République **A**lgérienne **D**émocratique et **P**opulaire
Ministère de l'**E**nseignement **S**upérieur et de la **R**echerche **S**cientifique

UNIVERSITE **M**OULOUD **M**AMMERI DE **T**IZI-**O**UZOU

FACULTE DU **G**ENIE **E**LECTRIQUE ET D'INFORMATIQUE
DEPARTEMENT D'INFORMATIQUE

Mémoire de Fin d'Etudes de MASTER ACADEMIQUE

Domaine : **Mathématiques et Informatique**
Filière : **Informatique**
Spécialité : **Systèmes Informatiques**

Présenté par
Celia ANÇA
Noria BOUSSAD

Thème

Reconnaissance de mots manuscrits en utilisant les réseaux de neurones Application : aux noms d'auteurs arabes

Mémoire soutenu publiquement le 17/ 07 / 2016 devant le jury composé de :

Président : Mr HABET Mohammed Said

Encadreur : Mr SOUALAH Mohammed Ourabah

Examinateur : M^{elle} Yesli Yassmine

Examinateur : Mr Kerbiche Mohand

REMERCIEMENTS

Nous tenons à remercier sincèrement notre promoteur Mr SOUALAH qui nous a proposé ce sujet, orienté, soutenu et conseillé tout au long de sa réalisation ;

Nos sincères remerciements vont aussi aux membres du jury pour avoir accepté d'évaluer notre travail.

Pour finir, nos dernières pensées vont vers nos familles, et surtout à nos parents, pour les sacrifices et les moyens qu'ils ont mis à notre disposition afin de nous permettre de suivre nos études dans les meilleures conditions ;

Noria ; Celia… ✍

DÉDICACES

Je dédie ce travail

A ma très chère maman qui m'a aidé et soutenu tout au long de mon parcours scolaire et universitaire, que dieu la garde pour moi.

A mes chers frères : IDIR et MEHDI,

A mes chères sœurs : KAHINA et VANESSA ,

A tous mes cousins et cousines,

A ma binôme Noria et sa famille,

A ma très chère voisine Nassima et son adorable petite Mayles que j'adore,

A mes copines: Mélina, Fariza, Wassila, Hanane, Dehbia, Rania,Djamila, Malika.

A tous ceux qui m'ont aidés est soutenus de près ou de loin,

A tous ceux qui m'aiment et que j'aime,

CELIA

DÉDICACES

Je dédie ce travail

A mes très chères parents qui m'ont aidés et soutenus tout au long de mon parcours scolaire et universitaire, que dieu les garde pour moi.

A mes chers frères : Rachid , Omar et Anes,

A ma sœur : Sarah,

A mon fiancé Idir, ma belle mère et mon beau père,

A toute ma famille,

A ma binôme Celia sa famille,

A mes copines: Yasmine, Djamila, Fazia, Wassila, Hanane, Mélina,

A tous mes amis,

A tous ceux qui m'aiment et que j'aime,

Noria

Table des matières

Partie III : Les réseaux de neurones artificiels

Chapitre II : Analyse et conception

Chapitre III : Réalisation

Liste des figures

Partie II : La reconnaissance des mots manuscrits

Partie III : Les réseaux de neurones artificiels

Chapitre II : Analyse et Conception

Chapitre III : Réalisation

Liste des tableaux

Partie II : La reconnaissance des mots manuscrits

Partie III : Les réseaux de neurones artificiels

Chapitre II : Analyse et Conception

Introduction générale

Introduction générale

L'écriture est l'un des grands fondements des civilisations, de conservation et de transmission du savoir.

La reconnaissance de l'écriture manuscrite ou imprimée reste encore un sujet de recherche et d'expérimentation, le problème n'est pas encore entièrement résolu bien que l'on sache atteindre des taux assez élevés dans certaines applications pour lesquelles soit le vocabulaire est limité, soit la fonte est unique ou en nombre restreint.

Il existe cependant plusieurs domaines pour lesquels la reconnaissance de l'écriture estappliquée avec un certain succès : le tri automatique du courrier, le traitement automatique de dossiers administratifs, des formulaires d'enquêtes, ou encore l'enregistrement des chèques bancaires. La reconnaissance de l'écriture manuscrite est beaucoup plus complexe que celle de l'écriture imprimée due à son extrême variabilité: variabilité des formes, des espacements entre mots et caractères, fluctuation des lignes.

Les travaux de recherche en reconnaissance de l'écriture arabe manuscrite, bien que moinsavancés que pour d'autres langues, deviennent de plus en plus intensifs.Mais les chercheurs sont confrontés à un problème difficile et incontournable, celui de la segmentation. La segmentation fait partie du processus de prétraitement et d'extraction de l'information. Cependant nous entamons notre travail en utilisant une segmentation manuelle, puis dans ce qui suivra nous présenterons les résultats de nos expérimentations sur la segmentation.

Parmi les outils les plus efficaces pour le traitement des problèmes de reconnaissances et classifications (dans notre cas il s'agit de reconnaitre des mots manuscrits arabes) : les réseaux de neurones artificiels qui sont inspirés du fonctionnement du système nerveux.

Ces réseaux ont la capacité de mémoriser la connaissance à partir d'exemples(dans notre cas ces exemples sont constitués de plusieurs caractères extraits à partir de versés coraniques anciens). Leur fonctionnement ressemble à celui d'un cerveau humain par le fait que la connaissance est acquise grâce à un processus d'apprentissage et que les poids des connexions inter-neuronales sont utilisés pour mémoriser cette connaissance sous une forme réduite.

Notre projet s'inscrit donc dans le cadre de reconnaissance de l'écriture manuscrite arabe par les réseaux de neurones

Pour mener à terme notre travail, nous avons adopté la structure suivante:

Chapitre I : Etat de l'art.

Ce chapitre est subdivisé en trois parties, dans la première partie nous présenterons quelques généralités sur le traitement d'images et l'intelligence artificielle, dans la deuxième partie parlerons de l'approche de la reconnaissance de mots, et enfin dans la troisième partie nous présenterons les réseaux de neurones artificiels.

Chapitre II : Analyse et conception.

Dans ce chapitre, nous présenterons une analyse de notre système, puis nous le modélisons en concevant les différents diagrammes UML, et enfin nous présenterons une description détaillée de notre système.

Chapitre III :Réalisation.

Ce chapitre traitera la mise en œuvre de notre système.

Et enfin nous terminons par une conclusion générale et une annexe portant un aperçu sur les outils, et algorithmes utilisés dans quelques parties de notre application.

CHAPITRE I :Etat de L'art

Etat de l’art

Partie I : Généralités sur les traitements d’images

I.1. Image

L'image est une Représentation d'un objet par les arts graphiques ou plastiques : la sculpture, la photographie, le dessin, le film, etc. C'est aussi un ensemble structuré d'informations qui, après affichage sur l'écran, ont une signification pour l’œil humain. **[1]**

I.2. Pixel [3]

Une image numérique est constituée d'un ensemble de points appelés **pixels** (abréviation de **PIC**ture**El**ement) pour former une image. Le pixel représente ainsi le plus petit élément constitutif d'une image numérique. L'ensemble de ces pixels est contenu dans un tableau à deux dimensions constituant l'image.

I.3. L’image numérique [1]

Contrairement aux images obtenues à l'aide d'un appareil photo ou dessinées sur du papier, les images manipulées par un ordinateur sont numériques (représentées par une série de bits). L'image numérique est l'image dont la surface est divisée en éléments de tailles fixes appelés cellules ou pixels, ou calculé à partir d'une description interne de la scène à représenter.

I.4. Histogramme

Un histogramme est un graphique statistique permettant de représenter la distribution des intensités des pixels d’une image. Il fournit diverses informations comme les statistiques d'ordre (moyenne, variance), et peut permettre d'isoler des objets. **[2]**
L'histogramme est donc une fonction qui donne la fréquence d'apparition de chaque niveau de gris (couleur) dans l'image. **[1]**

I.5. Contours et textures [1]

Les contours représentent la frontière entre les objets de l'image, ou la limite entre deux pixels dont les niveaux de gris représentent une différence significative. Les textures décrivent la structure de ceux-ci. L'extraction de contour consiste à identifier dans l'image les points qui séparent deux textures différentes.

I.6. Images à niveaux de gris [5]

Le codage dit en niveaux de gris permet d’obtenir plus de nuances que le simple noir et blanc. Il offre des possibilités supplémentaires pour coder le niveau de l’intensité lumineuse. La couleur est codée souvent sur un octet soit 8 bits ce qui offre la possibilité d’obtenir 256 niveau de gris (0 pour le noir et 255 pour le blanc). On peut aussi le faire avec 16 niveaux de gris (4 bits).

Plus le niveau de gris est élevé, meilleur est la distinction des détails sur l’image. L’usage de ce codage est utilisé fréquemment pour la presse écrite ou l’envoi par messagerie électronique de fichier d’image de taille réduite avec une perte de lisibilité de l’image moindre.

I.7. Images en couleurs [2]

Chaque pixel possède une couleur décrite par la quantité de rouge (R), vert (G) et bleu (B). Chacune de ces trois composantes est codée sur l’intervalle [0, 255], ce qui donne 255^3 = 16 777 216 couleurs possibles. Il faut 24 bits pour coder un pixel.

I.8. Le traitement d’image

I.8.1. Définition [7]

Le traitement d'images est une discipline de l'informatique et des mathématiques appliquées qui étudie les images numériques et leurs transformations, dans le but d'améliorer leur qualité ou d'en extraire de l'information.
Il s'agit d'un sous-ensemble du traitement du signal dédié aux images et aux données dérivées comme la vidéo (par opposition aux parties du traitement du signal consacrées à d'autres types de données : son et autres signaux monodimensionnels notamment), tout en opérant dans le domaine numérique (par opposition aux techniques analogiques de traitement du signal, comme la photographie ou la télévision traditionnelles).

Dans le contexte de la vision artificielle, le traitement d'images se place après les étapes d'acquisition et de numérisation, assurant les transformations d'images et la partie de calcul permettant d'aller vers une interprétation des images traitées. Cette phase d'interprétation est d'ailleurs de plus en plus intégrée dans le traitement d'images, en faisant appel notamment à l'intelligence artificielle pour manipuler des connaissances, principalement sur les informations dont on dispose à propos de ce que représentent les images traitées (connaissance du « domaine »).

I.8.2. Quelques techniques fondamentales en traitement d’images [2]

Une panoplie de traitements peut être appliquée à l’image numérique, dans ce qui suit nous donnant quelques exemples de ces traitements.

I.8.2.1. Acquisition de l’image [2]

L’acquisition d'images est une mesure spatiale d'une interaction entre une onde et de la matière. L'onde est émise par une source et reçu par un capteur.
La matière occupe de l'espace et possède une masse.
Elle a pour objet de passer de la scène physique à une forme numérique observée.

I.8.2.2. La segmentation [2]

C’est une procédure permettant de partitionner l’image en ses constituants ou objets. La segmentation automatique est la tâche la plus difficile en traitement d’images. Plus la segmentation est meilleur plus l’étape de reconnaissance d’objets est réussite.

I.8.2.3. La reconnaissance [2] [1]

Est le traitement qui affecte une étiquette (exemple : route, voiture) à un objet en se basant sur ses descripteurs.
La base de connaissance contient la connaissance du domaine du problème en cours du traitement. Dans son aspect le plus simple, elle peut consister en coordonnées de l’objet à traiter, ceci permet de réduire l’espace de recherche. Comme elle peut être complexe contenant toutes les défaillances que peut présenter un produit manufacturé.

I.9. Conclusion

Dans cette première partie du premier chapitre, nous avons abordé quelques notions de base surl’image et le traitement d’image.
Nous avons donné un petit aperçu sur le concept de l’intelligence artificielle, nous avons parlé de ses domaines de recherche ainsi que de quelques-unes de ses facettes.

Partie II : La reconnaissance des mots manuscrits

II.1. Introduction

Dans les dix dernières années, des progrès considérables ont été réalisés dans le domaine de la reconnaissance de l’écriture manuscrite. Ce progrès est dû d’une part aux nombreux travaux effectués dans ce domaine et d’autre part à la disponibilité de bases de données internationales standards relatives à l’écriture manuscrite qui permettait aux chercheurs de rapporter de façon crédible les performances de leurs approches dans ce domaine, avec la possibilité de les comparer avec d’autres approches, vu qu’ils utilisent les mêmes bases. La langue arabe n’a pas eu cette chance, contrairement au latin, elle reste encore au niveau de la recherche et de l’expérimentation **[11],** c'est-à-dire que le problème reste encore un pari ouvert pour les chercheurs. L’écriture arabe étant par nature cursive, elle pose de nombreux problèmes aux systèmes de reconnaissance automatique Le problème le plus difficile lors de la conception d’un système de reconnaissance de l’écriture manuscrite est la segmentation des mots manuscrits en vue de leur reconnaissance, qui n’est pas toujours triviale et demande beaucoup de temps et de calcul**. [12]**

L'objectif de cette deuxième partie du premier chapitre consiste à introduire et de présenter un état de l'art du domaine de la reconnaissance de mot, ce qui nous permettra de situer le problème d'OCR.

II.2. Définition de la reconnaissance de caractères [13]

La reconnaissance optique de caractères (OCR) est une opération informatique rapide permettant de réaliser la transformation d’un texte écrit sur papier en un texte sous forme d’un fichier informatique en représentation symbolique (par exemple pour les écritures latines, le codage opéré est le code ASCII (American Standard code forinformation interchange), tandis que pour l’arabe on utilise généralement le code ASMO (Arabic Standard MetrologyOrganization).

II.3. Caractéristiques morphologiques de l'écriture Arabe [14]

L'écriture arabe est semi-cursive dans sa forme imprimée ainsi que manuscrite. Les caractères d'une même chaîne (ou pseudo-mots) sont ligaturés horizontalement et parfois verticalement (dans certaines fontes deux, trois et même quatre caractères peuvent être ligaturés verticalement), occultant ainsi toute tentative de segmentation en caractères. De plus, la forme d'un caractère diffère selon sa position dans les pseudo-mots et même dans certains cas, selon le contexte phonétique. En outre, plus de la moitié des caractères arabes incluent dans leur forme des points diacritiques. Ces points peuvent se situer au-dessus ou au-dessous du caractère, mais jamais en haut et en bas simultanément. Plusieurs caractères peuvent avoir le même corps mais un nombre et /ou une position de points diacritiques différents. D'autre part, le caractère arabe présente une forme cursive voyellée nécessitant, pour la majorité des lettres, des matrices de dimensions importantes. Ceci laisse jusqu'à présent les formes informatisées des caractères arabes non encore normalisées.

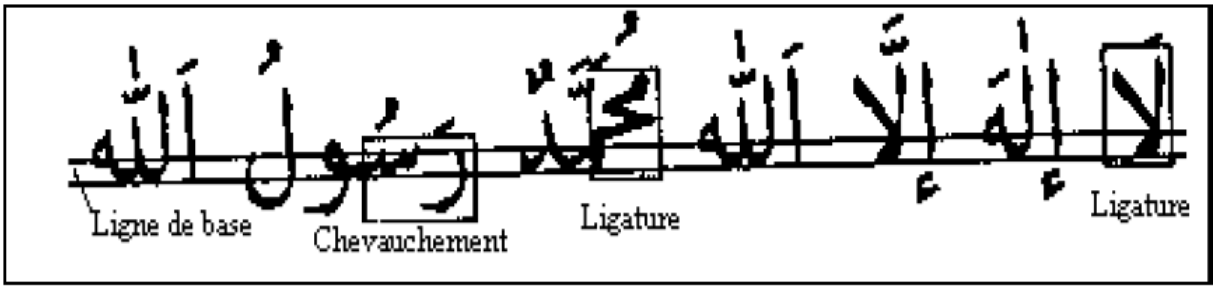


Figure II.1 : un échantillon de l'écriture arabe illustrant certaines de ces caractéristiques **[15]**

Des points nécessaires pour différencier les lettres

Dans l'alphabet arabe, 15 lettres parmi les 28 possèdent un ou plusieurs points. Ces signes diacritiques sont situés soit au-dessus, soit en dessous de la forme à laquelle ils sont associés, mais jamais les deux à la fois.

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

Tableau II.1 : L'alphabet arabe.

II.4. Différents aspects de la Reconnaissance Automatique de l'Ecriture [16]

Devant la difficulté d'établir un système universel traitant tous les cas d'écriture, la solution à court terme consiste à particulariser le cadre puis à construire l'approche la plus adaptée pour ce cadre. On peut répartir les systèmes de reconnaissance de l'écriture dans plusieurs catégories différentes : mode d'acquisition « En-ligne » ou « Hors-ligne », Approche de reconnaissance « Globale » ou « Analytique » et enfin Méthode de reconnaissance « Statistiques » ou « Structurelle ».

II.4.1. Mode d'acquisition (En-ligne / Hors-ligne) [16]

Il est d'usage de distinguer deux activités en RAED, les reconnaissances en-ligneet hors-ligne. Dans le cas de la REM, il existe deux champs d'applications suivant le mode de saisie. Si le mode de saisie est dynamique, on parle de reconnaissance en temps réel appelée en-ligneou on-line. Dans le cas de la saisie statique on parle alors de reconnaissance en temps différé appelée hors-ligne ou off-line.

II.4.1.1. Systèmes de reconnaissance en ligne [17]

Dans ce type de systèmes, la reconnaissance est effectuée en temps réel, c'est-à-dire elle est effectuée au fur et à mesure que le caractère est tracé, ce qui permet d'obtenir une large marge de correction et modification selon la réponse donnée à la phase de reconnaissance chevauchée à la phase d'acquisition.

Ce mode d'acquisition est réservé généralement à l'écriture manuscrite. C'est une approche « signal » où la reconnaissance est effectuée sur des données à une dimension.

L'écriture est représentée comme un ensemble de points dont les cordonnées sont fonction du temps. Où on s'intéresse aux méthodes et techniques de traitement du message tel qu'il est écrit, en prenant en compte les informations relatives au mécanisme d'écriture telles la position des points, la vitesse et l'accélération qui sont des fonctions du temps.

Les moyens de saisie en ligne sont nombreux où la tablette graphique avec un stylo électronique et l'écran tactile sont couramment utilisés.

Parmi les plus récentes plateformes disposant d'un système de reconnaissance de l'écriture, nous trouvons le Palm et l'agenda électronique. Ces deux appareils regroupent une tablette à numériser et un programme procédant à la reconnaissance de l'écriture. De ce fait, son utilisation est plus attrayante puisqu'elle épargne à l'utilisateur le besoin de « scanner » a priori son écriture. Ceci a remis la reconnaissance en ligne au centre d'intenses efforts de développement au sein de la communauté de l'écrit et du document.

(a)Écran tactile ou tablette PC, ipad | *(b) Smartphone, iphone, Assistant personnel* | *(c)Stylo doppler, stylo camera*

Figure II.2 : Exemples des systèmes en-ligne **[18]**

II.4.1.2. Systèmes de reconnaissance hors ligne [17]

L'écriture hors-ligne (ou en différé, u encore statique) est obtenue par la saisie d'un texte déjà existant, obtenue par un scanner ou une caméra. Dans ce cas, on dispose d'une image binaire ou en niveaux de gris, ayant perdu toute information temporelle sur l'ordre des points. De plus, ce mode introduit une difficulté supplémentaire relative à la variabilité du tracé en épaisseur et en connectivité, nécessitant l'application de techniques deprétraitement.

Les domaines d'application les plus typiques sont principalement associés au traitement automatique des adresses postales, du montant des chèques, des formulaires, des feuilles de soins…

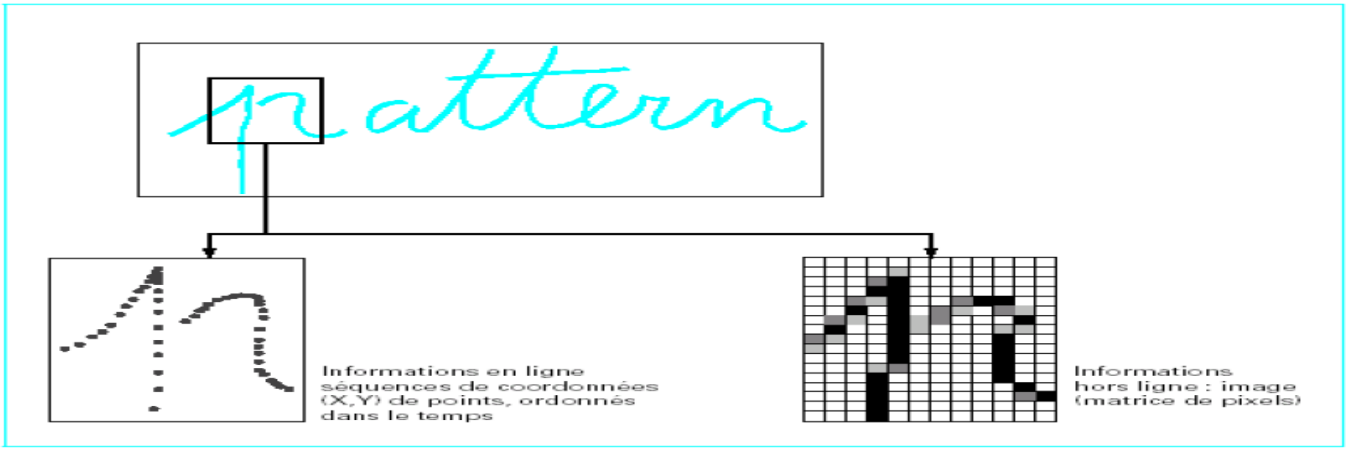


Figure II.3: Écriture en ligne et hors ligne **[17]**

II.4.2. Approches de reconnaissance [19]

Durant les dernières décennies, beaucoup de méthodes de segmentation ont été développées dans le but d'avoir un système de reconnaissance de caractères plus robuste. Malgré tous les efforts, la situation reste loin d'atteindre les ambitions. En se basant sur le processus de segmentation, deux approches ont été appliquées :

- Approche globale.
- Approche analytique.

II.4.2.1. L'approche globale ou holistique

L'approche globale essaye de reconnaître la représentation intégrale des mots de l'image d'entée et de le décrire indépendamment des caractères qui le constituent. Cette approche présente l'avantage de garder le caractère dans son contexte avoisinant, ce qui permet une modélisation plus efficace des variations de l'écriture et des dégradations qu'elle peut subir **[22].** Cependant cette méthode est pénalisante par la taille mémoire, le temps de calcul et la complexité du traitement qui croit linéairement avec la taille du lexique considéré, d'où une limitation du vocabulaire **[49].**

II.4.2.2. L'approche analytique ou locale [39].

Contrairement à l’approche globale, l’approche analytique cherche à identifier les caractères ou sous-caractères (graphèmes) issus de la segmentation (séparation de mots, des caractères) pour reconstituer les mots.
La difficulté d'une telle approche a été clairement évoquée par Sayre en 1973 et peut être résumée par le dilemme suivant : "pour reconnaître les lettres, il faut segmenter le tracéet pour segmenter le tracé, il faut reconnaître les lettres". Cette approche est la seule applicable dans le cas de grands vocabulaires. Elle peut s'adapter facilement à un changement de vocabulaire. Elle permet théoriquement une discrimination plus fine des mots car elle se base sur la reconnaissance des lettres qui la composent et il est possible de récupérer l'orthographe du mot reconnu. Son inconvénient principal demeure la nécessité de l'étape de segmentation avec les problèmes de sous- ou de sur-segmentation que cela implique.

Certaines des approches actuelles se proposent de tirer avantage des deux méthodes, réduisant la complexité de l'approche globale en l'appliquant sur des entités plus petites (lettres). L'approche analytique recherche la séquence de lettres contenues dans l'image à reconnaître. Certains modèles permettent de combiner ces deux niveaux en un seul et peuvent ainsi s'affranchir de la segmentation préalable de l'image.

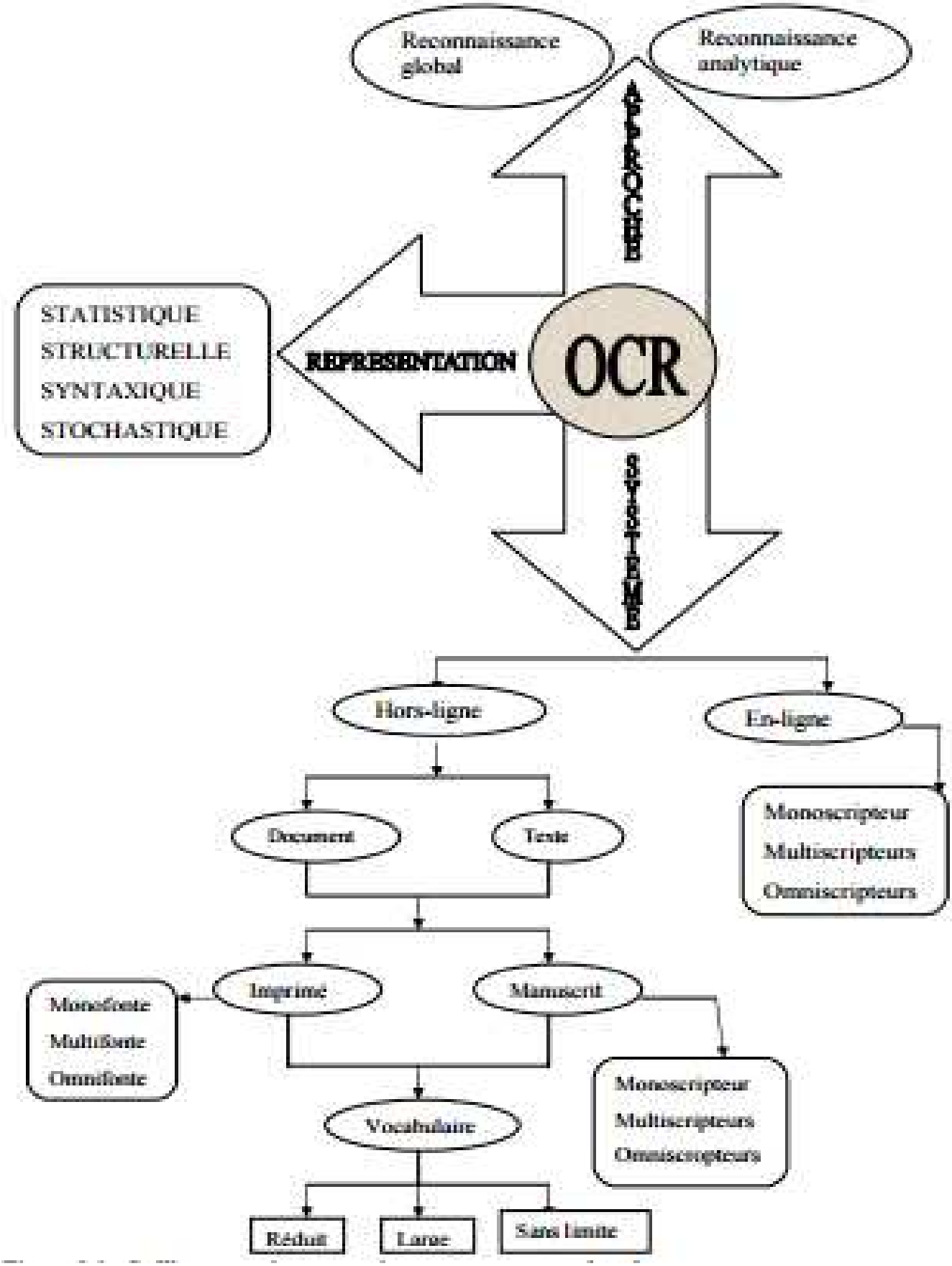


Figure II.4: Différents systèmes, représentations et approches de reconnaissance.

II.5. Organisation générale d'un système de reconnaissance [24]

La reconnaissance de l'écrite manuscrite s'intéresse à identifier correctement l'entrée d'une image du texte écrit sur papier scannée ou photographié, en la convertissant en un texte sous forme d'un fichier informatique en format d'édition telle HTML ou Latex. **[24]** Typiquement, quelque soit le système de reconnaissance du manuscrit, il fait appel des phases suivantes :**[25]**

- Acquisition (scanning, Numérisation),
- Prétraitement,
- Segmentation à des caractères séparés ou segments reliés à un caractère,
- Extraction des caractéristiques,
- Classification, suivis éventuellement d'une phase de post-traitement.

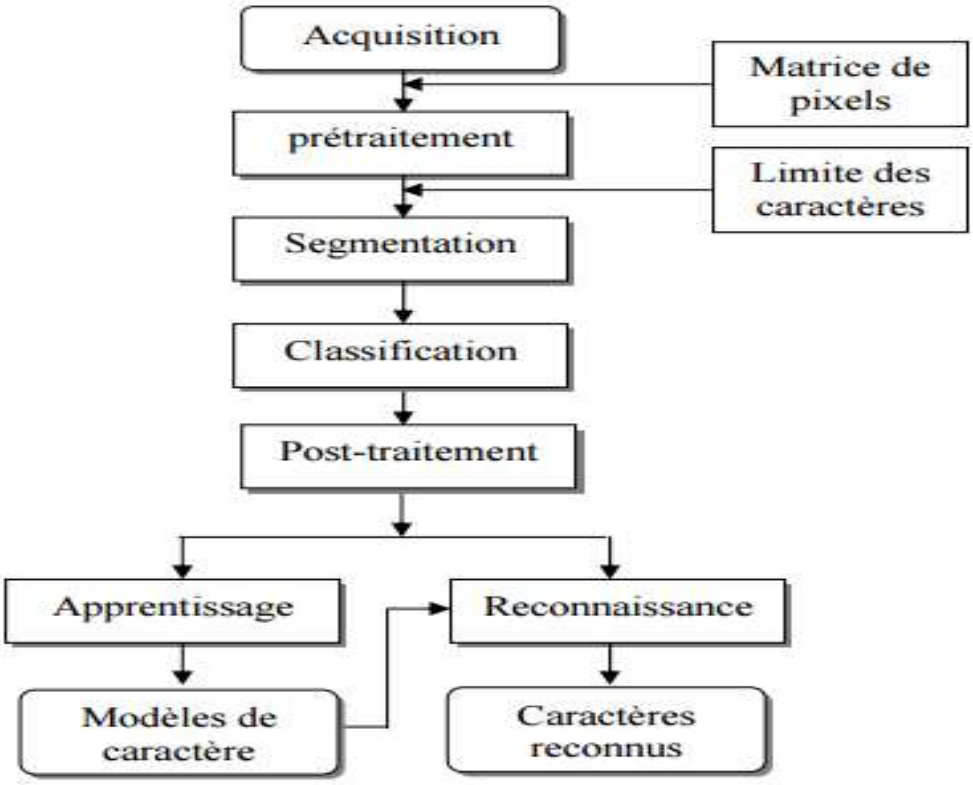


Figure II.5 : Schéma général d'un système de reconnaissance de caractères.

II.5.1. Phase d'acquisition [16]

L'opération d'acquisition constitue l'interface entre le système et le monde réel. Cette étape consiste à capter l'image d'un texte au moyen des capteurs physiques (scanner, caméra) et de la convertir en grandeurs numériques adaptés au système de traitement, avec un minimum de dégradation possible. La numérisation consiste à **[24]**

- Eliminer les bruits.
- Bouchage des trous (gap filling).
- Translation d'échelle.

- Normalisation.
- Binairisation.

Cette étape est assez simple mais très importante car elle influence sérieusement les étapes suivantes **[16]**

Il y a deux paramètres important :

- **Résolution :**la résolution normale est 300 dpi. Pourtant, quand la taille de l'écriture est petite, il faut augmenter la résolution. La résolution du scanner désigne sa capacité à digitaliser les traits fins.
- **Niveau d'éclairage** : si on ajuste le scanneur pour que l'image soit plus claire, le bruit est réduit mais des traits minces disparaissent aussi.

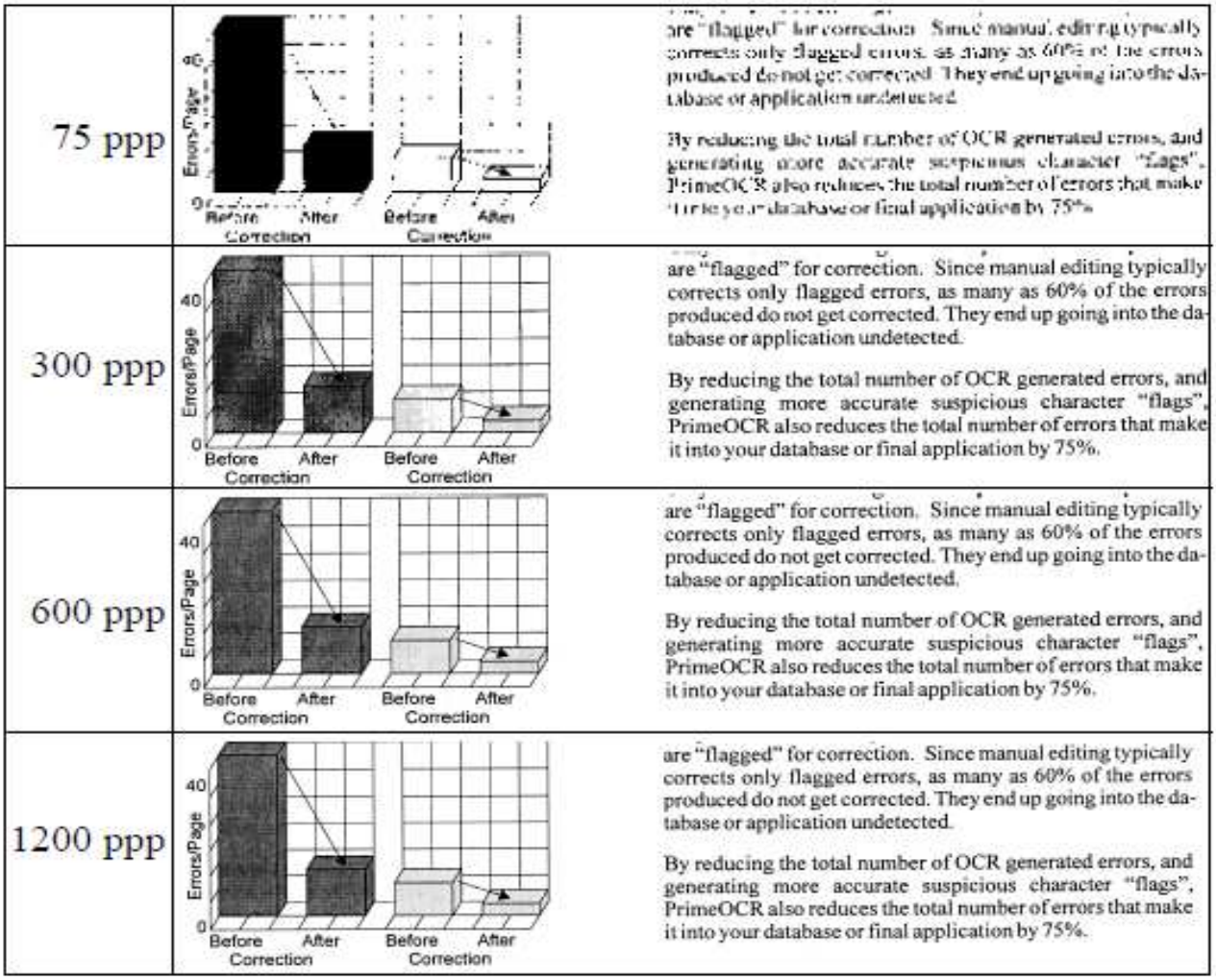


Figure II 6 : Différents niveaux de résolution**[26].**

II.5.2. Phase de prétraitements

Le prétraitement est le processus qui couvre de la forme originale de l'image d'entrée à une forme squelettique ce qui permet de simplifier la phase d'extraction de caractéristiques **[24]**. Il s'agit essentiellement de réduire le bruit superposé aux données et essayer de ne garder que l'information significative de la forme représentée. Le bruit peut être dû aux conditions

d'acquisition (éclairage, mise incorrecte du document,…) ou encore à la qualité du document d'origine**[27] [28].**

Parmi les opérations de prétraitement généralement utilisées nous pouvons citer :

- L'extraction des composantes connexes,
- Le redressement de l'écriture,
- Le lissage,
- La normalisation, et la squelettisation.

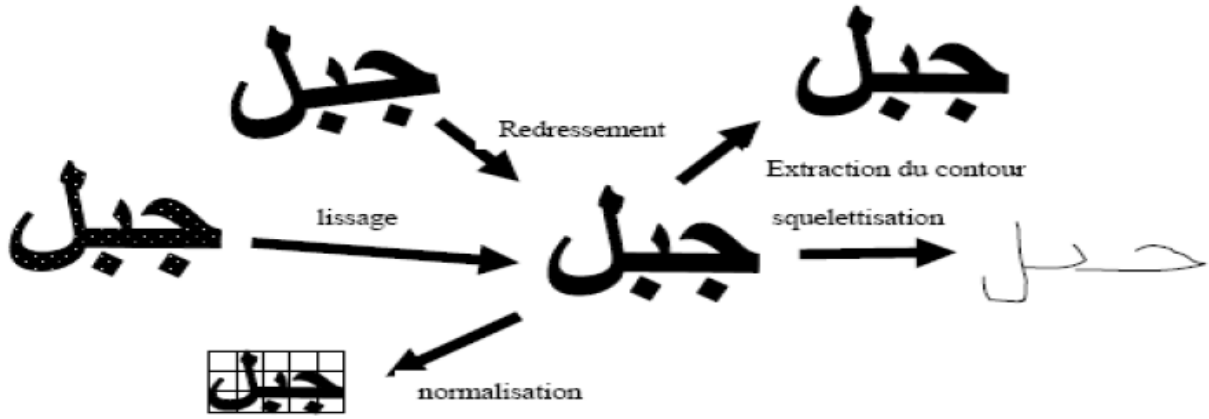


Figure II.7: Effet de certaines opérations de prétraitement **[22].**

II.5.2.1. Extraction de composantes connexes

L'écriture arabe est cursive et les mots sont séparés par des espaces. Toutefois, un mot peut contenir plusieurs pseudo mots (PAW) qui sont une portion du mot incluant une ou plusieurs lettres liées. Une segmentation typique est basée sur une analyse d'histogramme de projection, et le regroupement des composantes connexes **[21].**

Une composante connexe (CXX) est un ensemble de points dans le plan. Elle peut correspondre à un point diacritique, un accent, au corps d'un caractère ou d'une chaîne de caractères… Une fois localisés les CXX sont regroupées pour former les mots. Cette technique est utilisée pour le repérage des points diacritiques dans les images de textes arabes **[22]**.

II.5.2.2. Redressement de l'écriture

L'idée est de rendre horizontaux les mots à l'aide d'une transformation géométrique de type rotation isométrique (angle – α si α est l'angle d'inclinaison) des points de l'image du mot **[21]**.

L'inclinaison peut provenir de la saisie, si le document a été placé en biais, ou être intrinsèque au texte **[22]** :

$$X' = x \cos \alpha + y \sin \alpha$$
$$Y' = y \cos \alpha + x \sin \alpha$$

II.5.2.3. Lissage de l'écriture

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 **[29]**.

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.

Plusieurs autres techniques similaires sont utilisées dont la méthode statistique, une méthode basée sur la morphologie mathématique … (pour plus de détail sur ces technique le lecteur peut se référer à) **[30].**

II.5.2.4. Normalisation

La normalisation peut être indispensable pour certains types de systèmes (comme les réseaux de neurones), Elle permet de ramener les images de mots à des tailles standard. La différentielle pousse le principe de normalisation à un degré plus fin en essayant de normaliser localement différentes parties du mot, de manière à augmenter la similarité d'une image à une autre **[21] [27]**

Les parasites, les hampes et les jambages provoquent des décalages verticaux des mots qui désynchronisent la présence des informations, par exemple : la hampe des caractères [ا،ظ،ل] peut être éliminée **[22].**

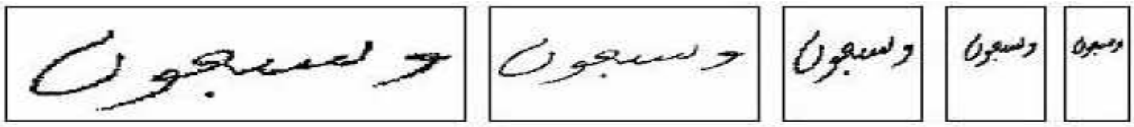


Figure II.8: Exemples de mots manuscrits avec des tailles différentes.

II.5.2.5. Squelettisation

Le but de cette technique est de simplifier l'image du caractère en une image à « ligne » plus facile à traiter en la réduisant au tracé du caractère. Les algorithmes de squelettisation se basent sur des méthodes itératives. Le processus s'effectue par passes successives pour déterminer si un tel ou tel pixel est essentiel pour le garder ou non dans le tracé **[31]**

La squelettisation des caractères arabes peut induire en erreur : deux points diacritiques sont souvent confondus avec un seul **[30]**

Remarque

Selon la qualité du document à traiter, le type de l'écriture et la méthode d'analyse adoptée, une ou plusieurs techniques de prétraitement sont utilisées. Mais pas forcément toutes.

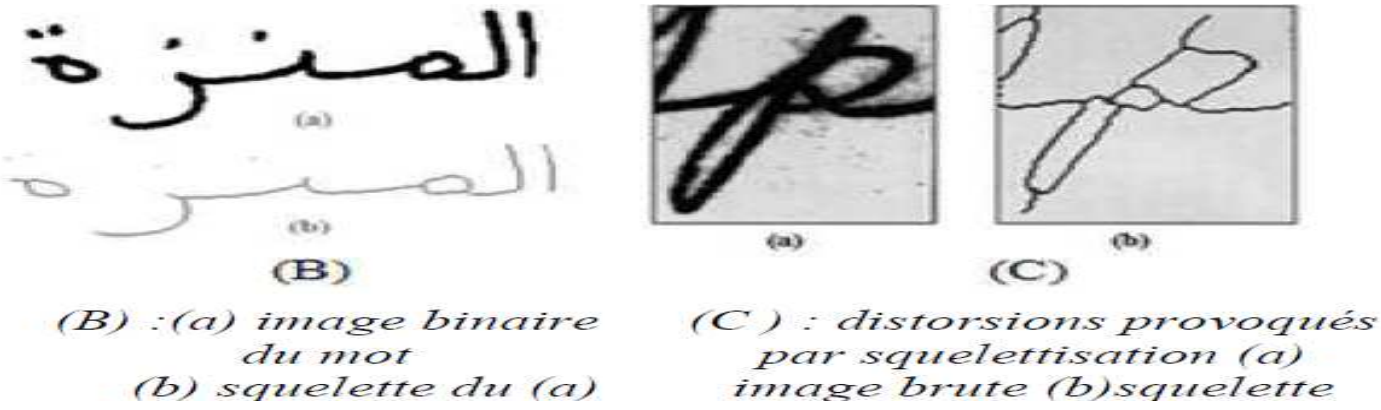


Figure II.9: Exemples de squelettisation **[32]**

II.5.3. Phase de segmentation

Dans cette phase les différentes parties logiques d'une image sont extraites. A partir d'une image acquise il y'a d'abord séparation des blocs de texte et des blocs graphiques, puis à partir d'un bloc de texte il y'a extraction des lignes, ensuite à partir de ces lignes sont extraits les mots puis les caractères (ou parties du caractère) **[33]**

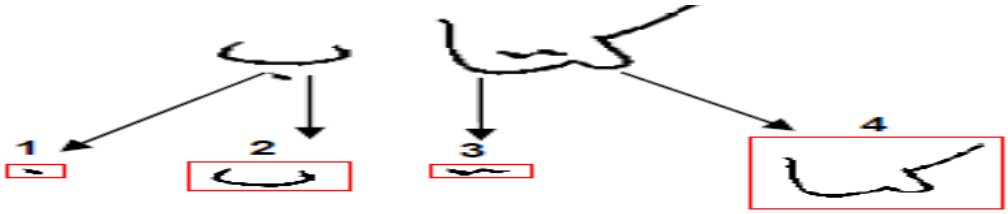


Figure II.10: Exemple de segmentation de l'image du mot **"كتاب " [13]**

Il existe deux techniques permettant la mise en œuvre de la segmentation : segmentations explicite et implicite.

- **Segmentation explicite** : segmentation sur des critères topologiques, elle consiste à utiliser des points caractéristiques dans le mot **[34]**, tels que les minima 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**.**
- **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 **[35].** 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'ya 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 de graphèmes qui correspondent à ses classes de lettres **[35].**

- **Sans segmentation** : détection de la présence d'une lettre, certains systèmes se basent sur la forme globale des mots. Elles s'avèrent cependant indispensable lorsque la taille du vocabulaire est importante **[16].**

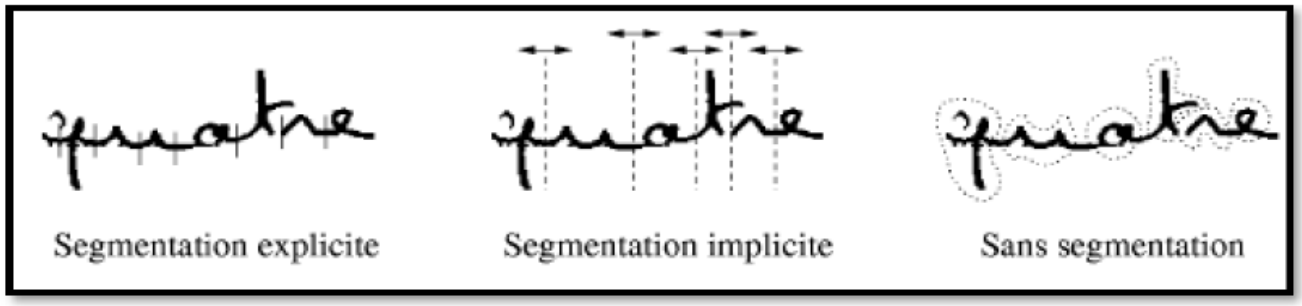


Figure II.11 : Les différents types de segmentation **[35].**

II.5.4. Phase d'analyse ou d'extraction des caractéristiques

C'est l'une des étapes les plus délicates et les plus importantes en OCR. La reconnaissance d'un caractère passe d'abord par l'analyse de sa forme et l'extraction de ses traits caractéristiques (primitives) qui seront exploités pour son identification.

II.5.5. Phase de Classification

Le choix du classifieur est très important. Il constitue l'élément de décision dans un système de RdF. Le but des traitements antérieurs était d'extraire l'information essentielle caractérisant les objets et de la présenter sous la forme la plus condensée possible au classifieur.

La classification dans un SREM hors-ligne consiste à déterminer la classe d'appartenance de l'objet en entrée, elle regroupe deux tâches : l'apprentissage et la reconnaissance et décision. A cette étape les caractéristiques de l'étape précédente sont utilisées pour identifier un segment de texte et l'attribuer à un modèle de référence **[37].**

II.5.5.1. Apprentissage [16]

Cette étape permet de construire un dictionnaire de prototype. Il s'agit de regrouper en classes plusieurs prototypes dont les caractéristiques se rapprochent. Il existe 2 types d'apprentissages : supervisé et non supervisé.

1) Apprentissage supervisé [16]

L'apprentissage est dit supervisé si les différentes familles des formes sont connues à priori et si la tâche d'apprentissage est guidée par un superviseur ou professeur. On choisit un sous-ensemble de formes, chacune est analysée puis le professeur indique la classe dans laquelle il souhaite la voir rangée. L'apprentissage consiste alors à analyser les ressemblances entre formes de la même famille et les dissemblances entre formes de familles différentes pour en déduire les classes avec les meilleures séparatrices possibles. Les paramètres décrivant cette partition sont stockés dans une base d'apprentissage.

2) Apprentissage non supervisé [16]

Il s'agit de construire automatiquement les classes, sans intervention de professeur, à partir d'échantillons de référence et de règles de regroupement. Ce mode nécessite un nombre élevé d'échantillons et des règles de construction précises et non contradictoire, mais n'assure pas toujours une classification correspondant à la réalité de l'utilisateur.
En fait le rôle du module d'apprentissage consiste à ajuster ses paramètres afin de donner une réponse lors de la phase de décision.

II.5.5.2. Reconnaissance et décision

La décision est l'ultime étape de reconnaissance. A partir de la description en paramètres du caractère traité, le module de reconnaissance cherche parmi les modèle de référence en présence, ceux qui lui sont les plus proches.
La reconnaissance peut conduire à un succès si la réponse est unique (un seul modèle répond à la description de la forme du caractère). Elle peut conduire une confusion si la réponse multiple (plusieurs modèles correspondent à la description). Enfin elle peut conduire à un rejet de la forme si aucun modèle ne correspond à sa description. Dans les deux premiers cas, la décision peut être accompagnée d'une mesure de vraisemblance, appelée aussi score ou taux de reconnaissance**[30].**
Voici quelques approches de reconnaissance :

- **L'approche bayésienne**

L'approche bayésienne consiste à choisir parmi un ensemble de caractères, celui pour lequel la suite de primitives extraites a la plus forte probabilité à posteriori par rapport aux caractères préalablement appris**[40].**

- **Les réseaux de neurones**

Un réseau de neurones est un graphe orienté pondéré. Les nœuds de ce graphe sont des automates simples appelés neurones formels. Les neurones sont dotés d'un état interne, l'activation, par lequel ils influencent les autres neurones du réseau. Cette activité se propage dans le graphe le long d'arcs pondérés appelés liens synaptiques **[42].**
En OCR, les primitives extraites sur une image d'un caractère (ou de l'entité choisie) constituent les entrées du réseau. La sortie activée du réseau correspond au caractère reconnu. Le choix de l'architecture du réseau est un compromis entre la complexité des calculs et le taux de reconnaissance **[43].**
Par ailleurs, le point fort des réseaux de neurones réside dans leur capacité de générer une région de décision de forme quelconque, requise par un algorithme de classification, au prix de l'intégration de couches de cellules supplémentaires dans le réseau **[44].**

- **Approche stochastique**

Contrairement aux méthodes précédemment décrites, l'approche stochastique utilise un modèle pour la reconnaissance, prenant en compte la grande variabilité de la forme. La distance communément utilisée dans les techniques de « comparaison dynamique » est remplacée par des probabilités calculées de manière plus fine par apprentissage. La forme est considérée comme un signal continu observable dans le temps à différents endroits constituant des états « d'observations ». Le modèle décrit ces états à l'aide de probabilités de transitions d'états et de probabilités d'observation par état. La comparaison consiste à chercher dans ce graphe d'états, le chemin de probabilité forte correspondant à une suite d'éléments observés

dans la chaîne d’entrée **[30].** Ces méthodes sont robustes et fiables du fait de l’existence d’algorithmes d’apprentissage efficaces **[46].** Si l’apprentissage est lent, la reconnaissance est par contre très rapide car les modèles comprennent généralement peu d’états et le calcul est relativement immédiat. Les méthodes les plus répondues dans cette, approche sont les méthodes utilisant les modèles de Markov cachés (HMM).

- **Approche hybride**

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

II.5.6. Phase de post-traitement

Cette étape aide à réduire considérablement des erreurs. Cependant, ce n'est pas une étape complètement séparée des étapes précédentes. Comme le processus de reconnaissance l'écriture de l'humain, l'étape de post-traitement est intégrée strictement en ces étapes.
Le post-traitement comprend la vérification, l'exécution de l'action et l'adaptation. Cette étape peut être rajouté à un système de reconnaissance de l’écriture à pour but d’améliorer le taux de la reconnaissance, en introduisant des informations contextuelles permettant de lever l’ambiguïté dans la reconnaissance de certains mots ou caractères, parmi ces informations en citant :

- Les connaissances pragmatiques sur la longueur moyenne de chacune des lettres, ou sur le nombre de lettres constituant un mot.
- Les algorithmes de correction orthographiques ou morphologiques à l'aide de dictionnaires de digrammes, trigrammes ou n-grammes.
- Les connaissances linguistiques quand il s'agit de la reconnaissance de phrases entières, on fait intervenir des contraintes de niveaux successifs : lexical, syntaxique ou sémantique.
- Lexical : pour valider la reconnaissance effectuée en ne retenant que des mots du dictionnaire, et en rejetant les listes de lettres inconsistantes.
- Syntaxique et sémantique : pour réduire la liste des mots candidates et valider ceux qui ont été retenus à l’étape précédente **[47] [48].**

Il peut être aussi envisagé si la reconnaissance ne donne pas les résultats escomptés, et sachant que le problème n’est en fait pas résultant de la reconnaissance elle-même mais d’un autre module, de l’acquisition, du prétraitement ou même de l’extraction des caractéristiques, ou ça peut bien être engendré par une base d’apprentissage pas assez complète**[16].**

II.6. Conclusion

Dans cette deuxième partie du premier chapitre, nous avons vus quelques Caractéristiques morphologiques de l'écriture Arabe, Différents aspects de la Reconnaissance Automatique de l’Ecriture, Approches de reconnaissance. En plus, les principaux problèmes rencontrés dans ce domaine. Ensuite nous avons abordé les différentes étapes intervenant dans la conception d'un système de reconnaissance de caractères.

Partie III : Les réseaux de neurones artificiels

III.1. Introduction et historique

Le cerveau humain est un appareil de calcul très complexe qui reste inaccessible.
Sa grande capacité de penser, de mémoriser et de résoudre les problèmes a inspiré les scientifiques, qui ont essayé de modéliser son fonctionnement durant ces opérations.
C'est ainsi, qu'un groupe de chercheurs a créé un modèle de calcul, qui tend à imiter le cerveau humain. Ce qui a introduit l'étude du calcul neuronal (réseaux de neurones).
La première période a commencé il ya une dizaine d'années avec William James, et s'est terminé par la publication du perceptron et c'est ainsi que les progrès ont commencé.
En 1943, Mac-Culloch et Pitts ont constitué le premier modèle simplifié du neurone biologique **[50],** appelé aussi neurone formel.
En 1949, le psychologue Donald O. Hebb a introduit le terme de connexionnisme, pour parler des modèles massivement parallèles et connectés. Il a été le premier à définir la règle de mise à jour des poids synaptiques. Connue sous le nom "règle de Hebb " **[50]**.
En 1958, le psychologue Frank Rosenblatt a développé le modèle du perceptron **[51]**.Il s'agit d'un réseau de neurones, capable d'apprendre à différencier des formes simples, et à calculer certaines fonctions logiques. Il est inspiré du système visuel. Il a réussi à l'appliquer pour la reconnaissance de formes.
Les travaux de Rosenblatt (1958), ont vécu un grand intérêt dans le milieu Scientifique.Il a défini une structure de réseau appelée Perceptron, qui a la caractéristique de traiter les problèmes linéairement séparables.
Au début des années 60, Bernard Widrow et MarcianHoff ont inventé le réseau Adaline (Adaptive Linear), il se distingue par un algorithme d'apprentissage plus rapide que de celui du perceptron. Cet algorithme ajuste les poids par la minimisation de l'erreur quadratique, cette minimisation appelée aussi 'la descente du gradient'.
En 1969, deux scientifiques américains, appelés Minsky et Papert**[50]** ont publié un livre, dans lequel ont démontré les limites du perceptron proposé par Rosenblatt. En particulier, son incapacité de traiter des problèmes non linéaires.
De son coté, TeuvoKohonen (1972) a introduit un modèle, il l'a appelé Mémoire Associative. Son apprentissage est non supervisé, il a trouvé son application, dans la reconnaissance de forme et de parole.
Les travaux se sont ensuite, ralentis considérablement jusqu'aux années 80. Précisément en 1982, où John Hopfield a donné un nouveau souffle aux réseaux de neurones, en publiant un article introduisant un nouveau modèle de réseaux « nommé réseau complètement connecté » **[50].** Parallèlement, Werbos a conçu un mécanisme d'apprentissage, pour les réseaux multicouches de type perceptron : la rétro propagation (Back-Propagation).
Cet algorithme qui permet de propager l'erreur vers les couches cachées sera Popularisé en 1986 dans un livre "ParallelDistributedProcessing" par Rumelhart et al **[52]**. Les réseaux de neurones ont par la suite connu un essor considérable, et plusieurs applications sont apparues, ce qui a donné un avancement significatif à la science neuronale **[53].**

Actuellement, Les réseaux de neurones sont exploités sous forme de circuit intégré, et qui sont utilisés dans des domaines très variés, citons à titre d'exemple, la reconnaissance de forme, les problèmes d'optimisation, la prédiction, l'évaluation et le contrôle.

III.2. Définition [54]

Un réseau de neurones artificiels est un modèle de calcul dont la conception est très schématiquement inspirée du fonctionnement des neurones biologiques.

Les réseaux de neurones sont généralement optimisés par des méthodes d'apprentissage de type probabiliste, en particulier bayésien. Ils sont placés d'une part dans la famille des applications statistiques, qu'ils enrichissent avec un ensemble de paradigmes permettant de créer des classifications rapides, et d'autre part dans la famille des méthodes de l'intelligence artificielle auxquelles ils fournissent un mécanisme perceptif indépendant des idées propres de l'implémenteur, et fournissant des informations d'entrée au raisonnement logique formel.

En modélisation des circuits biologiques, ils permettent de tester quelques hypothèses fonctionnelles issues de la neurophysiologie, ou encore les conséquences de ces hypothèses pour les comparer au réel.

III.3. Principes des réseaux de neurones [55]

III.3.1. Du neurone biologique au neurone artificiel [55]

Le cerveau humain contient de l'ordre de 10 milliards de neurones.

Un neurone biologique est une cellule vivante, consacrée au traitement de l'information. De son corps cellulaire ou soma, rayonnent de nombreuses dendrites (jusqu'à 100000) qui reçoivent des signaux provenant d'autres neurones ou cellules sensorielles**.**

Ces signaux sont traités par le neurone, qui transmet à son tour un signal, si certaines conditions sont réunies le long de son axone à d'autres neurones, ou à de cellules effectrices (cellule musculaire par exemple) : On dit que le neurone est alors activé. Chaque neurone artificiel est un processeur élémentaire. Il reçoit un nombre de variables d'entrées, en provenance des neurones en amont. A chacune de ces entrées, est associé un poids w représentatif de la force de connexion. Chaque processeur élémentaire est doté d'une sortie unique, qui se ramifie ensuite, Pour alimenter un nombre de neurones en avals, à chaque connexion est associé un Poids.

Le modèle général du neurone artificiel, est composé des éléments suivants :

– Une ou plusieurs entrées pondérées.

– Un sommateur; Une fonction de transfert ; Une sortie. S ; et une entrée x_i.

– w_{ij} est la valeur du poids synaptique reliant l'entrée i au neurone j.

– Σ est la somme pondérée des entrées x_i.

– $F()$ est la fonction de transfert.

– s est la sortie du neurone. $S = F(\Sigma w_{ij}. x_i)$.

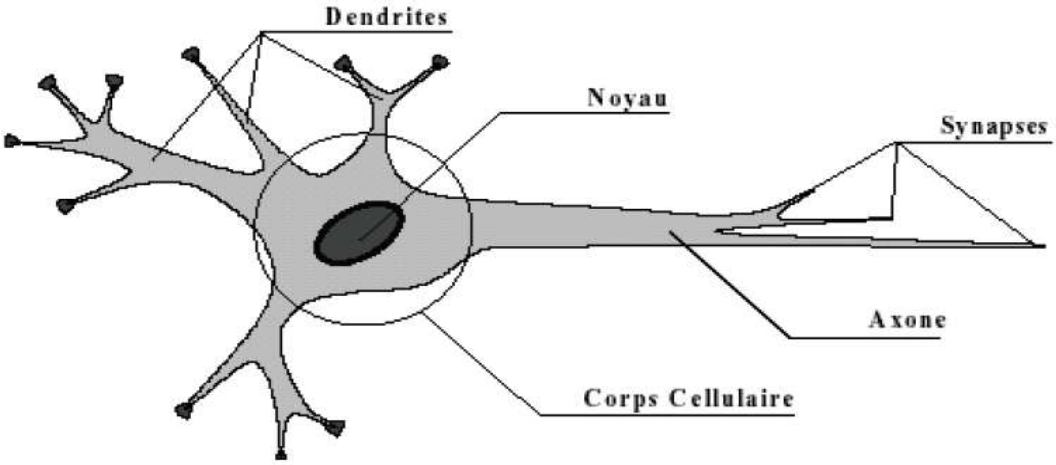


Figure III.1 : La représentation du neurone biologique.**[56]**

Un neurone biologique contient :

III.3.1.1. Un corps cellulaire [57]

Qui joue le rôle d'un sommateur à seuil. Il effectue une sommation des influx nerveux par ses dendrites ; si la somme est supérieure à un seuil donné, le neurone répond par un flux nerveux ou potentiel d'action qui se propage le long de son axone ; si la somme est inférieure au seuil, il reste inactif.

III.3.1.2. Des prolongements [57]

Qui reçoivent les signaux en provenance d'autres cellules. Ce sont les **dendrites.**

III.3.1.3. Un prolongement unique

Appelé **axone**, diffuse le signal du neurone vers d'autres cellules. **[57]**

L'axone, qui est à proprement parler la fibre nerveuse, sert de moyen de transport pour les signaux émis par le neurone Pour former le système nerveux, les neurones sont connectés les uns aux autres suivants des répartitions spatiales complexes, les connexionsentre deux neurones se font en des endroits appelés synapses où ils sont séparés par un petit espace synaptique de l'ordre d'un centième de micron. D'une façon simple, on peut dire que le soma du neurone traite les courants électriques qui lui proviennent de ses dendrites, et qu'il transmet le courant électrique (sous forme d'impulsions chaque une de durée d'environ 1ms et une amplitude d'environ 100mv) résultant de ce traitement aux neurones auxquels il est connecté par l'intermédiaire de son axone.

Le schéma classique présenté par les biologistes est celui d'un soma effectuant une sommation des influx nerveux transmis par ses dendrites Si la sommation dépasse un certain seuil, le neurone répond par un influx nerveux au potentiel d'action qui se propage le long de son axone. Si la sommation est inférieure à ce seuil, le neurone reste inactif. **[58]**

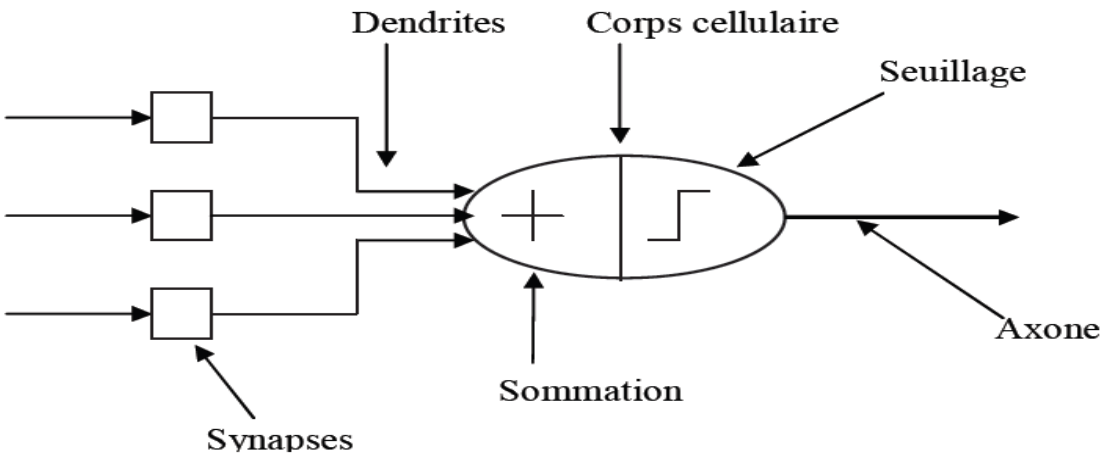


Figure III.2 : Le modèle d'un neurone biologique.**[56]**

III.3.1.4. Un élément de jonction [57]

Appelé **synapse**. Les synapses permettent aux cellules de communiquer entre elles, de plus il joue un rôle dans la modulation des signaux qui transitent le système nerveux.
Schéma d'une synapse :

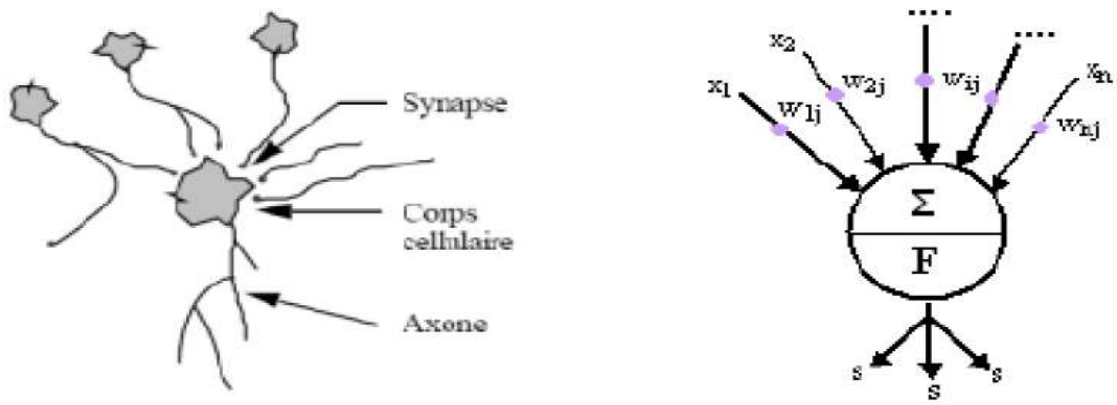


Figure III.3 : Mise en correspondance neurone biologique/ neurone artificiel. **[55]**

III.3.2.Modélisation d'un neurone formel [60]

La modélisation consiste à mettre en œuvre un système de réseau neuronal sous un aspect non pas biologique mais artificiel, cela suppose d'après le principe biologique, on aura une correspondance pour chaque élément composant le neurone biologique, donc une modélisation pour chacun d'entre eux.
On pourra résumer cette modélisation par le tableau (III.1), qui nous permettra de voir clairement la transition entre le neurone biologique et le neurone formel.

Neurone biologique	Neurone artificiel
synapses	Poids de connexions
Axones	Signal de sortie
Dendrites	Signal d'entrée
Noyau	Fonction d'activation

Tableau III.1 : Analogie entre le neurone biologique et le neurone formel. **[60]**

III.3.2.1. Les entrées :Elles peuvent être :

- Booléennes.
- Binaires (0 ,1) ou bipolaires (-1 ,1).
- Réelles.

III.3.2.2. Fonction d'activation

Cette fonction permet de définir l'état interne du neurone en fonction de son entrée totale, citons à titre d'exemple quelques fonctions souvent utilisées :

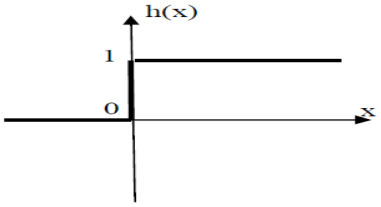
a. Fonction binaire à seuil

a.1. Fonction Heaviside définie par la fonction h(x), qui est illustrée par la figure III.4

$$h(x) = \begin{cases} 1 & si\ x \geq 0 \\ 0 & sinon \end{cases}$$

a.2. Fonction signe définie par la fonction sng(x), appelée aussi fonction de transfert à seuil, et qui est illustrée par la figure III.5

$$sgn(x) = \begin{cases} +1 & si\ x \geq 0 \\ -1 & sinon \end{cases}$$



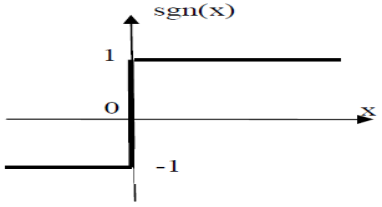


Figure III.4 : Fonction Heavisidale. **Figure III.5**: Fonction signe.

Le seuil introduit une non-linéarité dans le comportement du neurone, cependant il limite la gamme des réponses possibles à deux valeurs.

b. Fonction linéaire

C'est l'une des fonctions d'activation les plus simples, sa fonction est définie par :

$$F(x) = x$$

Elle est représentée par la figure III.7

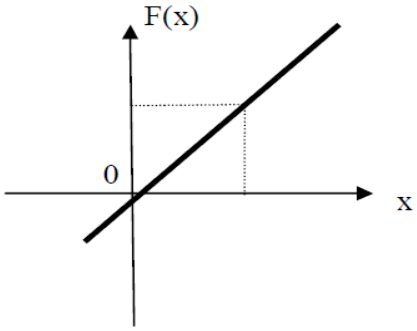


Figure III.6 : fonction linéaire

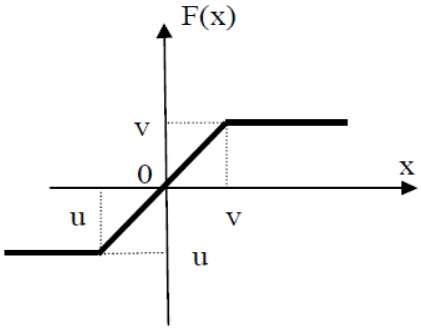


Figure III.7 : fonction linéaire à seuil.

c. Fonction linéaire à seuil ou multi-seuils

On peut la définir comme suit :

$$F(x) = \begin{cases} x & x \in [u\ v] \\ v & si\ x \geq v \\ u & si\ x \leq u \end{cases}$$

Cette fonction représente un compromis entre la fonction linéaire et la fonction à seuil :
Entre ces deux barres de saturations, elle confère au neurone une gamme de réponse possible.
En modulant la pente de linéarité, on affecte la plage de neurone.
Cette fonction est représentée par la figure III.7.

d. Fonction sigmoïde

Elle est l'équivalent continu de la fonction linéaire. Etant continue, elle est dérivable, d'autant plus que sa dérivée est simple à calculer, elle est définie par la fonction f (x):

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

La tracée de cette fonction est donnée par la figure III.8

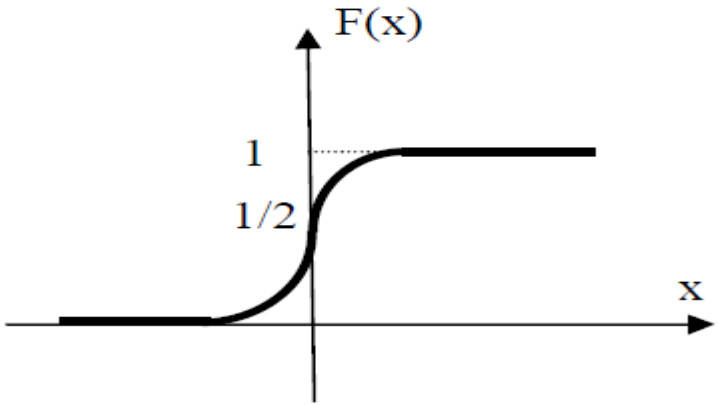


Figure III.8 : Fonction sigmoïde.

III.3.2.2. Fonction de sortie

Elle calcule la sortie d'un neurone en fonction de son état d'activation. En général, cette fonction est considérée comme la fonction identité. Elle peut être :

- Binaire (0 ,1) ou bipolaire (-1 ,1)
- Réelle

III.4. Apprentissage des réseaux de neurones [55]

Pour un réseau de neurones, l'apprentissage peut être considéré comme le problème de la mise à jour des poids de connexions au sein du réseau, afin de réussir la tâche qui lui est demandée.

L'apprentissage est la caractéristique principale des RNA et il peut se faire de différentes manières et selon différentes règles **[65].**

III.4.1. L'apprentissage supervisé [55]

L'apprentissage "supervisé" pour les réseaux de neurones formels, consiste à calculer les coefficients synaptiques, de telle manière que les sorties du réseau soient, aussi proches que possible des sorties "désirées". Ils peuvent être la classe d'appartenance de :

– La forme que l'on veut classer.

– La valeur de la fonction que l'on veut approcher.

– La sortie du processus que l'on veut modéliser.

– La sortie souhaitée du processus à commander.

On connaît donc, en tout point, ou seulement en quelques points, les valeurs que doit avoir la sortie du réseau en fonction des entrées correspondantes : C'est en ce sens que l'apprentissage est "supervisé". Cela signifie qu'un "professeur"**[61]**peut fournir au réseau des "exemples", de ce que celui-ci doit faire.

Ce type d'apprentissage, sera adopté par la suite dans l'algorithme de prédiction.

La plupart des algorithmes d'apprentissage des réseaux de neurones formels, sont des algorithmes d'optimisation : Ils cherchent à minimiser par des méthodes d'optimisation non Linéaires, une fonction de coût qui constitue une mesure de l'écart entre les réponses réelles du réseau et les réponses désirées.
Cette optimisation se fait de manière itérative, en modifiant les poids en fonction du gradient de la fonction de coût :
Le gradient est estimé par une méthode spécifique aux réseaux de neurones, dite méthode de rétro propagation, puis il est utilisé par l'algorithme d'optimisation proprement dit.
Les poids sont initialisés aléatoirement avant l'apprentissage, puis modifiés itérativement, jusqu'à obtention d'un compromis satisfaisant, entre la précision de l'approximation sur l'ensemble d'apprentissage, et la précision de l'approximation sur un ensemble de validation disjoint du précédent.
C'est ce type d'apprentissage que nous allons utiliser par la suite dans notre travail.

III.4.2. L'apprentissage non supervisé

Un réseau de neurones non supervisé peut être également utilisé dans un but de visualiser ou d'analyser des données : On dispose d'un ensemble de données, représentées par des vecteurs de grande dimension. Et l'on cherche à les regrouper selon des critères de ressemblance qui sont inconnus à priori. Ce type de tâches est connu en statistique sous le nom de méthodes "d'agrégation".
On peut utiliser les réseaux de neurones bouclés pour réaliser une tâche assez voisine : à partir des données décrites par des vecteurs de grandes dimensions, on cherche à trouver une représentation de ces données dans un espace de Dimension beaucoup plus faible, (Typiquement de dimension 2) tout en conservant les "proximités" ou "ressemblances" entre ces derniers.
Il n'y a pas là, donc de "professeur", puisque c'est au réseau de découvrir les ressemblances entre les éléments de la base de données, et de les traduire par une proximité dans la "carte" de dimension 2 qu'il doit produire.

III.4.3. L'apprentissage hybride

L'apprentissage hybride utilise les deux approches. Puisqu'une partie des poids est déterminée par l'apprentissage supervisé, et l'autre partie par l'apprentissage non supervisé.

III.5. Application [66]

Les principales applications des réseaux de neurones sont l'optimisation et l'apprentissage. En apprentissage, les réseaux de neurones sont essentiellement utilisés pour :

- L'apprentissage supervisé ;
- L'apprentissage non supervisé ;
- L'apprentissage par renforcement.

Pour ces trois types d'apprentissage, il y a également un choix traditionnel entre :

- L'apprentissage « off-line »: toutes les données sont dans une base d'exemples d'apprentissage qui sont traités simultanément ;
- L'apprentissage « on-line » : Les exemples sont présentés les uns après les autres au fur et à mesure de leur disponibilité.

Nous nous limitons, à l'apprentissage supervisé « hors line » à partir d'une base d'exemples. Dans ce cadre, l'apprentissage à l'aide de réseaux de neurones est bien adapté pour l'apprentissage à partir de données complexes (images sur une rétine, sons) mais aussi à partir de données symboliques. Les entrées peuvent être représentées par de nombreux attributs à valeurs réelles ou symboliques, les attributs pouvant être dépendants ou non. La ou les sorties peuvent être réelles ou discrètes.

L'apprentissage à l'aide de réseaux de neurones est tolérant au bruit et aux erreurs. Le temps d'apprentissage peut être long, par contre, après apprentissage, le calcul des sorties à partir d'un vecteur d'entrée est rapide. La critique principale est que le résultat de l'apprentissage, c'est-à-dire le réseau de neurones calculé par l'algorithme d'apprentissage, n'est pas interprétable par l'utilisateur : on ne peut pas donner d'explication au calcul d'une sortie sur un vecteur d'entrée. On parle de << boîte noire >>.

Ceci est la principale différence entre réseaux de neurones et arbres de décision. Si l'utilisateur a besoin de pouvoir interpréter le résultat de l'apprentissage, il choisira un système basé sur les arbres de décision, sinon les deux méthodes sont concurrentes.

Dans ce qui a précédé nous avons vu la notion du perceptron, brique de base des modèles plus complexes, et le perceptron multi-couches (PMC).

L'accent sera mis sur les algorithmes d'apprentissage pour ces deux modèles, en particulier sur l'algorithme de rétropropagation du gradient appliqué aux PMC. Cet algorithme est, en effet, le premier algorithme d'apprentissage convaincant dans un modèle suffisamment puissant et cet algorithme à de nombreuses applications.

III.6. Le perceptron

III.6.1. Définition

Le perceptron était la première signification complète de l'architecture adaptive, inventé par Frank Rosenblatt en 1957, le perceptron est actuellement une classe entière des architectures. Il suit un apprentissage supervisé selon la règle de Hebb. Il est limité à deux couches (couche d'entrée et couche de sortie), il a été connu pour son utilisation dans la reconnaissance de forme, dans la classification et pour résoudre des opérations logiques ('ET' ou 'OU') **[13]**.

Sa principale limitation est qu'il ne peut résoudre, que les problèmes linéairement séparables. **[4]**

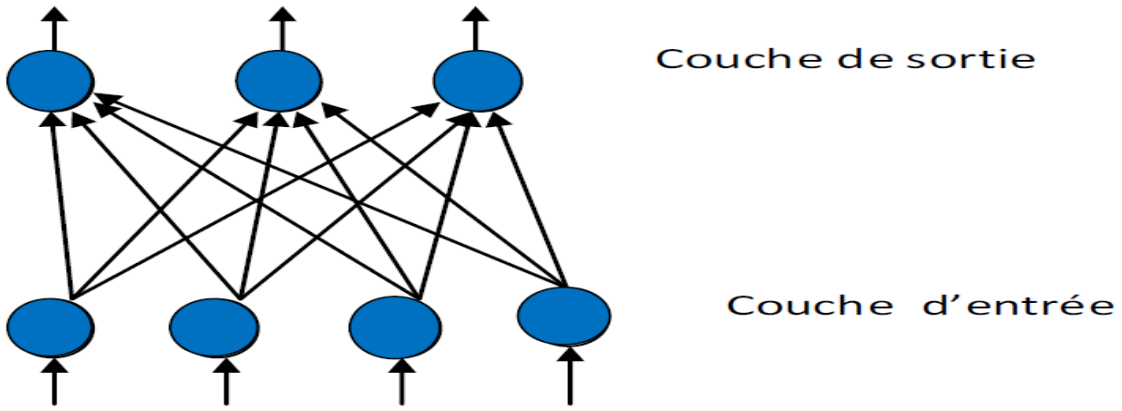


Figure III.9 : Perceptron à deux couches.

Dans sa version simplifiée, le perceptron est mono-couche et n'a qu'une seule sortie à laquelle toutes les entrées sont connectées. Les entrées et la sortie sont booléennes.

Le potentiel post-synaptique biaisé est représenté par $Z = \sum W_i X_i - \theta$

Θ définit le seuil (ou biais) à dépasser pour que la sortie Y soit à 1, est le W_i poids

de l'entrée X_i.

La fonction d'activation est la fonction de Heaviside (la fonction signe est parfois utilisée) : **[67]**

$$Y = H(Z) = \begin{cases} 0 & \text{si} \quad Z < 0 \\ 1 & \text{si} \quad Z \geq 0 \end{cases}$$

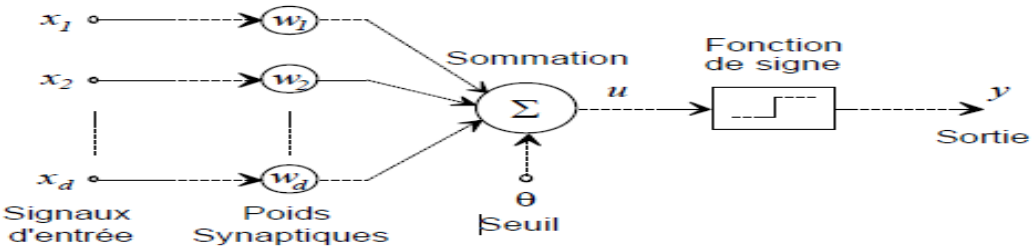


Figure III.10 : Représentation du Perceptron.**[68]**

III.6.2. La règle de Hebb[67]

La règle de Hebb établie par Donald Hebb est une règle d'apprentissage des réseaux de neurones artificiels dans le contexte de l'étude d'assemblées de neurones.
Cette règle suggère que lorsque deux neurones sont excités conjointement, il se crée ou renforce un lien les unissant.

Dans le cas d'un neurone artificiel seul utilisant la fonction signe comme fonction d'activation cela signifie que : $W_i' = W_i + \alpha(Y.X_i)$

Où W_i' représente le poids ì corrigé, et α représente le pas d'apprentissage.

Cette règle n'est malheureusement pas applicable dans certains cas bien que la solution existe.

III.6.3. Règle d'apprentissage du perceptron [67]

Le perceptron de Frank Rosenblatt est très proche de la règle de Hebb, la grande différence étant qu'il tient compte de l'erreur observée en sortie.

$$W_i' = W_i + \alpha(Y_t - Y)X_i$$

Où :

Y_t Représente la sortie attendue.

W_i' Le poids i corrigé, et α le pas d'apprentissage.

III.6.4. Apprentissage par méthode de gradient [69]

Plutôt que d'obtenir un perceptron qui classifie correctement tous les exemples, il s'agira maintenant de calculer une erreur et d'essayer de minimiser cette erreur. Pour introduire cette notion d'erreur, on utilise des poids réels et on élimine la notion de seuil (ou d'entrée supplémentaire), ce qui signifie que la sortie sera égale au potentiel post-synaptique et sera donc réelle.

Un perceptron linéaire prend en entrée un vecteur $\vec{x}$ de **n** valeurs $x_1,\ldots, x_n$ et calcule une sortie **o.** Un perceptron est défini par la donnée d'un vecteur $\vec{w}$ de **n** constantes :

Les coefficients synaptiques $w_1, \ldots, w_n$.La sortie o est définie par :

$$o = \vec{x} \cdot \vec{w} = \sum_{i=1}^{n} w_i x_i$$

L'erreur d'un perceptron **P** défini par $\vec{w}=(w_1,\ldots,w_n)$ sur un échantillon d'apprentissage **S** d'exemples $(\vec{x}^s, c^s)$ est définie en utilisant la fonction erreur quadratique par

$$E(\vec{w}) = 1/2 \sum_{(\vec{x}^s, c^s) \in S} (c^s - o^s)^2$$

Où o^s est la sortie calculée par P sur l'entrée $\vec{x}^s$**.** L'erreur mesure donc l'écart entre les sorties attendue et calculée sur l'échantillon complet. On remarque que $E(\vec{w}) = 0$ si et seulement si le perceptron classifie correctement l'échantillon complet.

On suppose **S** fixé, le problème est donc de déterminer un vecteur $\vec{w}$ qui minimise $E(\vec{w})$**.** Une méthode qui permet de rechercher le minimum d'une fonction est d'utiliser la méthode du gradient.

III.6.4.1. Méthode du gradient [66]

Soit f une fonction d'une variable réelle à valeurs réelles, suffisamment dérivable dont on recherche un minimum. La méthode du gradient construit une suite x qui doit en principe s'approcher du minimum. Pour cela, on part d'une valeur quelconquex_0 et l'on construit la suite récurrente par :

pour tout n>0, $x_{n+1} = x_n + \Delta x_n$ avec $\Delta x_n = -\varepsilon f(x_n)$ où ε est une valeur « bien » choisie.

On a : $f(x_{n+1}) = f(x_n - \varepsilon f(x_n)) \cup f(x_n) - \varepsilon (f(x_n))^2$ d'après le théorème des approximations finies si ε $f(x_n)$ « suffisamment » petit. On voit que, sous réserve de la correction de l'approximation, $f(x_{n+1})$ est inférieur à $f(x_n)$

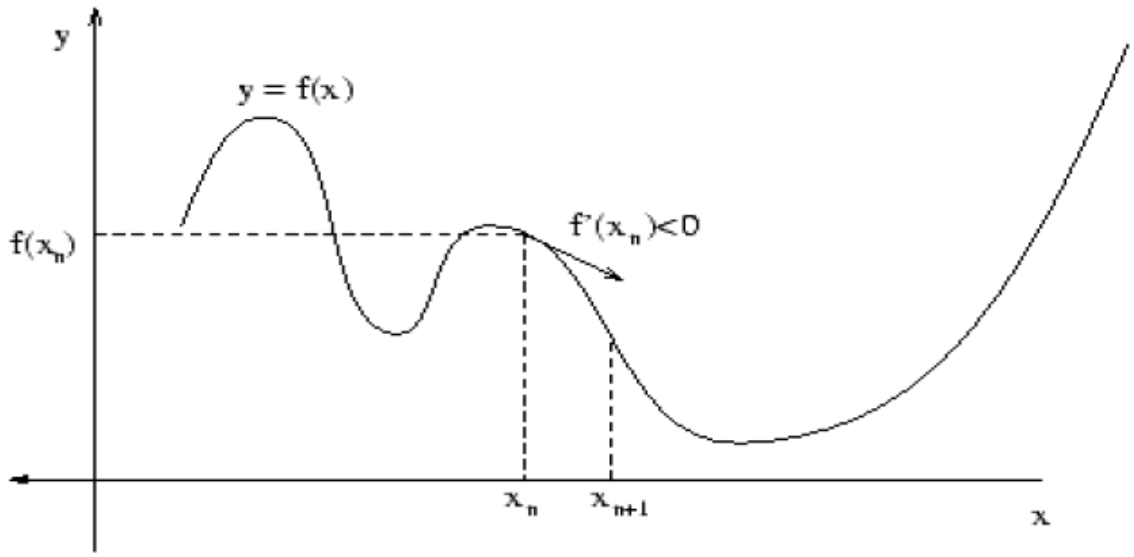


Figure III.11 : La méthode du gradient.

On remarque que x_{n+1} est d'autant plus éloigné de x_n ue la pente de la courbe en x_n est grande. On peut décider d'arrêter l'itération lorsque cette pente est suffisamment faible.

Les inconvénients bien connus de cette méthode sont :

1.le choix de e est empirique.
2. si e est trop petit, le nombre d'itérations peut être très élevé.
3. si e est trop grand, les valeurs de la suite risquent d'osciller autour du minimum sans converger.
4. rien ne garantit que le minimum trouvé soit un minimum global.

III.6.5. Algorithme d'apprentissage par descente de gradient [66] [70]

E est une fonction des n variables. La méthode du gradient a été rappelée dans le cas d'une variable w_ie, mais cette méthode peut être étendue au cas de fonctions de plusieurs variables réelles. Pour mettre en œuvre la méthode appliquée à la fonction erreur quadratique E, nous allons, tout d'abord, évaluer la dérivée partielle de E par rapport à w_i , pour tout i. On a :

$$\frac{\partial E(W)}{\partial wi} = \frac{1}{2}\frac{\partial}{\partial wi}\sum_S (c-o)^2 = \frac{1}{2}\sum_S \frac{\partial}{\partial wi}(c-o)^2$$

$$\frac{\partial E(W)}{\partial wi} = \frac{1}{2}\sum_S 2(c-o)\frac{\partial}{\partial wi}(c-o) = \sum_S (c-o)\frac{\partial}{\partial wi}(c-\sigma(x\cdot w))$$

$$Or : \frac{\partial}{\partial wi}(c-\sigma(x.w)) = \frac{\partial}{\partial wi}(c-\sigma(y)) = \frac{\partial}{\partial y}(c-\sigma(y)) * \frac{\partial y}{\partial wi} = -\sigma'(y) * xi$$

$$\frac{\partial E(W)}{\partial wi} = \sum_S (c-o)(-xi)\sigma'(x\cdot w)$$

$$D'où : \Delta wi = -\varepsilon * \frac{\partial E(W)}{\partial wi} = \varepsilon \sum_S xi * (c-o) * \sigma'(x\cdot w)$$

- **Algorithme**

- Initialiser aléatoirement les coefficients wi.
- Répéter :
  - Pout tout i :
    - $\Delta wi = 0$
  - Fin Pour
  - Pour tout exemple (x, c) dans S
    - Calculer la sortie o du réseau pour l 'entrée x
    - Pout tout i :
      - $\Delta wi = \Delta wi + \varepsilon * (c - o) * xi * \sigma'(x.w)$
    - Fin Pour
  - Fin Pour
  - Pour tout i :
    - $wi = wi + \Delta wi$
  - Fin Pour
- Fin Répéter

➢ **Variante de la Règle Delta généralisée :**

• On ne calcule pas les variations de coefficients en sommant sur tous les exemples de S mais on modifie les poids à chaque présentation d'exemple.

Initialiser aléatoirement les coefficients wi.
- Répéter :
  - Prendre un exemple (x, c) dans S
  - Calculer la sortie o du réseau pour l 'entrée x
  - Pout i de 1 à n :
    - $wi = wi + \varepsilon * (c - o) * xi * \sigma'(x.w)$
  - Fin Pour
- Fin Répéter

III.6.6. Algorithme d'apprentissage de Widrow-Hoff (adaline / règle delta)

L'algorithme de Widrow-Hoff s'écarte de l'algorithme du gradient sur un point important : on modifie les poids après présentation de chaque exemple en fonction de l'erreur locale et non de l'erreur globale. Rien ne prouve donc que la diminution de l'erreur en un point ne va pas être compensée par une augmentation de l'erreur pour les autres points. La justification empirique de cette manière de procéder est commune à toutes les méthodes adaptatives : le champ d'application des méthodes adaptatives est justement l'ensemble des problèmes pour lesquels des ajustements locaux vont finir par converger vers une solution globale.
L'algorithme de Widrow-Hoff est très souvent utilisé en pratique et donne de bons résultats. La convergence est, en général, plus rapide que par la méthode du gradient. Il est fréquent pour cet algorithme de faire diminuer la valeur en fonction du nombre d'itérations comme pour l'algorithme du gradient. **[66] [70]**

La fonction de transfert est linéaire (purelin) : $\sigma(y) = y$
Donc : $\sigma'(y) = 1$

- Initialiser aléatoirement les coefficients wi.
- Répéter :
  - Prendre un exemple (x, c) dans S
  - Calculer la sortie o du réseau pour l 'entrée x
  - Pout i de 1 à n :
    - $wi = wi + \varepsilon * (c - o) * xi$
  - Fin Pour
- Fin Répéter

- **Remarques[70]**

• ε : pas d'apprentissage.
• Règles « delta » et « delta généralisée » : les algorithmes convergent vers la solution des moindres carrés.
• La règle « delta généralisée », par la non-linéarité de la fonction de transfert σ, permet de minimiser l'importance d'un élément étranger (erreur de mesure, bruit trop important, …).

- **Conclusion [66]**

En conclusion, l'apprentissage par perceptron ou par la méthode du gradient ne sont rien d'autre que des techniques de séparation linéaire qu'il faudrait comparer aux techniques utilisées habituellement en statistiques. Ces méthodes sont non paramétriques, c'est-à-dire qu'elles n'exigent aucune autre hypothèse sur les données que la séparabilité.
On peut montrer que « presque » tous les échantillons de moins de 2n exemples sont linéairement séparables lorsque n estle nombre de variables. Une classification correcte d'un petit échantillon n'a donc aucune valeur prédictive. Par contre, lorsque l'on travaille sur

suffisamment de données et que le problème s'y prête, on constate empiriquement que le perceptron appris par un des algorithmes précédents a un bon pouvoir prédictif.
Il est bien évident que la plupart des problèmes d'apprentissage qui se posent naturellement ne peuvent pas être résolus par des méthodes aussi simples : il n'y a que très peu d'espoir que les exemples « naturels » se répartissent « sagement » de partet d'autre d'un hyperplan. Une manière de résoudre cette difficulté serait soit de mettre au point des séparateurs non-linéaires, soit (ce qui revient à peu près au même) de complexifier l'espace de représentation de manière à linéaires le problème initial.
C'est ce que permettent de faire les réseaux multicouches.

III.7. Le perceptron multicouche

Bien que les réseaux monocouches, soient déjà largement utilisés dans plusieurs applications, comme la reconnaissance de visages et le traitement de signal, leur efficacité pour traiter les problèmes compliqués est très limitée. Les théorèmes de limitation du perceptron mis au point par minsky et papert**[9]** , qui ont conclu que les cellules d'association dans le perceptron sont codées à la main, et pas par un algorithme d'apprentissage **[4]**.
L'idée de base des réseaux multicouches, est tout simplement de décomposer une tache, en plusieurs étages, dont chacun est censé être plus simple à résoudre. La différence entre le perceptron et le perceptron multicouche, réside dans la façon de génération des cellules intermédiaires, elles sont à la main dans le perceptron, tandis que la fonction des cellules intermédiaires dans un réseau multicouche, pourrait être générer automatiquement par un algorithme d'apprentissage **[9]**.
La caractéristique fondamentale de l'apprentissage est la règle du gradient, les unités de ces réseaux sont organisées en couches, les neurones d'une couche sont connectés à la couche suivante seulement (il n'ya pas de rétroaction).

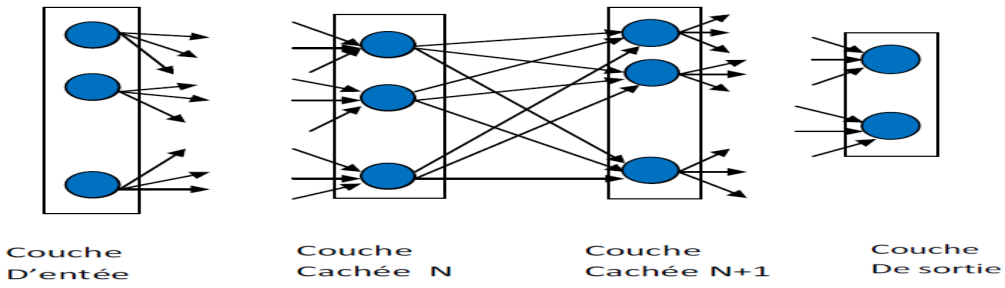


Figure III.12 : Architecture d'un perceptron multicouche.

Les réseaux multicouches comportent une couche d'entrée, une de sortie et une ou plusieurs couches cachées **[61]**.
L'apprentissage de ce type de réseau est un apprentissage supervisé.
Son architecture est arbitraire et dans la plupart des cas, il faut essayer plusieurs configurations, pour y arriver à l'architecture convenable au problème envisagé.

Dans notre travail, nous allons utiliser des réseaux de neurones de type perceptron multicouches.

III.7.1. La rétro propagation du gradient d'erreur

Un des points important dans la conception d'une application à base de réseaux de neurones concerne la méthode d'apprentissage. Celle-ci doit idéalement nous permettre de converger rapidement vers le minimum global de la fonction de coût choisie. Cependant les méthodes permettant de trouver le minimum global d'une fonction sont en général très lentes en considérant le nombre de paramètres que nous devrons prendre en compte. Nous avons donc utilisé l'algorithme de rétro-propagation du gradient qui ne garantit pas d'aboutir au minimum global de la fonction C, mais celui-ci est beaucoup plus rapide et est donc utilisable dans le cadre de notre application.
Le but de l'algorithme de rétro-propagation du gradient est de minimiser notre fonction de coût (i.e. de minimiser l'erreur). Pour cela il utilise le concept de dérivées partielles et calcule le gradient de la fonction de coût qui fournit une indication sur la direction et l'amplitude du changement à effectuer sur les poids w pour minimiser C. En effet cet algorithme utilise le

fait qu'en tout point le gradient est normal aux lignes de niveau ce qui permet de connaître la direction d'un des minimum de C.

La modification des poids suivra donc une loi du type :

$$\Delta w = -\eta \frac{\partial C}{\partial w}$$

La complexité de cet algorithme est donc fonction du nombre de paramètres (poids) à traiter et augmente très vite avec la taille du réseau neuronal.

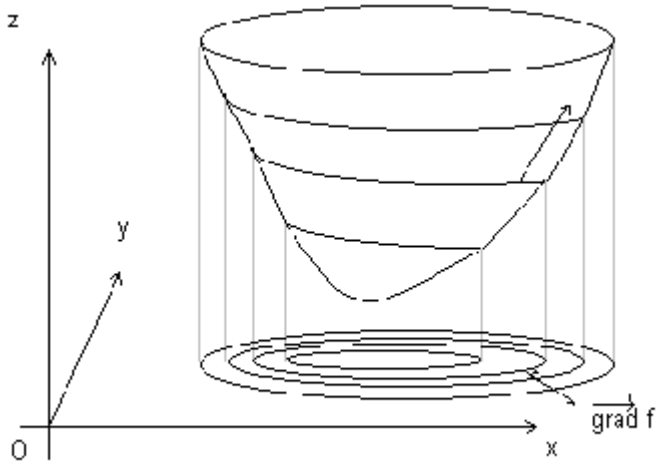


Figure III.13 : Graphe représentant la rétro propagation du gradient d'erreur.

III.7.2. Rapport : Capacité/Généralisation [6]

Nous ne pouvons pas parler d'apprentissage sans évoquer le problème de la généralisation.

La généralisation est la faculté d'étendre une compétence à des exemples non appris.Nous avons obtenu une machine à apprendre qui a appris une approximation. Comment se comporte cette approximation sur des points différents des N appris ?

Avant d'approfondir cette question, il est nécessaire d'introduire la notion de capacité.En effet, des résultats théoriques permettent d'établir une relation entre la généralisation d'une machine à apprendre et sa capacité.

III.7.2.1. Capacité [6]

La capacité a été introduite par Vapink(1995) dans la théorie de l'apprentissage.Pour illustrer la notion de capacité, nous considérons une suite de points que nous cherchons à interpoler avec un polynôme. D'un côté, si le polynôme a trop de coefficients (capacité forte)), il apprendra par cœur tous les points. Ainsi, si nous considérons une machine à apprendre dans le cadre des problèmes de classification, la capacité représente le nombre maximum d'exemples que la machine peut classifier, quel que soit l'étiquetage.

III.7.2.2. Généralisation [72]

Les réseaux de neurones devant généraliser sur les exemples de l'ensemble d'apprentissage, ils ne sont qu'une approximation des fonctions que l'on recherchait vraiment. Quelques problèmes de l'approximation de réseaux de neurones sont identifiés ici et des solutions y sont proposées.
Le paramètre à minimiser dans le cas de l'entraînement d'un réseau de neurone est l'erreur sur les résultats donnés par le réseau de neurone. Cependant, celle-ci ne tient compte que des valeurs de l'ensemble d'entraînement, alors qu'on devrait tenir compte de l'erreur sur toutes les données à traiter. Ceci étant impossible, on base plutôt l'optimisation sur un risque moyen. Le risque quadratique moyen est un bon paramètre à optimiser. Ce risque se décompose en trois termes : l'erreur bayésienne, le biais et la variance. L'erreur bayésienne tient à l'apprentissage, mais est indépendante de la procédure d'apprentissage. La difficulté

de cette optimisation est de contrôler à la fois le biais et la variance. Alors que la variance est monotone croissante, amis converge, le biais a un comportement non-monotone. Il commence par décroître, puis croît lentement. Il faut donc trouver un compromis où la somme du biais et de la variance est minimale. Cela revient à accepter un certain biais pour maintenir la variance relativement faible. Dans le cas des réseaux de neurones, c'est le contrôle de la complexité du réseau de neurone qui permet de trouver l'estimateur réalisant le bon compromis biais-variance.

III.7.3. L'algorithme de rétro propagation du gradient [74]

- La fonction sigmoïde de paramètre k>0 est définie par :

$$\sigma_k(x) = \frac{e^{kx}}{e^{kx}+1} = \frac{1}{1+e^{-kx}} \qquad (1.1)$$

Cette fonction est une approximation indéfiniment dérivable de la fonction à seuil de Heaviside, d'autant meilleure que k est grand. Nous prendrons k=1 dans la suite, soit la fonction dσ nie par :

$$\sigma(x) = \frac{e^{x}}{e^{x}+1} = \frac{1}{1+e^{-x}} \qquad (1.2)$$

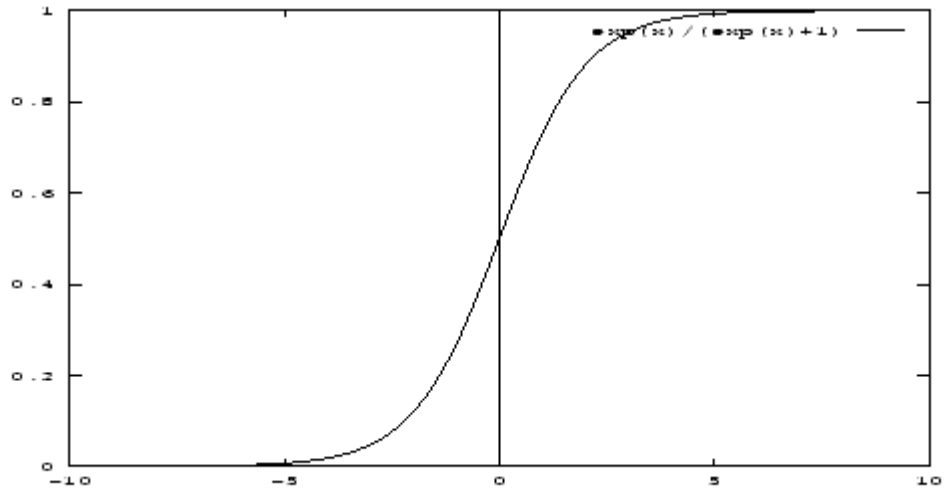


Figure III.14 : La fonction sigmoïde.

On peut remarquer que la dérivée de la fonction σ est simple à calculer :

$$\sigma'(x) = \frac{e^x}{(1+e^x)^2} = \sigma(x)(1-\sigma(x)) \qquad (1.3)$$

- Un neurone élémentaire à n entrées réelles $\vec{x} = (x_1,\ldots,x_n)$ est défini par les poids synaptiques réels $\vec{w} = (w_1,\ldots,w_n)$ et la sortie o est calculée par la formule suivante :

$$\vec{o(x)} = \frac{1}{1+e^{-y}} \qquad (1.4)$$

Où :

$$y = \vec{x}.\vec{w} = \sum_{i=1}^{n} w_i x_i \qquad (1.5)$$

Un perceptron multi-couches (PMC) est un réseau de neurones à couches cachées avec les neurones élémentaires ainsi définis.

III.7.3.1. Introduction de l'algorithme

Le principe de l'algorithme est, comme dans le cas du perceptron linéaire, de minimiser une fonction d'erreur. Il s'agit ensuite de calculer la contribution à cette erreur de chacun des poids synaptiques. C'est cette étape qui est difficile. En effet, chacun des poids influe sur le neurone correspondant, mais, la modification pour ce neurone va influer sur tous les neurones des couches suivantes.

Soit un PMC défini par une architecture à n entrées età p sorties, soit $\vec{w}$ le vecteur des poids synaptiques associés à tous les liens du réseau. L'erreur du PMC sur unéchantillon d'apprentissage S d'exemples $(\vec{x^s},\vec{c^s})$ est définie par :

$$\vec{E(w)} = 1/2 \sum_{(\vec{x^s},\vec{c^s}) \in S} \sum_{k=1}^{p} (c_k^s - o_k^s)^2 \qquad (1.6)$$

Où o_k est la k-ième composante du vecteur de sortie $\vec{o}$ calculé par le PMC surl'entrée $\vec{x}$. L'erreur mesure donc l'écart entre les sorties attendue et calculéesur l'échantillon complet. Nous supposons S fixé, le problème est donc de déterminer un vecteur $\vec{w}$ qui minimise $E(\vec{w})$. Cependant, de la même façon que pour le perceptron avec la règle de Widrow-Hoff, plutôt que de chercher à minimiser l'erreur globale sur l'échantillon complet, Nous cherchons à minimiser l'erreur sur chaque présentation individuelle d'exemple. L'erreur pour un exemple est :

$$E_{(\vec{x},\vec{c})}(\vec{w}) = 1/2 \sum_{k=1}^{p} (c_k - o_k)^2 \qquad (1.7)$$

Nous notons E la fonction $E_{(\vec{x},\vec{c})}$, E est une fonction des poids synaptiques, pour appliquer la méthode du gradient, il nous faut évaluer les dérivées partielles de cette fonction E par rapport aux poids synaptiques. Les calculs qui suivent sont faciles. La seule complication provient de la complexité des notations et des indices utilisés, complication due à la structure du PMC. Nous utilisons les notations suivantes :

- chaque cellule est définie par un indice,
- le réseau comporte p cellules de sortie,
- si i est l'indice d'une cellule de sortie, c_i est la sortie attendue pour cette cellule sur l'entrée $\vec{x}$,
- w_{ij} est le poids synaptique associé au lien entre cellule j vers la cellule i, ce qui implique qu'elles se trouvent sur deux couches successives par définition de l'architecture,
- x_{ij} est l'entrée associée au lien entre cellule j vers cellule i,
- Pred(i) est l'ensemble des cellules dont la sortie est une entrée de la cellule i ; ceci implique que la cellule n'est pas une cellule d'entrée et que tous les éléments de Pred(i) appartiennent à la couche précédente de celle à laquelle appartient la cellule i,
- y_i l'entrée totale de la cellule i, soit $y_i = \sum_{j \in Pred(i)} w_{ij} x_{ij}$,
- o_i est la sortie de la cellule i, soit $o_i = \sigma(y_i)$,Succ(i) est l'ensemble des cellules qui prennent comme entrée la sortie de la cellule i, ceci implique la cellule n'est pas une cellule de sortie et que tous les éléments de Succ(i) appartiennent à la couche suivante de celle à laquelle appartient la cellule i.

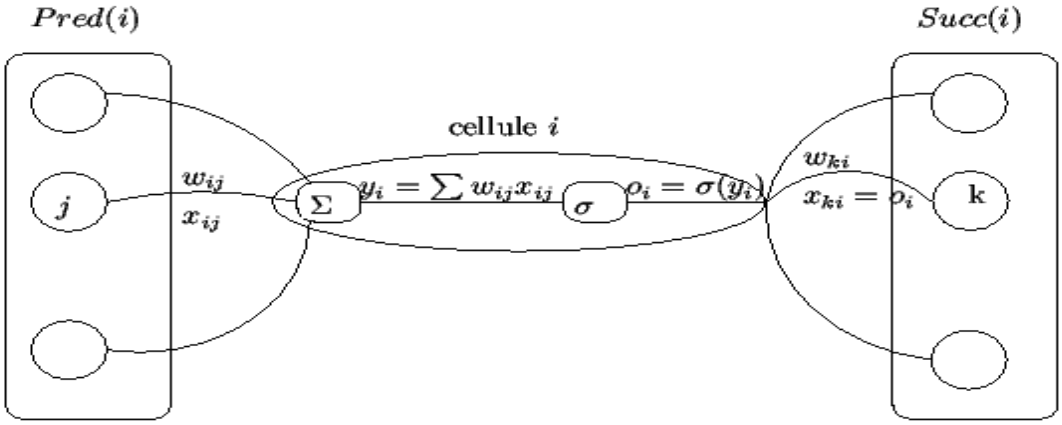


Figure III.15 : Schéma représentant la définition d'un réseau de neurones.

Il nous reste maintenant à évaluer $\partial E(w^{\rightarrow})/\partial w_{ij}$ que nous noterons $\partial E/\partial w_{ij}$. Tout d'abord remarquons que w_{ij} ne peut influencer la sortie du réseau qu'à travers le calcul de la quantité y_i,ce qui nous autorise à écrire que :

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial y_i}\frac{\partial y_i}{\partial w_{ij}} = \frac{\partial E}{\partial y_i}x_{ij} \qquad (1.8)$$

- Il nous suffit donc de calculer $\partial E/\partial y_i$, pour cela, nous allons distinguer deux cas : le cas où le neurone i est un neurone de sortie et le cas où c'est une cellule interne.

Le neurone i est une cellule de sortie :

Dans ce cas, la quantité y_i ne peut influencer la sortie du réseau que par le calcul de o_i. Nous avons donc :

$$\frac{\partial E}{\partial y_i} = \frac{\partial E}{\partial o_i}\frac{\partial o_i}{\partial y_i} \qquad (1.9)$$

Nous allons maintenant calculer chacune des deux dérivées partielles apparaissant dans l'équation (1.8).Pour la première de ces deux dérivées nous avons :

$$\frac{\partial E}{\partial o_i} = \frac{\partial}{\partial o_i}\frac{1}{2}\sum_{k=1}^{p}(c_k - o_k)^2 \qquad (1.10)$$

Seul le terme correspondant à k=i a une dérivée non nulle, ce qui nous donne finalement :

$$\frac{\partial E}{\partial o_i} = \frac{\partial}{\partial o_i}\frac{1}{2}(c_i - o_i)^2 = -(c_i - o_i) \qquad (1.11)$$

Pour la seconde des deux dérivées de l'équation (1.9), en utilisant la définition du calcul de la sortie d'une cellule élémentaire et la formule de calcul de la dérivée de la fonction sigmoïde donnée en (1.3), nous avons :

$$\frac{\partial o_i}{\partial y_i} = \frac{\partial \sigma(y_i)}{\partial y_i} = \sigma(y_i)(1-\sigma(y_i)) = o_i(1-o_i) \qquad (1.12)$$

En substituant les résultats obtenus par les équations (1.11) et (1.12) dans l'équation (1.9), nous obtenons :

$$\frac{\partial E}{\partial y_i} = -(c_i - o_i)o_i(1-o_i) \qquad (1.13)$$

- Le neurone i est une cellule interne :

  Dans ce cas, la quantité y_i va influencer le réseau par tous les calculs des cellules de l'ensemble Succ(i). Nous avons alors : (1.14)

$$\frac{\partial E}{\partial y_i} = \sum_{k \in Succ(i)} \frac{\partial E}{\partial y_k}\frac{\partial y_k}{\partial y_i} = \sum_{k \in Succ(i)} \frac{\partial E}{\partial y_k}\frac{\partial y_k}{\partial o_i}\frac{\partial o_i}{\partial y_i} = \sum_{k \in Succ(i)} \frac{\partial E}{\partial y_k} \times w_{ki} \times o_i(1-o_i)$$

Soit encore :

$$\frac{\partial E}{\partial y_i} = o_i(1-o_i) \sum_{k \in Succ(i)} \frac{\partial E}{\partial y_k} \times w_{ki} \qquad (1.15)$$

Par l'étude de ces deux cas, nous avons obtenu deux équations (1.13) et (1.15) qui nous permettent de calculer les dérivées partielles $\partial E/\partial y_i$ pour toute cellule i. Le calcul devra être fait pour les cellules de sortie puis des cellules de l'avant-dernière couche jusqu'aux cellules de la première couche. C'est pour cette raison que l'on parle de « rétropropagation ». Grace à l'équation (1.8), nous pouvons calculer toutes les dérivées partielles $\partial E(\vec{w})/\partial w_{ij}$. Enfin, pour en déduire la modification à effectuer sur les poids synaptiques, il nous reste simplement à rappeler que la méthode du gradient nous indique

$$\Delta w_{ij} = -\varepsilon \frac{\partial E(\vec{w})}{\partial w_{ij}} \qquad (1.16)$$

que : Tous les éléments sont donc en place pour nous permettre de définir l'algorithme de rétro- propagation du gradient.

III.7.3.2. L'Algorithme de rétro propagation

Pour écrire l'algorithme, nous allons simplifier quelques notations. Nous appelons δ_i la quantité $-\partial E/\partial y_i$. En utilisant les équations (1.8), (1.13), (1.15) et (1.16), nous obtenons les formules suivantes :
pour une cellule i de sortie, nous avons :

$$\delta_i = o_i(1-o_i)(c_i-o_i) \quad (2.1)$$

Pour une cellule i interne, nous avons :

$$\delta_i = o_i(1-o_i) \sum_{k \in Succ(i)} \delta_k w_{ki} \quad (2.2)$$

La modification du poids w_{ij} est alors définie par :

$$\Delta w_{ij} = \varepsilon x_{ij} \delta_i \quad (2.3)$$

Nous pouvons faire les remarques suivantes :

- La règle de modification des poids pour le perceptron linéaire est : $w_i \longrightarrow w_i + \varepsilon (c-o)x_i$. Dans le cas du PMC, cette règle est : $w_{ij} \longrightarrow w_{ij} + \varepsilon \delta_i x_{ij}$. Ces deux règles sont très similaires, le terme d'erreur c-oest remplacé par un terme plus compliqué δ_i,
- Pour une cellule i de sortie, la quantité δ_i correspond à l'erreur usuelle $c_i - o_i$ multipliée par la dérivée de la fonction sigmoïde,
- Pour une cellule i interne, le calcul de δ_i dépend de la somme pondérée deserreurs des cellules de la couche suivante,
- Après présentation de l'entrée $\vec{x}$ et calcul de la sortie $\vec{o}$, le calcul deserreurs δ_i sera effectué de la couche de sortie vers la couche d'entrée.

Il ne reste plus qu'à écrire l'algorithme.

Algorithme de rétropropagation du gradient :

Entrée : un échantillon S de $R^n \times R^p$: ε

un PMC avec une couche d'entrée C_0, q-1 couches cachées C_1, ..., C_{q-1},

une couche de sortie C_q, n cellules.

Initialisation aléatoire des poids w_i dans [-0.5,0.5] pour i entre 1 et n

Répéter

Prendre un exemple $(x^{\rightarrow}, c^{\rightarrow})$ de S et calculer $o^{\rightarrow}$

- - calcul des δ_i par rétropropagation

Pour toute cellule de sortie i $\delta_i \leftarrow o_i(1-o_i)(c_i-o_i)$ **finPour**

Pour chaque couche de q-1 à 1

Pour chaque cellule i de la couche courante

$\delta_i = o_i(1-o_i) \sum_{k \in Succ(i)} \delta_k w_{ki}$

finPour

finPour

- - mise à jour des poids

Pour tout poids $w_{ij} \leftarrow w_{ij} + \varepsilon \delta_i x_{ij}$ **finPour**

finRépéter

Sortie : Un PMC défini par la structure initiale choisie et les w_{ij}

Remarques

- L'algorithme de rétropropagation du gradient est une extension de l'algorithme de Widrow-Hoff. En effet, dans les deux cas, les poids sont mis à jour à chaque présentation d'exemple et donc on tend à minimiser l'erreur calculée pour chaque exemple et pas l'erreur globale.
- La méthode donne de bons résultats pratiques. Dans la plupart des cas, on rencontre peu de problèmes dus aux minima locaux, peut être grâce au fait que l'on minimise une erreur locale.
- Cependant, les problèmes de minima locaux existent. Pour améliorer l'algorithme vis à vis de ce problème, mais aussi pour essayer d'améliorer la vitesse de convergence, une variante couramment utilisée consiste à pondérer la modification des poids en fonction du nombre d'itérations déjà effectué. Plus formellement, on fixe une constante $\alpha \in [0,1[$ appelée moment (momentum) ; soit t un compteur du nombre d'itérations de la boucle principale, la règle de modification des poids devient :

$$w_{ij} \leftarrow w_{ij} + \Delta w_{ij}(t)$$
$$\Delta w_{ij}(t) = \varepsilon \delta_i x_{ij} + \alpha \Delta w_{ij}(t-1)$$

- L'intérêt de cette règle est de prendre en compte les modifications antérieures des poids dans le but d'éviter des oscillations perpétuelles.
- Le critère d'arrêt n'est pas précisé dans l'algorithme. Ce critère peut être : « arrêter dès que l'erreur estimée passe sous un seuil prédéfini ». On retrouve alors le problème toujours présent en apprentissage de la sur-spécialisation. En effet, le critère d'arrêt

dépend de l'erreur observée mesurée sur l'ensemble d'apprentissage et non de l'erreur réelle. Les méthodes pour palier à ce problème sont identiques à celles utilisées pour les arbres de décision. Par exemple, utiliser un ensemble test, quand c'est possible, pour estimer l'erreur réelle. Ou encore, utiliser des techniques d'élagage qui tentent de diminuer la taille du réseau.

- Le mode de présentation des exemples est également absent de notre algorithme. En règle générale, on choisit une présentation équitable.
- Le choix de l'architecture initiale du réseau reste un problème difficile. Ce choix peut être fait par l'expérience. Des méthodes dites « autoconstructives » existent : il s'agit d'ajouter des cellules au cours de l'apprentissage pour que l'apprentissage se fasse bien. Mais ces méthodes rencontrent souvent le problème de la sur-spécialisation. L'architecture peut aussi être choisie à l'aide de méthodes basées sur les algorithmes génétiques.
- Le modèle et l'architecture étant choisis, un problème est également de choisir les « bonnes » valeurs pour les paramètres e et a. Pour cela, on découpe l'ensemble d'apprentissage en un ensemble d'apprentissage, un ensemble de validation et un ensemble test. Lors de la phase d'apprentissage, on arrête périodiquement l'apprentissage, on estime l'erreur réelle sur l'ensemble de validation, on met à jour les paramètres e et a en fonction de la variation de cette erreur estimée. L'ensemble test sert à estimer l'erreur réelle à la fin de l'apprentissage.
- L'algorithme de rétropropagation du gradient a pu être étendu à certaines classes de réseaux récurrents pour lesquels il y a rétroaction. Ces réseaux sont très utiles pour la prédiction de séries temporelles.

III.8. Conclusion

Le développement qu’ont connu les réseaux de neurones artificiels a suscité un grand enthousiasme mais aussi des critiques. Beaucoup d’études comparatives comme celle effectuée dans cette thèse ont fournies une vue optimiste pour les réseaux de neurones artificiels, tandis que d’autres offrent une vue pessimiste. Pour de nombreuses tâches et problèmes, tel que la reconnaissance de formes, il n’y a pas une seule approche et qui pourrait dominer les autres. Le choix, le choix de la technique doit être en fonction de l’application en question, c’est pour ça qu’on doit comprendre et connaître les capacités, suppositions, la possibilité d’appliquer plusieurs approches dans différentes disciplines.

CHAPITRE II :Analyse et Conception

IV.1. Introduction

Avant l'implémentation et la réalisation de toute application informatique,il convient de suivre une démarche méthodologique pour arriver à la mettre en place.
Pour réaliser notre application, nous commençons par une analyse profonde et bien réfléchie, suivie d'une conception en se basant sur le formalisme UML.

IV.2. Présentation de l'UML
IV.2.1. Description

UML (UnifiedModelingLanguage ou Langage de modélisation unifié), est la notation standard qui s'est imposée pour la modélisation des systèmes informatiques.
Elle est née de la fusion de trois méthodes qui ont le plus influencé la modélisation objet au milieu des années 90 :l'OMT (Object Modeling Technique) de James Rumbaugh, Booch de GradyBooch, et OOSE (Object Orient Software Engineering) d'Ivar Jacobson.
L'UML permet de spécifier, de visualiser, de construire et de documenter l'ensemble des artefacts (graphisme) du système. Elle devient aujourd'hui un standard dans le domaine d'analyse et de conception orienté objet.

IV.3.Objectif de notre travail

L'objectif principal de notre travail est de réaliser un système de reconnaissance de mots manuscrits arabes anciens en utilisant les réseaux de neurones.

IV.4.Analyse

La phase d'analyse débute par la mise en évidence des différents acteursqui interviennent dans le système, et leurs besoins. Ensuite, modéliser les objectifs à atteindre, dans la phase de conception et ce en s'appuyant sur la phase d'analyse.

IV.4.1.Quelques définitions de base

- **Tâche**

Est l'ensemble logique d'opérations permettant l'exécution d'unprogramme ou d'une partie d'un programme.

- **Acteur**

Les acteurs d'un système sont les entités externes à ce dernier quiinteragissent (saisie de données, réception d'informations, …) avec lui. Les acteurs sont donc à l'extérieur du système et dialoguent avec lui. Ces acteurs permettent de cerner l'interface que le système va devoir offrir à son environnement.
Oublier des acteurs ou en identifier de faux conduit donc nécessairement à se tromper sur l'interface et donc la définition du système à produire.

- **Cas d'utilisation**

En anglais **use case**, permet de mettre en évidence les relations fonctionnelles entre les acteurs et le système étudié. Le format de représentation d'un cas d'utilisation est complètement libre mais UML propose un formalisme et des concepts issu de bonnes pratiques.
Le diagramme de cas d'utilisation permet de représenter visuellement la séquence d'actions réalisées par un système. Il est représenté par une boîte rectangulaire, produisant un résultat sur un acteur, appelé acteur principal, et ceci indépendamment de son fonctionnement interne.

- **Relation entre cas d'utilisation**

Qui propose trois types de relations standards :

- **Include**

Le cas d'utilisation incorpore explicitement et de manière obligatoire un autre cas d'utilisation à l'endroit spécifié.

- **Extend**

Le cas d'utilisation incorpore implicitement de manière facultative un autre cas d'utilisation à l'endroit spécifié.

- **Généralisation**

Les cas d'utilisation descendants héritent des propriétés de leurs parents.

- **Scénario**

Il représente une succession particulière d'enchainement qui s'exécute du début à la fin du cas d'utilisation, un enchainement étant l'unité de description séquence d'actions.
Un cas d'utilisation contient, en général, un scénario nominal et plusieurs scénarios alternatifs (qui se terminent d'une façon normale) ou d'erreurs (qui se terminent en échec).

IV.4.2. Identification des acteurs

- ✓ **Utilisateur**

C'est la personne qui désire reconnaitre des caractères (ou mots) manuscrits arabes anciens.

IV.4.3. Spécification des tâches

Les tâches effectuées sont résumées dans le tableau ci- dessous :

Acteur	Tâches
Utilisateur	T1 : Accéder à l'interface principale. T2 : Faire prétraitement. T3 : Faire apprentissage. T4 : Faire reconnaissance d'un caractère. T5 : Faire segmentation des lignes. T6 : Faire segmentation en caractères. T7 : Faire reconnaitre un mot. T8 :Déconnection.

Tableau IV.1 : Spécification des tâches.

IV.4.4. Spécification des scénarios pour l'utilisateur

Scénario

Succession particulière d'enchaînements, s'exécutant du début à lafin du cas d'utilisation.

Chacune des tâches effectuées par l'utilisateur est décrite par un ensemble de scénarios, ces scénarios sont illustrés dans le tableau ci-dessous.

Tâche	Scénarios
T1 : Accéder à l'interface principale.	S1 : Lancer l'application.
T2 : Faire prétraitement.	S2 :Appuyer sur le bouton « Prétraitement » sur l'interface principale. S3 : Sélectionner la lettre à prétraiter en appuyant sur le bouton « Sélection ». S4 : Appuyer sur le bouton « Lancement du prétraitement ».
T3 : Faire apprentissage.	S5 : Appuyer sur le bouton « apprentissage » sur l'interface principale. S6 :Saisir le nombre de neurones d'entrée. S7 : Saisirle nombre de neurones de sortie.

	S8 : Saisir le nombre de neurones dans la couche cachée. S9 : Saisir le pas d'apprentissage. S10 : Saisir le momentum. S11 : Cliquant sur le bouton « Input » et choisir l'image d'une lettre manuscrite. S12 : Sélectionner le type correspondant à la lettre manuscrite dans la Liste. S13 : Cliquer sur le bouton « Commencer l'apprentissage ». S14 : Sauvegarder les paramètres et résultats de l'apprentissage dans la base de données en cliquant sur le bouton « Ajouter à la base de données ».
T4 : faire reconnaissance d'un caractère.	S15 : Appuyer sur le bouton reconnaissance d'un caractère sur l'interface principale. S16 : Sélectionner l'image de la lettre manuscrite à reconnaitre en cliquant sur le bouton « Selection ». S17 : Entrainer la lettre en cliquant sur le bouton « Entrainer ». S18 : Reconnaitre la lettre en cliquant sur le bouton « Reconnaissance ».
T5 : faire segmentation des lignes.	S19 : Cliquer sur le bouton « Segmentation des lignes » dans l'interface Segmentation. S20 : Choisir l'image des lignes à segmenter en cliquant sur le bouton « Selection ». S21 : Choisir le répertoire de sortie en cliquant sur le bouton « Selection ». S22 : Extraire les lignes en cliquant sur le bouton « Extraction des lignes ».
T6 :faire Segmentation en caractères.	S23 : Cliquer sur le bouton « Segmentation des caractères » dans l'interface Segmentation. S24 : Choisir l'image du mot à segmenter en cliquant sur le bouton « Selection ». S25 : Choisir le répertoire de sortie en cliquant sur le bouton

	« Selection ». S25 : Saisir la taille en pixels des caractères de sortie dans le champ zone de texte. S26 : Extraire les caractères en cliquant sur le bouton « Extraction des caractères ».
T7 : Faire reconnaissance d'un mot.	S27 : Sélectionner l'image du mot à reconnaitre en cliquant sur le bouton « Selection ». S28 : Reconnaitre le mot en cliquant sur le bouton « Reconnaissance ».
T8 : Déconnection.	S29 : Se déconnecterde l'interface principale en cliquant sur le bouton «Quitter».

Tableau IV.2 : Spécification des scénarios pour l'utilisateur.

IV.5.Conception

Cette étape consiste à représenter des objets d'interface et de contrôle en page client et serveur.

IV.5.1. Diagramme de classes

Le diagramme de classes constitue un élément très important de la modélisation : il permet de définir de manière générale la structure statique d'un système : il ne permet en revanche pas de définir le nombre et l'état des instances individuelles. Néanmoins, on constate souvent qu'un diagramme de classes proprement réalisé permet de structurer le travail de développement de manière très efficace, il permet aussi, dans le cas de travaux réalisés en groupe (ce qui est pratiquement toujours le cas dans les milieux industriels), de séparer les composantes de manière à pouvoir répartir le travail de développement entre les membres de groupe.
Enfin, il permet de construire le système de manière correcte (**Build the system right**).

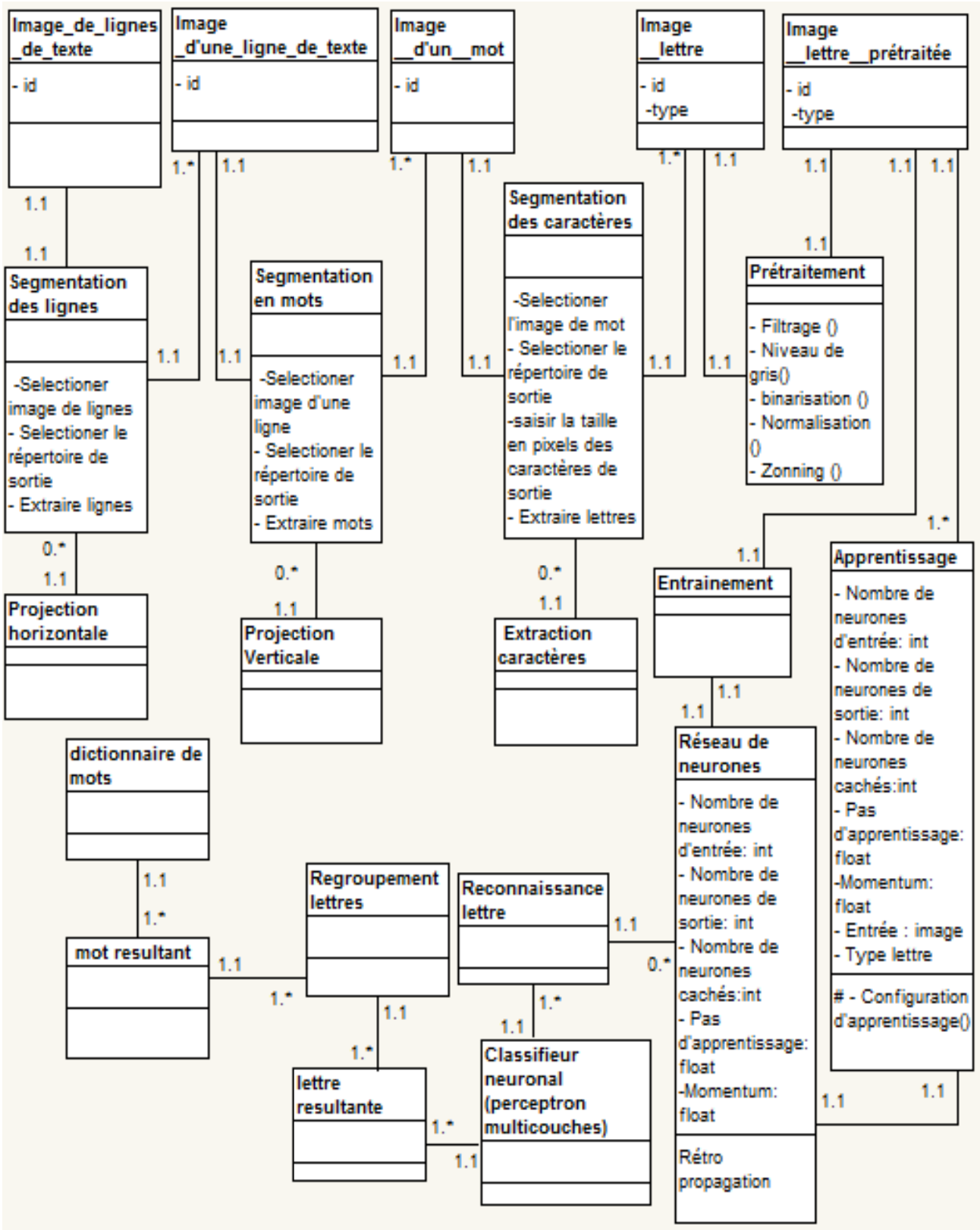


Figure IV.1 : Diagramme de classes.

IV.6. Description de notre système

Le système de reconnaissance de mots arabes manuscrits que nous avons développé se basesur la segmentation des images de texte en lignes ,puis la segmentation des lignes en mots ,et des mots en caractères, et sur les réseaux de neurones qui participent à l'apprentissage et la reconnaissance des caractères arabes manuscrits afin de reconnaitre tout le mot.

IV.6.1. Construction de la base d'apprentissage et de la base test des lettres

Dans le but de la construction de notre système de reconnaissance de mots arabes manuscrits, nous nous sommes intéressés à des images (scannées) de texte, traitant des versés coraniques manuscrits anciens. En effet, la qualité de ces images ainsi que le type de l'écriture qui les composent s'avèrenttrès complexe.

La figure suivante illustre un exemple de l'une de ces images :

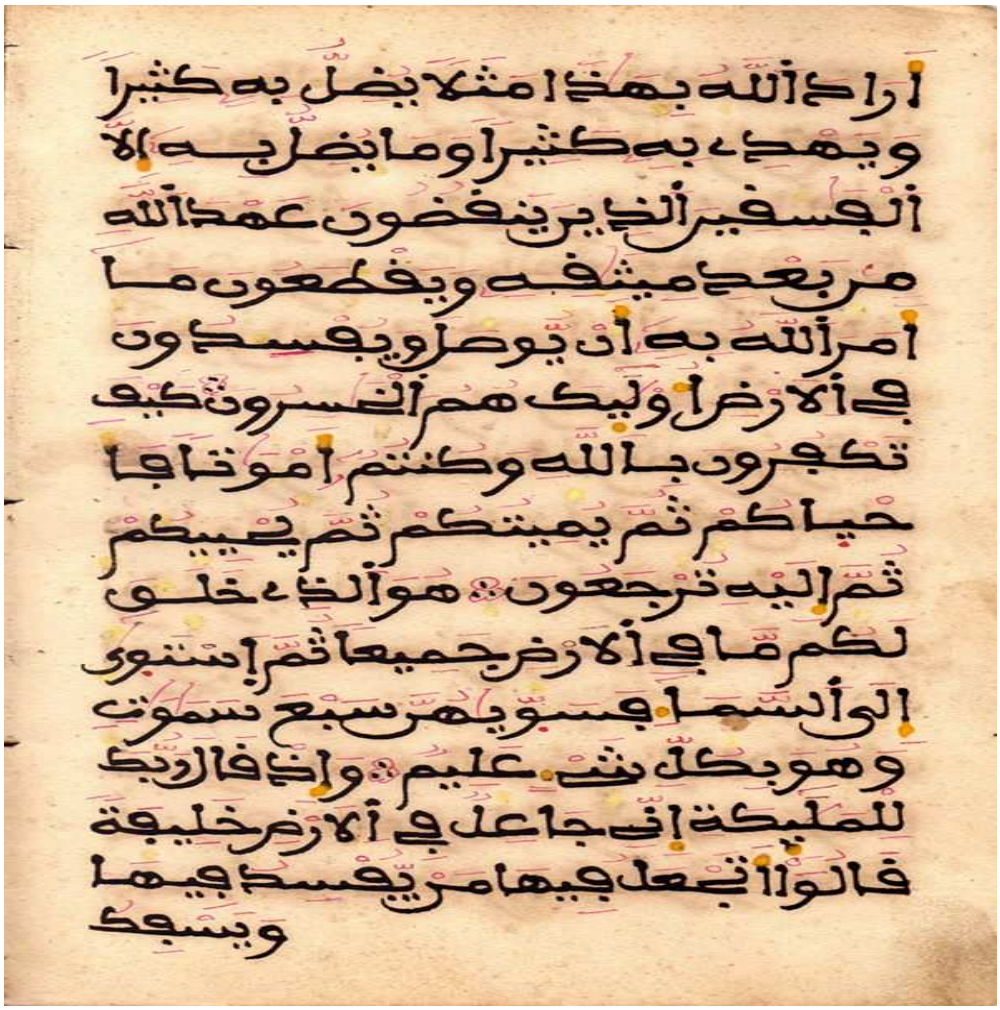


Figure IV.2 : Image de texte manuscrit arabe et ancien.

- **Quelques caractéristiques de l'écriture des images étudiées**

- L'espacement entre les différentes lignes n'étant pas régulier.
- Dans la plus part des lignes chaque caractère d'une même ligne est interrompu par une ou plusieurs extensions des caractères de la même ligne, **exemple** : ligne 2, 3, 5 et 11. Ou presque interrompu par une extension d'un caractère de la ligne en dessus ou en dessous, **exemple** :Entre les lignes 4, 5 et lignes 8, 9(de l'image présentée en haut).
- Espacements entre caractères des mots non régulier et souvent étroit.
- Adaptation d'une certaine syntaxe dans l'écriture de quelques caractères,

Exemples :

-La non inclusion de la « Hamza » habituelle dans cette écriture, notamment dans les deux caractères « alif Début » et « alif Fin », et le remplacement de celle-ci par un grand point jaune rempli.

- Les deux points diacritiques hauts habituels de la lettre « qa » sont remplacés par un seul.

- Le point diacritique haut habituel de la lettre « fa » est remplacé par un point diacritique bas.

- La suppression du point diacritique haut habituel de la lettre « na Fin » et « na Extrémité ».

- la non inclusion de la « senna » habituelle dans les lettres « ssa », « ddha » dans cette écriture.

- Ecriture des lettres « ya Fin », « ya Extrémité »avec une architecture très complexe et pas très facile à reconnaitre (même pas pour l'être humain), différente de l'architecture des lettres « ya Fin » et « yaExtrimité » habituelle.

- Al'aide de l'outil **« Paint »**, Nous avons segmenté des mots de ces images et avons extrait un ensemble de caractères pour chaque type des lettres de l'alphabet arabe. Puis, nous avons classé chaque lettre selon son **type** (son nom dans l'alphabet arabe et sa position dans un mot : début, milieu, fin ou isolée), sachant que l'alphabet arabe est constitué de 28 lettres différentes.

Ceci a donc donné lieu à la construction de notre base de caractères à présenter pour notre base d'apprentissage ainsi que notre base de caractères à conserver pour notre future base de test. Ces caractères sont classés dans 28 dossiers différents, chaque dossier contient des sous dossiers des différentes positions dans un mot pour la lettre correspondante.

Exemple

Pour la lettre **« alif »**, nous avons deux possibilités de son emplacement dans un mot :

- **« alif D »** pour « alif » au début d'un mot,

- **« alif F »** pour « alif » à la fin d'un mot,

- L'objectif de notre travail est la reconnaissance des mots extraient à partir des manuscrits arabes, notre système fonctionne donc en deux phases : l'apprentissage et la reconnaissance.

IV.6.2. Sous système d'apprentissage

Dans l'apprentissage de nos lettres manuscrites arabes, nous avons utilisé un apprentissage supervisé avec réseaux de neurones de type perceptron multicouches auquel nous avons appliqué l'algorithme de retro propagation du gradient, fonction de transfert sigmoïde.

- Mais avant l'étape de l'apprentissage, l'image de la lettre manuscrite doit d'abord

passer par un processus de prétraitement qui comporte quatre sous modules : un sous module de niveaux de gris et de binarisation qui permet de passer d'une image en niveaux de gris à une image binaire composée de deux valeurs 0 et 1. La sortie de ce sous module sera l'entrée du sous module d'encadrement pour localiser la lettre et enfin nous allons passer l'image de la lettre à un sous module de normalisation afin d'obtenir une lettre adaptée à une dimension fixée par le système. A la fin de ces quatre sous modules les primitives de l'image de la lettre prétraitée sont extraites à l'aide de la méthode de zoning.

1. Niveau de gris [89]

Le niveau de gris est la valeur de l'intensité lumineuse en un point. La couleur du pixel peut prendre des valeurs allant du noir au blanc en passant par un nombre fini de niveaux intermédiaires.

La représentation des images en niveaux de gris se fait en attribuant à chaque pixel de l'image une valeur correspondant à la quantité de lumière renvoyée. Cette valeur peut être comprise entre 0 et 255 **[83]**.

Pour ce faire on peut utiliser la formule suivante :

$$Gris = \frac{Rouge + Vert + Bleu}{3} \quad (1)$$

1.1. Algorithme de transformation d'une image en niveau de gris

Ouvrir l'image 1
Créer une image résultat de même taille que l'image 1.
Pour chaque colonne faire
Pour chaque ligne faire
Extraire les canaux Rouge, Vert et Bleu du pixel de l'image 1.
Calculer la valeur de niveau de gris correspondante (1).
Ecrire la valeur de l'entier RGB dans le pixel de l'image résultat.
Fin pour
Fin pour

2. Binarisation [89]

La binarisation est effectuée dans le but de simuler le processus d'impression à l'encre noir sur du papier blanc. C'est une opération qui produit deux classes de pixels, en général, ils sont représentés par des pixels noirs et des pixels blancs.
Ainsi les pixels correspondant à des points élevés doivent être binarisés en noire (valeur=0) et ceux dans les creux doivent être binarisés en blanc (valeur=1).
Il existe plusieurs algorithmes de binarisation. Nous avons utilisé l'algorithme de binarisation Seuillage globale fixe **[84]** ;
C'est une technique très simple, elle consiste à comparer le niveau de gris de chaque pixel $\mathbf{x_i}$ de l'image avec un seuil global fixe **T** (par exemple 127). On note $\mathbf{b_i}$la nouvelle valeur du pixel, le seuillage est donné par l'expression suivante :

$$b_i = 255 \text{ si } x_i >= T \text{ et } b_i = 0 \text{ si } x_i < T$$

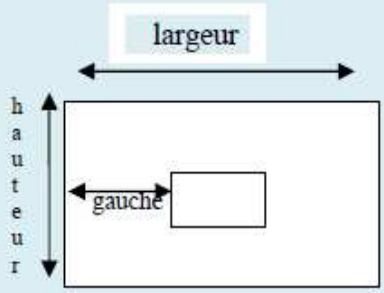
2.1.Algorithme de binarisation

```
image1 : image d'entrée, Image2 :image de sortie
Pour i de 1 à largeur image1 Faire
        Pour j de 1 à hauteur image1 Faire
         pixel = Image1.pixel (i, j)
         SI pixel < seuil ALORS
                val=0
         sinon
                val=255
         fin si
         Image2 .MettrePixel (i, j, val)
        Fin pour
Fin pour
Fin.
```

Le niveau de gris de chaque pixel de l'image initiale est comparé à un seuil. S'il est inférieur à ce seuil, on met en noir le pixel qui lui correspond dans l'image résultat (noir et blanc), sinon il est mis en blanc.

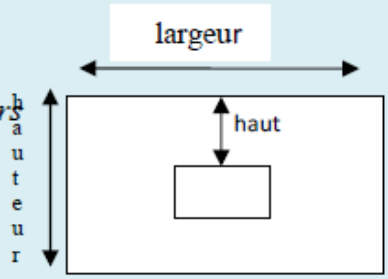
3. L'encadrement [89]

L'encadrement est le processus de localisation de la lettre, il permet de définir les coordonnées de la lettre dans l'image. Pour cela nous avons réalisé des algorithmes qui permettent de donner les propriétés suivantes : haut, bas, gauche, droite, afin de passer à l'encadrement de la lettre.
Les algorithmes utilisés sont représentés ci-dessous :
Initialisation haut = -1, gauche = -1 ; droit = -1 ; bas = -1 ;
Entierindiced, indiceb;
Image img;

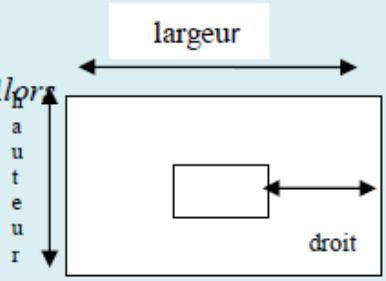



```
// A gauche
Pour  x = 0 à x < img.largeur  faire
      Pour y = haut à y < img.hauteur faire
          // recuperer  la couleur de chaque pixel
          Si ((img.getRGB(x, y) != blanc) et (img.getRGB(x, y) != -1)) Alors
               gauche=x;
               sortir de la boucle ;
          FinSi
     Fin pour
     Si gauche!=-1 Alors
        Sortir de la boucle
     FinSi
Fin pour
```

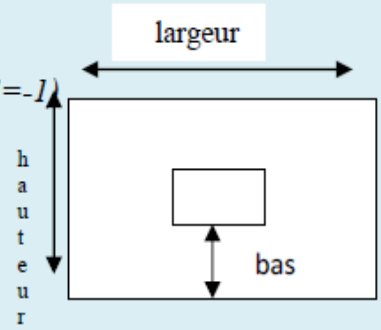
```
// En hauteur
Pour j = 0 à j < img.hauteur  faire
   Pour i = 0 à  i <  img.largeur faire
    // recuperer couleur de chaque pixel
  Si (img.getRGB(i, j) !=blanc) et (img.getRGB(i, j) !=-1) Alors
                        haut=j;
                        Sortir de la boucle;
Fin Si
Fin Pour
Si  haut !=-1 Alors
      Sortir de la boucle
FinSi
FinPour
```




```
// A Droite
Pour  x =img.largeur-1; x >= gauche, x++  faire
  Pour  y = img.getHeight()-1; y >= haut; y-- faire
      // recuperer la couleur de chaque pixel
      Si (img.getRGB(x, y) !=blanc) et (img.getRGB(x, y) !=-1) Alors
                        Indiced=x ;
                        droit =img.largeur - x;
                        Sortir de la boucle
      Fin Si
    Fin Pour
     Si (droit!=-1) Alors
          Sortir de la boucle
     Fin Si
Fin Pour.
```



```
//En bas
Pour y = img.hauteur-1à y >haut Faire
    Pour x =0 à x <img.largeur  Faire
        // recuperer couleur de chaque pixel
        Si (img.getRGB(x, y) !=blanc) et (img.getRGB(x, y) !=-1)
            Indiceb=y ;
            bas=img.hauteur-y;
            sortir de la boucle;
        Fin Si
Fin Pour
   Si bas!=-1  Alors
            Sortir de la boucle
  Fin Si
Fin Pour
```




```
//L'encadrement
Entiers w=0, h=0;

Pour col = gauche à col < indiced  faire

  Pour row = haut à row < indiceb faire

      // Recopie de l'image 1 dans l' image résultant

       img2. setRGB( w, h , imgb.getRGB( col , row )) ;

            Incrémentation de h;

   Fin Pour

  Incrémentation de w;

  h=0;

Fin Pour.
```

Ces algorithmes permettent donc de localiser les pixels noirs dans une image contenant seulement une lettre ou un mot en parcourant toute l'image. Donc cette phase est vraiment la plus intéressante dans notre sous système de prétraitement, puisqu'elle nous offre un gain de temps que ce soit dans l'apprentissage ou dans la reconnaissance du mot et ceci par le traitement de l'image encadrée résultante.

La figure suivante montre un exemple de l'encadrement d'une lettre dans une image.

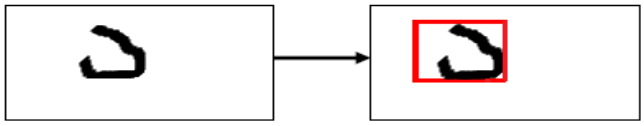


Figure IV.3 : Exemple d'encadrement d'une lettre dans une image.

4. Normalisation [89]

La normalisation consiste à transformer la taille de l'image et l'adapter à une dimension fixée à priori par le réalisateur de système. Pour cela nous avons proposés une procédure qui permet de normaliser l'image encadrée dans une dimension de 64x64 pixels. Cette procédure copie le contenu d'une première image pixel par pixel et la copie dans une seconde image rétrécie si l'image dépasse 64x64, sinon elle sera agrandie.

Nous avons utilisé une échelle qui sera calculée automatiquement en fonction de la dimension de la lettre encadrée dans l'image par rapport à la dimension 64*64 pixels.

Alors, on note les différentes échelles : Echelle<1 : Pour agrandir,

Echelle >1 : Pour rétrécir,
Echelle =1 : Pour garder la même taille,

4.1. L'algorithme de normalisation

```
Procédure Normalisation (Image1 : Bitmap, Image2 : Bitmap, Echelle : Réel): Bitmap
Variable  X, Y: Réel ;
          c : entier ;
          i, j : Réel ;
          Hauteur, Largeur : Réel;
Début
   Hauteur = bas - haut
   Largeur= droit- gauche
  EchelleH = Hauteur / Image2 .hauteur
  EchelleW = Largeur / Image2 .largeur
 j = 0 ;
 Pour  y de haut à  bas (pas de EchelleH) Faire
   i = 0
  Pour x de droit à gauche (pas de EchelleW) Faire
       c = Image1. Point (x, y)
       Image2 . PSet (i, j ) , c
        i = i + 1
  Fin Pour
  j = j + 1 ;
 Fin Pour
Fi
```

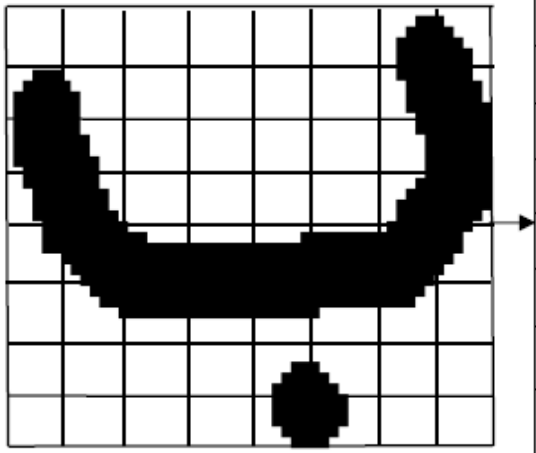
5. Extraction des primitives [89]

Après les prétraitements sur l'image de la lettre manuscrite, nous nous intéressons à l'étape de sa caractérisation (extraction des primitives), qui a pour objectif de déterminer le vecteur représentatif de cette image. Ce vecteur servira à représenter de manière compacte la séquence des formes (la lettre) contenues dans l'image.

Dans notre système, nous extrayons plutôt des primitives de type statistiques qui conviennent pour l'utilisation des réseaux de neurones. A partir de l'image normalisée, nous tirons des primitives du type **zoning [85] [86] [87].**

La méthode zoning consiste à partitionner une image en M sous-images de mêmes tailles appelées zones (8x8). Nous calculons pour chaque zone le pourcentage de pixels noir en obtenant une matrice de pourcentage qui représente le vecteur de primitives dont les valeurs

seront normalisées entre l'intervalle 0 et 1 puis transformé en vecteur ligne, celui-ci sera utilisé pour l'apprentissage et la reconnaissance.
La figure suivante illustre le principe de zoning pour la lettre arabe (ba) :



0%	0%	0%	0%	0%	0%	74%	45%
85%	30%	0%	0%	0%	0%	60%	96%
96%	30%	0%	0%	0%	0%	40%	100%
70%	85%	2%	0%	0%	0%	60%	94%
15%	94%	97%	95%	97%	100%	98%	35%
0%	15%	47%	47%	47%	43%	25%	0%
0%	0%	0%	0%	47%	30%	0%	0%
0%	0%	0%	0%	47%	36%	0%	0%

Figure IV.4 : Représentation de la lettre arabe (ba) sous forme d'une matrice.

- Mais comme dit en haut, l'apprentissage se fera avec les réseaux de neurones avec une fonction d'activation sigmoïde et comme les fonctions sigmoïdes fonctionnent seulement avec des valeurs comprises entre l'intervalle 0 et 1, alors nous avons normalisé notre matrice de pixels noirs à cet intervalle(cela a été réalisé sous matlab en appliquant les deux formules « **A=A-min(A(:)) »**et **« A=A/max(A(:)) »** ; telle que A est notre matrice représentative du pourcentage de pixels noirs pour une lettre prétraitée, déjà déclarée) , et comme les réseaux de neurones prennent des données sous forme de vecteurs lignes pour chaque donnée étudiée (des matrices de pourcentage de pixels noirs correspondant aux images prétraitées des lettres manuscrites arabes dans notre cas), ceci nécessite donc la transformation de notre matrice normalisée (pour chaque image d'un caractère) en vecteur ligne(nous avons aussi réalisé cela sous matlab en appliquant laformule « **r=reshape(A',1,64)** » tel que r est une variable quelconque) .

Remarque :
Le traitement de quelques images (mais de chaque type) de lettres de notre base de lettres d'apprentissage, associées aux images de lettres imprimées (idéales) correspondant à chaque image de lettre manuscrite; en passant par toutes les étapes que nous venant de citer nous a conduit à construire notre base d'apprentissage que nous devant passer comme entrées et sorties désirées aux réseaux de neurones, quant aux images de lettres restantes de notre vocabulaire(le reste des caractères que nous avons extrait à partir des images de texte considérées), elle seront utilisées comme une base de test.

- Dans la phase d'apprentissage, nous avons suivi les techniques d'implémentation et d'apprentissage proposées par**Encog 2.3.0**[1]**,** en important la librairie **encog-core-2.3.0**qui contient différents packages nécessaires à la modélisation de notre réseau de neurones**.**
- On a commencé par la récupération du paquet Encog, et on ajoutant les directives suivantes :

```
12  import org.encog.neural.networks.BasicNetwork;
13  import org.encog.neural.networks.layers.BasicLayer;
14  import org.encog.neural.networks.training.Train;
15  import org.encog.neural.networks.training.propagation.back.Backpropagation;
16  import org.encog.neural.activation.ActivationSigmoid;
17  import org.encog.neural.data.basic.BasicNeuralDataSet;
18  import org.encog.util.logging.Logging;
19  import org.encog.neural.data.NeuralData;
20  import org.encog.neural.data.NeuralDataPair;
21  import org.encog.neural.data.NeuralDataSet;
```

Nous avons implémenté un réseau de neurones de type perceptron multicouches (car d'après les chercheurs, c'est le type de réseau le mieux adapté à la reconnaissance et donnant des résultats plus performant), constitué :

- **D'une couche d'entrée :** dont le nombre de neurones est variable, mais à 64 neurones dans notre cas, correspondant au nombre d'entrées (vecteur ligne de 64 éléments).
- **D'une seule couche cachée à fonction d'activation sigmoïde :** dont le nombre de neurones reste variable.
- **D'une couche de sortie à fonction d'activation sigmoïde :** dont le nombre de neurones reste variable.

```
602
603             BasicNetwork network = new BasicNetwork();
604             network.addLayer(new BasicLayer(new ActivationSigmoid(), true,nbi));
605             network.addLayer(new BasicLayer(new ActivationSigmoid(), true, nbc));
606             network.addLayer(new BasicLayer(new ActivationSigmoid(), true,nbs));
607             network.getStructure().finalizeStructure();
608             network.reset();
```

On a appliqué à ce réseau la méthode de décente du gradient d'erreur qui fait appel à l'algorithme de rétro propagation du gradient,cet algorithme utilise deux paramètres :

```
613  final Train train = new Backpropagation(network, trainingSet,pasap,errmin);
```

[1]Framework de machine à apprendre pour java et .NET, documentée en anglais, elle supporte et implémente des bibliothèques destinées à la création des réseaux de neurones et à leur utilisation dans différents domaines (classification, reconnaissance, clustering…), développée par l'excellent Jeff Heaton.

- **Le pas d'apprentissage :**

C'est un pourcentage qui directement détermine comment la décente du gradient doit être appliquée à la matrice des poids. Le gradient est multiplié par le pas d'apprentissage puis y est ajouté à la matrice des poids. Ceci optimise lentement les poids à des valeurs qui produisent une erreur minimale.
Sa valeur varie entre 0.1 et 0.9.

- **Le momentum (minimum local de l'erreur) :**

Sa valeur varie entre 0.1 et 0.9, il permet d'améliorer la vitesse de convergence.

- La base d'apprentissage a été construite, en effectuant les opérations de prétraitement, zoning, normalisation des valeurs entre 0 et 1 et la transformation de la matrice de pourcentage de pixels noirs pour quelques images (mais de chaque type) de lettre manuscrite de notre vocabulaire, cela nous a donné les entrées que nous devons fournir au réseau de neurones, et l'application de ces même opérations aux même type d'images de lettres mais imprimées (caractère idéal pour chaque caractère manuscrit) nous a donné les sorties désirées que nous devons fournir au réseau de neurones. De cette façon on serai passéà l'entrainement du réseau de neurones en mettant en place un objet « BasicNeuronalDataSet », son constructeur nécessite deux tableaux, un pour les entrées et un pour les sorties idéales.

Voici une capture d'écran qui montre une petite partie de la base d'apprentissage:

```
80    public static  double carInput[][] =
81
82    {
83     //alif D
84     {0.0,0.0,0.0,0.3900,0.6200,0.3900,0.0,0.0,0.0,0.0,0.0,0.0,0.3900,1.0000,0.9500,0.3200,0.0,0.0,0.0,0.0,0.0,0.0,0.6700,0.6200,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.9000,0.230
85     {0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.4500,0.9200,0.9500,0.8200,0.0900,0.0,0.0,0.0,0.0,0.0,0.0700,0.3900,1.0000,0.8100,0.0400,0.0,0.0,0.0,0.0,0.0,0.4000,1.0000
86     {0.0,0.4200,0.6200,0.5900,0.3100,0.0,0.0,0.0,0.0,0.0,0.5100,0.7500,1.0000,1.0000,0.4300,0.0,0.0,0.0,0.0,0.0,0.0,0.0,1.0000,0.8100,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,1.0000,0.2
87     {0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.4667,0.7444,0.4333,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.1556,0.9444,0.9000,0.0,0.0,0.0,0.0,0.0,0.0,0
88     {0.0,0.0,0.1800,0.2500,0.0600,0.0,0.0,0.0,0.0,0.0,0.6400,1.0000,1.0000,0.9500,0.4200,0.0,0.0,0.0,0.0,0.0,0.0,0.6200,1.0000,0.4600,0.0,0.0,0.0,0.0,0.0,0.0,0.6200,1.0
89     {0.0,0.0400,0.2300,0.2500,0.2500,0.0600,0.0,0.0,0.0,0.0,0.3200,0.8500,1.0000,1.0000,0.9600,0.2800,0.0,0.0,0.0,0.0,0.0,0.0,0.6200,1.0000,0.9300,0.1800,0.0,0.0,0.0,0.0,0.
90     {0.0,0.0700,0.3700,0.3700,0.0700,0.0,0.0,0.0,0.0,0.0,0.5600,1.0000,1.0000,1.0000,0.4200,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.8100,0.8500,0.0400,0.0,0.0,0.0,0.0,0.0,0.0,0.9500,
91     {0.0,0.1400,0.3700,0.3700,0.2300,0.0,0.0,0.0,0.0,0.0,0.2300,0.8900,1.0000,1.0000,0.6500,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.6200,1.0000,0.3200,0.0,0.0,0.0,0.0,0.0,0.0,0.6200,
92     {0.0,0.0,0.0,0.0,0.1200,0.0700,0.0,0.0,0.0,0.0,0.0,0.0,0.6500,0.9300,1.0000,1.0000,0.3700,0.0,0.0,0.0,0.0,0.0,0.0,0.1000,0.8500,1.0000,0.1800,0.0,0.0,0.0,0.0,0.0,0.0,0.0,1.0000,0.4
93     {0.1500,0.4800,0.6200,0.6200,0.5300,0.0600,0.0,0.0,0.0,0.0300,0.5400,0.8100,1.0000,1.0000,0.5300,0.0,0.0,0.0,0.0,0.0,0.0,0.7500,1.0000,0.2500,0.0,0.0,0.0,0.0,0.0,0.
94     {0.0,0.0400,0.5600,0.4500,0.6400,0.2300,0.0,0.0,0.0,0.0,0.1800,0.8200,1.0000,1.0000,0.9800,0.3200,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.6500,0.9300,0.1800,0.0,0.0,0.0,0.0,0.0,0
95    //alif F
96     {0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0306,0.2041,0.1735,0.0,0.0,0.0,0.0,0.0,0.0,0.4592,1.0000,0.9694,0.0714,0.0,0.0,0.0,0.0,0.0,0.0,0.6327,0.6939,0.0,0.0,0.0,0.0,
97     {0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.1458,0.9688,1.0000,0.2604,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.5833,0.7917,0.0104,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.5208,0.5938,0.0,0.0,0.0,0.0,
98     {0.0,0.0,0.0,0.1563,0.4792,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.9271,1.0000,0.0417,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.2708,0.8438,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.1250,0.8750,0.0,0.0,0.0,0.0,
99     {0.0,0.0400,0.0400,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.8100,1.0000,0.3100,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.3700,0.7800,0.5600,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.2800,0.2800,0.0,0.0,0.0,0.0
100    // ba D
101    {0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.2900,0.2600,0.0300,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.9600,0.9600,0.7300,0.0,0.0,0.0100,0.1200,0.1000,0.0400,0.0700,0.
102    {0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.2347,1.0000,0.5714,0.0306,0.0,0.0,0.0,0.0,0.0,0.0,0.0102,0.2653,0.8571,0.4082,0.0,0.3469,0.3571,0.2143,0.2551,0.3
103    // ta D
104    {0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0, 0.0900,0.2300,0.1000,0.7300,0.0300,0.0,0.0,0.0,0.0,0.5100,0.6500,0.0300,0.4000,0.0600,0.0,0.0,0.0,0.0,0.0,0.0900,0.100
105    {0.0,0.0,0.0,0.0,0.0,0.1400,0.0400,0.0,0.0,0.0,0.0,0.0,0.9000,0.0,0.4000,0.9500,0.0900,0.0,0.0,0.0,0.0,0.5600,0.1800,0.0,0.3500,0.0400,0.0,0.0,0.0,0.0,0.0900,0.6200,0.4
106    {0.0,0.0,0.3400,0.0600,0.2900,0.1700,0.0,0.0,0.0,0.0,0.6500,0.4300,0.5100,0.3200,0.0,0.0,0.0,0.0,0.0,0.0900,0.3100,0.1500,0.0,0.0,0.0,0.0,0.0,0.0,0.2800,0.8100,
107    {0.0,0.0,0.0,0.0,0.0,0.0,0.1500,0.0,0.0,0.0,0.0,0.0,0.4600,0.5600,0.0,0.8700,0.4000,0.0,0.0,0.0,0.0,0.1500,0.4300,0.0,0.1800,0.2800,0.0,0.0,0.0,0.0,0.0,0.0,0.5600,0.560
108    //ta E
109    {0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.1400,0.1400,0.0,0.0,0.0900,0.0900,0.0,0.2600,0.3700,0.5000,0.0,0.2500,0.9500,0.6200,0.0,0.6200,0.4200,0.4200,0.2300,1.0000,0.
```

```java
    public static double  carIdeal[][] =


    {
   //alif D bon
    {0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.1957,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.1957,0.0,0.0,0.0,0.0,0.0,0.0,0.1957
    {0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.1957,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.1957,0.0,0.0,0.0,0.0,0.0,0.0,0.1957
    {0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.1957,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.1957,0.0,0.0,0.0,0.0,0.0,0.0,0.1957
    {0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.1957,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.1957,0.0,0.0,0.0,0.0,0.0,0.0,0.1957
    {0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.1957,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.1957,0.0,0.0,0.0,0.0,0.0,0.0,0.1957
    {0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.1957,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.1957,0.0,0.0,0.0,0.0,0.0,0.0,0.1957
   //alif F bon
    {0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.8000,0.0933,0.0,0.0,0.0,0.0,0.0,0.0,1.0000,0.1600,0.0,0.0,0.0,0.0,0.0,0.0,0.0,1.00
    {0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.8000,0.0933,0.0,0.0,0.0,0.0,0.0,0.0,1.0000,0.1600,0.0,0.0,0.0,0.0,0.0,0.0,0.0,1.00
    {0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.8000,0.0933,0.0,0.0,0.0,0.0,0.0,0.0,1.0000,0.1600,0.0,0.0,0.0,0.0,0.0,0.0,0.0,1.00
    {0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.8000,0.0933,0.0,0.0,0.0,0.0,0.0,0.0,1.0000,0.1600,0.0,0.0,0.0,0.0,0.0,0.0,0.0,1.00
  // ba D bon
    {0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.3086,0.6543,0.0,0.0,0.0,0.0,0.0,0.0,0.3086,0.7654,0.0,0.9259,0.925
    {0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.3086,0.6543,0.0,0.0,0.0,0.0,0.0,0.0,0.3086,0.7654,0.0,0.9259,0.925
  // ta D bon
    {0.0,0.0,0.0,0.0,0.0100,0.1800,0.2800,0.0100,0.0,0.0,0.0,0.0,0.0,0.0600,0.6500,0.5600,0.0600,0.0,0.0,0.0,0.0,0.0,0.0,0.3200,0.0,0.0,0.0,0.0,0.0,0.0,0.120
    {0.0,0.0,0.0,0.0,0.0100,0.1800,0.2800,0.0100,0.0,0.0,0.0,0.0,0.0,0.0600,0.6500,0.5600,0.0600,0.0,0.0,0.0,0.0,0.0,0.0,0.3200,0.0,0.0,0.0,0.0,0.0,0.0,0.120
    {0.0,0.0,0.0,0.0,0.0100,0.1800,0.2800,0.0100,0.0,0.0,0.0,0.0,0.0,0.0600,0.6500,0.5600,0.0600,0.0,0.0,0.0,0.0,0.0,0.0,0.3200,0.0,0.0,0.0,0.0,0.0,0.0,0.120
    {0.0,0.0,0.0,0.0,0.0100,0.1800,0.2800,0.0100,0.0,0.0,0.0,0.0,0.0,0.0600,0.6500,0.5600,0.0600,0.0,0.0,0.0,0.0,0.0,0.0,0.3200,0.0,0.0,0.0,0.0,0.0,0.0,0.120
  // ta E bo
    {0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0323,0.0,0.0,0.0,0.3011,0.4194,0.0,0.0968,0.3978,0.0645,0.0645,0.0,0.4946,0.4624,0.0,0.182
  // ta F bon
    {0.0,0.0,0.0,0.1400,0.1400,0.0,0.0,0.0,0.0,0.0,0.3100,0.7600,0.3200,0.0,0.0,0.0,0.0,0.0,0.3100,0.1800,0.0,0.0,0.0,0.0,0.0900,0.0,0.0,0.0,0.0,0.2100,0.0,0.0,(
    {0.0,0.0,0.0,0.1400,0.1400,0.0,0.0,0.0,0.0,0.0,0.3100,0.7600,0.3200,0.0,0.0,0.0,0.0,0.0,0.3100,0.1800,0.0,0.0,0.0,0.0,0.0900,0.0,0.0,0.0,0.0,0.2100,0.0,0.0,(
    // ta M bon
```

- Enfin, on effectue l'entrainement du réseau.

```java
                  int epoch = 1;

                  do {
 train.iteration();
  jTextArea1.setText (jTextArea1.getText () +"\nEpoch #" + epoch + " \n,Error:" + train.getError()+"      " );
 System.out.println("\nEpoch #" + epoch + " \n,Error:" + train.getError());
                          epoch++;
                  } while(train.getError() > 0.01);
```

- Pour l'affichage des sorties fournies :

```java
                System.out.println("Neural Network Results:");


                for(NeuralDataPair pair: trainingSet ) {

final NeuralData output = network.compute(pair.getInput());

 // jTextArea4.setText (jTextArea4.getText  () +pair.getInput().getData(0) + "," + pair.getInput().getData(1)+ pair.getInput().getData(2)+"," +", actual=" +
 jTextArea4.setText (jTextArea4.getText  () +pair.getInput().getData(0)+","+pair.getInput().getData(1)+","+pair.getInput().getData(2)+","+pair.getInput().ge
System.out.println(pair.getInput().getData(0)+","+pair.getInput().getData(1)+","+pair.getInput().getData(2)+","+pair.getInput().getData(3)+","+ pair.getInp
//System.out.println(pair.getInput().getData(0) + "," + pair.getInput().getData(1)+ ", actual=" + output.getData(0) + ",ideal=" + pair.getIdeal().getData((

                }
```

- On fournit au réseau de neurones une image de lettre de notre vocabulaire, (quisera bien-sûr soumise au prétraitement et zoning habituels, normalisée et transformée en vecteur ligne par l'intermédiaire de l'objet de normalisation automatisée

« NormalizedField » d'encog, et on sélectionne son type dans la liste pour apprendre la lettre désignée par l'image.

- Le résultat de l'apprentissage nous donne l'erreur minimale de chaque itération et les sorties fournies pour chaque élément de la donnée en entrée, si les sorties fournies s'approchent bien des sorties désirées, on dira alors que la lettre a bien été apprise.

Remarque :
Les réseaux de neurones ne sont pas entrainés pour donner des résultats exacts, mais ils sont plutôt entrainés pour une erreur de 1%, donc chacun des résultats sera fourni avec 1% moins que sa sortie attendue.Puisque le réseau de neurones est initialisé à des valeurs aléatoires, alors une seconde exécution avec les mêmes paramètres du réseau donnera des résultats différents.

- Après avoir eu les résultats de l'apprentissage, ils seront sauvegardés dans la base de données ainsi que les paramètres du réseau et le type de la lettre sélectionné, tous sur la même ligne.

IV.6.2.1. La base de données de notre système

La base de données utilisée est construite à partir des lettres arabes imprimées, chacune selon ses différentes formes qu'elle peut prendre dans un mot (début, milieu, fin, isolée), ainsi que les paramètres du réseau neuronal créé pendant l'apprentissage de chaque lettre manuscrite correspondant à chaque type lettre imprimée et bien-sûr des résultats de l'apprentissage de chaque lettre.

Id_lettre	Nombre_de neurone_d'entrée	Nombre_deneurone_de sortie	Nombre_deneurone_dans_la_couche_cachée	Pas d'apprentissage	Momentum	Erreur local de chaque iteration	Erreur actual de sortie fournie	Type lettre
1	64	variable	Variable	Entre 0.1 et 0.9	Entre 0.1 et 0.9	Errlocal1	ErreurAcSor1	أ
2	64	Variable	Variable	Entre 0.1 et 0.9	Entre 0.1 et 0.9	Errlocal2	ErreurAcSor 2	ـأ
3	64	Variable	Variable	Entre 0.1 et 0.9	Entre 0.1 et 0.9	Errlocal3	ErreurAcSor 3	ب
4	64	variable	Variable	Entre 0.1 et 0.9	Entre 0.1 et 0.9	Errlocal4	ErreurAcSor 4	ـب
5	64	variable	Variable	Entre 0.1 et 0.9	Entre 0.1 et 0.9	Errlocal5	ErreurAcSor 5	بـ
6	64	variable	Variable	Entre 0.1 et 0.9	Entre 0.1 et 0.9	Errlocal6	ErreurAcSor 6	ـبـ
7	64	variable	Variable	Entre 0.1 et 0.9	Entre 0.1 et 0.9	Errlocal7	ErreurAcSor 7	ت
8	64	variable	Variable	Entre 0.1	Entre	Errloca	ErreurAcS	ـت

				et 0.9	0.1 et 0.9	l8	or 8	
9	64	variable	Variable	Entre 0.1 et 0.9	Entre 0.1 et 0.9	Errloca l9	ErreurAcS or 9	ذ
10	64	variable	Variable	Entre 0.1 et 0.9	Entre 0.1 et 0.9	Errloca l 10	ErreurAcS or 10	ذ
11	64	variable	Variable	Entre 0.1 et 0.9	Entre 0.1 et 0.9	Errloca l 11	ErreurAcS or 11	ج
12	64	variable	variable	Entre 0.1 et 0.9	Entre 0.1 et 0.9	Errloca l 12	ErreurAcS or 12	ج
13	64	variable	Variable	Entre 0.1 et 0.9	Entre 0.1 et 0.9	Errloca l 13	ErreurAcS or 13	ج
14	64	variable	Variable	Entre 0.1 et 0.9	Entre 0.1 et 0.9	Errloca l 14	ErreurAcS or 14	ج
15	64	variable	Variable	Entre 0.1 et 0.9	Entre 0.1 et 0.9	Errloca l 15	ErreurAcS or 15	ج
16	64	variable	Variable	Entre 0.1 et 0.9	Entre 0.1 et 0.9	Errloca l 16	ErreurAcS or 16	ج
17	64	variable	Variable	Entre 0.1 et 0.9	Entre 0.1 et 0.9	Errloca l 17	ErreurAcS or 17	ح
18	64	variable	Variable	Entre 0.1 et 0.9	Entre 0.1 et 0.9	Errloca l 18	ErreurAcS or 18	ح
19	64	variable	Variable	Entre 0.1 et 0.9	Entre 0.1 et 0.9	Errloca l 19	ErreurAcS or 19	خ
20	64	variable	Variable	Entre 0.1 et 0.9	Entre 0.1 et 0.9	Errloca l 20	ErreurAcS or 20	خ
21	64	variable	Variable	Entre 0.1 et 0.9	Entre 0.1 et 0.9	Errloca l 21	ErreurAcS or 21	خ
22	64	variable	Variable	Entre 0.1 et 0.9	Entre 0.1 et 0.9	Errloca l 22	ErreurAcS or 22	خ
23	64	variable	Variable	Entre 0.1 et 0.9	Entre 0.1 et 0.9	Errloca l 23	ErreurAcS or 23	د
24	64	variable	Variable	Entre 0.1 et 0.9	Entre 0.1 et 0.9	Errloca l 24	ErreurAcS or 24	د

25	64	variable	Variable	Entre 0.1 et 0.9	Entre 0.1 et 0.9	Errlocal 25	ErreurAcSor 25	ذ
26	64	variable	Variable	Entre 0.1 et 0.9	Entre 0.1 et 0.9	Errlocal 26	ErreurAcSor 26	ـذ
27	64	variable	Variable	Entre 0.1 et 0.9	Entre 0.1 et 0.9	Errlocal 27	ErreurAcSor 27	ر
28	64	variable	Variable	Entre 0.1 et 0.9	Entre 0.1 et 0.9	Errlocal 28	ErreurAcSor 28	ـر
29	64	variable	Variable	Entre 0.1 et 0.9	Entre 0.1 et 0.9	Errlocal 29	ErreurAcSor 29	ز
30	64	variable	Variable	Entre 0.1 et 0.9	Entre 0.1 et 0.9	Errlocal 30	ErreurAcSor 30	ـز

IV.6.3. Sous système de reconnaissance

Le sous système de reconnaissance d'un mot manuscrit consiste à reconnaitre chaque lettre du mot, cette reconnaissance se fait par comparaison aux apprentissages des réseaux de neurones de la base de données. Ainsi la reconnaissance de toutes les lettres du mot conduira à la reconnaissance du mot entier.
Dans ce qui suit nous allons détailler les différentes parties de la reconnaissance.

IV.6.3.1. Prétraitement

Dans les phases d'apprentissage et de reconnaissance des mots, le module de prétraitement contient en plus des sous modules que nous avons cité au part avant, deux autres sous modules : le sous module de la correction de l'inclinaison des lignes et le sous modulede la correction de l'inclinaison des caractères qui sont deux opérations très nécessaires puisque l'inclinaison des lignes et l'inclinaison des caractères sont des sources d'erreur classique, relativement gênantes pour la phase de segmentation donc pour la reconnaissance.Mais dans notre cas, nous allons considérer que des images de lignes et de caractères non inclinés.

IV.6.3.2. La segmentation

La segmentation permet à partir d'une image acquise (image de texte de lignes) l'extraction de ses lignes (segmentation en lignes), ensuite chaque imaged'une ligne est segmentée en mots (segmentation en mots). Et enfin chaque image d'un mot extrait est segmenté en caractères (segmentation en caractères). Cette étape est indispensable, puisqu'elle produit des caractères en sortie, et que qui les caractères sont la base pour la reconnaissance des mots.

IV.6.3.2.1. La segmentation en lignes

La segmentation en lignes consiste à extraire les lignes à partir d'une image de lignes acquise.
Le principe de notre algorithme est d'utiliser les lignes blanches (les espaces inter-lignes) pour distinguer et extraire les régions (les lignes) contenant des pixels noirs (les parties deslignes contenant des pixels noirs).
Notre algorithme est applicable dans le cas où les lignes ne sont pas chevauchées.

La figure suivante montre un exemple de segmentation en lignes, en cet appliquant l'algorithme :

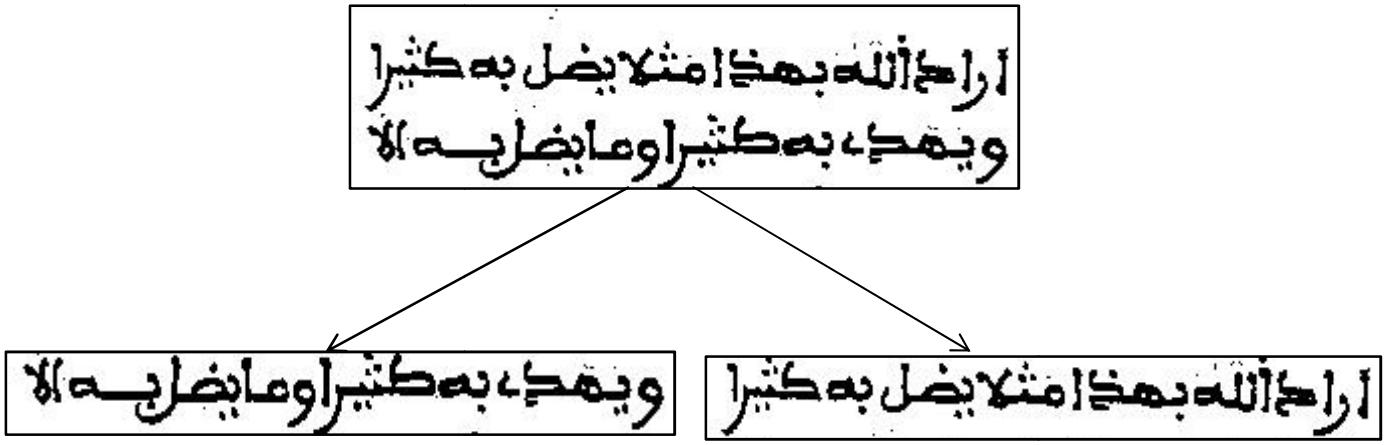


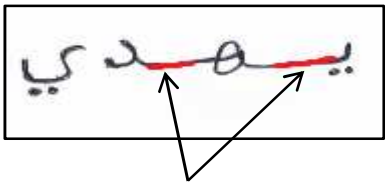
Figure IV.5 : Exemple de segmentation en lignes.

IV.6.3.2.2. La segmentation en mots

Le but de la segmentation en mots est de déterminer (approximativement) où débute et où prend fin chaque mot dans une ligne de texte.
Pour chaque ligne, une projection verticale en détectant les valeurs nulles donnera chaque mot de cette ligne.

IV.6.3.2.3. La segmentation en caractères

Le but de la segmentation en caractères est de déterminer (approximativement) où débute et où prend fin chaque caractère dans un mot.
Vu que les caractères des mots arabes sont généralement liés par des éléments d'épaisseur uniforme (voir la figure suivante), nous avons proposé un algorithme de segmentation quipermet la séparation des caractères en éliminant ces éléments (pour plus de détail sur cet algorithme voir annexe).



Eléments d'épaisseur uniforme.

Figure IV.6 : Représentation des éléments d'épaisseur

- **Principe de fonctionnement de l'algorithme de segmentation en caractères [89]**

Nous avons en entrée une image contenant un mot, cette image sera parcourue de droite à gauche en détectant les colonnes blanches en utilisant un seuil. Telle que si le nombre de pixel blanc dans cette colonne est supérieur ou égale à ce seuil, elle sera considérée une colonne blanche. Ce seuil est obtenu empiriquement comme suit : hauteur de l'image * 0.95f.

Dans chaque itération, on calcul aussi le nombre de colonnes blanches successives. Si ce nombre est supérieur ou égale à un seuil, cette suite de colonne est considérée comme un bloque blanc, et ses indices de début et de fin serons sauvegarder dans une liste « al ». Ce seuil est obtenu empiriquement comme suit : maximum (1, hauteur * 0.05f).
A l'aide de cette liste, nous déterminons les caractères à extraire de l'image d'entrée en parcourant la liste par pas de deux (2).
Dans ce qui suit, nous présentons les détails de cet algorithme.

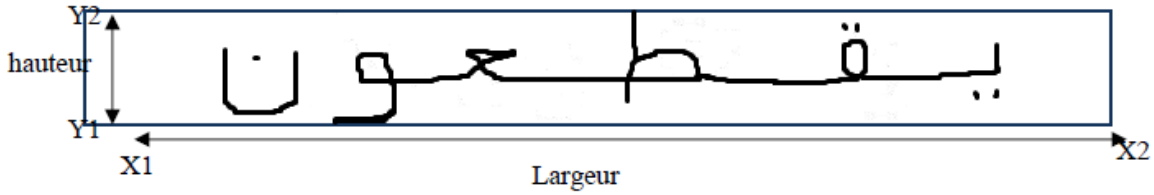


Figure IV.7: Image d'entrée résultante de la segmentation en ligne.

Entrée : l'image contenant le nom ou une partie de nom,
Sortie : images des trames résultantes de la segmentation de nom,

// hauteur de l'image d'entrée
Entier : hauteur = y2 - y1,

//seuil déterminant si un bloque peut être négligé (blanc)
Entier : SeuilBlocBlanc = maximum (1, hauteur * 0.05f),

//seuil déterminant si une colonne d'un pixel d'épaisseur peut être négligée (blanche)
Entier : SeuilColonneBlanc = hauteur * 0.95f ,

//Liste résultante contenant les indices où se débute et où se termine chaque trame à extraire dans l'image d'entrée
Liste d'entier : al,
Booleen : Separateur = **Vrais,**
Entier : CharX1 = 0, prevCharX1 = -1,
Booleen: EspaceBlancLibre = **Faux**,
Entier : NumBlancConsecutive = 0,

Etape 1 : Le but de cette étape est de trouver les indices des bloques considérés blancs, puis les sauvegarder dans la liste al.

```
Pour  x = 1  à  x < (largeur - 1),   pas de 1 Faire
    // Teste effectuer dans le cas il a trouvé un bloque de largeur supérieur à la hauteur et il n'a
        pas marqué des colonnes blanches.
    Si EspaceBlancLibre = Faux et NumBlancConsecutive == 0 et  x - charX1>= hauteur  alors
        x = charX1,
        EspaceBlancLibre = Vrais,
    FinSi



    //Nombre de pixels blancs dans une colonne
    Entier :  numPixelsBlancColumn = 0,
    Booléen :  EspaceBlanc = Vrais,

    Pour  y = y1   à    y < y2,   pas de 1   Faire
        //tester si le pixel de coordonnée x, y est blanc ou non
      Si   pixels(x, y) >= 128
        //Incrémenter le nombre de pixel blanc d'une colonne x
          Incrimenter    numPixelsBlancColumn,
      FinSi
      Sinon
        // le cas où le pixel de coordonnée x, y est noir (<128)
        Si  EspaceBlancLibre = Faux Alors
            EspaceBlanc = Faux
            Sortir de la boucle pour,
        FinSi
      FinSinon,
    Fin pour,
```

```
//tester si la colonne x est négligeable (> SeuilColonneBlanc) ou non (<SeuilColonneBlanc)
Si EspaceBlancLibre = Vrais et numPixelsBlancColumn < SeuilColonneBlanc Alors
    EspaceBlanc = Faux
FinSi

Si EspaceBlanc = Vrais Alors
    //incrémenter le nombre de colonne blanche successive
    Incrimenter NumBlancConsecutive,
    //test si le bloque (succession de colonnes) est négligeable (>= SeuilBlocBlanc) ou non
      (<SeuilBlocBlanc)
    Si NumBlancConsecutive >= SeuilBlocBlanc)
        Si Separateur= Faux Alors
            Separateur=Vrais,
            //sauvegarder dans la liste al les indices des bloques blancs à supprimer
            Ajouter dans al(CharX1), //début de bloque
            Ajouter dans al(x- (NumBlancConsecutive - 1)), // fin de bloque
        FinSi
    FinSi
FinSi
// Le cas où la colonne x n'est pas blanche
Sinon
    NumBlancConsecutive = 0,
    Si Separateur = Vrais Alors
        Separateur = Faux,
        prevCharX1 = charX1,
        charX1 = x,
        EspaceBlancLibre = Faux,
    FinSi
FinSinon
Fin Pour
```

```
//à  la fin de nom, si le dernier bloque n'est pas un blanc alors il faut sauvegarder son indice
pour   le localiser
Si NumBlancConsecutive = 0 Alors
    |       Ajouter dans al(CharX1),
    |       Ajouter dans al (largeur),
FinSi
```

La figure suivante illustre le principe de l'étape 1 :

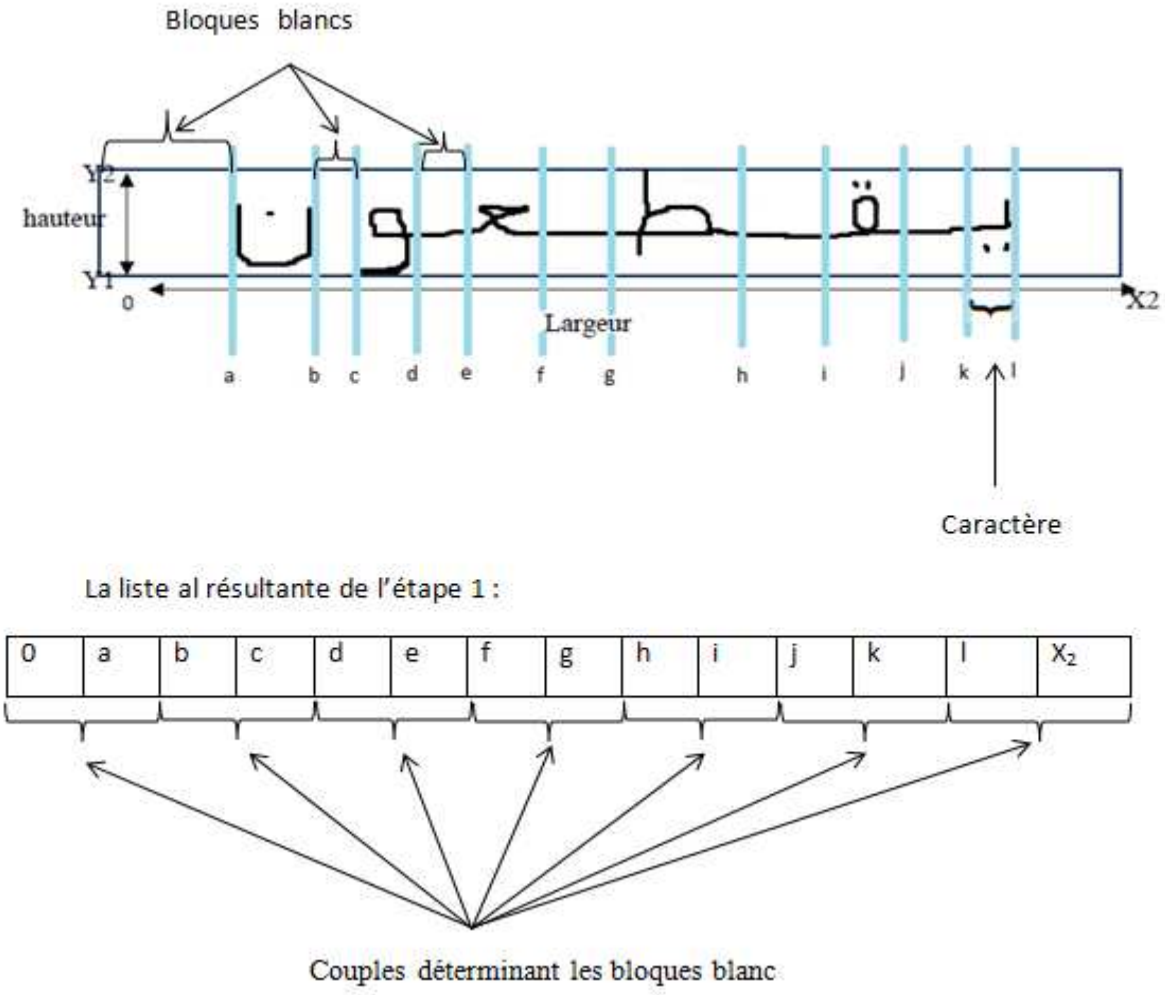


Figure IV.8 : Illustration de l'étape 1.

Étape 2: l'objectif est de fusionner les bloques blanc successifs les plus proche en utilisant le seuil MinlargeurChar.

```
Entier :  MinlargeurChar = hauteur * 0.15f,
Si MinlargeurChar < 1 Alors
    MinlargeurChar=1,
FinSi

//Test permettant de déterminer si deux bloques blanc successifs dans la liste peuvent être
  fusionnés

Pour  i = 0 à  (i + 4) < taille de la liste al, pas de  2 Faire
    Entier LargerChar = récupérer de al l'élément (i + 2) - récupérer de al l'élément (i),
    Si largeurChar < MinlargeurChar ou largeurChar < 2))
            Supprimer de la liste al l'élément (i + 2),
            Supprimer de la liste al l'élément (i + 1);
            Décrémenter  i de 2,
    FinSi
FinPour
```

Etape 3 : Après l'obtention de la liste al optimisée qui contient les indices des blocs blancs, elle sera utilisée pour l'extraction des caractères de l'image d'entrée. Cette liste est utilisée comme suit :

La liste seras parcourus de début jusqu'à la fin par pas de deux, dans chaque itération le contenu de la liste dont les indices sont (i+1) et (i+2) délimite où commence et où termine le caractère à extraire dans l'image d'entrée.

Si par exemple i=2 alors le contenu des indices 3 et 4 délimite une trame à extraire.

Comme il est illustré dans la figure suivante :

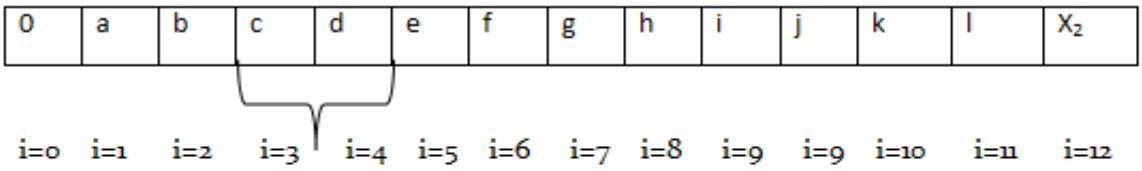


Figure IV.9 : Illustration de l'étape 3.

```
Pour i = 0  à i < (taille de la liste al – 2), pas de 2 Faire
    Si (al (0)=0) Alors
        Entier cx1 = récupérer de la liste al l'élément (i +1),
        Entier cx2 = récupérer de la liste al l'élément (i+2),
        Extraire de l'image d'entrée trame de largeur (cx1, cx2),
    FinSi
    Sinon
        Si al(0) <> 0 Alors
            Entier cx1 = récupérer de la liste al l'élément (i),
            Extraire de l'image d'entrée trame de largeur (0, cx1),
        FinSi
        Si i <> 0 Alors
            Entier cx1 = récupérer de la liste al l'élément (i +1),
            Entier cx2 = récupérer de la liste al l'élément (i+2),
            Extraire de l'image d'entrée trame de largeur (cx1, cx2),
        FinSi
    FinSinon
Fin Pour
```

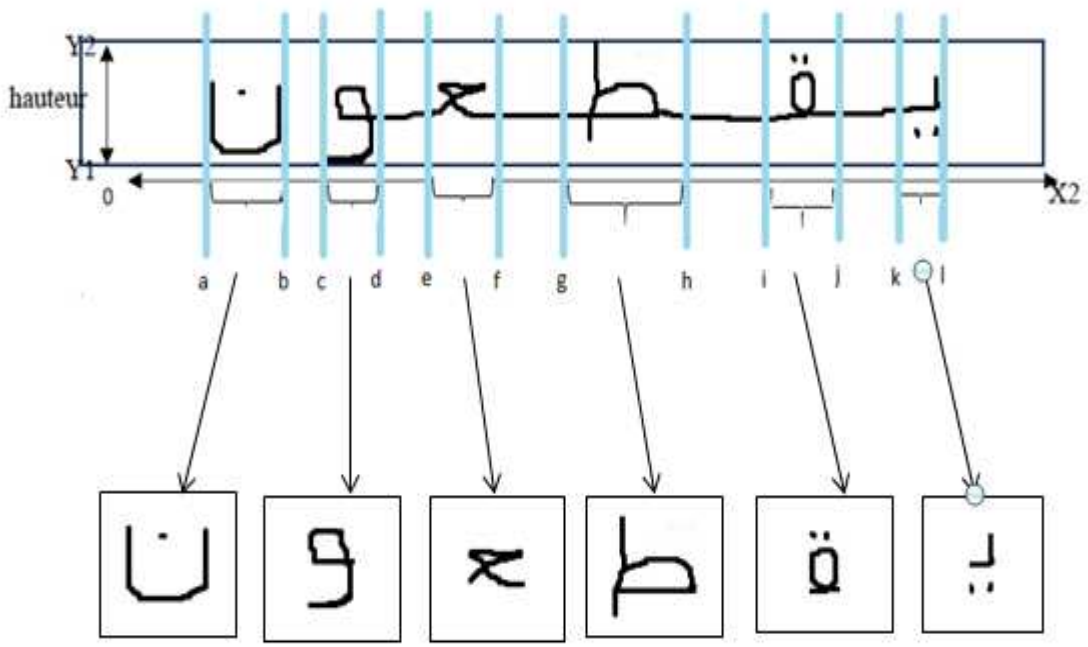


Figure IV.10: illustre le résultat de l'algorithme de la segmentation.

L'application de l'algorithme de segmentation en caractères sur l'écriture arabe manuscrite peut ne pas être fiable, vu les caractéristiques de l'écriture arabe manuscrites par exemples : les problèmes de discontinuité de l'écriture, les ligatures, de chevauchement et d'accolement de pseudo mots, de grandes variabilité inter et intra scripteur, de variation de dimension des pseudos mots ainsi que les signes diacritiques...

« ***Il n'est pas évident de juger de l'efficacité d'un algorithme de segmentation en lettres, le résultat peut être décevant*** » **[88].**

IV.6.3.3. Normalisation et extraction des caractéristiques

Après l'étape de segmentation du mot en caractères, chaque caractère doit être normalisé pour l'adapter à une dimension de 64*64 pixels, puis nous allons construire son vecteur de primitives qui va être par la suite normalisé (entre l'intervalle 0 et 1) puis transformé en vecteur lignes, pour enfin être classifié par le réseau de neurones pour trouver sa lettrecorrespondante dans la base de données de données.

IV.6.3.4. la reconnaissance par le perceptron multicouche (réseau de neurones)

Soit L_M notre ensemble d'apprentissage de M exemplesà partir duquel notre classifieur neuronal va déterminer ses paramètres, chaque exemple étant un couple formé d'un vecteur d'entrée et sa classe(dans notre cas les vecteurs d'entrée sont ceux correspondant aux lettres manuscrites et ceux de sortie ceux des lettres idéales) tel que :

$L_M = \{(x^1,y^1),(x^2, y2)\ldots,(x^m,y^m)\}$

Où l'entrée $x^k = [x_1^k, x_2^k, \ldots, x_m^k]^T$ est un vecteur de M composantes à valeurs réelles comprises entre 0 et 1 décrivant l'exemple k(k=1, 2,…, M), et $y^k \in \{-1, +1\}$ représente la classe de x^k (le type de chaque lettre, c'est à noter que nous avons 98 types différents).
La sortie du classifieur dépend de son entrée **x** et de ses paramètres que l'on note **w** ; sortie est notée $\sigma^k(x ; w)$, ou simplement $\sigma^k(\sigma^k \in \{-1, +1\})$. Le classifieur est en mesure de classer correctement l'exemple $\mathbf{x}^k$ si $\sigma^k = y^k$ autrement dit si la condition suivante est satisfaite :

Condition de classification correcte

$\sigma^k y^k > 0$
En effet, si l'exemple est mal classé, on a $\sigma^k \neq y^k$, et alors $\sigma^k y^k < 0$.

Notre apprentissage par réseau de neurones avait pour objet d'apprendre à bien classer les exemples de L_M, et de déterminer les paramètres qui permettent de classer correctement des entrées nouvelles (on dira alors que le classifieur généralise correctement).
Si l'on considère que le vecteur des entrées **x** est une réalisation d'un vecteur aléatoire à valeurs réelles **X,** et que la sortie y (qui est le code de la classe) est la réalisation d'une variable aléatoire discrète **Y,** on peut faire l'hypothèse qu'il existe une densité de probabilité $P_{X,Y}(x,y) \equiv p_X(x)P_Y(y|x)$ que l'on ignore, d'où sont tirées :

- Les entrées et les classes de l'ensemble d'apprentissage.
- Les nouvelles entrées, dont la classe, donnée par $P_Y(y|x)$, est inconnue.

L'erreur que nous aimerons minimiser lors de l'apprentissage est l'erreur de généralisation $\varepsilon_g(w)$:

$$\varepsilon_g(w) = \sum_{y=\pm 1} \int \Theta(-y\,\sigma(x ; w)) p_{X,Y}(x,y)\,dx$$

Où :

$\Theta(u)$ est la fonction Heaviside qui prend la valeur 1 si son argument u est positif ou nul, et 0 s'il est négatif.

σ est la classe attribuée à l'entrée **x** par le classifieur. L'erreur de généralisation est la probabilité que le classifieur, de paramètre w, commette une erreur de classification sur une entrée **x,** tirée avec la densité de probabilité $\mathbf{P_Y(y|x)}$. L'expression de $\varepsilon_g(w)$ ne peut pas être calculée dans les applications, car $P_Y(y|x)$ est inconnue.

IV.6.3.5. Regroupement des lettres et la reconnaissance du mot

Le système regroupe toutes les lettres reconnues, en fonctions de ces lettres il sélectionne lemot correspondant dans le dictionnaire des mots arabes. La sélection s'effectue en choisissant le mot possédant le maximum des lettres reconnues.

IV.7. Conclusion

Dans ce chapitre nous avons modélisé notre application par l'introduction du langage de modélisation « UML » et nous avons conçu le digramme de classe général.
A la fin de la modélisation, nous sommes passés à la description du fonctionnement de notre système de reconnaissance de mots manuscrits arabes.
Le chapitre qui suit sera consacré à l'implémentation et la réalisation de notre application.

CHAPITRE III :Réalisation

Réalisation

V.1. Introduction

Après avoir présenté dans le chapitre précédent les différentes étapes d'analyse et de conception, nous allons présenter dans ce dernier chapitre l'étape de réalisation qui consiste à traduire la conception en code exécutable. Nous allons, tout d'abord, commencer par la description des outils de développement et d'implémentation, ainsi que les différents langages nécessaires à la réalisation de notre système. Et enfin nous terminons par une représentation de quelques interfaces graphiques de notre application.

V.2. Présentation des outils de développement

V.2.1. Quelques définitions de base

1. Définition d'une base de données Une base de données (BDD) est un ensemble de structures créées à l'image d'un modèle de données et gérées pour stocker les informations dans le but d'effectuer subséquemment des recherches et des mises à jour. C'est en quelque sorte la représentation structurée et codée de l'interprétation d'une réalité organisationnelle qui est en constante évolution dans le temps.

2. Définition d'un SGBD (système de gestion de fichier)

Le logiciel SGBD est un ensemble de procédures partagées par tous les utilisateurs pour la création et la manipulation des données avec une garantie de cohérence, de consistance dans les opérations ou des persistances des ajouts et des modifications transactionnelles.

3. Le serveur de données [78]

C'est un système de gestion de base de données. Son rôle est de stocker et de gérer une grande quantité de données en les organisant sous forme de tables, et de permettre la manipulation de ces données à travers le langage de requêtes**SQL**. On ne s'occupe plus alors de la manière dont les données sont stockées sur le disque dur, de simples instructions permettent d'ajouter, de supprimer, de mettre à jour et surtout de rechercher des données dans une base de données.

- **Les points forts de MySQL**

- implémentation libre et populaire ;
- facile à mettre en œuvre ;
- rapide à apprendre ;
- support multiplateformes ;
- fiable et rapide.

- **Les points faibles de MySQL**

Ses principaux points faibles sont :

- ne possède pas de mécanisme transactionnel dans sa version 3 ;
- n'implémente pas les références d'intégrité relationnelles ;
- absence de procédures stockées.

V.2.2. Caractéristiques de MySQL [78]

1. Vitesse: Bien sûr, la vitesse à laquelle un programme côté serveur fonctionne dépend principalement du matériel serveur. Étant donné que le matériel du serveur est optimal, MySQL fonctionne très rapidement. Il supporte les serveurs en cluster pour des applications exigeantes.

2. Facilité d'utilisation: MySQL est d'une haute performance, le système de base de données relativement simple. Dès le début, MySQL a été configuré normalement, surveillé et géré à partir de la ligne de commande. Cependant, plusieurs interfaces graphiques de MySQL sont disponibles tel que décrit ci-dessous:

- MySQL Administrateur: cet outil permet aux administrateurs de mettre en place, évaluer et ajuster leur base de données MySQL. Ceci est conçu comme un remplacement pour mysqladmin.
- MySQL Query Browser: Fournit aux développeurs de bases de données et les opérateurs avec une interface graphique de fonctionnement de la base. Il est particulièrement utile pour voir les plans de requêtes multiples et des ensembles de résultats dans une seule interface utilisateur.
- Assistant de configuration: les administrateurs peuvent choisir parmi une liste prédéfinie de paramètres optimaux, ou en créer leur propre.
- System Tray MySQL: Fournit Windows administrateurs une vue unique de leur instance de MySQL, y compris la capacité de démarrer et d'arrêter leurs serveurs de bases de données.

3. Coût: MySQL est disponible sans frais. MySQL est un "Open Source" base de données. MySQL est une partie de LAMP (Linux, Apache, MySQL, PHP / Perl / Python), une croissance rapide pile open source d'entreprise. De plus en plus de personnes utilisent LAMP comme une alternative aux coûteuses piles de logiciels propriétaires en raison de son coût, faible fiabilité, et la documentation.

4. Capability : Beaucoup de clients peuvent se connecter au serveur en même temps. Les clients peuvent utiliser plusieurs bases de données simultanément. Vous pouvez accéder à MySQL en utilisant plusieurs interfaces telles que la ligne de commande des clients, les navigateurs Web.

5. Connectivité et sécurité: MySQL est entièrement en réseau, et base de données peut être consulté de n'importe où sur Internet, ainsi vous pouvez partager vos données avec n'importe qui, n'importe où. La connectivité pourrait être atteinteavec les programmes Windows en utilisant les pilotes ODBC. En utilisant le connecteur ODBC pour MySQL, toute application cliente ODBC (par exemple, Microsoft Office, auteurs du rapport, Visual Basic) peut se connecter à MySQL.

6. Portabilité: MySQL fonctionne sur de nombreuses variétés d'UNIX, ainsi que sur d'autres systèmes non-Unix, tels que Windows et OS / 2.MySQL fonctionne sur du matériel d'un PC à domicile au serveur haut degamme. MySQL peut être installé sur Windows XP, Windows Server 2003, RedHat Linux Fedora, Debian Linux, et autres.

V.3. Présentation du langage de développement

V.3.1. Le langage de programmation JAVA [79]

Le langage **Java** est un langage de programmation informatique orienté objet créé par James Gosling et Patrick Naughton employés de Sun Microsystems avec le soutien de Bill Joy (cofondateur de Sun Microsystems en 1982), présenté officiellement le 23 mai 1995 au Sun World.

Le langage Java a la particularité principale que les logiciels écrits avec ce dernier sont très facilement portables sur plusieurs systèmes d'exploitation tels que l'UNIX, Windows, Mac OS ou GNU/Linux avec peu ou pas de modifications. C'est la plate-forme qui garantit la portabilité des applications développées en Java qui s'agit de la machine virtuelle (JVM).

1. **Caractéristique de java [80]**

- **Robuste et sûr *:*** ce qui a permet de dire que java est un langage robuste et sur est l'effet que sont compilateur est très strict car toutes les valeurs doivent être initialisé

ainsi que le traitement des exceptions est obligatoire sans oublié que les pointeurs et les fonctions d'arguments variables n'existe pas.

- **Sécurisé :**Java assure une certaine sécurité au système à travers des tests qui vérifient en permanence la conformité du pseudo-code à certaines règles.
- **Simple :**le code source en java est simple, même si java est basé sur le C/C++mais certaine concept à l'origine de nombreux bugs ont été supprimé comme les pointeurs, gestion de la mémoire, héritage multiple.
- **Portable :**le compilateur java génère du baye-code, ce code est conçu pour être rapidement interprété, facilement traduit en instructions assembleur lors de son exécution et vérifiable, ce dernier est exécuté par la machine virtuelle qui dépend de la plat forme. Java est donc un langage portable.
- **Orienté objet :**Inspiré du C++, Objective C et Small talk, Java est un ensemble de structures de données qu'on appelle objets, chaque objet regroupant à la fois des données et des méthodes manipulant les données. Java est un langage fortement objet.

2. Environnement de programmation de java [81]

Il se base sur trois principaux logiciels qui se présentent comme suit :

- **Le JDK** (Java Development Kit), qui contient javac, le compilateur qui transforme votre programme source en byte code, java, l'interpréteur de byte code, pour exécuter les applications, l'Applet Viewer, pour exécuter les applets, java doc, un programme permettant de créer automatiquement la documentation de vos programmes au format HTML, et d'autres utilitaires.
- **La documentation**, qui contient la liste de toutes les classes Java. Cette documentation est absolument indispensable au programmeur. Elle est au format HTML et doit être consultée à l'aide d'un navigateur.
- **Le JRE** (Java RuntimeEnvironment), qui contient un interpréteur de 'byte code' et tout ce qui est nécessaire pour diffuser vos applications aux utilisateurs (y compris un compilateur JIT).

V.3.2.Langage de requêtes SQL [82]

SQL signifie StructuredQueryLanguage, ou langage de requêtes structuré. C'est le langage le plus répandu pour accéder à des systèmes de gestion de bases de données relationnelles (SGBD). Il permet :

- La manipulation des tables : création, suppression, modification de la structure des tables.
- La manipulation des bases de données : sélection, modification et suppression d'enregistrements.
- La gestion des droits d'accès aux tables : contrôles des données et validation des modifications.

V.3.3.NetBeans IDE 7.3.1 [78]

NetBeansest un environnement de développement intégré (EDI), placé en *open source* par Sun en juin 2000,il permet également de supporter différents autres langages, comme Python, C, C++, JavaScript, XML, Ruby, PHP et HTML.

NetBeans constitue par ailleurs une plateforme qui permet le développement d'applications spécifiques (bibliothèque Swing (Java)). L'IDE NetBeans s'appuie sur cette plateforme.

En 1997, NetBeans naît de Xelfi, un projet d'étudiants dirigé par la Faculté de mathématiques et de physique de l'Université Charles de Prague. Plus tard, une société se forme autour du projet et édite des versions commerciales de l'EDI NetBeans, jusqu'à ce qu'il soit acheté par Sun en 1999. Sun place le projet sous double licence CDDL et GPL v2 en juin de l'année suivante.

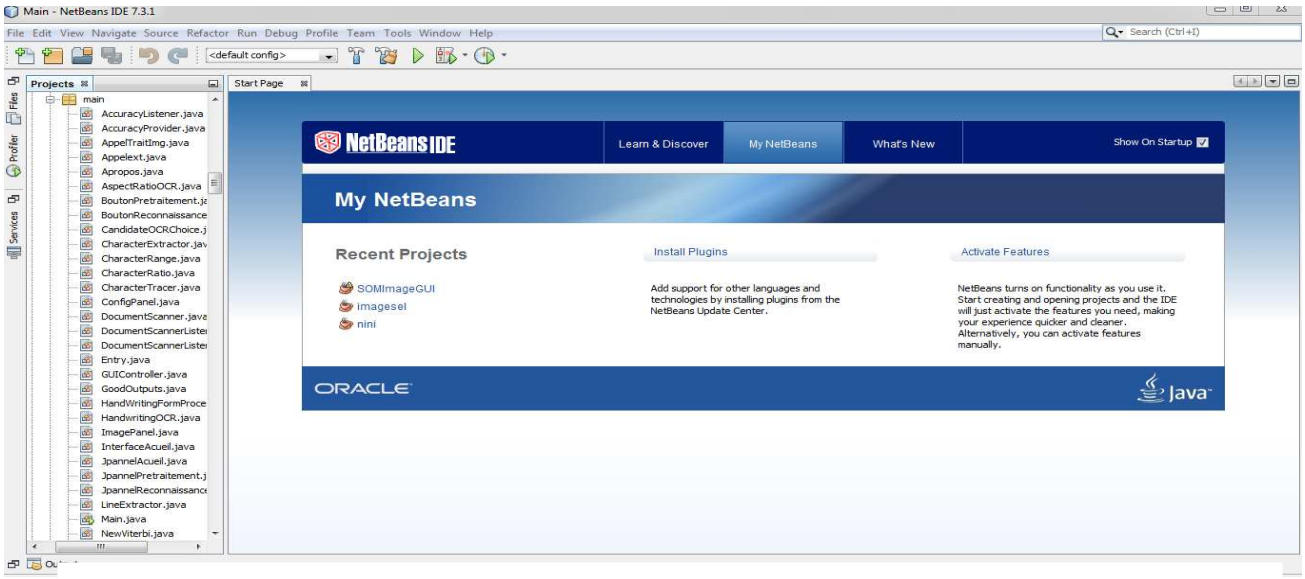


Figure V.1 : Interface d'accueil JAVA Net Beans.

V.4. Présentation de notre système

Notre application consiste à mettre en évidence le fonctionnement d'un réseau de neurones artificiel dans une reconnaissance hors ligne de mots arabes manuscrits, c'est-à-dire on fournit pour le système des images de lettres manuscrites pour l'apprentissage et pour la reconnaissance, avec un apprentissage supervisé qui consiste à fournir à l'application au préalable les sorties qu'on veut atteindre (désirées), pour qu'elle puisse faire la comparaison avec les sorties obtenues, ensuitele système boucle tout en ajustant ses poids synaptiques avec l'algorithme de rétro propagation, jusqu'à ce qu'il s'approche au mieux des résultats voulus.

V.4.1. Interface d'accueil

L'exécution de notre application mène directement à la page d'accueil représentée par la figure suivante :

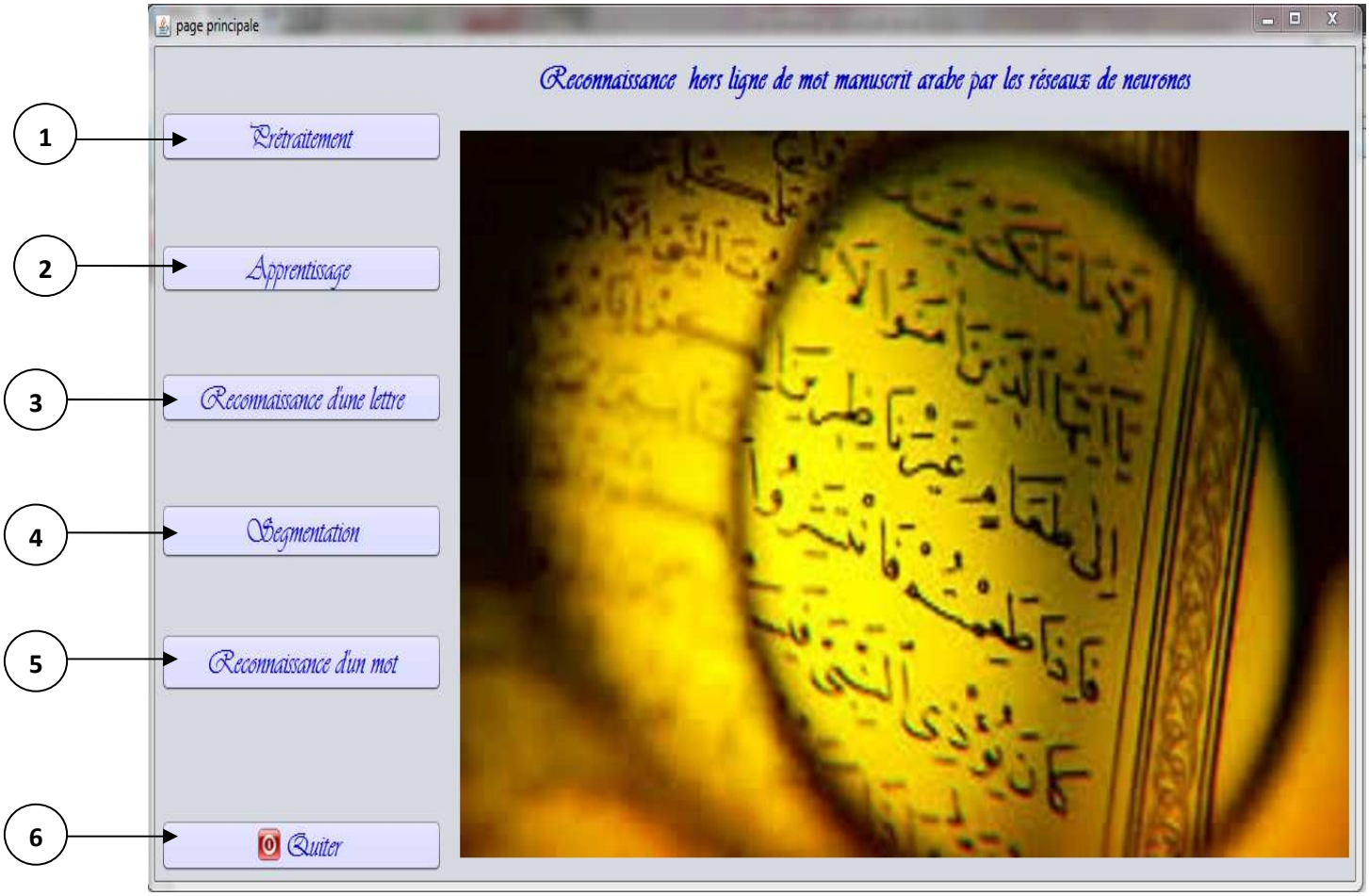


Figure V.2 : Interface d'accueil.

1) **Bouton prétraitement :** permettant d'afficher l'interface de prétraitement.
2) **Bouton apprentissage :** permettant d'afficher l'interface d'apprentissage.
3) **Bouton reconnaissance d'une lettre :** permettant d'afficher l'interface de reconnaissance d'une lettre.
4) **Bouton segmentation:** permettant d'afficher l'interface de segmentation enlignes, l'interface de segmentation en mots ainsi que l'interface de segmentationen caractères.
5) **Bouton reconnaissance d'un nom:** permettant d'afficher l'interface de reconnaissance d'unmot.
6) **Bouton quitter :** permettant de quitter l'interface d'accueil principale.

V.4.2. Interface de prétraitement

Elle permet d'effectuer les prétraitements sur l'image de la lettre et l'extraction de ses primitives (matrice d'observations). Cette interface est représentée par la figure suivante :

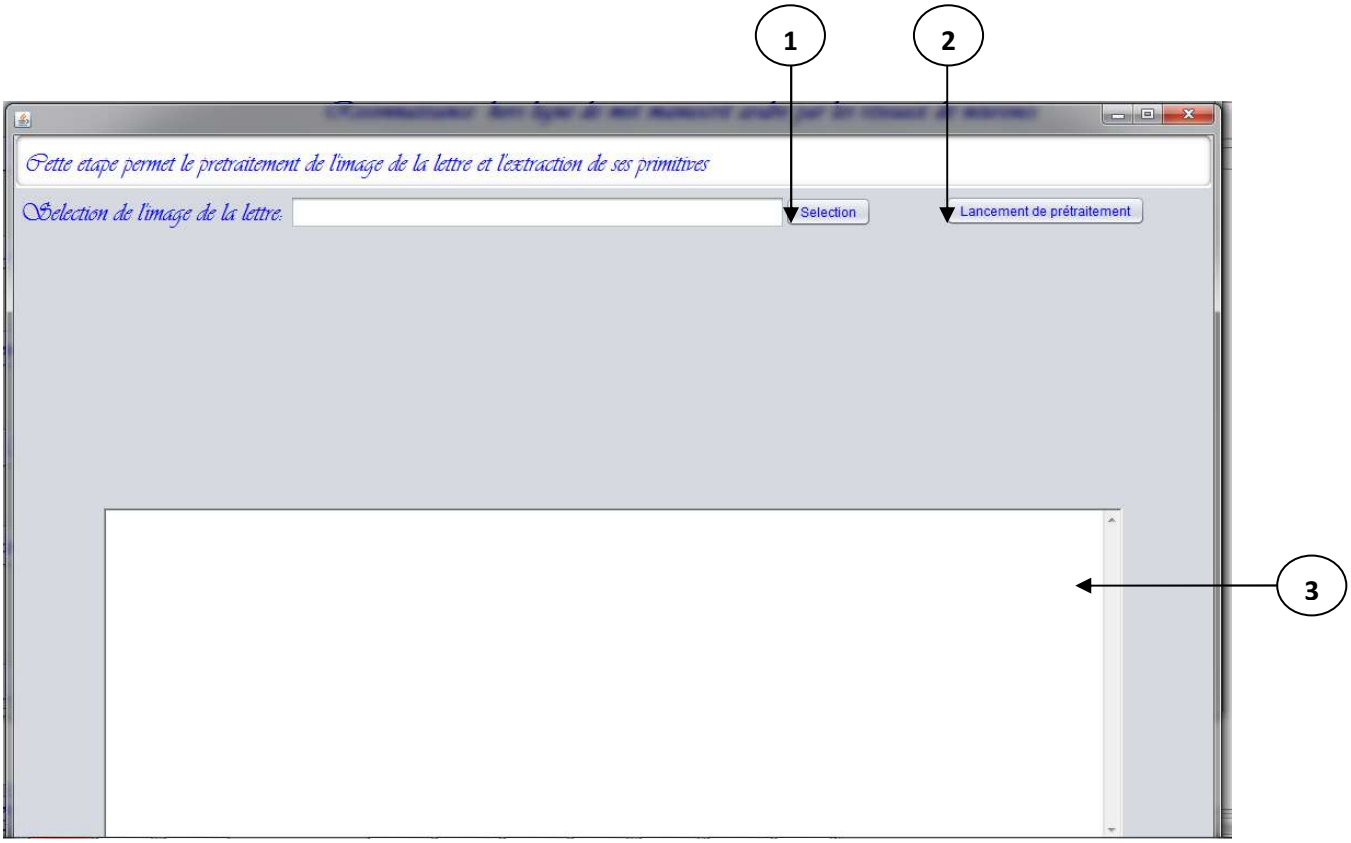


Figure V.3 : Interface de prétraitement.

1) **Bouton sélection :** il permet le chargement de l'image.
2) **Bouton lancement du prétraitement** : il permet de déclencher les opérations de prétraitement et l'extraction des primitives.

Le résultat de l'extraction des primitives sera affiché dans la zone de texte (3), et l'image prétraitée sera affichée sur écran.

V.4.3. Interface d'apprentissage

Cette interface permet l'apprentissage d'une image d'une lettre prétraitée en appliquant l'algorithme de rétro propagation du gradient appliqué sur un réseau de neurones de type perceptron multicouches. Cette interface est représentée par la figure suivante :

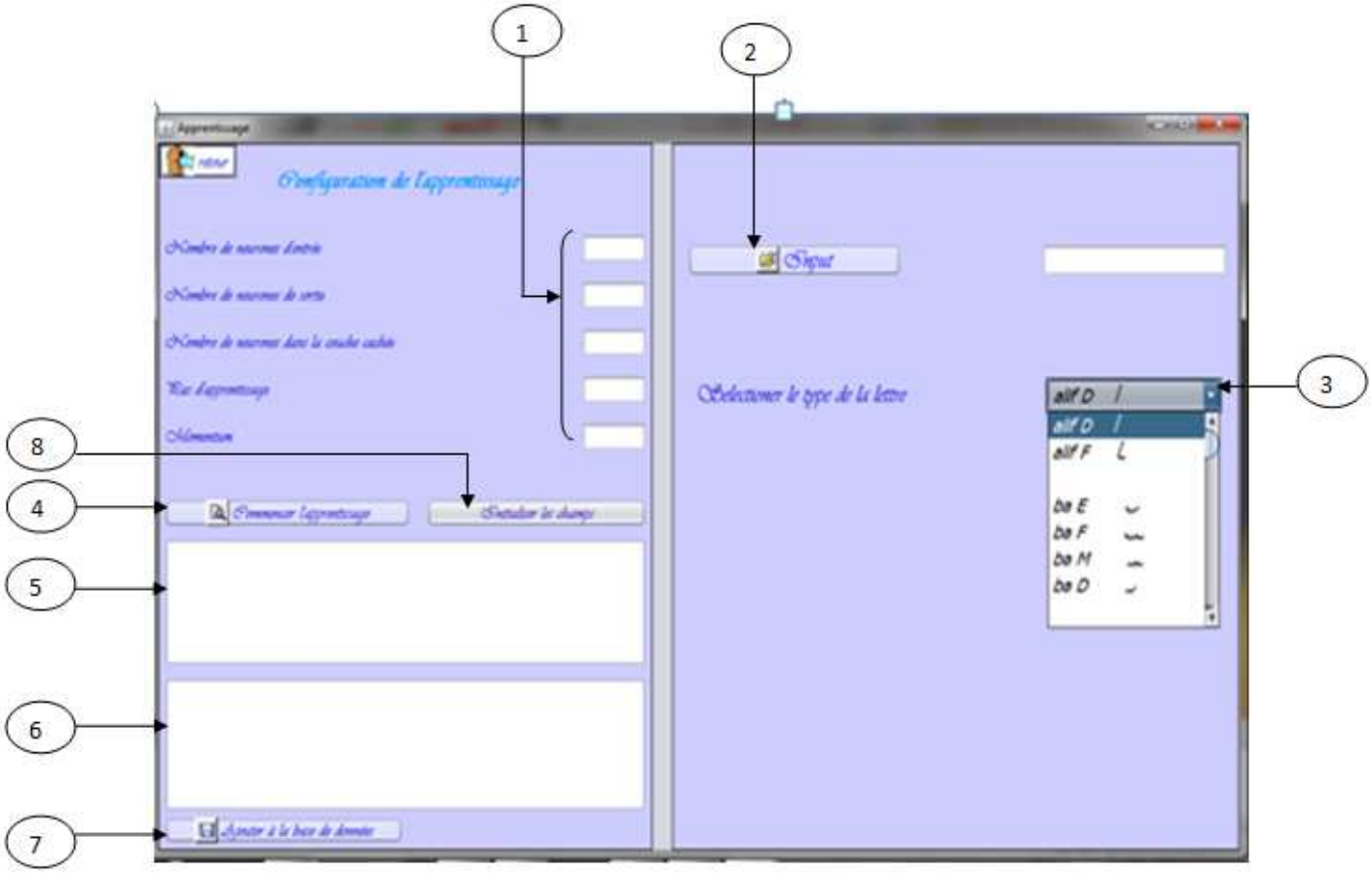


Figure V.4 : Interface d'apprentissage.

1) **Partie confuguration des pramètres du réseau de neurones** : en saisissant les champs correspendant respectivement au :

   - Nombre de neurones dans la couche d'entrées.
   - Nmobre de neurones dans la couche de sortie.
   - Nombre de neurones dans la couche cachée.
   - Pas d'apprentissage.
   - Momentum.

2) **Un bouton Input :** pour charger une image d'une lettre à apprendre.
3) **Une liste de choix :** pour choisir le type de la lettre à apprendre.
4) **Un bouton commencer apprentissage :** qui déclenche l'opération d'apprentissage de la lettre.
5) **Une zone de texte :** pour afficher l'erreur locale minimale de chaque itération.
6) **Une zone de texte :**pour afficher le vecteur de la sortie fournie ou bien calculée par le réseau (pour chaque élément du vecteur d'entrée correspondent à l'image de le lettre pour laquelle l'apprentissage été lancé).

7) **Un bouton ajouter à la base de données :** qui permet d'enregistrer les paramètres et résultats de l'apprentissage dans la base de données.
8) **Un bouton initialiser les champs :** qui permet dinitialiser les champs **1)** et **2)**

V.4.4. Interface de reconnaissance d'une lettre

Cette interface permet d'effectuer la reconnaissance d'une lettre en appelant le classifieur neuronal (perceptron multicouches). Elle est représentée par la figure suivante :

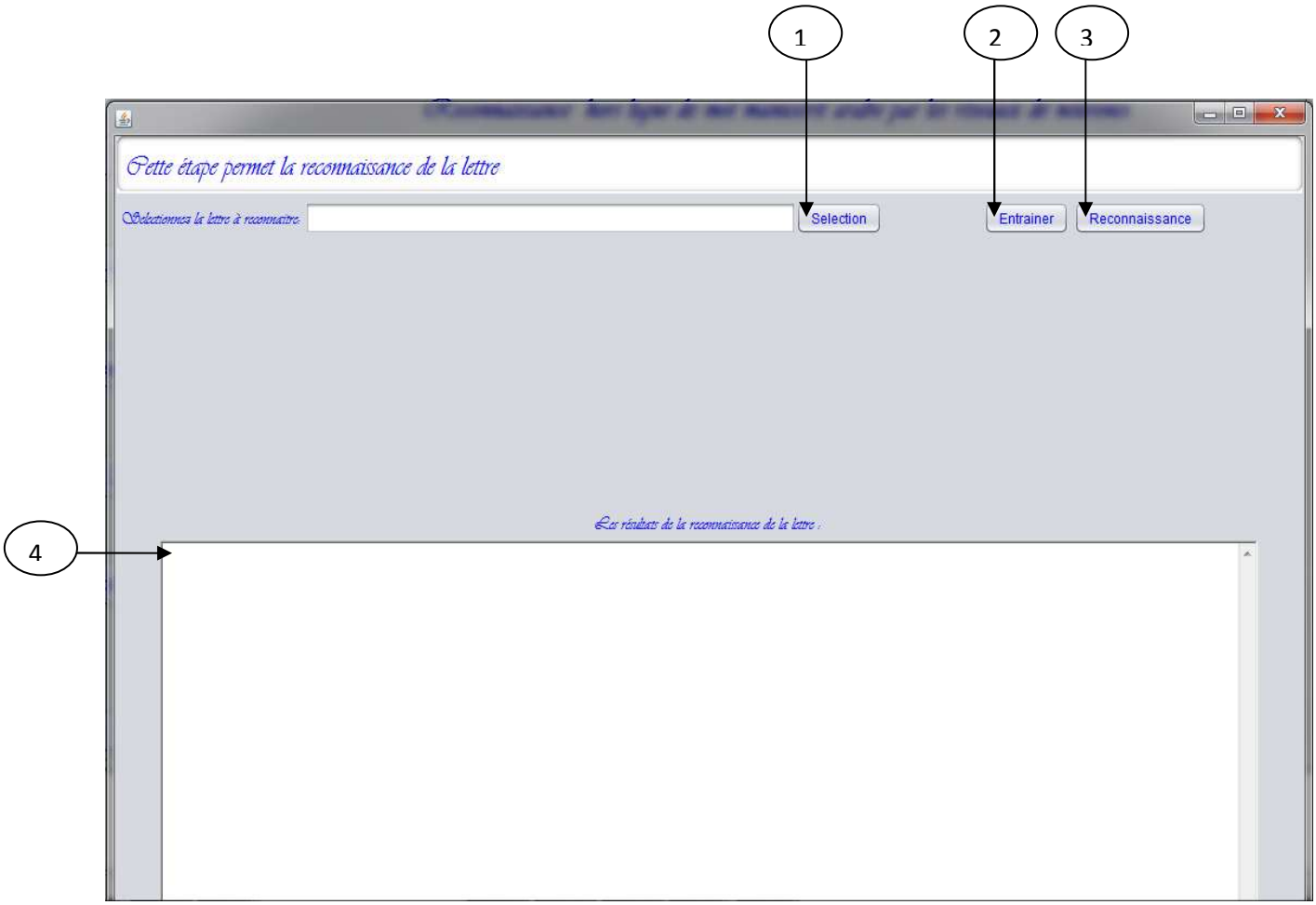


Figure V.5 : Interface de reconnaissance d'une lettre.

1) **Un bouton sélection :** pour charger l'image de la lettre à reconnaitre.
2) **Un bouton entrainer :** pour entrainer la lettre.
3) **Un bouton reconnaissance :** qui déclenche l'opération de reconnaissance de la lettre.
4) **Une zone de texte :** pour afficher les résultats de la reconnaissance de la lettre.

V.4.5. Interface de segmentation en lignes

Cette étape permet la segmentation d'un texte de lignes en lignes. Elle est représentée par la figure suivante :

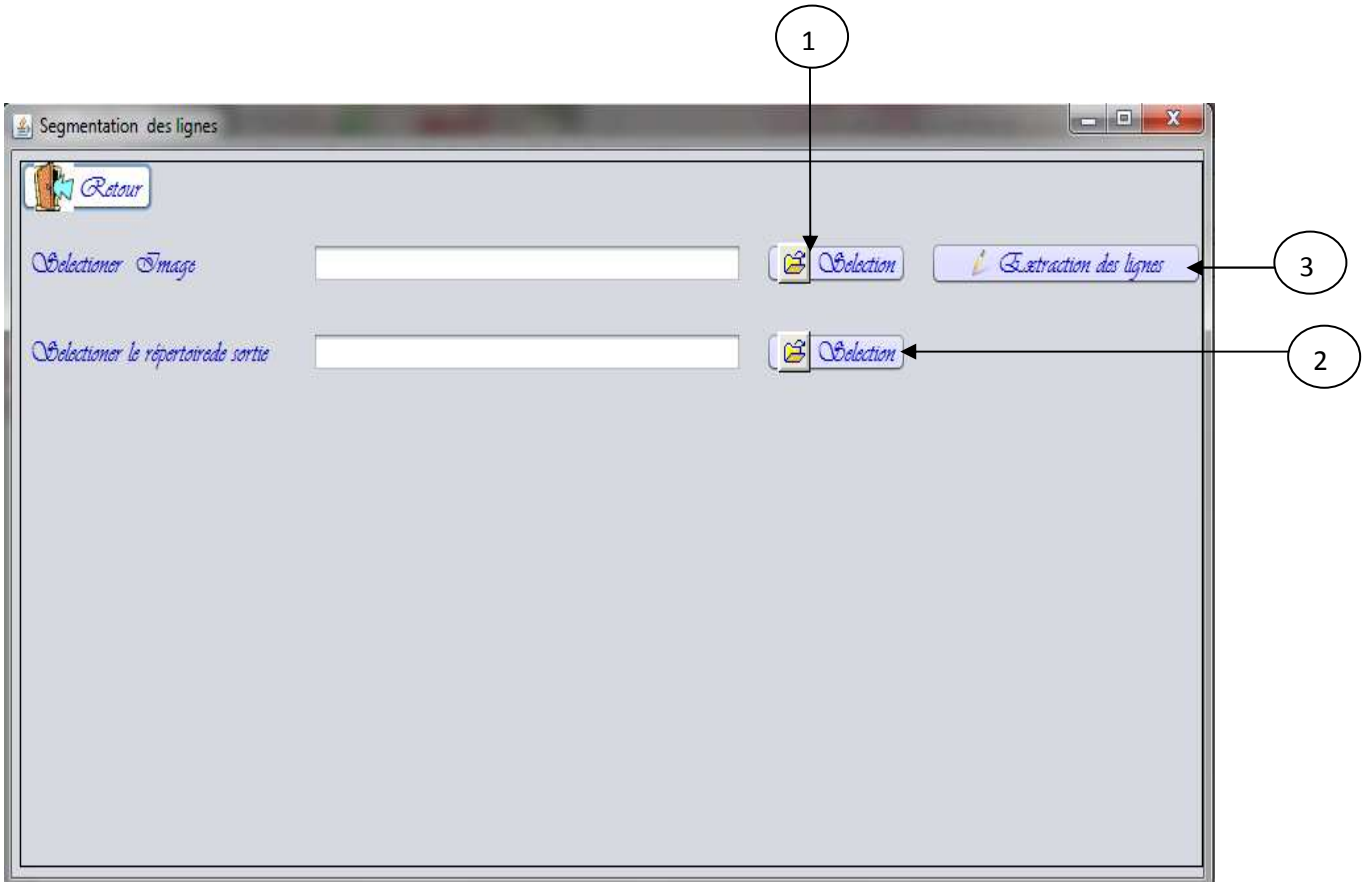


Figure V.6 : Interface de segmentation des lignes.

1) **Un bouton Selection :** pour charger l'image de lignes à segmenter.
2) **Un autre bouton Selection :** pout choisir le répertoire de sortie pour les résultats de la segmentation.
3) **Un bouton Extraction des lignes :** pour déclencher l'opération de la segmentation des lignes.

V.4.6. Interface de segmentation en mots

Cette étape permet la segmentation en mots d'une image d'une ligne de texte en mots. Elle est représentée par la figure suivante :

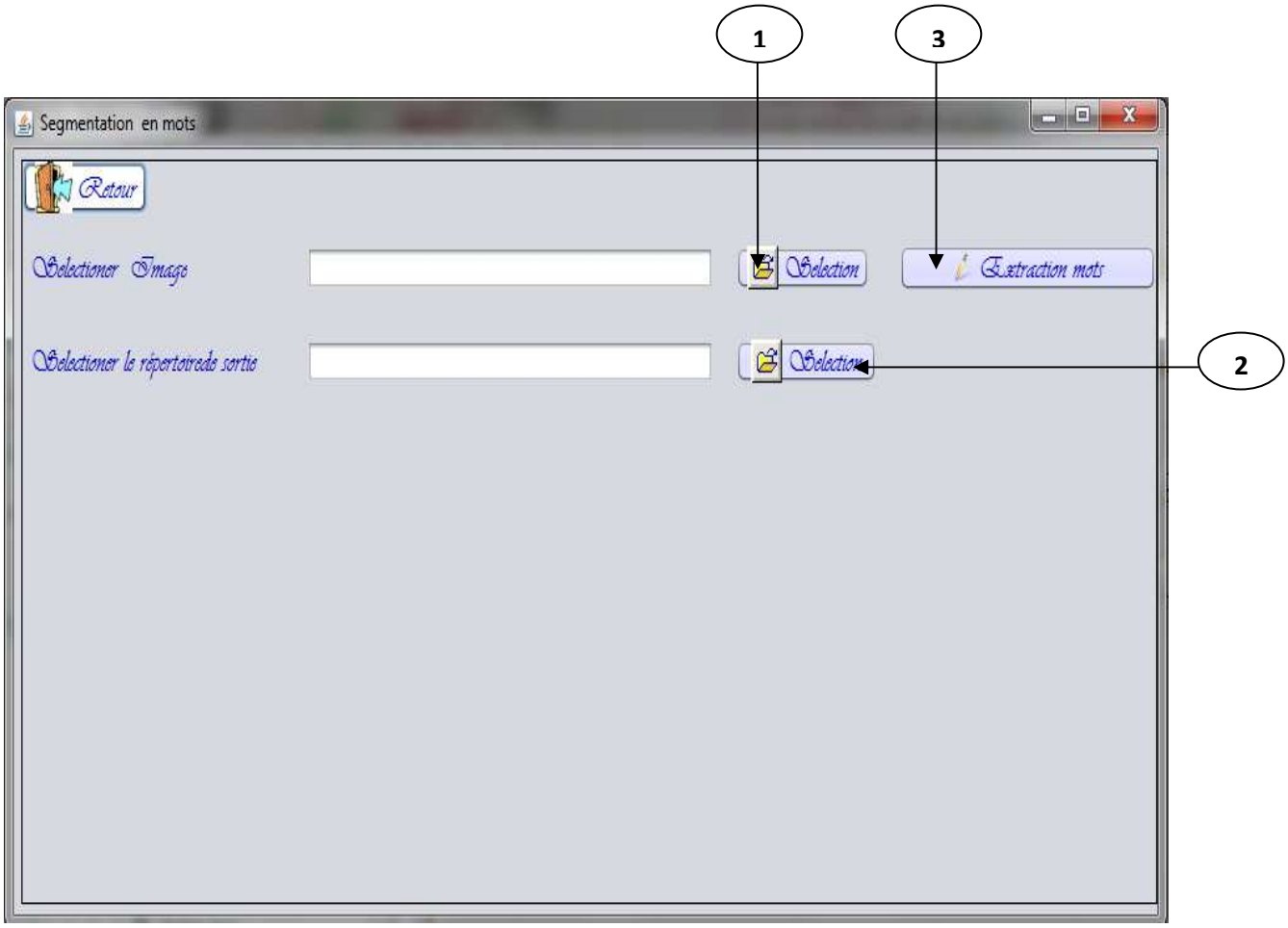


Figure V.7 : Interface de segmentation en mots.

1) **Un bouton Selection :** pour charger l'image de la ligne de texte à segmenter en mots.
2) **Un autre bouton Selection** : pour choisir le répertoire de sortie pour les résultats de segmentation.
3) **Un bouton Extraction des mots :** pour déclencher l'opération de segmentation en mots.

V.4.7. Interface de segmentation en caractères

Cette étape permet la segmentation d'une image d'un mot en caractères. Elle est représentée par la figure suivante :

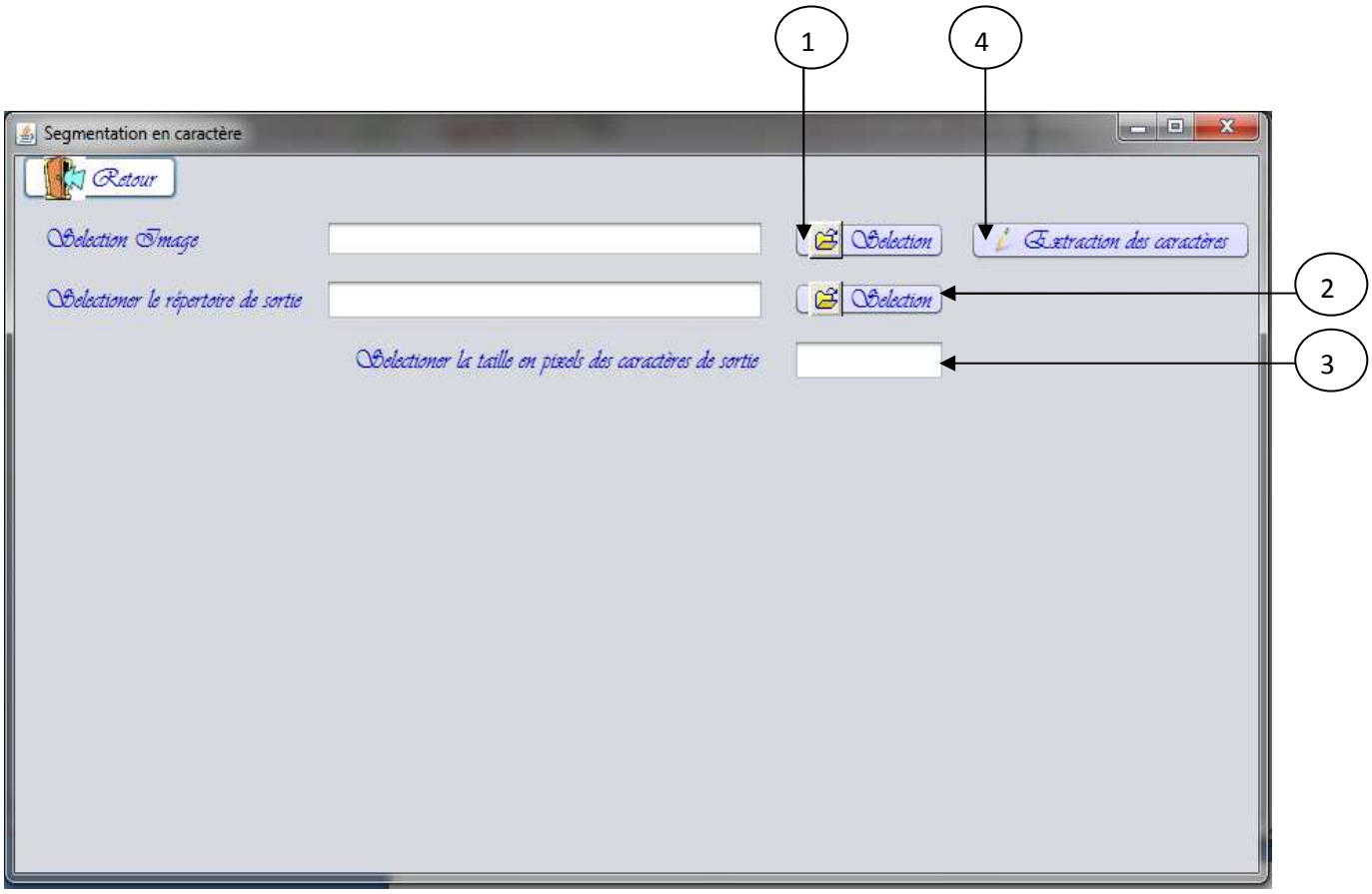


Figure V.8 : Interface de segmentation en caractères.

1) **Un bouton Selection :** pour charger l'image du mot à segmenter en caractères.
2) **Un autre bouton Selection :** pour choisir le répertoire de sortie pour les résultats de la segmentation.
3) **Une zone de texte :** pour saisir la taille en pixels des caractères de sortie.
4) **Un bouton Extraction des caractères :** pour déclencher l'opération de segmentation en caractères.

V.4.8. Interface de reconnaissance d'un mot

Cette interface permet d'effectuer la reconnaissance d'un mot et est représentée par la figure suivante :

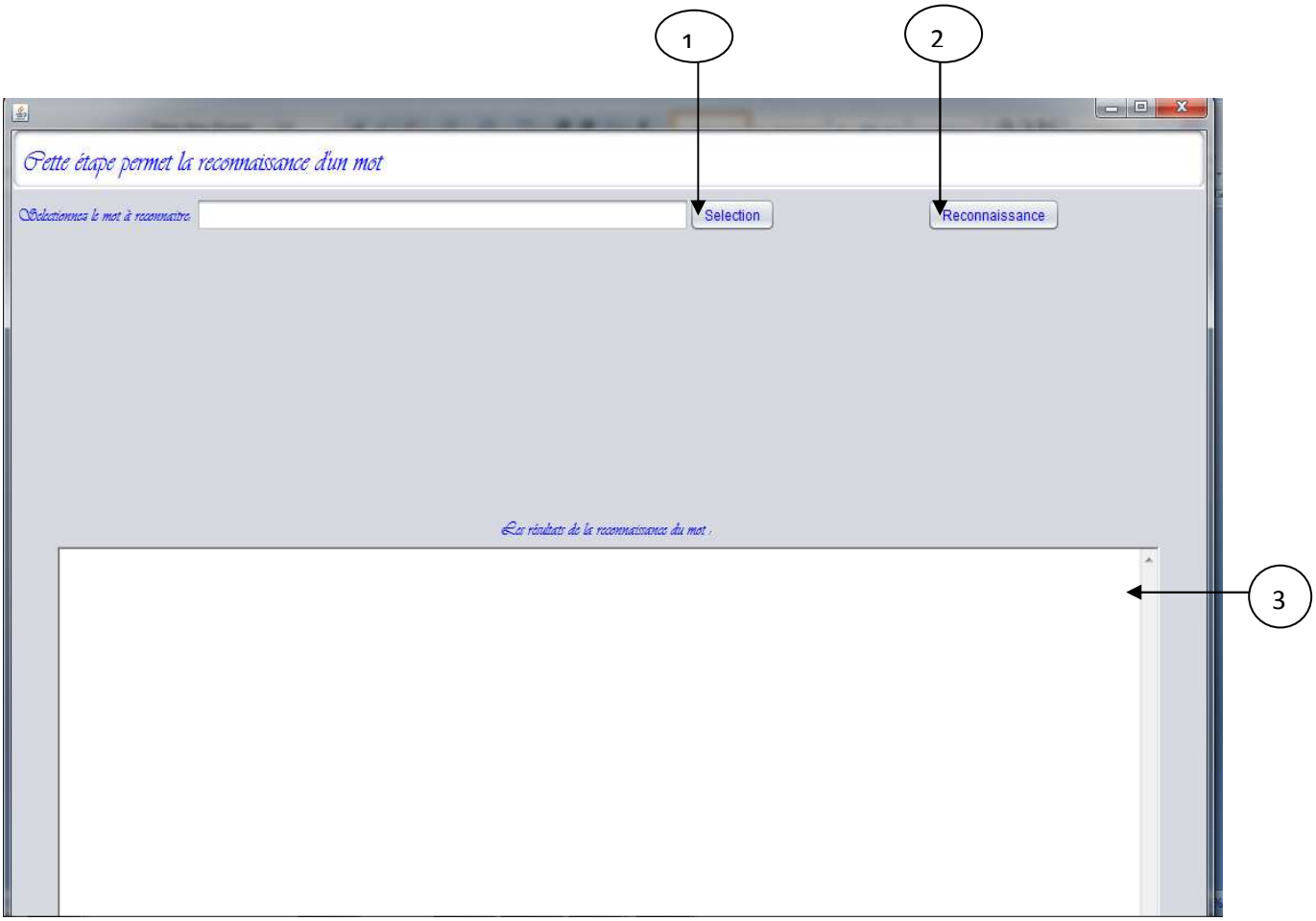


Figure V.9 : Interface de reconnaissance d'un mot.

1) **Un bouton Selection :** pour charger l'image du mot à reconnaitre.
2) **Un bouton Reconnaissance :** pour déclencher l'opération de reconnaissance d'un mot.
3) **Une zone de texte :** pour afficher les résultats de la reconnaissance du mot.

V.5. Fonctionnement de notre système

Dans ce qui suit, nous présenterons le déroulement d'un exemple sur notre système afin de comprendre son fonctionnement.

V.5.1. Sous système d'apprentissage

Cette étape consiste à faire apprendre au système les propriétés et caractéristiques des différentes lettres arabes des manuscrits dont nous avons parlé dans le chapitre 2 (page 98). Chaque lettre est représentée dans la base de données selon son type (chaque position différente pour un caractère dans un mot : début, milieu, fin ou isolée constitue un type) ainsi que les résultats de son apprentissage.

Mais avant tout apprentissage la lettre doit d'abord passée par un prétraitement et l'extraction de ses primitives.

Dans la suite nous présenterons un test sur la lettre « ma ».

a. Chargement de l'image de la lettre à prétraiter.

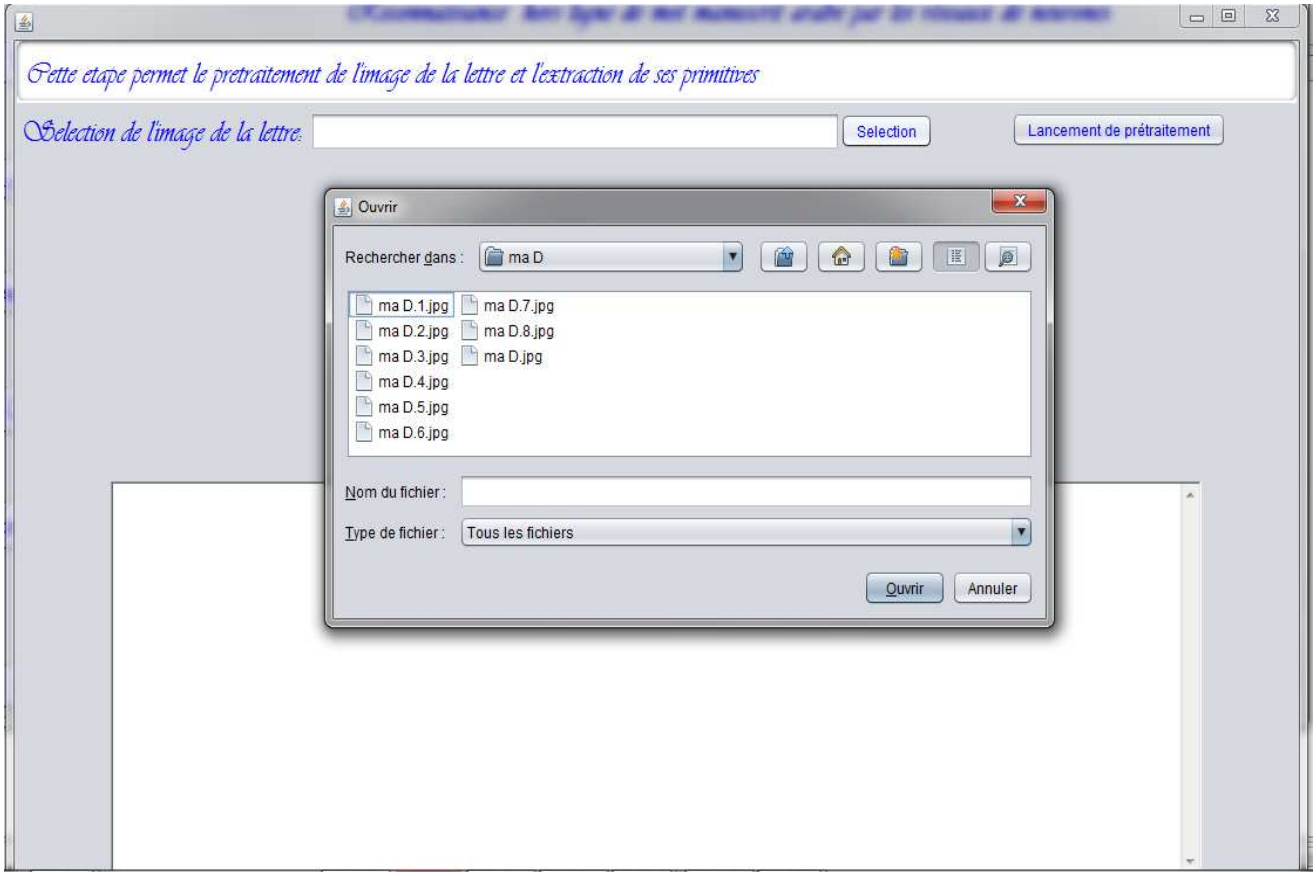


Figure V.10 : chargement de l'image de la lettre « ma D ».

b. Prétraitement et extraction des primitives

Ces opérations sont déclenchées en cliquant sur le bouton de « lancement du prétraitement », les résultats de cette étape sont l'image prétraitée affichée à l'écran et la matrice d'observations de la lettre affichée dans la zone de texte. La figure suivante illustre cet exemple.

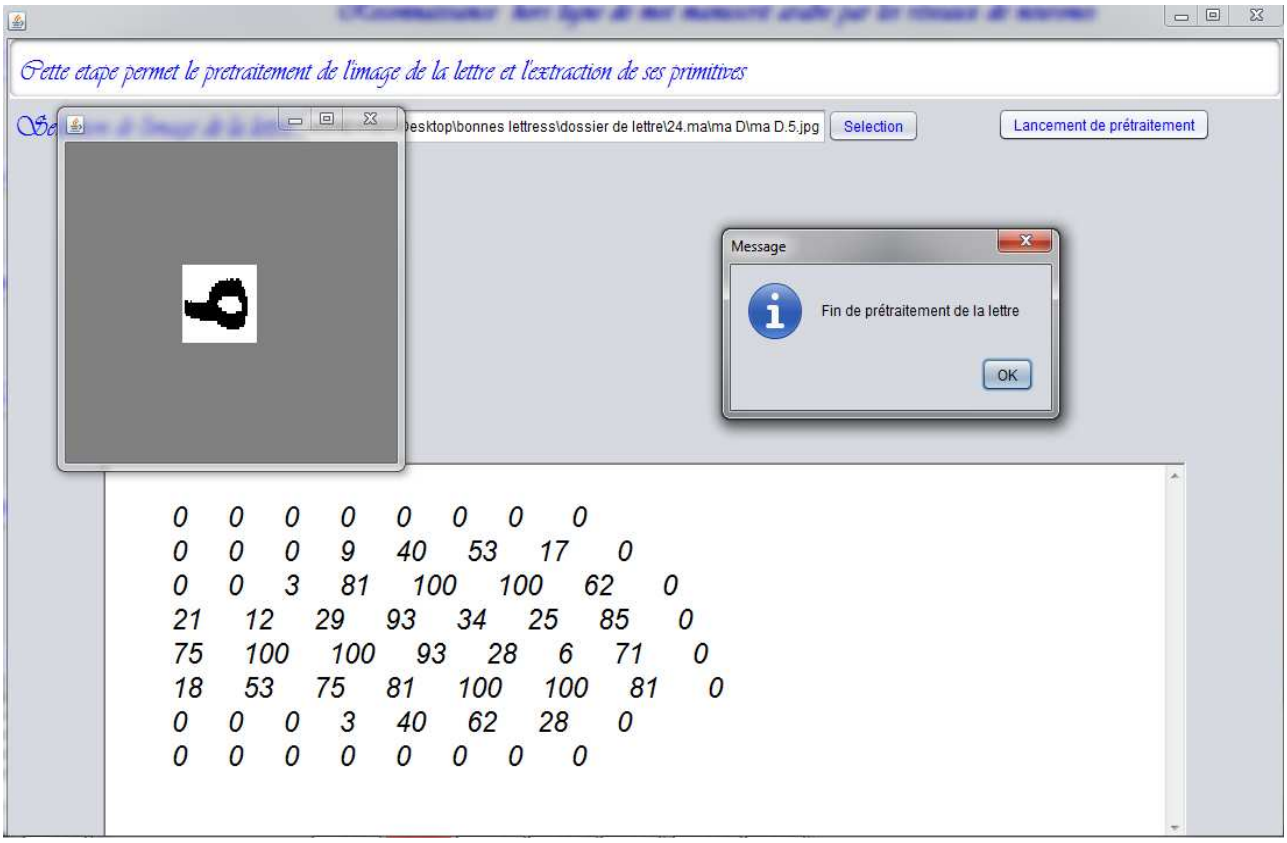


Figure V.11 : Les résultats de prétraitement et extraction des primitives de la lettre « ma D ».

c. L'apprentissage

Cette opération est effectuée après la configuration des paramètres du réseau de neurones artificiel cités en haut, le chargement de l'image de la lettre et la sélection de son type dans la liste de choix.

Les résultats de l'apprentissage consistent en l'erreur locale minimale de chaque itération[1], les éléments du vecteur ligne calculés pour chaque élément du vecteur d'entrée. Ces résultats seront enregistrés dans la Base de données pour représenter la lettre « ma D ».

La figure suivant illustre le résultat de l'apprentissage de la lettre « ma D ».

[1] C'est à noter que nous avons mis comme condition d'arrêt des itérations la suivante : la boucle d'exécution des itérations ne s'arrêtera pas jusqu'à atteindre une erreur minimale locale inférieure à 0.01 et le nombre d'itérations doit ne pas dépasser les 1000, pour éviter les exécutions très longues.

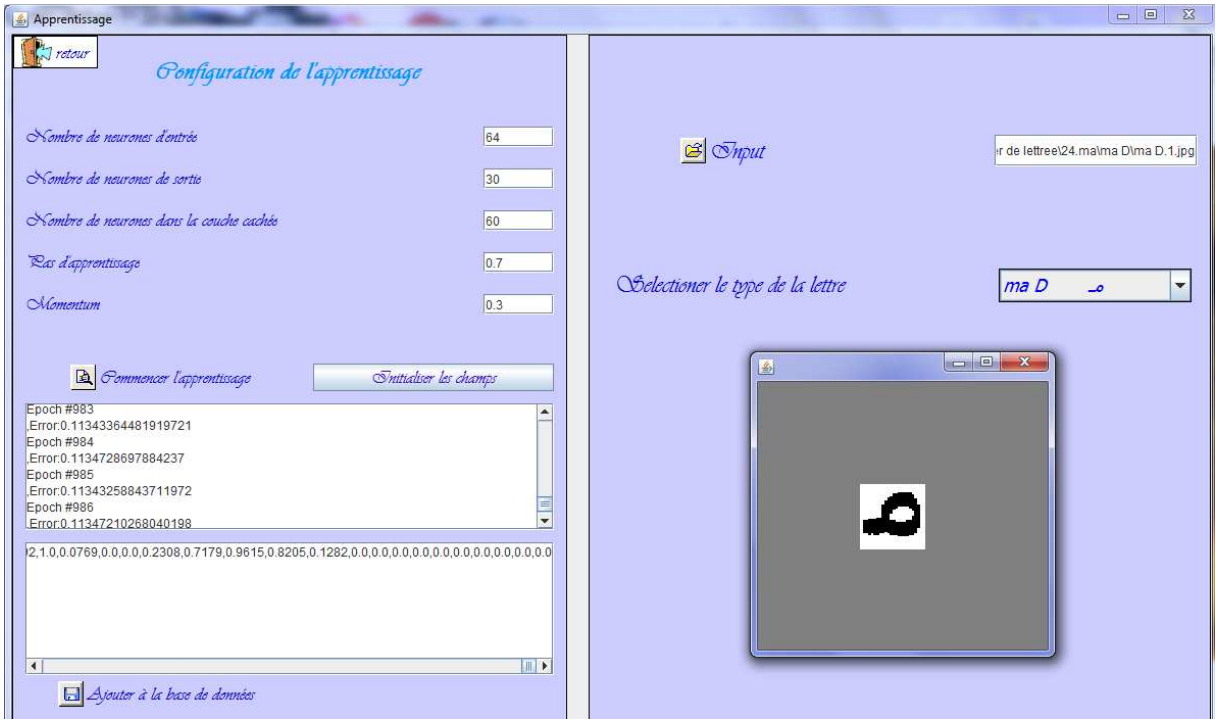


Figure V.12 : les résultats de l'apprentissage de la lettre « ma D ».

V.5.2. Sous système de reconnaissance

Le but de cette étape est la reconnaissance de mots. Afin de reconnaitre un mot, on doit d'abord passer par l'extraction de ce mot à partir de l'image où il est contenu, s'il figure sur image de lignes qui contiennent plusieurs mots (donc dans ce cas on devra passer par la segmentation en lignes puis en mots pour l'extraire), sinon on passe directement à la segmentation en caractères. Après la segmentation en caractères, on passe à la reconnaissance de ces caractères pour reconnaitre le mot en entier.

Dans ce qui suit nous présenterons le déroulement d'un exemple de reconnaissance d'un mot.

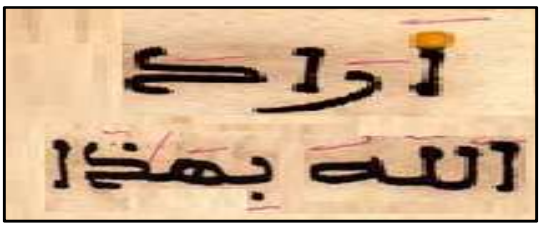


Figure V.13 : image de lignes contenant le mot à reconnaitre.

a. Segmentation en lignes

Après le chargement de l'image de lignes, on passe à l'extraction de ses lignes en cliquant sur le bouton « extraction des lignes » la segmentation sera effectuée. Les résultats seront sauvegardés dans un dossier créé par l'utilisateur afin d'être utilisé dans la segmentation en caractères.

Les résultats de la segmentation sont illustrés par la figure suivante :

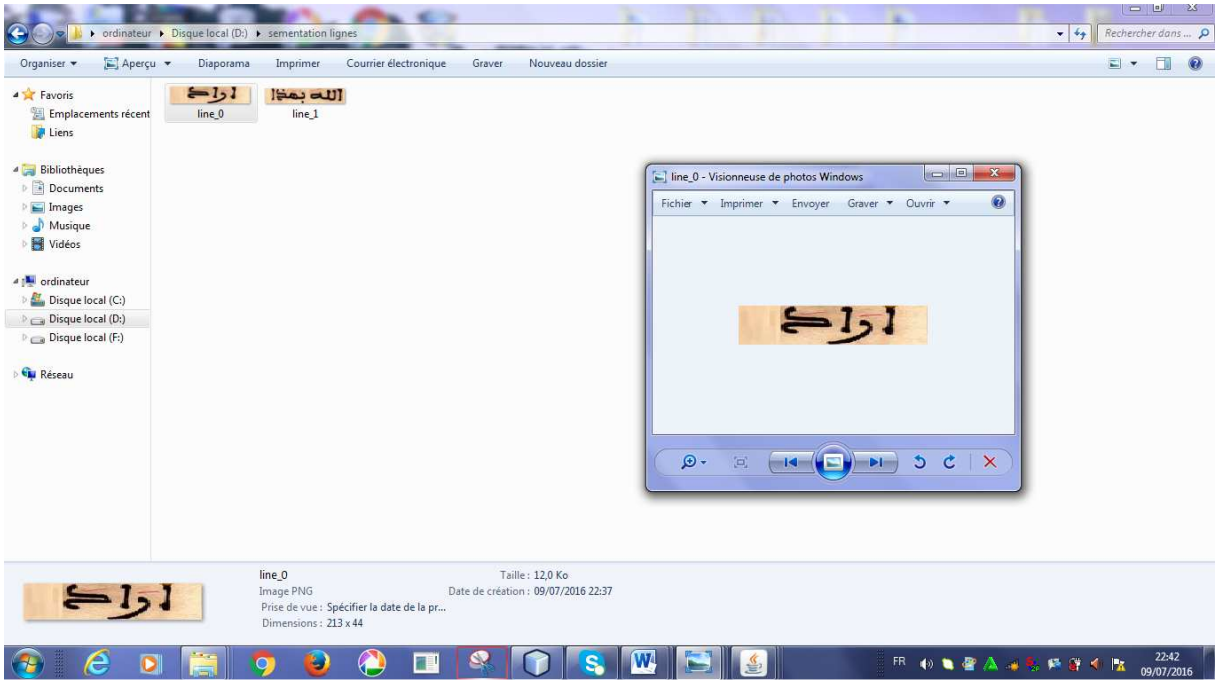


Figure V.14 : La segmentation en lignes.

b. Segmentation en caractères

Après le chargement de l'image du mot à segmenter en caractères, on passe à l'extraction de ses caractères en cliquant sur le bouton « extraction des caractères » la segmentation sera effectuée. Les résultats seront sauvegardés dans un dossier créé par l'utilisateur afin d'être utilisé pour la reconnaissance du mot.

Les résultats de la segmentation du premier mot sont illustrés par la figure suivante :

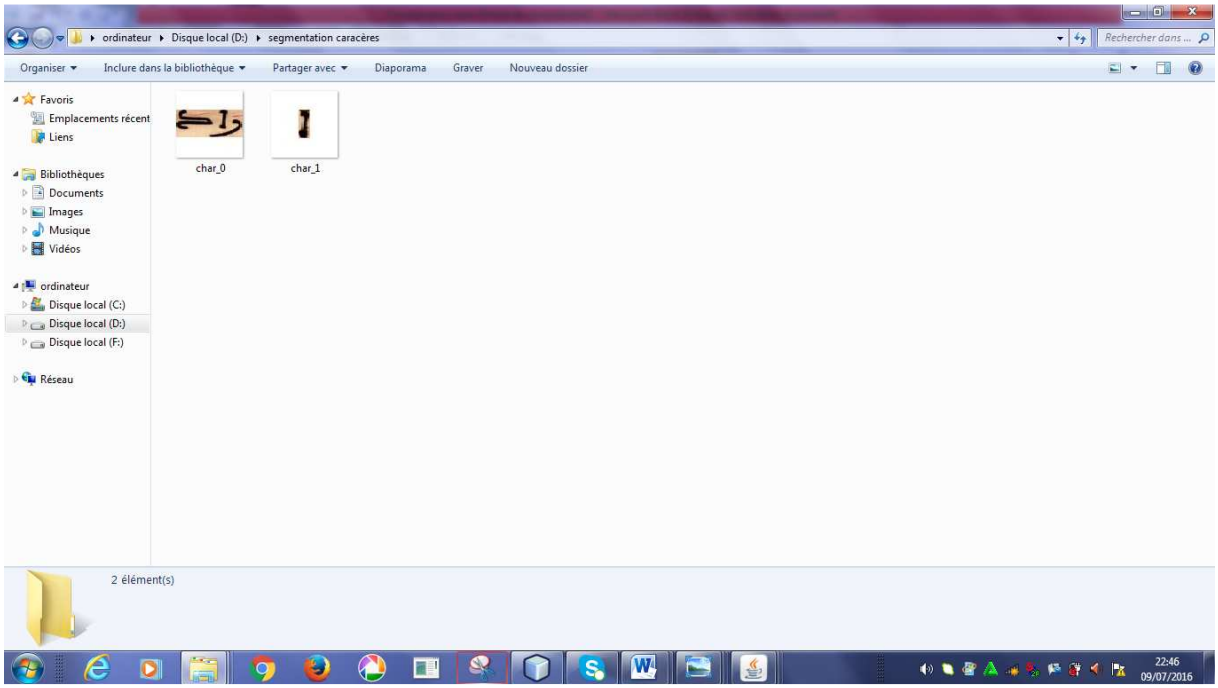


Figure V.15 :Résultats de l'extraction des caractères d'un mot.

Comme vous pouvez le constater, le mot n'a pas été entièrement segmenté en caractères. Ceci est dû à la complexité de cette écriture manuscrite ainsi qu'à la qualité des images qui les contiennent.

c. La reconnaissance des caractères

Cette étape permet la reconnaissance des caractèresdonnés en entrée.

Après le chargement de la lettre à reconnaitre, on passe à sa reconnaissance en cliquant sur le bouton« reconnaissance », l'opération de reconnaissance sera effectuée et le résultat sera affichédans la zone de texte :

- La lettre : elle représente la lettre reconnue en arabe.
- La lettre correspond à l'image : elle représente le nom de la lettre reconnue.

La figure suivante représente le résultat de la reconnaissance de la lettre « alif D » extraite du mot donné dans la figure V.14.

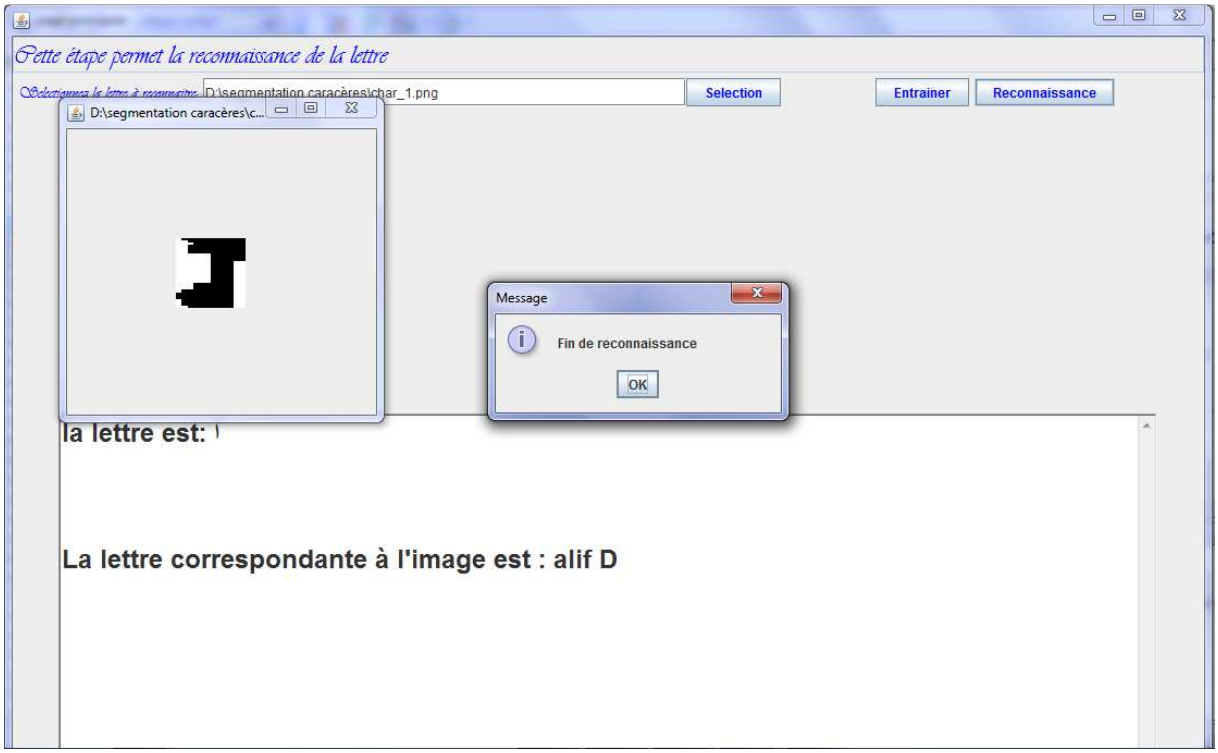


Figure V.16 : Résultats de la reconnaissance de la lettre « alif Début ».

d. Reconnaissance d'un mot

Cette étape doit permettre la reconnaissance d'un mot. Après le chargement de l'image du mot, on clique sur le bouton «Reconnaissance», et l'opération de reconnaissance sera effectuée, le résultat sera affiché dans la zone de texte.
Le résultat de la reconnaissance du mot est montré dans la figure suivante.

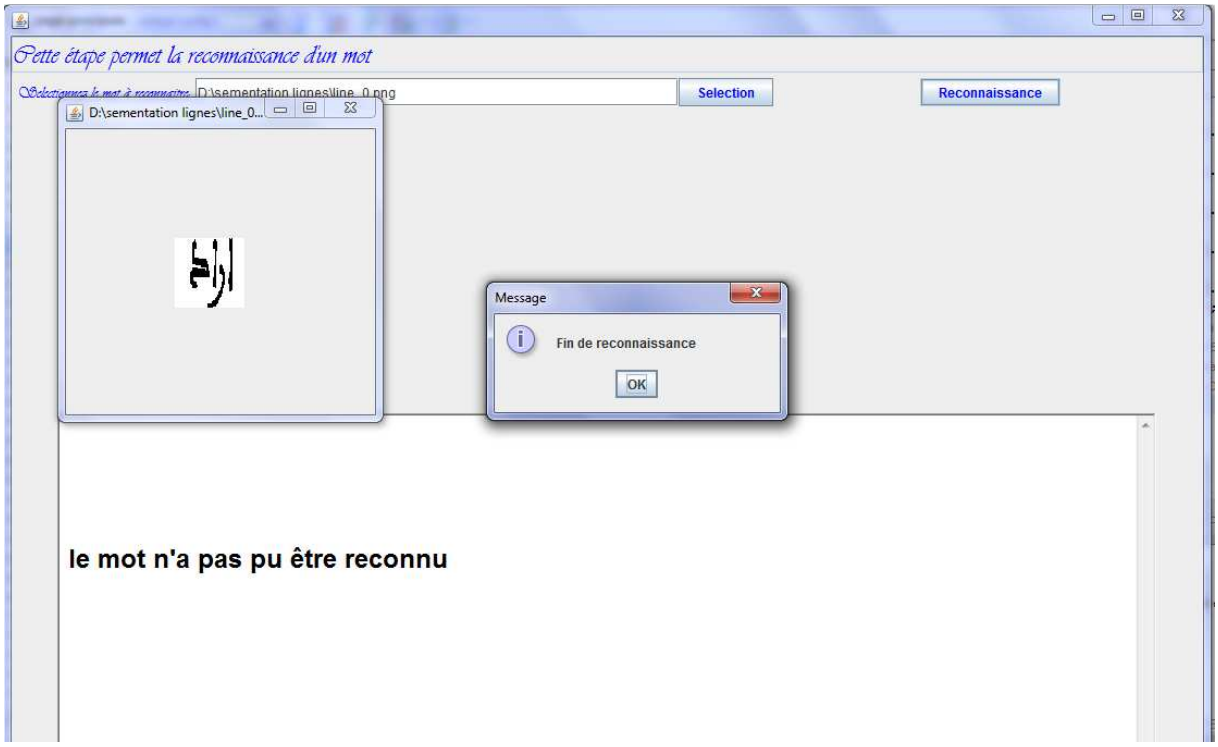


Figure V.17 : Résultats de la reconnaissance du mot.

Bien-sûre, le mot n'est pas reconnu à la fin et c'est normal, car le mot n'a pas bien été segmenté en caractères, ceci a conduit à reconnaitre que la première lettre du mot (alif Début), or la reconnaissance d'un mot en entier nécessite la reconnaissance de chaque lettre composant ce mot.

Remarque :

Le test de la reconnaissance appliqué sur d'autres mots a conduit au même résultat.

V.6. Résultats Expérimentaux

Dans le but d'évaluer notre système de reconnaissance de mots manuscritsarabes, nous avons effectué des expérimentations sur la tâche reconnaissance des lettres arabes manuscrites et sur lestâches de segmentation en lignes, segmentation en mots et de segmentationen caractères. Pour enfin passer à la tâche de reconnaissance de mots puisque celle-ci est très dépendante de la segmentation de mots en caractères et de la reconnaissance de ces derniers.
Les résultats de l'évaluation de chaque tache sont décrits ci-dessous.

V.6.1. Tâche de la reconnaissance des lettres arabes manuscrites

La performance de la reconnaissance de lettres peut être mesurée en calculant le taux de reconnaissance et le taux de substitution[2]:

- **Taux de reconnaissance=** Nombre de lettres reconnues / Nombre total de lettres.
- **Taux de substitution =** Nombre de lettres males reconnues / Nombre total de lettres.

[2]Mauvaise reconnaissance.

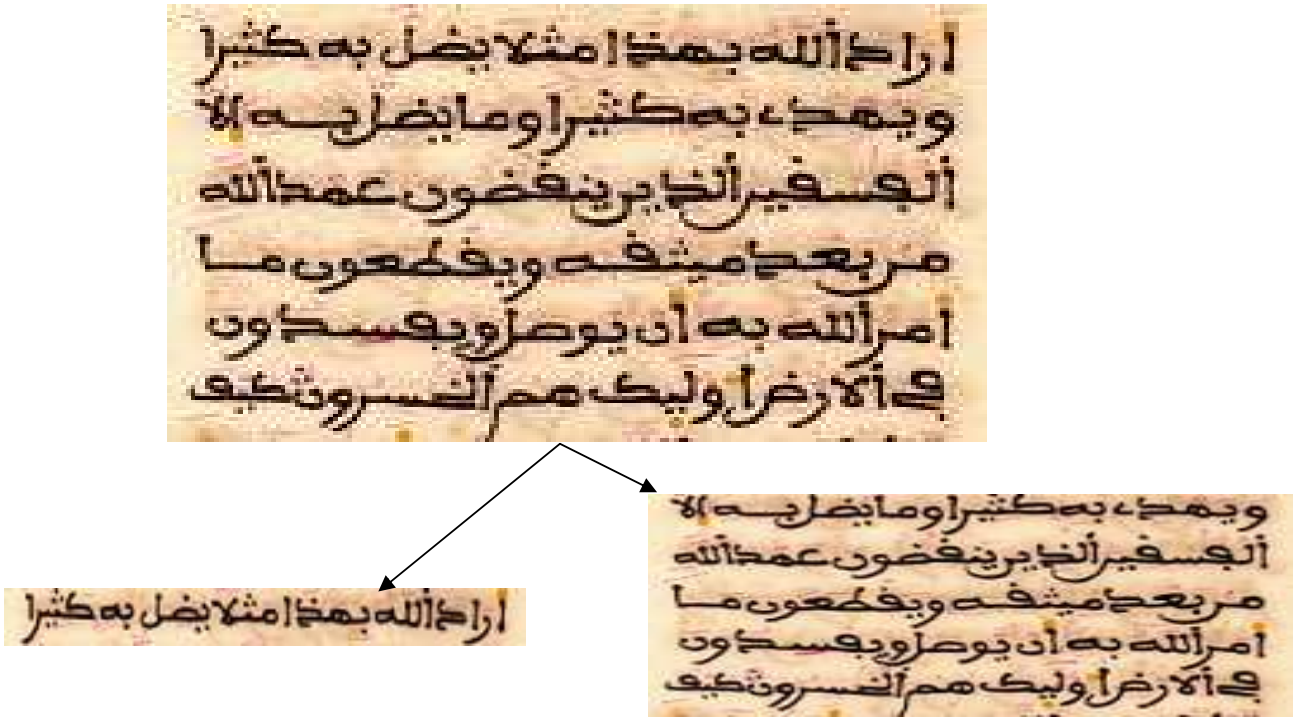
Après avoir testé la reconnaissance des lettres sur notre base de test composée de 70types de lettres arabes différentes, nous avons obtenu respectivement les taux de reconnaissance et de substitutionsuivants : 70% et 30%.
D'après nos tests, nous avons constaté que notre système fait des confusions entre quelqueslettres, comme par exemple : « kha Début » et « ha début », « tha début » et « ta début » « ssa début » et « ma début », « cha Milieu » et « sa Milieu », « dha Fin » et « da Fin », «qa Milieu » et « fa Milieu ».
Ceci est dû aux lettres qui se ressemblent en écriture, donc à l'étape de l'extraction de primitives qui est basé sur le calcule de pourcentage de pixels dans les zones de l'image, ces lettres sont confondues.

V.6.2. Tâche de la segmentation

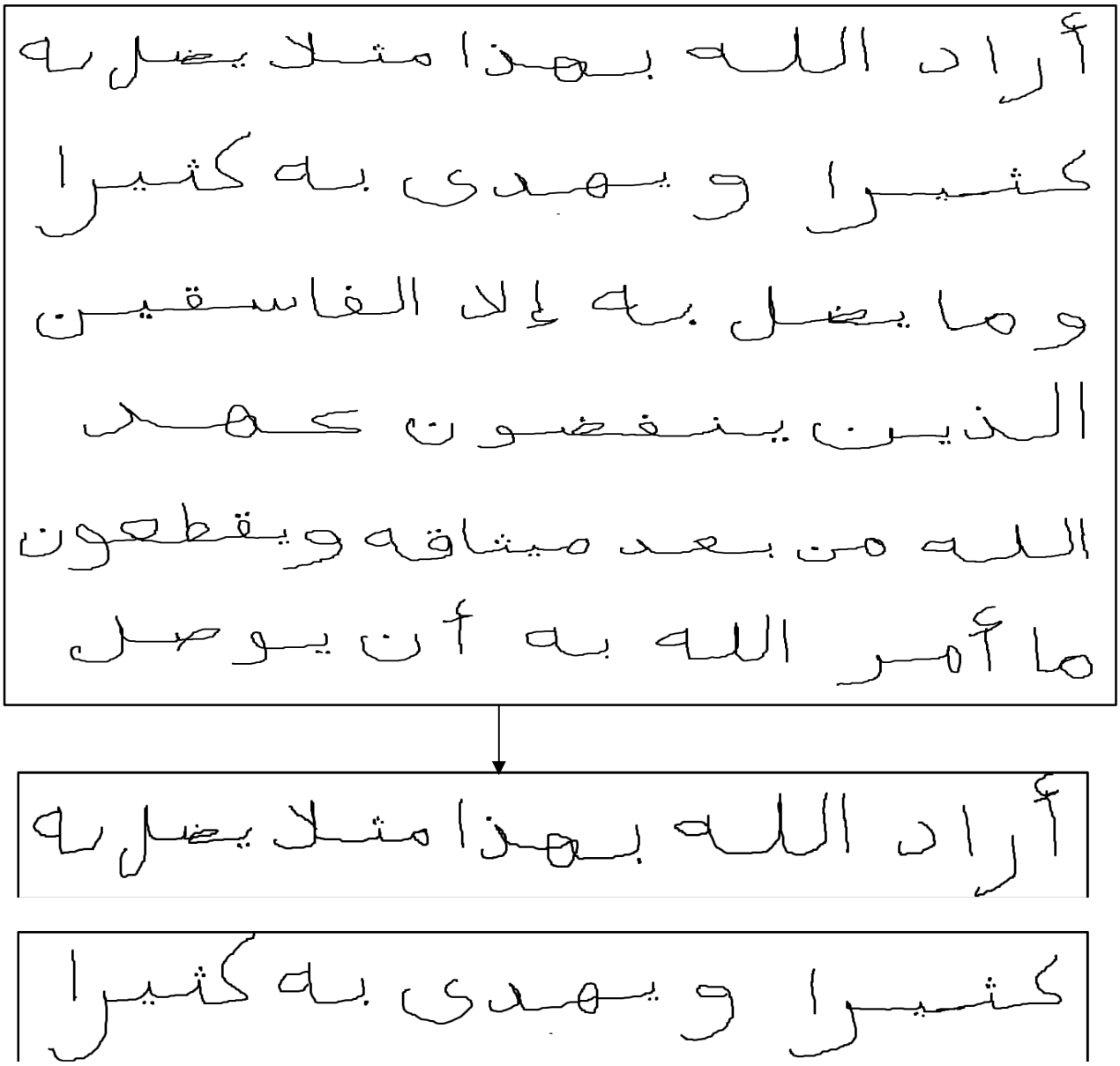
V.6.2.1. Segmentation des lignes

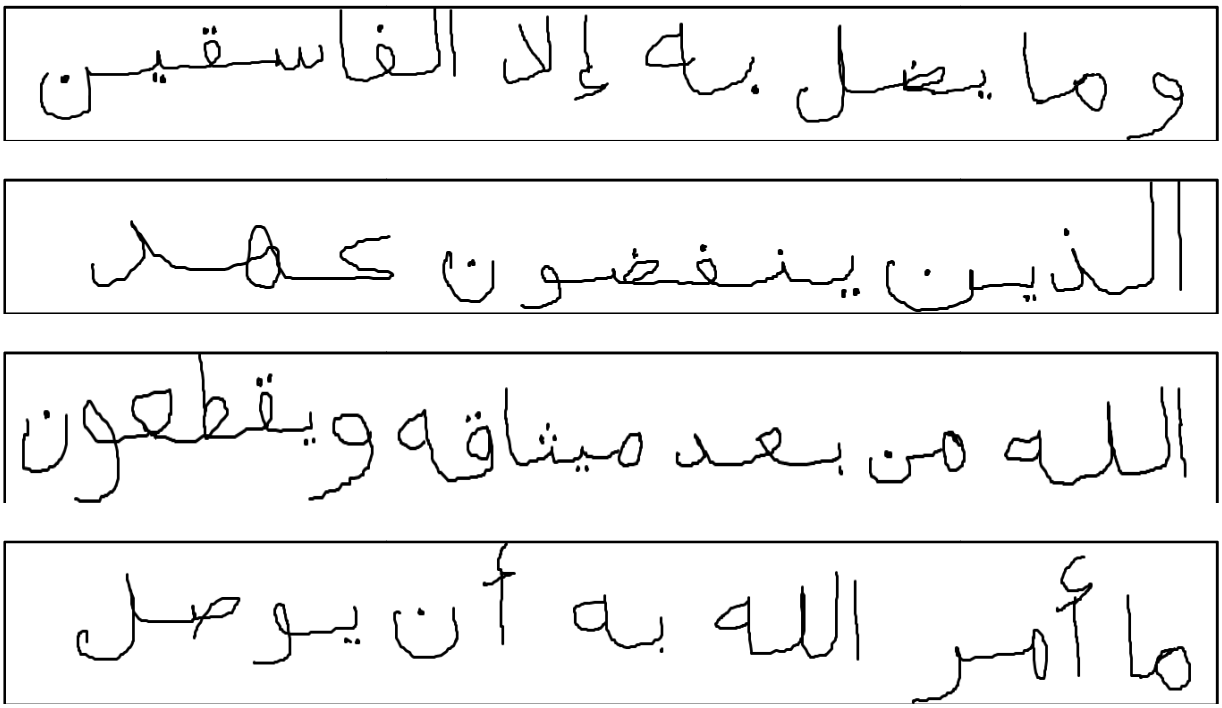
La performance de la segmentation en lignes est calculée comme suit :
Taux de segmentation des lignes = nombre de lignes bien segmentés / sur le nombre total de lignes.
Nous avons testé la segmentation des lignes sur des images manuscrites anciennes et sur des images manuscrites introduites par nos même (en utilisant l'interface « paint », et des images scannées écrites avec un stylo) ceci pour des images ayant toutes le même nombre de lignes,nous avons obtenu respectivement pour les images manuscrites anciennes et les images manuscrites récentes les taux de segmentation suivants:16.66% et 100%.

Exemple :
La segmentation en lignes de l'image manuscrite ancienne suivante, composée de 6 lignes, nous a donné le résultat suivant :



D’après nos tests, nous avons remarqués que les images contenant des lignesdont les caractères comportent des extensions qui touchent ou presque aux caractères de la ligne au-dessus ou au-dessous sont mal segmentées (l’espace interligne est chevauché).
La performance de notre segmentation dépend aussi de la taille de l’image considérée, tel que, dans certains cas où la taille de l’image est grande, les lignes seront mal segmentées.
Pour vérifier la validité de notre conclusion, nous avons saisie des images sur l’interface « paint » contenant les mêmes mots et même nombre de lignes (6), les résultats sont les suivants :





Toutes les lignes de notre image sont bien segmentées, ceci confirme donc que le problème n'est pas lié à notre système mais plutôt à la qualité des images prises, ainsi qu'à la qualité de leurs écritures.

V.6.2.2. Segmentation en caractères

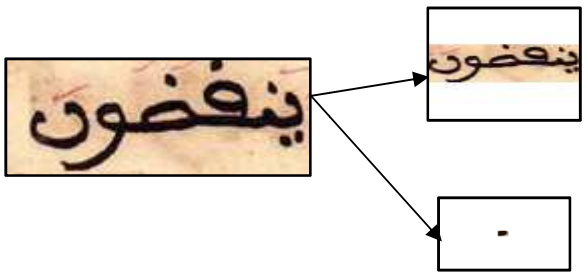
La performance de la segmentation en caractères est calculée comme suit :

Taux de segmentation en caractères = nombre de mots bien segmentés / sur le nombre total de mots

Nous avons testé la segmentation sur une base de teste composée de 10 mots arabes manuscrits anciens et 10 mots arabes manuscrits saisis avec « paint », nous avons obtenu respectivement les taux de segmentation suivants : 0%, 70%.

Exemple :

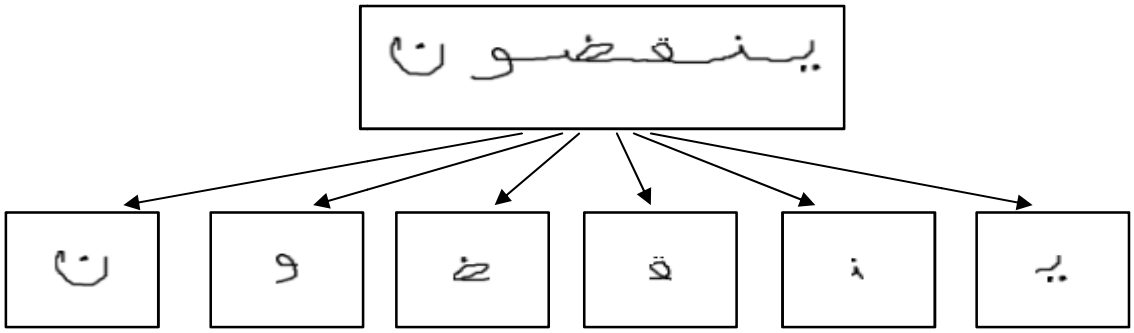
Le résultat de la segmentation d'un mot arabe extraitd'une image de texte arabe ancien est le suivant :



D'après nos tests, nous avons remarqué que les mots contenant des lettres chevauchées et ligaturées sont mal segmentés.

La performance de notre segmentation est dépendante aussi de la hauteur des mots, tel que, dans certains cas où la hauteur du mot est grande le mot sera mal segmenté ou ne sera pas segmenté du tout.

Le résultat de la segmentation du même mot que nous avons écrit avec « paint » nous a donné le résultat suivant :



V.6.3. Tâche de reconnaissance des mots

La performance de reconnaissance des mots est dépendante de la segmentation en caractèreset la reconnaissance des lettres.

Nous avons effectué des tests de reconnaissance des mots sur la base des mots utilisée dans la segmentation en caractères. Nous n'avons obtenu aucun mot reconnu, ce qui est normal vue les résultats de la segmentation en caractères de ces mot manuscrits anciens.

V.7. Comparaison des résultats obtenus dans la reconnaissance de mots manuscrits arabes en utilisant les réseaux de neurones aux résultats obtenus en utilisant le modèle de Markov caché (MMC)

La réalisation du même système de reconnaissance de mots manuscrits arabes (en prenant le même vocabulaire), mais en utilisant le modèle de Markov caché[3] réalisé par nos camarades a donné un taux de reconnaissance de caractères de 77%, et le même taux de reconnaissance de mots que celui donné par notre système (0%). Ces résultats qualifient donc les modèles de Markov cachés comme plus performants dans la phase de reconnaissance des caractères pour le cas étudié.

V.8. Conclusion

Dans ce chapitre, nous avons présenté notre système et nous avons décri son fonctionnement. Ceci en présentant les différentes interfaces aux quelles notre système fait appel, et en illustrant des exemples de déroulement de chaque étape. Enfin, nous avons évalué la performance de notre système pour chaque opération incluse. Nous pouvons dire en résumé que les résultats obtenus sont encourageants pour la reconnaissance des caractères, mais plus d'études, analyse et améliorations se portent nécessaires en ce qui concerne la reconnaissance des mots manuscrits arabes anciens, et ceci en raison de la complexité de l'écriture de ces mots.

[3] Les modèles de Markov cachés sont introduits par Baum et All dans les années 60. Ils sont des modèles stochastiques, largement utilisés en reconnaissance de la parole et plus tardivement en reconnaissance de l'écrit. Les MMC permettent de calculer la probabilité d'appartenance d'une forme à une classe en fonction de degré de distorsion subi par cette forme.

Conclusion Générale

Conclusion générale

L'objectif de notre travail était la réalisation d'un système de reconnaissance hors ligne de mots manuscrits arabes anciens.

Pour mener à terme notre travail, nous avons proposé l'utilisation de l'approche des réseaux de neurones artificiels dans les phases d'apprentissage et de reconnaissance.

La reconnaissance hors-ligne est un sujet qui a beaucoup de difficulté ce qui a mené plusieurs chercheurs à conduire plusieurs travaux pour remédier au problème de la reconnaissance.

Le problème posédans la reconnaissance pour les approches existantes est l'opération de la segmentation. Pour remédier à ce dernier nous avons utilisé un algorithme qui permet la segmentation de mot en caractères en éliminant les éléments d'épaisseur uniforme qui relient ces caractères, mais vue la complexité de l'écriture des manuscrits étudiés ceci s'est révélé insuffisant.

Nous avons implémenté notre système en langage Java. Le prototype réalisé respecte l'architecture que nous avons proposée pour la reconnaissance. Pour évaluer la performance de notre système en termes de précision de reconnaissance, nous avons mené une étude expérimentale portant sur la segmentation, la reconnaissance de caractères et la reconnaissance de mots. Les résultats obtenus sont prometteurs parlant de la tâche de reconnaissance des caractères. Mais le cas n'est pas le même pour la tâche de segmentation des mots en caractères, qui associée avec la tâche de reconnaissance des lettres représentent les étapes les plus cruciales pour l'aboutissement à la reconnaissance d'un mot complet. Donc ceci a conduit directement à la reconnaissance d'aucun mot des échantillons des mots testés, ce qui laisse l'intervalle ouvert pour plus d'études, analyse et améliorations concernant ce type de manuscrits, leurs segmentation en caractères et enfin leur reconnaissance.

Comme les résultats de la reconnaissance des caractères pris du même vocabulaire, mais réalisé par les modèles de Markov cachés se sont avérés encore plus prometteurs, à l'avenir nous proposons une combinaison des méthodes (à savoir la méthode des réseaux de neurones et celle de modèles de Markov cachés) pour améliorer le taux de la reconnaissance de tel type de caractères.

Références Bibliographiques

Références bibliographique :

[1] : MEMOIRE Présenté par Chérif TAOUCHE Pour l'obtention du diplôme de Magister en Informatique Option: Information & Computation ; Thème : Implémentation d'un Environnement Parallèle pour la Compression d'Images à l'aide des Fractales 2005, Université Mentouri Constantine Faculté des Sciences de l'Ingénieur Département d'Informatique

[2] :http://dspace.univ-tlemcen.dz/bitstream/112/5813/1/Developpement-dune application-de-traitement-dimages.pdf
Mémoire de fin d'études pour l'obtention du diplôme de Licence en Informatique Thème : Développement d'une application de traitement d'images Juin 2013, Université Abou BakrBelkaid– Tlemcen Faculté des Sciences Département d'Informatique

[3] :http://raphael.isdant.free.fr/traitement_numerique/2traitement_numerique_de_l'image.pdf

[4] : http://www.pixel-dargent-74.fr/documents/base.pdf.

[5] : http://www.ac-nice.fr/iencagnes/file/tice/tuto/Image_numerique.pdf.

[6] : Mémoire UMMTO ING.49.2010/1- Reconnaissance en ligne de chiffres manuscrits isolés par les réseaux de neurones.

[7] :http://www.ac-nice.fr/iencagnes/file/tice/tutoral/Images_numerique.pdf.

[8] :http ://www.image_pixel.org

[9] : http://www.Introduction à L'intelligence Artificielle- http://www.iro.umontreal.ca/-aimeur.

[10] :http://livre.fnac.com/a1073396/N-J-Nilsson-Principes-de-l-intelligence-artificielle.
N. NILSSON, Principes d'Intelligence artificielle, Cépadues, 1988.

[11]: J. Cao, M. Ahmadi and M. Shridar, A Hierarchical Neural Network Architecture For Handwritten Numeral Recognition (Pattern Recognition.Vol. 30, 1997).

[12] : Essoukhri Ben Amara, N., Belaïd, A., Ellouze, N.: Utilisation des modèles markoviens en reconnaissance de l'écriture arabe: Etat de l'art, CIFED'2000, Colloque International Francophone sur l'Ecrit et le Document, pp.181-191, Lyon, France, 2000.

[13] : S. Haitaamar :" segmentation de texte en caractère pour le reconnaissance optique de l'écriture arabe ".Université EL-HADJ LAKHDHAR Batna, Juillet 2007.

[14] :http://www.fsjegj.rnu.tn/Pages/bibliotheque/memoire_dea_informatique.htm

MEMOIRE DE MASTERE DONNÉES, CONNAISSANCES ET SYSTÉMES DISTRIBUÉS par Riadh BOUSLIMI Sujet Système de reconnaissance hors-ligne des mots manuscrits arabe pour multi scripteurs.

Références bibliographique :

[15] : A. Zahour, B.Taconet, A.Foure : « Méthode structurelle de reconnaissance de l'écriture arabe manuscrite ». Bigre n 68, Actes du 1er colloque national sur l'écrit et le document (CNED '90), pp.148-159, Nancy, 1990.

[16] : N. Ayat : "Sélection De Modèle Automatique Des Machines À Vecteurs De Support: Application À La Reconnaissance D'images De Chiffres Manuscrit", thèse présentée à l'Ecole de Technologie Supérieure comme Exigence Partielle à l'obtention du Doctorat en génie P.H.D. Montréal, Le 20 Janvier 2004.

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

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

[19]:Shubair A et al.:" Off-line Arabic handwritten word segmentation using rotational invariant segments features ". The international Arab journal of information technology, Vol.5, No. 2, April 2008.

[20]: G. Stamon, N. Vincent :" Bio-Imagery – Image Analysis Document Analysis – Pattern Recognition ". Rapport d'activité SIP, Laboratoire CRIP5, Université Paris Descartes, France, Avril 2008.

[21] : A. Belaïd :" Reconnaissance automatique de l'écriture et du document ". LORIA-CNRS, Campus scientifique B.P. 239, 54506 Vandoeuvre-Lès-nancy, France.

[22] : S. Haitaamar :" segmentation de texte en caractère pour le reconnaissance optique de l'écriture arabe". Université EL-HADJ LAKHDHAR Batna, Juillet 2007.

[23]:Tappert, C.C., "Adaptive on-line handwriting recognition", Proceedings of ICPR'84, 7th International Conference on Pattern Recognition, pp. 1004-1007,Montreal, Canada, July/August 1984.

[24]: Al-Rashaideh H:" Preprocessing phase for Arabic Word Handwritten Recognition ". Institut d'informatique et autimatisme, Tom 6, N0 1,2006, cmp.11-19, Russie, February 26, 2006.

[25] : L. Robadey :" 2(CREM): Une methode de reconnaissance structurelle de documents complexes basee sur des patterns bidimensionnels ". Thèse de doctorat soumise à la Faculté des Sciences de l'Université de Fribourg, Suisse, 2001.

[26] : Reconnaissance automatique de l'écriture et du document
Abdel Belaïd LORIA-CNRS Campus scientifique B.P. 239 54506 Vandoeuvre-Lès-nancy - email:abelaid@loria.fr

Références bibliographique :

[27]: A. Madaan, R. Mehta:"Comparative Evaluation of Learning Algorithms on Hand written Character Data". Cours CS 698, Mars 2004.

[28]: J. Park :" Hierarchical character recognition and it's use in handwritten word/phrase recognition". Thèse de phd, Université de New York, Novembre 1999.

[29]: P. Burrow : « Arabic handwriting recognition ». Master of science thesis. School of Informatics, university of Edinburg, England,,2004.

[30]: N. Benamara : « Utilisation des modèles de Markov cachés planaires en reconnaissance de l'écriture arabe imprimée ». Thèse de doctorat, spécialité Génie Electrique, Université des sciences, des Techniques et de médecine de Tunis II, 1999.

[31]: T. Steinherz, E. Rivlin, N. Intrator: «Off-line cursive word recognition: a survey ». International journal on document analysis and recognition, 2(2), pp. 90-110, 1999.

[32] : A.Benouareth. Reconnaissance de Mots Arabes Manuscrits par Modèles de Markov Cachés à Durée d'Etat Explicite, Thèse de doctorat, Université Badji Mokhtar - Annaba V, 2007.

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

[34]: R.M. Bozinovic and 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, January, 1989.

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

[36]: 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, Algérie, Février 2006.

[37]: S. Kermi : « Classifieur neuronal base connaissances, application à la reconnaissance des caractères arabes isolés manuscrits ». Thèse de magister, université Badji Mokhtar, Annaba, Algerie 1999.

[38]: B. Al-Badr , S.A. Mahmoud : « Survey and bibliography of Arabicoptical text recognition ». Signal processing , vol. 41, pp. 49-77,1995.

[39]: B. Al-Badr, R.M. Haralick: « Symbol recognition without prior segmentation ». Conference SPIE-EI 1994.

Références bibliographique :

[40]: J. Anigbogu : « Reconnaissance de textes imprimés mutifontes àl'aide de modèles stochastiques et métriques ». thèse de doctorat,Université de Nancy I, 1992.

[41]: P. Burrow : « Arabic handwriting recognition ». Master of sciencethesis. School of Informatics, university of Edinburg, England,2004.

[42]: J.L. Amat, G. Yahiaoui : « Techniques avancées pour le traitement de l'information». Edition CEPADUES 1996.

[43]: L. Souici, Z. Zmirli, M. Sellami : « Système connexionniste pour la reconnaissance de l'arabe manuscrit ». 1ères journées scientifiques et techniques (JST FRANCIL), pp. 383-388, Avignon, France, 1997.

[44]:R.P. Lippmann: « An introduction to computing with neural nets ». IEEE, ASSP magazine, April 1987.

[45]: T.M. Ha, G. Kaufmann, H. Bunke: « Text localization and handwriting recognition». Technical report, university of Berne, 1996.

[46]: K. Seymore, A. McCallum, R. Rosenfeld: « Learning Hidden Markov model structure for information extraction ». AAAI. Workshop on machine learning for information extraction, pp. 37-42, 1999.

[47]:A.Belaid : "La reconnaissance automatique de l'écriture et du document", pour la Science, octobre 2001.

[48]: G. Saon : "Modèles Markoviens uni-bidimensionnels pour la reconnaissance de l'écriture manusc-rite hors-ligne". Thèse de doctorat, Université Henri Poincaré Nancy 1, 1997.

[49]: P. Smrž et al :" Off-line Recognition of Cursive HandwrittenCzechtext". Université de Masaryk, Février 1998.

[50]: Davalo E., Naim P., Des Réseaux de Neurones, EYROLLES, Deuxième édition, 1993.

[51]:Rosenblatt F., The Perceptron: a probabilistic model for information storage.

[52]:Rumelhart D., Hinton G., Williams R., Parallel Distributed Processing, MIT Press, Vol.1, Cambridge, 1986.

[53]:Watous R. L., Learning algorithms for connectionist networks: Applied gradient methodscof non lineair optimization. IEEE, First International Conference on Neural Networks, San Diego, California, 1987.

[54]: http://www.Réseaux de neurones artificièl-Wikipedia.html.

Références bibliographique :

[55]:http://lab.univbatna.dz/lpea/index.php?option=com_content&view=article&id=113:memoires-de-magister-soutenus&catid=33:theses-et-memoires&Itemid=31
MEMOIRE Présenté AU DEPARTEMENT DE MECANIQUE FACULTE DES SCIENCES DE L'INGENIEUR UNIVERSITE DE BATNA Pour L'obtention du diplôme de MAGISTERE EN GENIE MECANIQUE Option : Construction Mécanique Par CHEFRI BELKACEM- Thème : Etude du comportement non linéaire de l'endommagement sous sollicitation thermo mécanique des structures mécaniques 2014.

[56]: http://bu.umc.edu.dz/theses/electronique/DAD4448.pdf
MEMOIRE Présenté pour l'obtention du diplôme de Magister en électronique-Thème : Utilisation des réseaux de neurones dans l'estimation des paramètres de la distribution Ki-2 Non Centrale Gamma-Option Traitement du signal ParOTHMANIMARABOUT Farouk 2005 (UNIVERSITE DE MENTOURI-CONSTANTINEFACULTE DES SCIENCES DE L'INGENIEUR DEPARTEMENT D'ELECTRONIQUE).

[57]: http://www.lb.refer.org/memoires/215282dea_falou.pdf
Université de Reims – Université de Rennes – IRISA Ecole Polytechnique Fédérale de Lausanne DEA Modélisation et ingénierie du logiciel scientifique- Thème : Reconnaissance de caractères manuscrits par réseau de neurones Réalisé Par : Al FalouWassim Sous la Direction de : Dr. El-Eter Bassam Décembre 1998.

[58]: http://archipel.univ-toulouse.fr/ipac20/ipac.jsp?full=3100001~!721128~!0
Eric DAVALO, Patrick NAIM, Des réseaux de neurones. Deuxième édition, deuxième tirage 1993.

[60]: MEMOIRE DE MAGISTERE- Spécialité : Automatique – Option : Automatique de systèmes continus et productique-présenté par : M. HAMMOUCHE Sofiane- Thème : Identification d'un modèle fractionnaire à l'aide des réseaux de neurones 2012 (UMMTO – Tizi-Ouzou Faculté du Génie Electrique et Informatique Département Automatique .

[61]: Bourret P. Reggia J. Réseaux neuronaux une approche connexionniste de l'intelligence artificielle. Teknea , Toulouse Marseille - Barcelone.1999]

[62]: Zurada J M, introduction to artificial neural system. West Publishing Company, USA.1992.

[63]: Dayoff J.E., Neural Network Architectures: An introduction .Van Nostrand Reinhold, Newyork.1990.

[64]: Davalo E. et Naim P. Des réseaux de neurone. Deuxième Edition.1993

[65]: Nerrand O., Roussel-Ragot P., Personnaz L., Dreyfus G., Neural networks and nonlinear adaptive filtering : unifying concepts and new algorithms, Neural Computation, Vol.5,p.165-199, 1993.

[66]: Apprentissage automatique : les réseaux de neurones___http://www.grappa.univ lille3.fr/polys/apprentissage/sortie005.html.

Références bibliographique :

[67]: http://www.Perceptron — Wikipédia.htm.

[68]: « Neural network » Tutorial slides of Andrew Morre http://www.autonlab.org/tutorials/neural.html

[69]: http://www.grappa.univ-lille3.fr/polys/apprentissage/sortie005.html

[70]: INTRODUCTION AUX RESEAUX DE NEURONS Gérald PETITJEAN gerald.petitjean@eurodecision.com

[71]: Bishop C., Neural Networks for Pattern Recognition, Oxford University Press, 1995.

[72] : « Apprentissage à partir d'exemples » Notes de cours F. Denis & R. Gilleron – Lille 3 http://www.grappa.univ-lille3.fr/polys/apprentissage/

[73]: 2003s-20 Comment améliorer la capacité de généralisation des algorithmes d'apprentissage pour la prise de décisions financières. Nicolas Chapados, YoshuaBengio.

[74]: http://www.grappa.univ-lille3.fr/polys/apprentissage/sortie005.html

[75]: http://www.nnwj.de/sample-applet.html.

[76]:http://aass.oru.se/~lilien/ml/seminars/2007_03_12c-Markus_Ingvarsson-RPROP.pdf

[77]: Thèse de Rufin VanRullen : Une Premiere Vague de Potentiels d.Action, Une Premiere Vague Idée de la Scène Visuelle Université Paul Sabatier (192 p.) 2000.

[78] :Mémoire de fin d'étude : Conception et Réalisation d'une application Client/serveur. Cas : Gestion commercial de l'ENIEM.

[79] : http ://www.netbeans.org

[80] : bouraoui@dpt-info.u-strasbg.fr

[81] : http://java.sun.com

[82] : Grand livre PHP 4 & MYSQL Edition Micro Application, 2000.

[83] : Elise Gabarra, « De la binarisation de document vers la reconnaissance de symboles dans l'analyse de schéma électriques », thèse de doctorat, université de Pau et des Pays de l'Adour, 2008.

[84] : Abderrahmane kefali, Toufik Sariet et Mokhtar Sellami, « Evaluation de plusieurstechniques de seuillage d'images de documents arabes anciens », Laboratoire de

Références bibliographique :

Recherche en Informatique (LRI), Département Informatique, BP 12 – Université Annaba, 23000, Algérie, 2000.

[85]:S. Impedovo, R. Modugno et D. Impedovo, «Advancements in Handwriting Recognition », Département Informatique, Université de degliStudi de Bari, 1999

[86] :Cinthia O. A. Freitas, Luiz S. Oliveira, Simone B. K. Aires et FlávioBortolozzi, « Metaclasses and Zoning Mechanism Applied to Handwriting Recognition», Pontifical Catholic University of Paraná – PUCPR, Brazil, 2008

[87]: Cinthia O. A. Freitas, Luiz S. Oliveira etFlavioBortolozzi, «Handwritten Character Recognition Using Nonsymmetrical Perceptual Zoning», Pontifical Catholic University of Parana (PUCPR), 2007

[88] : Najoua Ben Amara, Abdel Belaïd et Noureddine Ellouze, « utilisation des modèles markoviens en reconnaissance de l'écriture arabe : Etat de l'art », 1'école nationale d'ingénieurs de Monastir - 5019 Monastir – Tunisie, 2003.

[89] : Mémoire présenté par : Ben belkacem Farida et Amalou Kenza pour l'obtention du diplôme de Master académique en informatique. Option: systèmes informatiques. Thème : Reconnaissance automatique des noms arabes manuscrits anciens, Université de Mouloud Mammeri de Tiz-Ouzou. Promotion : 2010/2011.