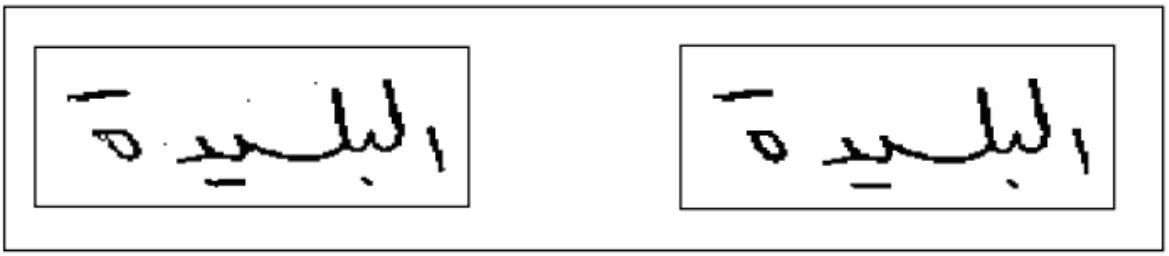


Figure 2.7. Exemple de lissage [21]

1.2.5. Normalisation

La méthode de normalisation est généralement utilisée dans la reconnaissance des mots pour réduire tous les types de variations, et pour obtenir des données normalisées. Cependant, il engendre également à une distorsion de la forme excessive et élimine quelques informations utiles. Les méthodes usuelles pour normaliser un caractère sont les suivantes :

- Normalisation de la taille ;
- Correction de l'inclinaison des lignes (Skew correction) ;
- Correction de l'inclinaison des caractères (Slant correction) ;
- Estimation de la ligne de base [19].

Figure 2.8. Exemples de mots manuscrits avec des tailles différentes [19]

1.3. Segmentation

Cette étape consiste à diviser l'image de l'écriture manuscrite en caractères ou en mots individuels pour permettre une reconnaissance précise. Cela peut être accompli en utilisant des techniques telles que la détection de contour, la segmentation en ligne, la segmentation en bloc ou la segmentation hybride [25].

1.3.1. Segmentation explicite

Segmentation sur des critères topologiques, elle consiste à utiliser des points caractéristiques dans le mot [26] ; tels que les minimas locaux du contour supérieur, les espaces ou encore les points d'intersection. Cette approche est antérieure à la reconnaissance et n'est pas remise en cause pendant la phase de reconnaissance. Elle doit d'être d'une grande fiabilité car la moindre erreur remet en cause la totalité des traitements ultérieurs.

1.3.2. Segmentation implicite

Segmentation d'après les modèles de lettres, elle consiste à effectuer un découpage a priori de l'image en intervalles de grandeur régulière [27]. Cette technique est similaire à celle utilisée en reconnaissance de la parole, où le signal est divisé en intervalle de temps régulier. Contrairement à la segmentation explicite, il n'y a pas de pré-segmentation du mot. La segmentation s'effectue pendant la reconnaissance et est guidée par cette dernière. Le système cherche dans l'image, des composantes ou des groupements des graphèmes qui correspondent à ses classes des lettres. [28].

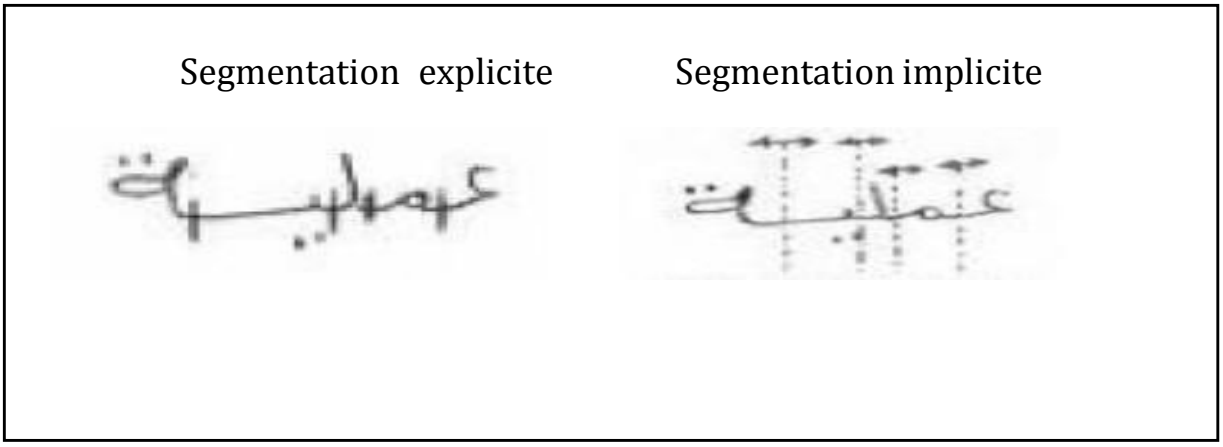


Figure 2.9. Les différents types de segmentation [19]

1.4. Extraction de caractéristiques

Cette étape consiste à extraire les caractéristiques clés des caractères ou des mots individuels pour les comparer à une base de données d'exemples connus. Les caractéristiques peuvent inclure des éléments tels que la forme, la taille, la position et l'orientation [22].

Les types de caractéristiques peuvent être classés en quatre groupes principaux :

Caractéristiques structurelles, caractéristiques statistiques, transformations globales, et superposition des modèles et corrélation [15].

1.4.1. Caractéristique structurelle

Les caractéristiques structurelles décrivent une forme en termes de sa topologie et sa géométrie en donnant ses propriétés globales et locales [22]. Parmi ces caractéristiques on peut citer :

- Les traits et les anses dans les différentes directions ainsi que leurs tailles.
- Les points terminaux.
- Les points d'intersections.
- Les boucles.
- Le nombre de points diacritiques et leur position par rapport à la ligne de base.
- Les voyellisions et les zigzags (hamza).
- La hauteur et la largeur du caractère.
- La catégorie de la forme (partie primaire ou point diacritique,…etc).
- Plusieurs autres caractéristiques peuvent être tirées, suivant qu'ils soient extraits d'une courbe, un trait ou un segment de contour.

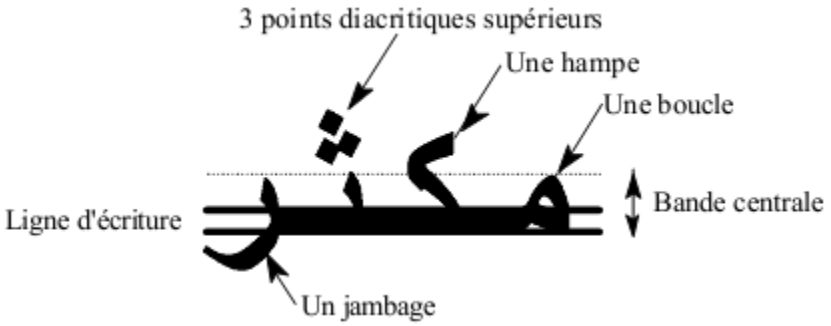


Figure 2.10. Caractéristiques structurelles dans un mot arabe

1.4.2. Caractéristique statistique

Ce type de caractéristiques décrivent une forme en termes d'un ensemble de mesures extraites à partir de cette forme. Les caractéristiques utilisées pour la reconnaissance de textes arabes sont : le zonage (zoning), les caractéristiques de lieu géométrique (Loci : pluriel de mot locus.En phonétique acoustique, zone des fréquences du spectre d'une consonne vers laquelle se dirigent les transitions des formants des voyelles adjacentes) et les moments [22].

- Le zonage consiste à superposer une grille $n \times m$ sur l'image du caractère et pour chacune des régions résultantes, calculer la moyenne ou le pourcentage de points en niveaux de gris, donnant ainsi un vecteur de taille $n \times m$ de caractéristiques.
- La méthode Loci (Venue de l'antiquité, cette méthode très facile fut seulement supplantée au XVIIIème siècle par la méthode du fichier numérique) est basée sur le calcul du nombre de segments blancs et de segments noirs le long d'une ligne verticale traversant la forme, ainsi que leurs longueurs [15].
- La méthode des moments : les moments d'une forme par rapport à son centre de gravité sont invariants par rapport à la translation et peuvent être invariants par rapport à la rotation [23]. Ils sont aussi indépendants de l'échelle. Ces caractéristiques peuvent être facilement et rapidement extraites d'une image de texte, ils peuvent tolérer modérément les bruits et les variations [24]. Les moments les plus utilisés pour l'écriture arabe sont les moments de Hu et les moments de Zernike.

1.4.3. Transformation globale

On parle de caractéristiques globales lorsque le codage ne fait pas intervenir la position spécifique d'éléments particuliers de l'image. L'image est considérée globalement sans chercher à distinguer les différentes zones.

1.4.4. Caractéristiques morphologiques

L'extraction des caractéristiques morphologiques s'appuie sur une étude des positions relatives des différentes composantes noires et blanches de l'image. On décrit alors le mot en termes de composantes blanches et noires, de cavités (parties blanches partiellement entourées de noir) et de boucles (parties blanches entièrement entourées de noir). La détection des

caractéristiques morphologiques peut être effectuée par des opérateurs morphologiques de dilatation selon les quatre directions, et d'intersection d'images.

1.4.5. Caractéristiques métriques

Cette catégorie comprend des caractéristiques basées sur des mesures physiques de l'image. Outre des caractéristiques assez simples, comme la hauteur, la largeur et le rapport de ces deux grandeurs, on peut utiliser des caractéristiques plus complexes comme le codage des profils. On peut définir les profils par rapport aux quatre axes naturels (gauche, droit, haut et bas), mais aussi par rapport à d'autres directions (des profils à 0°, 45°, 90° et 135°).

1.4.6. Caractéristiques adaptatives

Les caractéristiques adaptatives sont obtenues directement de l'image et requièrent une phase d'apprentissage. Autrement dit, le système opère sur une représentation proche de l'image d'origine et doit lui-même construire et optimiser l'extracteur de caractéristiques

1.5. Correspondance de modèles

Cette étape consiste à comparer les caractéristiques extraites aux exemples connus stockés dans une base de données pour identifier le caractère ou le mot correspondant. Cette étape peut être réalisée en utilisant des techniques telles que la reconnaissance optique de caractères (OCR), la reconnaissance de formes ou la classification basée sur l'apprentissage machine [21].

1.6. Post-traitement

Cette étape consiste à affiner les résultats de la reconnaissance d'écriture pour améliorer la précision du texte final. Cela peut inclure des techniques telles que la vérification orthographique, la correction de mots mal reconnus ou la fusion de mots mal segmentés.

En combinant ces étapes, un système de reconnaissance d'écriture peut convertir l'écriture manuscrite en texte numérique avec une précision élevée.

2. Certain travail de segmentation de mots arabes en caractères

Un certain nombre de méthodes de segmentation ont été développées en utilisant différentes stratégies, et dans la section suivante, le travail de divers auteurs sur la segmentation des textes arabes sera montré.

2.1. Sari et al dans [29]

Proposent un algorithme pour la segmentation des caractères arabes manuscrits appelé ACSA (Arabic Character Segmentation Algorithm). Après acquisition de l'image du texte, les mots sont normalisés puis lissés pour réduire le bruit et régulariser le contour des mots. Les pseudo-mots sont extraits de chaque mot par suivi du contour extérieur. L'étape suivante est l'extraction des caractéristiques et là ils utilisent différents types de caractéristiques dont on peut citer les boucles, hamza, hampes, jambages …, ces caractéristiques sont stockées dans une liste. Après détection de la ligne de base, en utilisant l'histogramme des projections horizontal. La ligne d'écriture est divisée en trois zones, une zone supérieure comportant les

hampes, une zone médiane comportant les jambages et une zone médiane comportant le corps principal de l'écriture. Chaque pseudo-mot est ensuite considéré séparément pour être segmenté en caractères. Les points de segmentation sont considérés comme les points minimaux du contour extérieur inférieur du pseudo-mot. Après détection de ces points, ils sont validés dans une autre étape en leur appliquant un ensemble de règles pour en générer un ensemble de points de segmentation valides (représentant une segmentation idéale en caractères). Ces segments sont remis au classifieur ainsi que leurs caractéristiques pour être reconnus.

2.2. Zoubeir Mouelhi [30]

Écrit en Visual Basic et tournant sous Windows, AraSeg est un ségmenteur à la fois à grammaire et à dictionnaire. L'analyse de chaque séquence de caractères est basée sur le modèle d'analyse du mot graphique en arabe dans lequel un mot graphique est considéré comme une suite de constituants immédiats. Le rôle principal de se segmenter est l'itémisation, la segmentation lexicale.

2.3. A.Hamid et R.Haraty dans [31]

Ils ont mis en avant une approche neuro- heuristic pour segmenter les mots manuscrits arabes. Il y a six étapes dans le processus. La première est l'acquisition de texte, et la seconde est sa binarisation en utilisant une technique heuristique qui crée une matrice avec une valeur de 1 pour un pixel noir et 0 pour un pixel blanc. La troisième étape consiste à extraire les pseudo-paroles. La quatrième étape implique la dérivation des caractéristiques de chaque colonne de la matrice de pixels.

Les caractéristiques sont topographiques en nature. La génération de points de pré-segmentation est la cinquième étape. Cette sur-segmentation est basée sur les caractéristiques extraterrestres de chaque colonne de matrice et une taille de caractère approximative. La sixième étape consiste à vérifier la validité des points de pré-segmentation. Un réseau neurones de 52 entrées, 4 couches cachés et 1 sortie est utilisé pour cela. Les attributs et les caractéristiques d'un point de pré-segmentation sont les entrées, et la validité de ce point est la sortie.

2.4. S.T. Masmoudi et al dans [32]

Présentent l'étape de segmentation associée à un modèle basé PHMM (modèles Markoviens cachés planaires) pour la reconnaissance hors-ligne des noms de villes tunisiennes manuscrites. Le procédé consiste en trois étapes. La première est une segmentation horizontale qui consiste à diviser l'image du texte en cinq zones logiques : la zone de points diacritiques supérieurs, la zone de points diacritiques inférieurs, la zone de hampes, la zone de jambages et la zone médiane. Après cela il y'a détection des boucles qui caractérisent la zone médiane. La seconde étape est appelée segmentation naturelle. Elle consiste à séparer les différents pseudo-mots en détectant l'espace entre les pseudo-mots dans la zone médiane. La troisième étape est appelée segmentation verticale. C'est l'étape la plus difficile dans le processus de segmentation. Elle consiste à examiner le contour supérieur du pseudo-mot pour localiser les points minimaux locaux, et les considérer comme points de

segmentation. Le résultat de cette étape est un ensemble de graphèmes qui peuvent correspondre à un caractère, un caractère sur-segmenté (une partie du caractère) ou à un caractère sou segmenté (deux caractères ou plus comme dans le cas de caractères ligaturés verticalement).

Conclusion

Dans ce chapitre nous avens explique le système de la reconnaissance des caractères arabes et les méthodes utilisées à cet effet.

Le principe de reconnaissance est basé sur l'obtention d'une base de données contenant des images de chaque caractère puis en passent par trois étapes nécessaires qui sont : le prétraitement puis l'extraction des attributs, puis la classification.

Nous avons choisi les réseaux de neurone comme classifieur qui seront détaillés ainsi que les algorithmes afférents dans les chapitrer qui suivent.

Chapitre 3

Les réseaux de neurones convolutionnels (CNN)

Introduction

Récemment, la disponibilité des données et l'amélioration considérable des puissances de calcul des GPU ont rendu l'apprentissage en profondeur, « Deep Learning », populaire par rapport aux autres techniques d'apprentissage automatiques. Un modèle de Deep Learning est simplement un modèle empilant un large nombre de couches de différents types de réseaux de neurones**.**

Dans ce chapitre, nous allons étudier un type particulier de réseaux de neurones : les réseaux de neurones convolutionnels CNN (convolutionnels Neural networks). Il s'agit d'une classe de convolution de neurones parfaitement adaptée aux données séquentielles.

1. Réseaux de neurones et neurosciences

Le cerveau contient environ 100 milliards de neurones. On compte de quelques centaines à plusieurs dizaines de milliers de contacts synaptiques par neurone. Le nombre total de connexions est estimé à environ 10 puissance 15.

La vitesse de propagation des influx nerveux est de l'ordre de 100m/s. C'est à dire bien inférieure à la vitesse de transmission de l'information dans un circuit électronique.

Propriétés collectives de réseaux, qui mettent en jeu des relations excitatrices ou inhibitrices.

Dans le cerveau, on distingue des sous-ensembles, des connexions internes et externes, des entrées et des sorties, réalisant certaines tâches.

Les recherches sur le fonctionnement du cerveau ont fait d'énormes progrès (exposition Cité des Sciences, 2002), tant du point de vue de la compréhension du fonctionnement qu'à l'échelle cellulaire et membranaire.

La connectique du cerveau ne peut pas être codée dans un "document biologique " tel l'ADN pour de simples raisons combinatoires. La structure du cerveau provient donc en partie des contacts avec l'environnement. L'apprentissage est donc indispensable à son développement.

2. Les réseaux de neurones artificiels

Haykin propose la définition suivante [33] :

Un réseau de neurones est un processeur massivement distribué en parallèle qui a une propension naturelle pour stocker de la connaissance empirique et la rendre disponible à l'usage. Il ressemble au cerveau sur deux aspects :

- La connaissance est acquise par le réseau au travers d'un processus d'apprentissage ;
- Les connexions entre les neurones, connues sous le nom de poids synaptiques servent à stocker la connaissance [34]. Selon [35] Un réseau de neurones est un circuit composé d'un nombre très important d'unités de calcul simples basées sur des

neurones. Chaque élément opère seulement sur l'information locale. Chaque élément opère de façon asynchrone ; il n'y a donc pas d'horloge générale pour le système [36].

3. Réseaux des neurones convolutionnels (CNN)

Nous allons présenter dans cette section les différents types de couches classiquement utilisés dans les CNN. Le but est d'exprimer leur connectivité indépendamment du modèle de neurone. Les réseaux de neurones convolutionnels sont à ce jour les modèles les plus performants pour classer des images.

Désignés par l'acronyme CNN, de l'anglais Convolutional Neural Network, ils comportent deux parties bien distinctes. En entrée, une image est fournie sous la forme d'une matrice de pixels. Elle a 2 dimensions pour une image en niveaux de gris. La couleur est représentée par une troisième dimension, de profondeur 3 pour représenter les couleurs fondamentales [Rouge, Vert, Bleu].

La première partie d'un CNN est la partie convolutive à proprement parler. Elle fonctionne comme un extracteur de caractéristiques des images. Une image est passée à travers une succession de filtres, ou noyaux de convolution, créant de nouvelles images appelées cartes de convolutions. Certains filtres intermédiaires réduisent la résolution de l'image par une opération de maximum local. Au final, les cartes de convolutions sont mises à plat et concaténées en un vecteur de caractéristiques, appelé code CNN. [37]

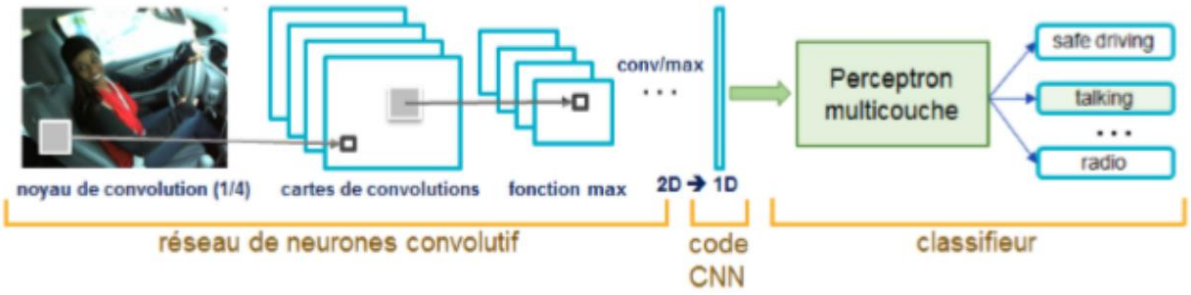


Figure 3.1. Architecture standard d'un réseau de neurone convolutionnel

Ce code CNN en sortie de la partie convolutive est ensuite branché en entrée d'une deuxième partie, constituée de couches entièrement connectées (perceptron multicouche). Le rôle de cette partie est de combiner les caractéristiques du code CNN pour classer l'image.

La sortie est une dernière couche comportant un neurone par catégorie. Les valeurs numériques obtenues sont généralement normalisées entre 0 et 1, de somme 1, pour produire une distribution de probabilité sur les catégories

3.1. Architecture de réseaux de neurone convolutionnels

3.1.1. La convolution

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

La convolution agit comme un filtrage. On définit une taille de fenêtre qui va se balader à travers toute l'image (rappelez-vous qu'une image peut être vue comme étant un tableau).

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

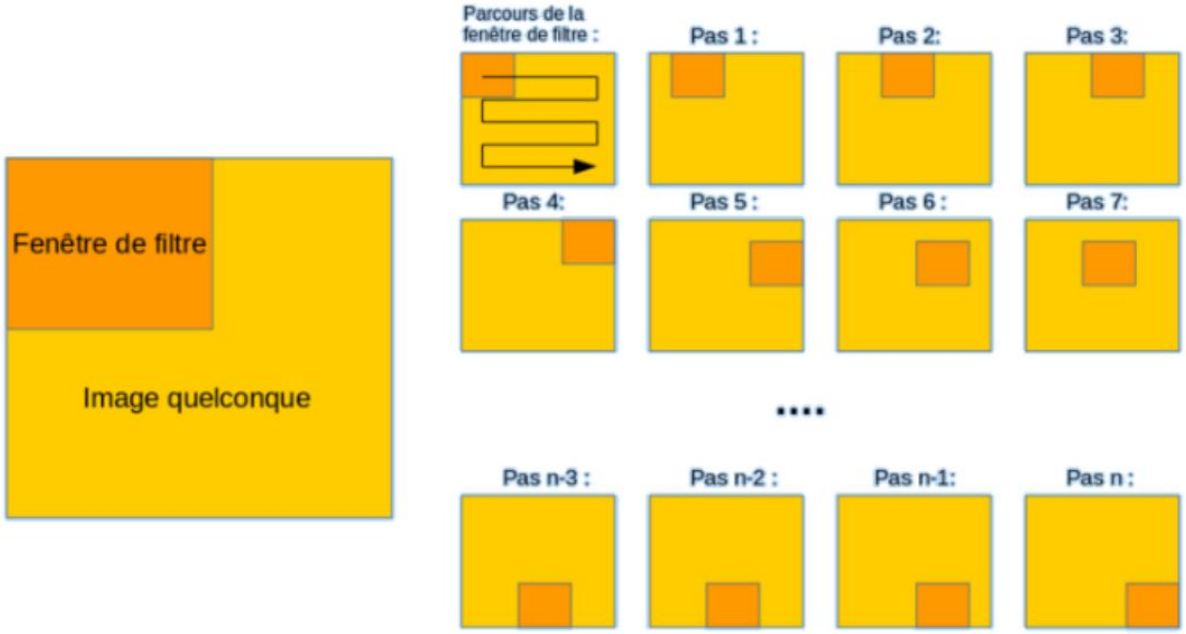


Figure 3.2. Schema du parcours de la fenêtre de filtre sur l'image.

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

La convolution va effectuer une opération avec ces 9 pixels. Il peut s'agir de n'importe quelle opération, par exemple on extrait la valeur la plus grande (soit le pixel avec la plus grande valeur).

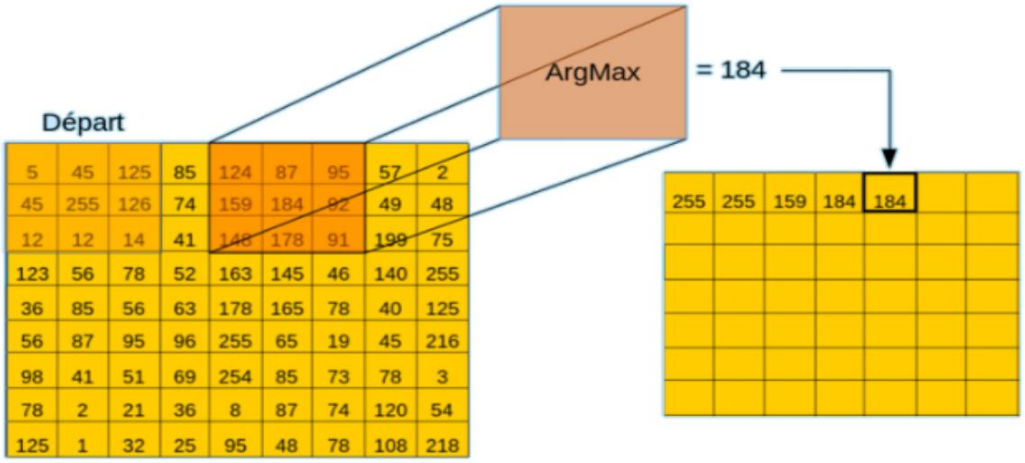


Figure 3.3. Exemple d'une convolution dont la configuration est : opération = argument maximale, pas horizontale = 1 pixel, pas vertical = 1 pixel.

On fait glisser la fenêtre en orange et à chaque pas on récupère la valeur la plus grande parmi les 9 valeurs de pixels.

On remarque que la sortie de la convolution, que l'on peut appeler « carte de caractéristiques », à des dimensions plus petites que celle de l'image en entrée.

3.1.2. Couche de mise en commun (pooling)

Ce type de couche est souvent placé entre deux couches de convolution : elle reçoit en entrée plusieurs feature maps, et applique à chacune d'entre elles l'opération de pooling.

L'opération de pooling (ou sub-sampling) consiste à réduire la taille des images, tout en préservant leurs caractéristiques importantes.

Pour cela, on découpe l'image en cellules régulière, puis on garde au sein de chaque cellule la valeur maximale. En pratique, on utilise souvent des cellules carrées de petite taille pour ne pas perdre trop d'informations. Les choix les plus communs sont des cellules adjacentes de taille 2 x 2 pixels qui ne se chevauchent pas, ou des cellules de taille 3 x 3 pixels, distantes les unes des autres d'un pas de 2 pixels (qui se chevauchent donc). On obtient en sortie le même nombre de feature maps qu'en entrée, mais celles-ci sont bien.

La couche de *pooling* permet de réduire le nombre de paramètres et de calculs dans le réseau. On améliore ainsi l'efficacité du réseau et on évite le sur-apprentissage.

Il est courant d'insérer périodiquement une couche Pooling entre les couches Conv successives dans une architecture ConvNet. Sa fonction est de réduire progressivement la taille spatiale de la représentation afin de réduire la quantité de paramètres et de calculs dans le réseau, et donc de contrôler également le sur-ajustement. La couche de pooling fonctionne indépendamment sur chaque tranche de profondeur de l'entrée et la redimensionne spatialement, en utilisant l'opération MAX.

Ainsi, la couche de pooling rend le réseau moins sensible à la position des features: le fait qu'une feature se situe un peu plus en haut ou en bas, ou même qu'elle ait une orientation

légèrement différente ne devrait pas provoquer un changement radical dans la classification de l'image.

3.1.3. Couche ReLU

Pour améliorer l'efficacité du traitement en intercalant entre les couches de traitement une couche qui va opérer une fonction mathématique (fonction d'activation) sur les signaux de sortie. Dans ce cadre on trouve ReLU (*Rectified Linear Units*) désigne la fonction réelle non-linéaire définie par $ReLU(x) = max(0, x)$.

La couche de correction ReLU remplace donc toutes les valeurs négatives reçues en entrées par des zéros. Elle joue le rôle de fonction d'activation Souvent, la correction Relu est préférable, mais il existe d'autre forme

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

3.1.4. Couche fully-connected

La couche fully-connected constitue toujours la dernière couche d'un réseau de neurones, convolutif ou non-elle n'est donc pas caractéristique d'un CNN.

Ce type de couche reçoit un vecteur en entrée et produit un nouveau vecteur en sortie. Pour cela, elle applique une combinaison linéaire puis une fonction éventuellement d'activation aux valeurs reçues en entrée.

La couche fully-connected permet de classifier l'image en entrée du réseau : elle renvoie un vecteur de taille N, où N est le nombre de classes dans notre problème de classification d'images. Chaque élément du vecteur indique la probabilité pour l'image en entrée d'appartenir à une classe.

Par exemple, si le problème consiste à distinguer les chats des chiens, le vecteur final sera de taille 2: le premier élément (respectivement, le deuxième) donne la probabilité d'appartenir à la classe "chat" (respectivement "chien"). Ainsi, le vecteur [0.90.1] signifie que l'image a 90% de chances de représenter un chat.

Chaque valeur du tableau en entrée "vote" en faveur d'une classe. Les votes n'ont pas tous la même importance : la couche leur accorde des poids qui dépendent de l'élément du tableau et de la classe.

Pour calculer les probabilités, la couche fully-connected multiplie donc chaque élément en entrée par un poids, fait la somme, puis applique une fonction d'activation (logistique si $N = 2$, softmax si $N > 2$):

Il est possible de remplacer les couches entièrement connectées d'un CNN par des couches convolutionnelles, ce qui le rend entièrement convolutionnel. All-convolutional network est une bonne idée je vous invite à le découvrir.

4. Les limites des architectures CNN

Dans la première partie de ce chapitre, nous avons découvert les réseaux de neurones convolutionnels, dans lesquels l'information transite des entrées vers les sorties. Les perceptrons multicouches (PMC, ou MLP pour MultiLayer Perceptrons, en anglais) .

Aujourd'hui, ces modèles sont la référence pour l'apprentissage machine, car ils ont de nombreux avantages :

- Performances à l'état de l'art ;
- Rapides en décision (mais longs en apprentissage !) ;
- Supportent très bien des hautes dimensions en entrée et/ou en sortie ;
- Capables d'extraire des caractéristiques automatiquement, notamment sur les images avec les CNN ;
- Existence d'un algorithme efficace d'apprentissage : la rétropropagation du gradient.

4.1. Limites liées à la taille des données

Malheureusement, les modèles CNN a également une contrainte importante : la taille de l'entrée et la taille de sortie doivent être identiques pour tous les exemples à traiter, aussi bien en apprentissage qu'en décision.

Si l'on prend l'exemple d'un problème de reconnaissance d'image de chiffres manuscrits, les architectures CNN imposent que toutes les images de chiffres soient de même taille par exemple 28×28 pixels. Bien évidemment, ce n'est pas du tout représentatif de la réalité, comme on peut le constater ci-dessous !

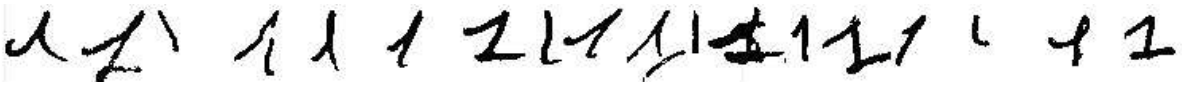


Figure 3.4. Exemples de chiffres manuscrits

La taille en pixels est variable selon les scripteurs et les exemples.

Dans ce dernier exemple, les données sont de taille variable aussi bien en entrée qu'en sortie du modèle. Dans certains problèmes, la taille des données d'entrée peut être différente de la taille des données de sortie. En restant dans l'exemple de la traduction automatique :

1. "How are you?" -> "Comment vas-tu ?" (3 mots -> 3 mots).
2. "I am fine, thank you" -> "Je vais bien, merci" (5 mots -> 4 mots).

Idéalement, nous aimerions donc avoir un modèle capable de prendre en compte des données de taille variable, aussi bien en entrée qu'en sortie. Pour tous ces problèmes, nous allons désormais parler de "signaux" d'entrée et de sortie

4.2. L'astuce de la fenêtre glissante

Afin de traiter les signaux de taille variable, une astuce très répandue consiste à utiliser une "fenêtre glissante" qui va se déplacer dans le sens de parcours du signal pour l'analyser. À chaque déplacement, le contenu de la fenêtre sera soumis à un classifieur statique, tel qu'un réseau de neurones convolutionnels.

Prenons l'exemple de la reconnaissance d'écriture manuscrite :

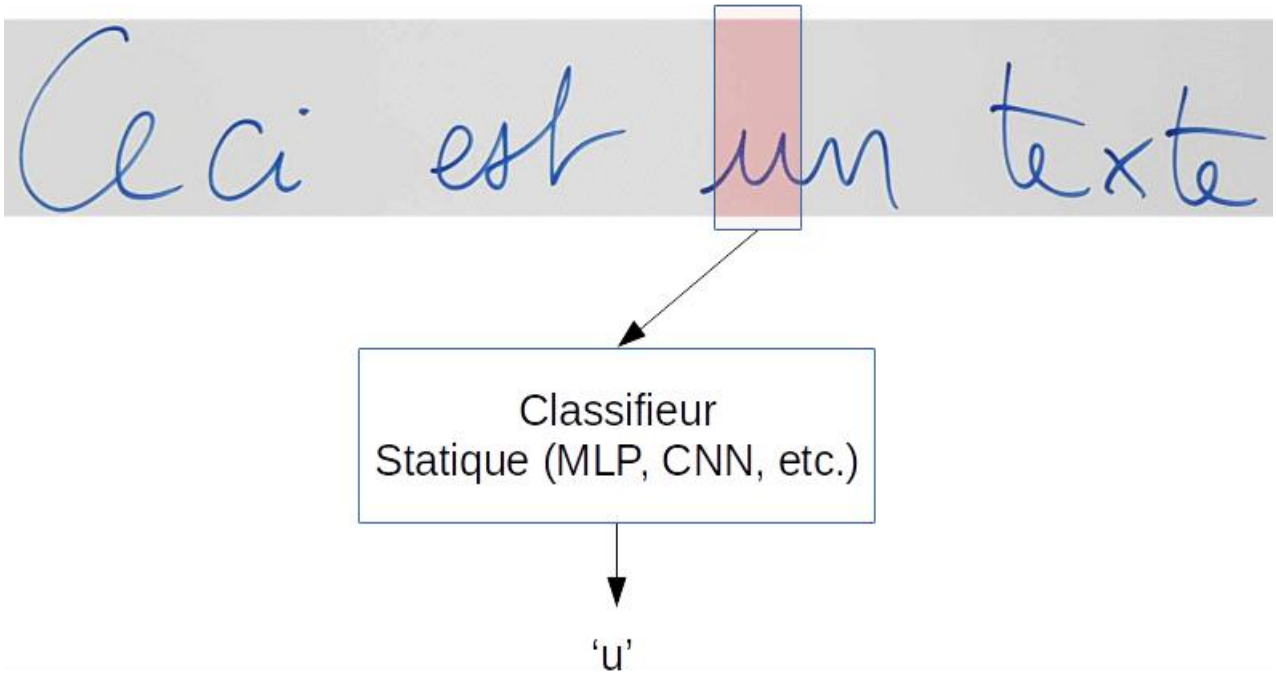


Figure 3.5. Fenêtre glissante pour la reconnaissance d'écriture

Dans cet exemple, un classifieur statique de type CNN ou MLP entraîné sur des caractères isolés devrait être capable de produire la séquence finale attendue :
"Ceci?est?un?texte"

Vous pouvez noter la présence d'une classe de caractère spécial '?' pour modéliser les espaces, et qui peut également modéliser un doute sur le caractère présent, notamment lorsque le caractère est mal écrit, ou lorsque la fenêtre est à cheval sur plusieurs caractères.

4.3. Limites liées aux dépendances entre les données

Lorsque l'on traite des signaux, il existe généralement des dépendances entre les données d'entrée et/ou les signaux de sortie.

Par exemple, essayez de reconnaître le signal suivant :

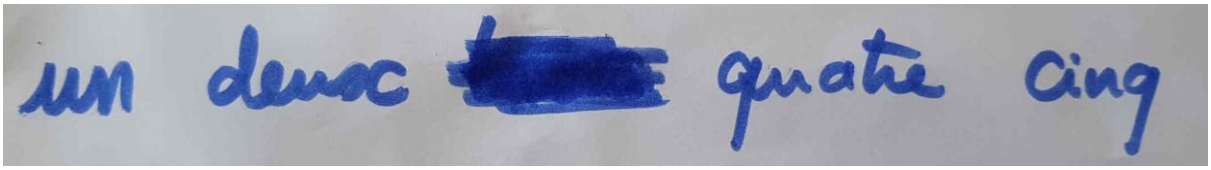


Figure 3.6. Exemples liés aux dépendances entre les données

Il est facile pour un être humain de reconnaître cette séquence malgré la rature, car nous sommes capables d'analyser le contexte du reste de la phrase. En revanche, pour une machine, la reconnaissance du mot raturé sera beaucoup plus compliquée ! En effet, l'astuce de la fenêtre glissante ci-dessus donnera probablement la séquence suivante :

"un?deux??????quatre?cinq", qui n'est pas la bonne réponse...

Idéalement, nous aimerions que les réseaux de neurones soient doués de notre faculté de mémoire !

Conclusion

Dans ce chapitre, il était question des réseaux de neurones et particulièrement des réseaux de neurones convolutionnels. Nous avons détaillé leur conception et leur utilisation dans le traitement de la reconnaissance des caractères arabes. Nous allons procéder dans le chapitre qui suit à l'application de ces réseaux de neurone convolutionnels dans la reconnaissance des caractères arabes.

Ces réseaux sont capables d'extraire des caractéristiques d'images présentées en entrée et de classifier ces caractéristiques. Ils implémentent l'idée de partage des poids permettant ainsi de réduire d'une part le nombre de paramètres libres de l'architecture, d'autre part de réduire les temps de calcul, l'espace mémoire nécessaire, et améliorer ainsi les capacités de généralisation du réseau.

Chapitre 4
Conception de l'architecture CNN pour la reconnaissance des caractères arabes

Introduction

Dans ce chapitre nous allons aborder la dernière partie qui représente la partie réalisation de notre projet, en se basant sur les mécanismes évoqués précédemment dans le chapitre de réseaux de neurones.

Ce chapitre est composé de deux parties : la première présente l'environnement notre logiciel, la deuxième présente les résultats des tests effectués.

1. Logiciel et librairie utilise dans l'implémentation

1.1. Dataset :

La base de données AHDC est une base de données d'images de lettres manuscrites arabes, créée pour la reconnaissance de caractères manuscrits arabes à des fins de recherche et de développement.

La base de données contient de 247517 images de lettres manuscrites arabes, représentant les 28 lettres de l'alphabet arabe. Les lettres se trouvent dans les différentes positions (isolée, début, milieu et fin) qui ne donne 40 classes de lettres.

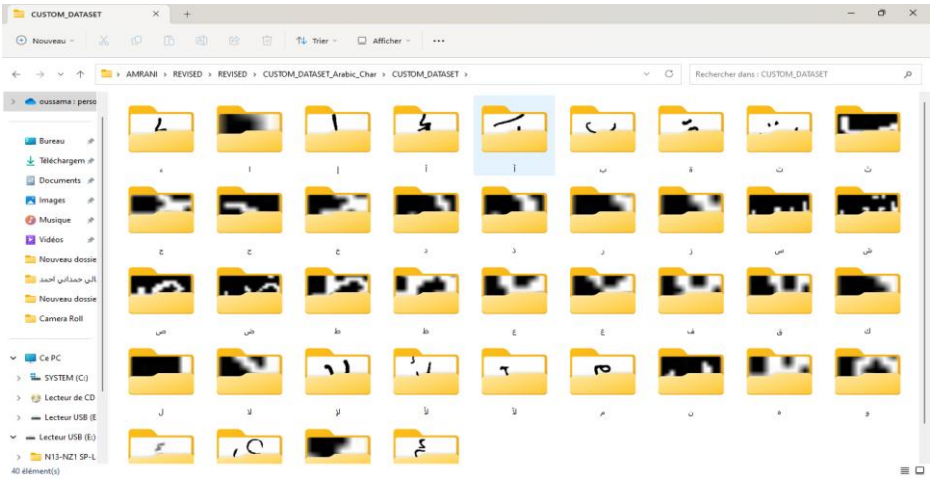


Figure 4.1. La base de donne des caractères.

Les images de la base de données AHDC ont été numérisées en niveaux de gris à une résolution de 300 dpi et ont été divisées en 40 classes différentes. Chacune de ces classes représente une lettre dans une position.

1.2. Python

Python est un langage de programmation de haut niveaux interprété (il n'y a pas d'étape de compilation) et oriente objet avec sémantique dynamique il est très sollicite par une large

communauté de développeurs et de programmeurs, python est un langages faciles à apprendre et permet une bonne réduction du cout de la maintenance des codes, les bibliothèque (packages) python encourageant la modularité et la réutilisabilité des codes.

Python est ses bibliothèque sons disponible (en source au en binaire) sans charger pour la majorette des plateformes et peuvent être redistribués gratuitement.

1.3. Modelés de Keras utiliser

Les modules de Keras que nous avons utilisés sont :

- Keras.models:
  - Sequential
- Keras.layers:
  - Conv2D
  - MaxPooling2D
  - Flatten
  - Dense
  - Dropout
- Keras.preprocessing.image:
  - ImageDataGenerator
  - image

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                 Output Shape              Param #
=================================================================
 conv2d (Conv2D)              (None, 25, 25, 32)        320

 conv2d_1 (Conv2D)            (None, 25, 25, 32)        9248

 max_pooling2d (MaxPooling2D  (None, 12, 12, 32)        0
 )

 batch_normalization (BatchN  (None, 12, 12, 32)        128
 ormalization)

 dropout (Dropout)            (None, 12, 12, 32)        0

 conv2d_2 (Conv2D)            (None, 12, 12, 64)        18496

 conv2d_3 (Conv2D)            (None, 12, 12, 64)        36928

 max_pooling2d_1 (MaxPooling  (None, 6, 6, 64)          0
 2D)

 batch_normalization_1 (Batc  (None, 6, 6, 64)          256
 hNormalization)
```

```
dropout_1 (Dropout)          (None, 6, 6, 64)          0

conv2d_4 (Conv2D)            (None, 6, 6, 128)         73856

conv2d_5 (Conv2D)            (None, 6, 6, 128)         147584

batch_normalization_2 (Batc  (None, 6, 6, 128)         512
                        hNormalization)

dropout_2 (Dropout)          (None, 6, 6, 128)         0

max_pooling2d_2 (MaxPooling  (None, 3, 3, 128)         0
                        2D)

batch_normalization_3 (Batc  (None, 3, 3, 128)         512
                        hNormalization)

dropout_3 (Dropout)          (None, 3, 3, 128)         0

flatten (Flatten)            (None, 1152)              0

dense (Dense)                (None, 512)               590336

dropout_4 (Dropout)          (None, 512)               0

dense_1 (Dense)              (None, 256)               131328

dropout_5 (Dropout)          (None, 256)               0

dense_2 (Dense)              (None, 40)                10280

=================================================================
                    Total params: 1,019,784
                Trainable params: 1,019,080
                Non-trainable params: 704
_________________________________________________________________
```

Figure 4.2. Model de CNN utilise

2. Configuration utiliser dans l'implémentation

La configuration de matériel utiliser dans notre implémentation est :

- PC portable DELL I5 CPU 8eme generation 2.70 GHz
- Disque Dure de Taille 512SSD
- RAM de Taille 8 GO DDR4.
- Système d'exploitation Windows 10

3. Architecture du réseau

L'architecture de notre réseau pour la reconnaissance des caractères arabes manuscrit est basé sur le réseau de neurones convolutionnels.

Le modèle est un réseau de neurones convolutifs pour la classification d'images en 40 classes. Le modèle prend une image en entrée de taille 32x32 et passe par Cinq couches de convolution avec des tailles de filtre de 3x3 et des filtres de 32, 64, 128 et 256. Après chaque

couche de convolution, une fonction d'activation ReLU est appliquée, suivie d'une couche de Maxpooling pour réduire la taille de l'image et le nombre de paramètres. Le modèle utilise également des couches de normalisation de batch pour accélérer l'apprentissage et des couches de dropout pour réduire le surapprentissage.

Après les cinq couches de convolution, le modèle utilise deux couches fully connected avec 512 et 256 neurones respectivement, où la fonction d'activation utilisée est ReLU. Les deux couches fully connected ont également une régularisation L1L2 pour contrôler le surapprentissage. La dernière couche est une couche de sortie avec une fonction d'activation softmax pour calculer la distribution de probabilité des 40 classes.

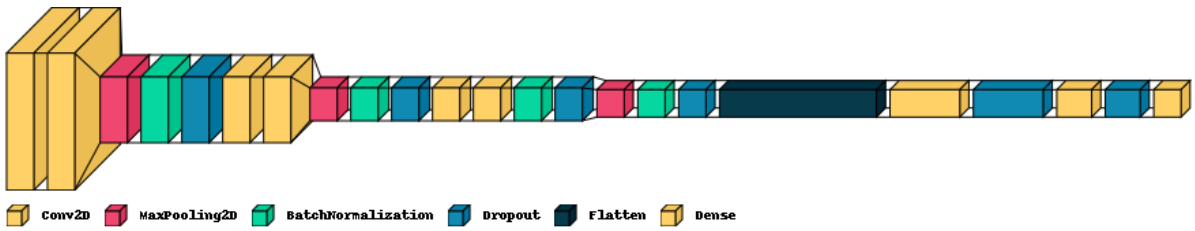


Figure 4.3. Architecture de réseaux.

Code Source de notre model proposé :

```
filtre = 5 #taille de filtre
strides=1 #le pas de convolution
convolution1=32 # nombre de filtre (32 features maps)
convolution2=64# nombre de filtre (64features maps)
convolution3 = 128 # nombre de filtre (128 features maps)
pool_size=2 # taille de pool
dropout1 = 0.25 #appliquer un dropout avec probabilité de 25 %
dropout2=0.50 #appliquer un dropout avec probabilité de 50 %
couchecache1= 256 #créer une première couche de 256 neurones
```

Trace la distribution des classes dans les données du train :

Pour trace la distribution des classes dans les données du train en utilise les commandes suivent :

```
fig, ax = plt.subplots(figsize=(10, 10))
g = sns.countplot(data=train_df, x="y", color="c")
g.set_xticklabels(characters);
```

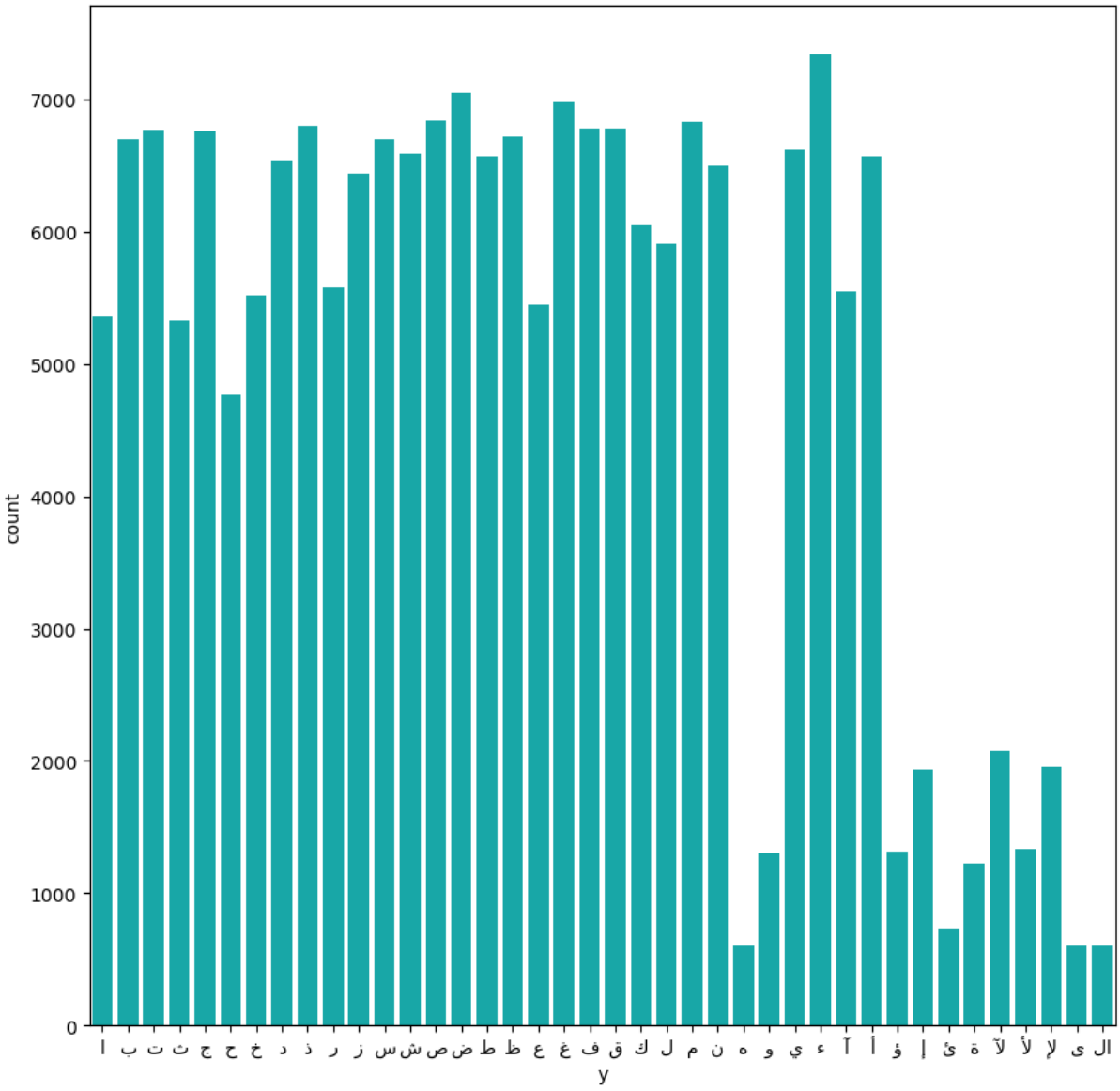


Figure 4.4. La distribution des classes dans les données

FONCTIONS DE PRÉTRAITEMENT DES IMAGES :

- Convertissez une image en niveaux de gris à l'aide de TensorFlow.
- Binarisez une image à l'aide de TensorFlow. Pixels d'une valeur supérieure à 0,5 sera défini sur 1,0 et les autres sur 0,0.
- Binarisez une image à l'aide d'OpenCV. Pixels d'une valeur supérieure à 127 sera réglé sur 255 (blanc) et les autres sur 0 (noir). Ensuite, l'image est remodelée pour ajouter une dimension supplémentaire et réduit d'un facteur 255.

```
def to_gray(img):

    return tf.image.rgb_to_grayscale(img)

def tf_binarize(img):

    return tf.cast(tf.greater(img, 0.5), tf.float32)

def pre_gen_binarize(img):

    img = img[:,:,0]
    _, binary_img = cv2.threshold(img, 127, 255, cv2.THRESH_BINARY)
    binary_img = binary_img.reshape(binary_img.shape[0], binary_img.shape[1], 1)
    return binary_img // 255
```

Cree le model :

```
def build_model(image_dim, num_classes, optimizer='adam', activation='relu', kernel_initializer='he_normal'):
    model = Sequential()
```

Pour crée le model on utilise la fonction build_model est en déclare les paramètres

```
model.add(Conv2D(filters = 32, kernel_size = (3,3),padding = 'same',
                 activation ='relu', input_shape = image_dim))
model.add(Conv2D(filters = 32, kernel_size = (3,3), padding = 'same',
                 activation ='relu'))
```

Cette commande ajoute deux couches de convolution avec 32 filtres de taille 3x3, une bordure de remplissage ("padding") égale et une fonction d'activation ReLU. L'entrée est une image de taille (img_size, img_size, 1)

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

Cette ligne de commande ajoute une couche de max pooling pour réduire la taille de l'image de moitié.

```
model.add(BatchNormalization())
```

Cette commande ajoute une couche de batch normalization pour normaliser les activations entre les couches. La normalisation par lots peut aider à accélérer la convergence de l'apprentissage en réduisant les fluctuations dans les activations.

```
model.add(Dropout(0.25))
```

Cette commande ajoute une couche de dropout avec un taux de 0,25 pour régulariser le modèle. Le dropout consiste à désactiver aléatoirement un certain pourcentage des neurones d'une couche pendant l'apprentissage, ce qui peut aider à prévenir le sur-apprentissage.

```
model.add(Conv2D(filters = 64, kernel_size = (3,3), padding = 'same',
                 activation ='relu'))
model.add(Conv2D(filters = 64, kernel_size = (3,3), padding = 'same',
                 activation ='relu'))
```

Cette commande ajoute deux couches de convolution avec 64 filtres de taille 3x3, une bordure de remplissage ("padding") égale et une fonction d'activation ReLU.

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

Cette ligne de commande ajoute une autre couche de max pooling pour réduire la taille de l'image de moitié.

```
model.add(BatchNormalization())
```

Cette commande ajoute une couche de batch normalization pour normaliser les activations entre les couches.

```
model.add(Dropout(0.25))
```

Cette commande ajoute une couche de dropout avec un taux de 0,25 pour régulariser le modèle.

```
model.add(Conv2D(filters = 128, kernel_size = (3,3), padding = 'same',
                 activation ='relu'))
model.add(Conv2D(filters = 128, kernel_size = (3,3), padding = 'same',
                 activation ='relu'))
```

Cette commande ajoute deux couches de convolution avec 128 filtres de taille 3x3, une bordure de remplissage ("padding") égale et une fonction d'activation ReLU.

```
model.add(BatchNormalization())
model.add(Dropout(0.25))

model.add(MaxPooling2D(pool_size=(2,2), strides=(2,2)))
model.add(BatchNormalization())
model.add(Dropout(0.25))
```

Cette ligne de commande ajoute une autre couche de max pooling pour réduire la taille de l'image de moitié et deux couches de batch normalization pour normaliser les activations entre les couches aussi ajoute deux couches de dropout avec un taux de 0,25 pour régulariser le modèle.

```
model.add(Flatten())
model.add(Dense(512, activation = "relu",
                kernel_regularizer=regularizers.L1L2(l1=1e-5, l2=1e-4),
                bias_regularizer=regularizers.L2(1e-4),
                activity_regularizer=regularizers.L2(1e-5)))
model.add(Dropout(0.5))
model.add(Dense(256, activation = "relu",
                kernel_regularizer=regularizers.L1L2(l1=1e-5, l2=1e-4),
                bias_regularizer=regularizers.L2(1e-4),
                activity_regularizer=regularizers.L2(1e-5)))
model.add(Dropout(0.5))
model.add(Dense(num_classes, activation = "softmax"))
```

Cette commande ajoute une couche de Flatten pour transformer les feature maps en un vecteur 1D.

Après une couche dense (fully connected) avec 512 neurones, une fonction d'activation ReLU et une régularisation L1L2. Cette régularisation utilise les paramètres l1 et l2 pour appliquer des pénalités sur les poids et les biais de la couche, ce qui peut aider à prévenir le sur-apprentissage en réduisant la complexité du modèle. Après on ajoute une couche de dropout avec un taux de 0,5 pour régulariser le modèle. Refaire la même chois on ajoute une autre couche dense avec 256 neurones, une fonction d'activation ReLU et une régularisation L1L2, une couche de dropout avec un taux de 0,5 pour régulariser le modèle. A la fine on ajoute une couche de sortie avec un nombre de neurones égal au nombre de classes (ici, num_classes) et une fonction d'activation softmax pour calculer la probabilité de chaque classe.

```
model.compile(loss='categorical_crossentropy',
              metrics=['accuracy'],
              optimizer=optimizer)
return model
```

Cette commande permettre de Compiler le model.

Avent de compile le model on a créé des générateurs pour augmenter les données de Train et rendre le model plus généralisable

```
generators = Generators(train_df, test_df)
```

```
Found 198013 validated image filenames belonging to 40 classes.
Train generator created
Found 49504 validated image filenames belonging to 40 classes.
Test generator created
```

Instanciation de générateurs de Train et de Test

```
num_epochs = 50
batch_size = 1024
validation_ratio = 0.1
```

Cette commande permettre de définir les paramètres de train.

```
history = model.fit(generators.train_generator,
                    epochs=50,
                    validation_data=generators.test_generator,
                    verbose=1)
```

Cette commande permettre de lancement de l'apprentissage du modèle sur 50 epoch.

4. Résultat :

Après l'analyse des résultats obtenus, On constate les remarques suivantes :

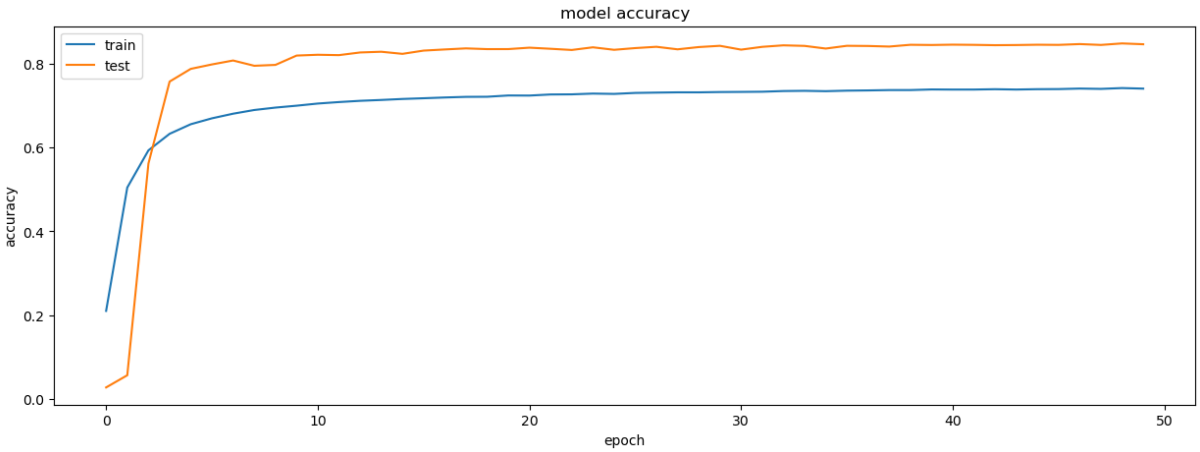


Figure 4.5. Le model accuracy

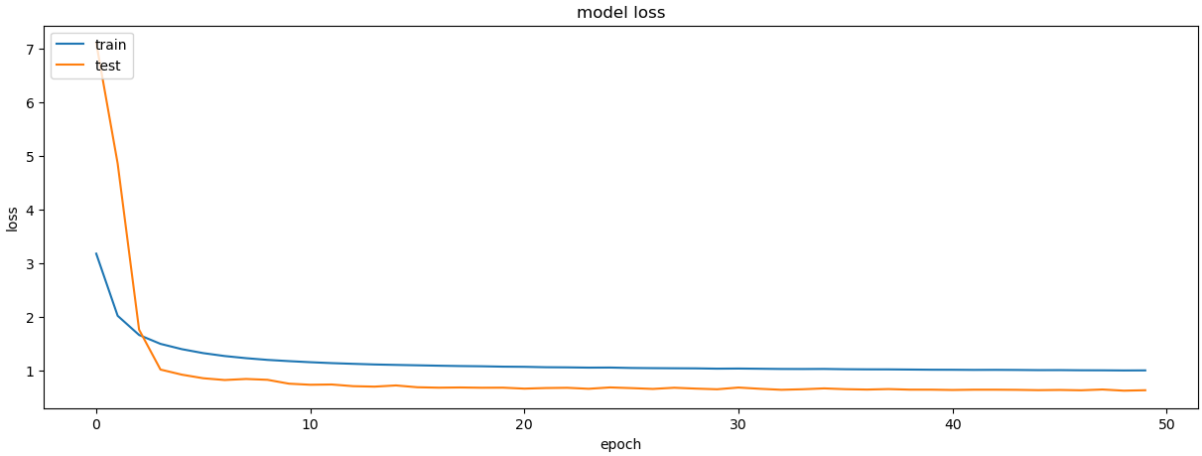


Figure 4.6. Le model LOSS

Test Accuracy : 0.84722

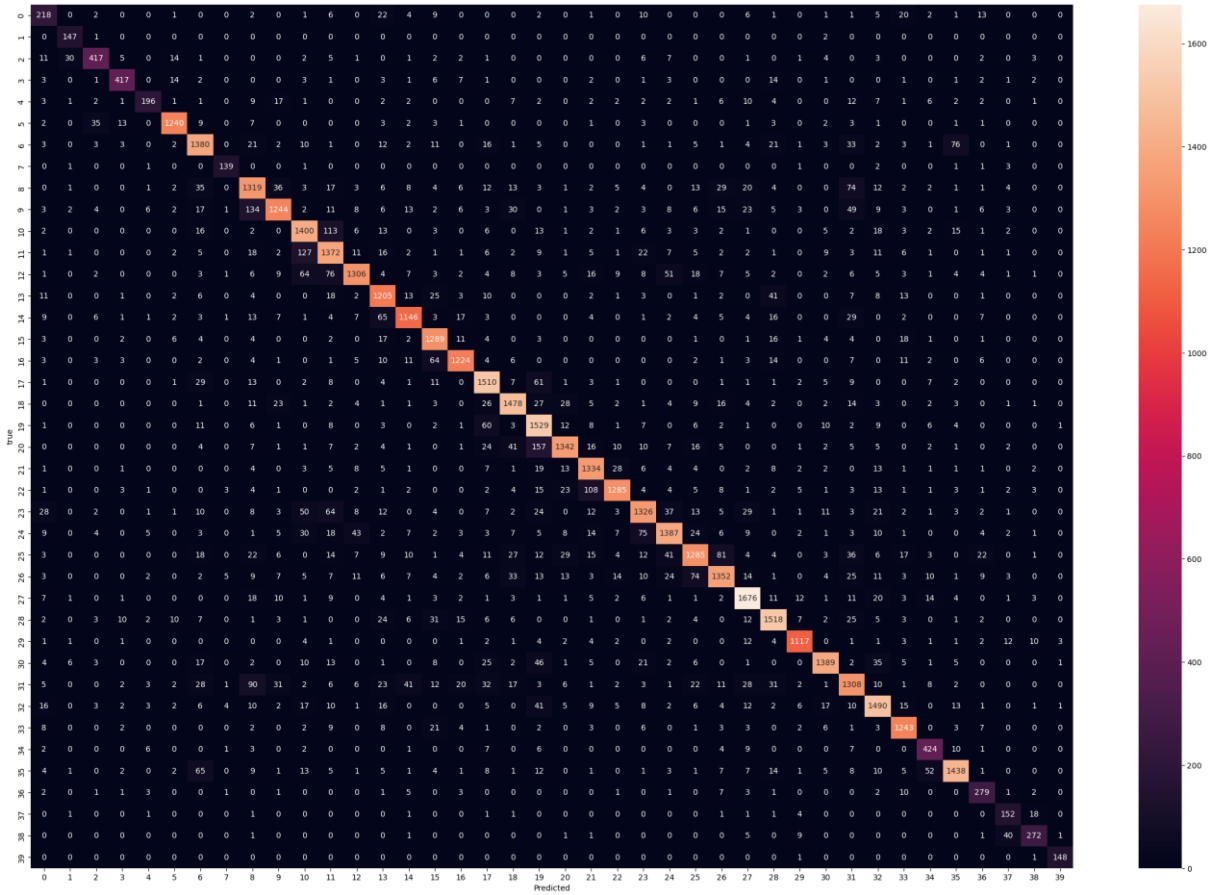


Figure 4.7. Test Accuracy

La courbe d'accuracy montre l'évolution de la précision du modèle sur les données d'entraînement et de validation au fil des epochs. L'accuracy d'entraînement augmente progressivement et atteint une valeur élevée de 0.8472

à la fin de l'entraînement. En revanche, l'accuracy de validation augmente également au début de l'entraînement, mais elle commence à stagner aux alentours de l'epoch 10 et n'augmente plus significativement par la suite. Cela indique que le modèle commence à sur-apprendre (overfitting).

La courbe de loss montre l'évolution de l'erreur du modèle sur les données d'entraînement et de validation au fil des epochs. La loss d'entraînement diminue progressivement et atteint une valeur faible de 0,7927 à la fin de l'entraînement. En revanche, la loss de validation diminue également au début de l'entraînement, mais elle augmente ensuite à partir de l'epoch 4, atteignant une valeur élevée de 4,6148 à l'epoch 3. Cela indique que le modèle commence à sur-apprendre les données d'entraînement.

```
visualize("test-1.png")
predicted= OCR("test-1.png")
```

A la fine on a utilisé la fonction prédéfinie OCR pour teste le programme et on a donné une image avec d'écrite manuscrite.

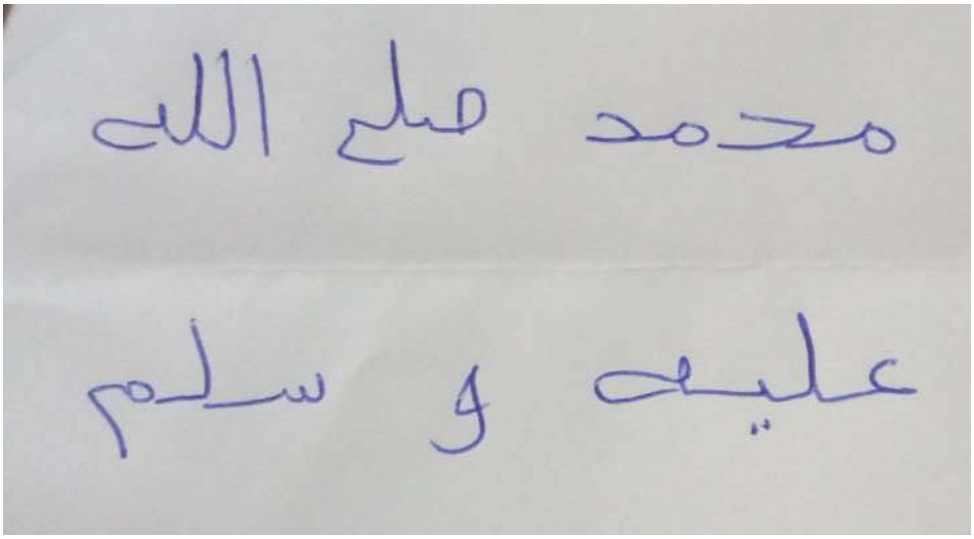


Figure 4.8. Image Test

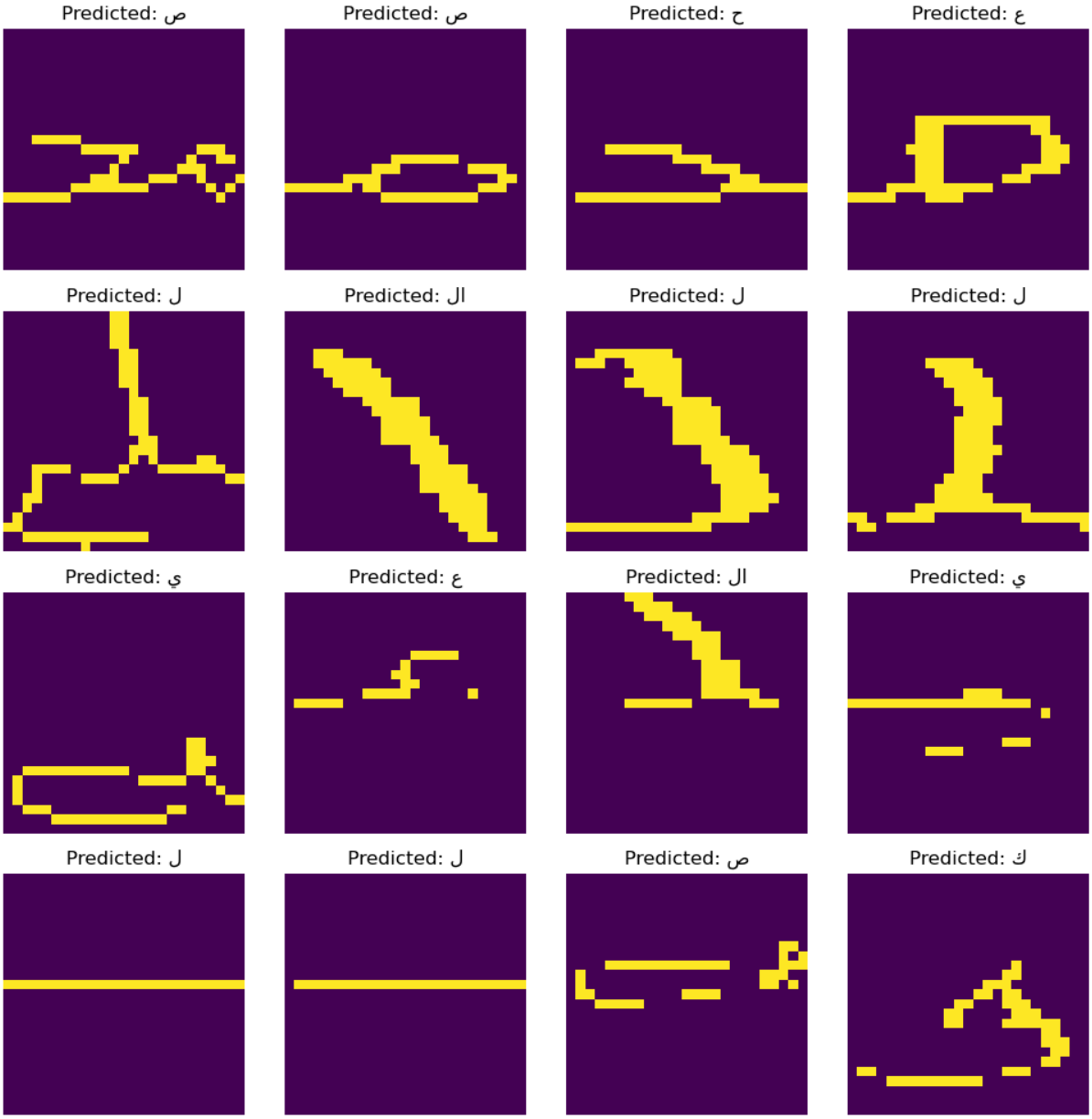


Figure 4.9. Résultat de test

Conclusion

Nous avons montré dans ce chapitre l'utilisation des réseaux de neurone pour la reconnaissance des caractères arabe manuscrits.

Les résultats obtenus sont très encourageants avec un taux d'exactitudes de classification de 80% et ce malgré la complexité de chaque caractère et la similitude entre les caractères.

Conclusion générale

La reconnaissance automatique de l'écriture arabe manuscrite demeure un domaine de recherche et d'exploration de méthodes et techniques modernes pour la conception de procédures, logiciels modernes optimisés à l'instar des autres langues, et ce en dépit de la portée de cette langue au travers le monde (400 millions de locuteurs /internet).

Nous constatons que les avancées technologiques n'ont pas pour autant fait avancer les recherches sur cette problématique qui nécessite plus d'effort et peut être plus de volonté.

Notre étude a consisté d'une part, à comprendre la problématique, d'analyser les différentes solutions proposées par les chercheurs pour la reconnaissance des caractères arabes manuscrits, d'explorer d'autres part les techniques et méthodes de traitement utilisées pour cette reconnaissance et de concevoir un système relativement fiable, optimisé pouvant contribuer à mieux cerner les difficultés relatives à la reconnaissance de l'écriture arabe compte tenu de la spécificité de ses caractères, tel qu'expliqué dans le document (caractère, au début d'un mot, au milieu du mot, à la fin d'un mot).

Nous rappelons les différentes méthodes et processus utilisés pour atteindre notre objectif avec une marge d'erreur acceptable, en référence aux moyens et disponibilité d'information.

La première phase de notre étude s'est articulée sur la compréhension des caractéristiques des caractères arabes en prenant connaissance des différentes méthodes de classification et processus de reconnaissance évoquées dans les différentes expériences y afférentes.

Nos différentes recherches et tests ont conclu en la méthode la plus efficace soit : le réseau des neurones convolutionnels.

Nous avons détaillé la méthodologie d'utilisation de ce processus en mettant en exergue les avantages liés à la reconnaissance de l'écriture arabe manuscrite ; cependant tel qu'indiqué sur les recherches ayant trait à notre sujet, il est recommandé d'examiner et de déterminer les règles et algorithme de recherche sur les réseaux des neurones convolutionnels, considérés assez importants pour le domaine de reconnaissance des caractères arabes.

Partant de ce constat, nous avons expliqué l'architecture de ce réseau et conçu la logique de notre logiciel et procédé aux tests nécessaires et a différentes simulations de cas pour aboutir à des résultats satisfaisants (objet de la démonstration) ; néanmoins, la comparaison de fiabilité entre la reconnaissance des caractères est assez mitigée, du fait de la spécificité des caractères arabes, tel que précisé plus haut :

Les lettres sont 28 et ils prennent des positions différentes.

La base de données des caractères est (198013 = training +49504 = test).

Considérant ces observations et les marges d'erreurs susceptibles de fausser les résultats, nous pouvons tirer les enseignements suivants :

Le réseau de neurone convolutionnel est basé sur une importante base de données pour l'expérimentation et l'apprentissage qui nécessite des techniques de pointe et des procédés

efficaces pour être conforme à son exploitation ; il reste certain que dans un avenir proche, les évolutions techniques et scientifiques vont découdre de ces lacunes et permettre un niveau de reconnaissance des caractères arabes assez élevé.

Ouverture vers l'avenir :

Ces recherches et études, très intéressantes, réalisées pour ce mémoire nous laissent envisager des perspectives de travaux à accomplir et des domaines à explorer pour mieux appréhender le processus le plus adéquat, optimisé et rationnel pour manager cette reconnaissance des caractères arabes.

Nous affirmons que sommes aptes à développer d’avantage cette application par un apport qualitatif et quantitaf des bases de données. Il faut aussi s'accorder du temps pour permettre un bon apprentissage du réseau de neurones convolutionnel. Sans oublier un matériel informatique performant.

Bibliographie

[1]. Lawgali, Ahmed, et al. "A Survey on Arabic Character Recognition." International Journal of Computer Applications, Volume 100, No. 7, August 2014, pp. 1-7.

[2]. Daoud Fouad et Louli Farouk « La reconnaissance des caractères arabes manuscrits par les reseaux convolutifs » Thèse magistère en informatique, Université Blida 2007/2008

[3]. 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

[4]. 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.

[5]. Zerdani Amina « Reconnaissance de caractères arabes manuscrits par réseau de Hopfield » mastère en informatique, Université Oum el Bouaghi Juin 2014

[6]. Zermi Narimane « Reconnaissance de mots manuscrits arabe par les modèles de markov » thèse de doctorat Université Annaba Juin 2014

[7]. Haitaamar Schahrazed « SEGMENTATION DE TEXTES EN CARACTERES POUR LA RECONNAISSANCE OPTIQUE DE L'ECRITURE ARABE » Thèse magistère en informatique, Université Batna 08 Juillet 2007

[10]. Baka Abdeladim et Fillali Hichal « Traitement et reconnaissance des caractères »

[11]. Benkhessirate Walid et Khenniche Oussama « L'elaboration d'un OCR basé sur les modéles de markov cachés : application au texte arabe imprimé non voyelle »

[13]. Nemouchi Soulef « Reconnaissance de l'Ecriture Arabe par Systèmes Flous »

[14]. N. Ben Amara, "Utilisation des modèles de Markov cachés planaires en reconnaissance de l‟écriture arabe imprimée" These de doctorat, Specialite genie electrique, Universite des sciences des Techniques et de medecine de Tunis II, 1999.

[15]. B. Al-Badr, S.A. Mahmoud " Survey and bibliography of Arabic optical text recognition ". Signal processing, vol. 41, pp. 49-77,1995.

[16]. F. Grandidier « Un nouvel algorithme de sélection de caractéristiques -Application à la lecture automatique de l‟écriture manuscrite,» Thèse de Doctorat, Ecole de Technologie Supérieure, Université du Québec, Canada, 2003.

[17]. P.Burrow, «Arabic handwriting recognition,» Memoire Master, University of Edinburg, England, 2004.

[18]. R. Bouslimi « Système de reconnaissance hors-lignedes mots manuscrits arabe pour multi-scripteurs,» Universite de jendouba, Jendouba, 2006.

[19]. S. Bouzariata « Segmentation des textes manuscrits» Memoire master, Université de Tébessa, 2014 .

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

[21]. S. Nemouchi « Reconnaissance de l'ecriture arabe par systèmes flous,» Memoire magister, Universite Badji Mokhtar, Annaba, 2010.

[22]. S. Kermi, Classifieur neuronal base connaissances, application à la reconnaissance des caractères arabes isolés manuscrits, thèse de magister, Université d'Annaba, Algérie, 1999.

[23]. J.J. De Oliveira, J. De Carvalho, C.O. De A. Freitas, and R. Sabourin, "Evaluating NN and HMM classifiers for handwritten word recognition", Brazilian Symposium on Computer Graphics and Image Processing, pp.210-217, Fortaleza-Brazil, 2002.

[24]. I.R. Tsang, Pattern recognition and complex systems, Doctorat thesis, 2000.

[25]. L. Chergui, « Combinaison de classifieurs pour la reconnaissance de mots arabes manuscrits,» Université de Mentouri, Constantine, 2013.

[26]. R.M. Bozinovic et S.N.Srihari, «" Off-line cursive script word recognition ",» IEEE Transaction on Pattern Analysis and Machine Recognition PAMI, Vol 11, NO. 1, pp: 68-83, 1989.

[27]. M. M. P.Gader, ««Handwritten word recognition using segmentation-free hidden Markov modeling and segmentation-based dynamic programming techniques",» IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 18, No. 5, pp. 548-554, 1996.

[28]. L.Souici-Meslati, « Reconnaissance des mots arabes manuscrits par intégration neuro-symbo-lique,» Thèse de Doctorat d"Etat, Labo. LRI, Département d'informatique, Université d'Annaba Annaba, Algérie 2006.

[29]. T. Sari, L. Souici, M. Sellami, " Off-line handwritten Arabic character segmentation algorithm : ACSA ". IEEE Proc. 8thinternational workshop on frontiers in handwriting recognition (IWFHR"02), 2002.

[30]. Zoubeir Mouelhi, AraSeg : un segmenteur semi-automatique des textes arabes In JADT 2008: 9es Journées internationales d"Analyse statistique des Données Textuelles.

[31]. A. Hamid, R.Haraty, "A neuro-heuristic approach for segmenting handwritten Arabic text". IEEE. Proc. Of International conference on computer systems and applications (ACS), pp. 110-113, 2001.

[32]. S.T. Masmoudi, N.E. Benamara, H. Amiri, " Segmentation stage of a PHMM-based model for off-line recognition of Arabic handwritten city names", IEEE. International conference on systems, Man and cybernetics, SMS 2002, vol. 4, 6-9 October 2002.

[33]. S. Haykin, "Neural Networks - A Comprehensive Foundation", Macmillan College Publishing Company, New York, 1994.

[34]. Bernard Gosselin, "Application de réseaux de neurones artificiels à la reconnaissance automatique de caractères manuscrits", Thèse De Doctorat, Faculté Polytechnique de Mon, Année académique 95-96.

[35]. A. Nigrin, Networks for Pattern Recognition Cambridge: MA: The MIT Press, 1993

[36]. Kamel Mohamed, "Reconnaissance de formes appliquée à l‟écriture arabe manuscrite par des multiclassifieurs", Thèse De Magister En Informatique, Université Mohamed Khider, Biskra, 2010, p65

[37]. Rumelhart DE, Hinton GE, Williams RJ (1986) Learning representations by backpropagating errors. Nature 323 :533–536.

[38]. LeCun Y, Bengio Y, Hinton G (2015) Deep learning. Nature 521 :436–444

[39]. Schmidhuber J (2015) Deep Learning in Neural Networks: An Overview. Neural Networks 61 :85–117.

[40]. Goodfellow I, Bengio Y, Courville A (2016) Deep learning. MIT Press.

[43]. Bengio Y, Simard P, Frasconi P (1994) Learning long-term dependencies with gradient descent is difficult. IEEE Transactions on Neural Networks 5 :157–166.

[44]. Hochreiter S, Schmidhuber J (1997) Long Short-Term Memory. Neural Computation 9 :1735–1780.

[45]. Gers F, Schmidhuber J, Cummins FA (2000) Learning to Forget: Continual Prediction with LSTM. Neural Computation. https ://doi.org/10.1162/089976600300015015.

[46]. Wikimedia

[47]. Cho K, van Merrienboer B, Gulcehre C, Bahdanau D, Bougares F, Schwenk H, Bengio Y (2014) Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. arXiv :1406.1078.