Abstarct

The automatic analysis and recognition of Arabic handwritten characters from images is an important problem in many applications. Even with the great progress of recent research in optical character recognition, a few problems still wait to be solved, especially for Arabic characters. The emergence of Deep Neural Networks promises a strong solution to some of these problems. We present a deep neural network for the handwritten Arabic character recognition problem that uses convolutional neural network (CNN) models with regularization parameters such as batch normalization to prevent overfitting. We applied the Deep CNN for the AHDD , IESK and the AHCD databases and the classification accuracies for the three datasets were 96%, 78% and 94%, respectively. Second model based on Capsule Neural Network (CapsNet) was also proposed, the aim of which is to improve the accuracy of recognition system. the results show that our model based on CapsNet improve the accuracy of our proposed system and outperform the first model ,based on CNN.

Résumé

L'analyse et la reconnaissance automatiques des caractères manuscrits arabes à partir d'images est un problème important dans de nombreuses applications. Même avec les grands progrès de la recherche récente en matière de reconnaissance optique de caractères, quelques problèmes attendent encore d'être résolus, en particulier pour les caractères arabes. L'émergence des réseaux neuronaux profonds promet une solution solide à certains de ces problèmes. Nous présentons dans notre travail un réseau neuronal profond pour le problème de la reconnaissance des caractères arabes manuscrits qui utilise des modèles de réseaux neuronaux convolutifs (CNN) avec des paramètres de régularisation tels que la normalisation par lots pour éviter le sur ajustement. Nous avons appliqué le CNN profond pour les bases de données AHDD, IESK et AHCD et les précisions de classification pour les trois ensembles de données étaient de 96%, 78% et 94%, respectivement. Un deuxième modèle basé sur le Capsule Neural Network (CapsNet) a également été proposé, dont le but est d'améliorer la précision du système de reconnaissance. Les résultats montrent que notre modèle basé sur CapsNet améliore la précision du système proposé et surpasse le premier modèle, basé sur CNN.

Dédicace

Je dédie ce modeste travail

A mes très chers parents pour leur soutient et encouragement durant toutes mes années d'études et sans lesquels je n'aurais jamais réussi, A mon frère sofiane et ma soeurs merieme. A tous mes amis azzedine , aziz , abdelsalam, oualie elddine ,mohamed dhia elddine ,sofiane,ahmed et tahar, et a darine qui m'ont aidés, soutenus et encouragés. A tous mes enseignants durant mes années d'études avec lesquels j'ai beaucoup appris et mon enseignants et encadrant tibarmacine ahmed.

Table des matières

Chapitre 01 : Apprentissage Machines

Chapitre 02 : Reconnaissance des formes

Chapitre 03 : Conception et implémentation

Liste des figures

Liste des symboles et notations

CD	Divergence Contrastive (Contrastive Divergence)
EM	Espérance-Maximisation (Expectation-Maximization)
NLL	Log-Vraisemblance Négative (Negative Log-Likelihood)
PCA	Analyse en Composantes Principales (Principal Component Analysis)
RBM	Machine de Boltzmann Restreinte (Restricted Boltzmann Machine)
SVM	Machine à Vecteurs de Support (Support Vector Machine)
$f(t)$	Valeur de la fonction f appliquée à t
$f(\cdot)$	Fonction f (ayant un argument)
$ln(\cdot)\ ou\ log(\cdot)$	Logarithme népérien
$argmax_u f(u)$	La valeur de u qui maximise f(u)
$argmin_u f(u)$	La valeur de u qui minimise f(u)
$\mathbb{R}$	Ensemble des nombres réels
$v^T w$	Produit scalaire entre les vecteurs v et w
M^T	Transposée de la matrice M
M_{ij}	Élément de la matrice M en i^{ème} rangée et j^{ème} colonne
$\mathbf{1}\cdot$	fonction indicatrice, par exemple $\mathbf{1}_{i<j}$vaut 1 si $i < j$, et 0 sinon
$\mathbb{I}$	Matrice identité
L	Matrice Laplacienne associée à un graphe
x_i	Partie ' entrée 'du i^{ème} exemple d'apprentissage
x_{ij}	La j^{ème} composante de la partie 'entrée ' du i^{ème} exemple d'apprentissage
y_i	Partie "étiquette" du i^{ème} exemple d'apprentissage
z_i	i^{éme} exemple d'apprentissage (si étiqueté: $z_i = (x_i, x_i)$, sinon: $z_i = (x_i)$)
$D = \{z_1, \ldots, z_n\}$	Ensemble d'entraînement
T	Ensemble de test
ℓ	Nombre d'exemples étiquetés dans l'ensemble d'entraînement
$K(\cdot,\cdot)$	Fonction noyau (ayant deux arguments)
$P(V)$	Distribution de probabilité d'une variable aléatoire V
$P(v)$	Raccourci pour $P(V = v)$ (probabilité discrète ou densité)
$P(v\|w)$	Raccourci pour $P(V = v\|W = w)$ (probabilité discrète ou densité)
$\hat{P}$	Distribution empirique
$v \sim P(V)$	Indique que la quantité v est tirée de la distribution $P(V)$
$E_V[f(V)]$	Espérance de $f(V)$,égale à $\int_V f(v)P(V = v)\,dv$
$E_V[f(V)\|w]$	Espérance conditionnelle de $f(V)$,égale à $\int_V f(v)P(V = v\|W = w)\,dv$
$D_{KL}(P\|\|Q)$	Divergence de Kullback-Leibler entre les distributions P et Q
$\mathcal{N}(\cdot; \mu, \Sigma)$	Densité d'une Gaussienne de moyenne μ et covariance Σ
$\mathcal{E}$	Fonction d'énergie dans une machine de Boltzmann restreinte

Introduction Générale

La reconnaissance des formes « RDF » est à la fois une discipline historique de l'intelligence artificielle et un domaine de recherches extrêmement dynamique qui a subi de multiples influences de domaines scientifiques variés depuis une période d'années. Elle est aujourd'hui partie intégrante de l'apprentissage automatique, domaine auquel elle a apporté de nombreuses idées et méthodes, en outre elle couvre un spectre large d'applications. L'objectif principal de notre travail consiste à concevoir automatiquement un système autonome à base réseau de neurone artificiels. Ce système est capable de reconnaître une forme spécifique de pattern « écriture arabe manuscrite ».

Au cours des dix dernières années, des progrès considérables ont été réalisés dans le domaine de la reconnaissance de l'écriture manuscrite. Ces progrès sont dus aux nombreux travaux réalisés dans ce domaine, mais aussi à la disponibilité de bases de données de normes internationales pour l'écriture manuscrite (mnis,nist database) qui ont permis aux chercheurs de rendre compte de manière crédible des performances de leurs approches d'apprentissage dans ce domaine, avec la possibilité de les comparer avec d'autres approches qu'ils utilisent sur les mêmes bases.

La langue arabe n'a pas eu cette chance, contrairement au latin, elle est encore au niveau de la recherche et de l'expérimentation, c'est-à-dire que le problème reste un défi ouvert pour les chercheurs.

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

Pour remédier à ces problèmes, des approches d'apprentissage approfondi (deep learning) ont été proposées pour la reconnaissance des mots arabes manuscrits. Un tel système nécessite la prise en compte d'un grand nombre de variabilités d'écriture.

Dans ce contexte, il convient de se poser la question suivante: Comment pouvons-nous utiliser le Deep Learning sur le l'écriture arabe manuscrite? Pour répondre à cette question et atteindre notre

objectif (reconnaissance des textes en arabe), nous avons proposé deux modèles basés sur deux approches d'apprentissage profond différentes: CNN et CapsNet. Les parties suivantes du mémoire expliquent ce fait plus en détail.

Le présent mémoire est organisé en trois chapitres dont les thèmes sont donnés ci-dessous: Dans un premier temps, nous définissons la notion de l'apprentissage automatique et ses caractéristiques et nous introduisons ses types les plus répandus. Le deuxième chapitre, nous introduisons la notion de reconnaissance des formes nous parlerons brièvement dans son historique, ses type d'application dans divers domaines, nous définirons aussi ses principales méthodes qu'ils ont été développées pour extraire automatiquement des informations des données sensibles afin de caractériser les classes de formes (apprentissage) et d'assigner automatiquement des données à ces classes (reconnaissance). Dans le troisième chapitre nous présentons l'architecture conceptuelle de notre système de reconnaissance d'écriture arabes, nous introduisons également les détails des étapes de modélisation et de l'implémentation, et à travers cela nous dirigeons vers les réseaux de neurones convolutionnels(CNN) et les réseaux de capsules (CapsNet). Dans cette dernière partie nous allons expliquer les différents bibliothèques et paquets utilisé pour atteindre notre objectif et nous analyserons les résultats obtenus.

Chapitre 01

Apprentissage Machines

1. Introduction

Les inventeurs rêvent depuis longtemps de créer des machines réfléchissantes, et ce désir remonte au moins à l'époque des Grecs. Lorsque les ordinateurs programmables ont été conçus, les utilisateurs se sont demandé si ces appareils pourraient devenir intelligents. L'intelligence artificielle est aujourd'hui un domaine florissant qui compte de nombreuses applications et sujets de recherche actifs. Attendez-vous à des programmes intelligents pour automatiser le travail de routine, comprendre la parole ou l'image et établir des diagnostics Médecine et soutien à la recherche scientifique fondamentale. Le véritable défi de l'intelligence émotionnelle réside dans le fait qu'elle résout des tâches facilement réalisables, mais il est difficile de décrire des problèmes formels que nous résolvons de manière intuitive, tels que la reconnaissance de mots prononcés ou de des formes spéciales dans des images comme l'écriture manuscrite

2. Apprentissage machine

L'apprentissage machine est une sous-discipline de l'intelligence artificiel dont l'objectif ultime est de reproduire chez l'ordinateur les capacités cognitives de l'être humain, par *le biais de l'apprentissage*. Ici, le terme apprentissage est à mettre en opposition avec des techniques d'intelligence artificielle basées sur des comportements intelligents "pré-codés", comme dans le fameux programme ELIZA [11], où l'ordinateur donnait l'illusion de pouvoir poursuivre une conversation intelligente avec un humain à partir d'un système en fait très basique de détection de mots-clés et de réponses prédéfinie. L'approche "apprentissage machine" consiste plutôt à programmer des mécanismes qui permettent de développer les connaissances c'est-à-dire d'apprendre à partir d'observations (*les exemples d'apprentissage*) de manière automatique [52].

Les êtres humains faisant preuve de capacités d'apprentissage impressionnantes, il est naturel que l'apprentissage machine s'inspire d'eux pour tenter de reproduire leurs comportements. En particulier, les travaux en neurosciences visant à comprendre les mécanismes d'apprentissage dans le cerveau. Une classe d'algorithmes d'apprentissage très populaire, les réseaux de neurones, a ainsi été inspirée de telles observations biologiques. Mais les détails du fonctionnement du cerveau restant pour l'instant en grande partie un mystère. Les algorithmes développés en apprentissage machine ont des objectifs moins ambitieux. Ils ont chacun leurs propres forces et faiblesses – souvent très différentes de celles des humains – et sont généralement prévus pour résoudre certains types de problèmes spécifique [52].

3. Différents types d'apprentissage

3.1. Apprentissage supervisé

La forme d'apprentissage qui est considérée comme la plus intuitive est celle dite d'apprentissage supervisé. Cela signifie que la réponse attendue est observée dans les données collectées. Formellement, si l'on note *z* un exemple d'apprentissage, alors on peut écrire $z = (x, y)$ avec *x* la partie "entrée", c'est-à-dire les données que l'algorithme est autorisé à utiliser pour faire une prédiction, et *y* la partie "étiquette", c'est-à-dire la valeur correcte pour la prédiction. Par exemple, si la tâche consiste à déterminer le sexe d'une personne à partir d'une photo d'identité, *x* serait la photo et *y* soit "homme", soit "femme" (en supposant qu'il s'agisse des deux seules possibilités). Dans un tel cas où les valeurs possibles de l'étiquette *y* ne sont pas interprétables comme des nombres, on parle de tâche de classification. Si par contre la tâche était de prédire l'âge de la personne plutôt que son sexe, *y* serait un nombre et on parlerait d'une tâche de régression. La figure 1.1 illustre ces deux situations [52].

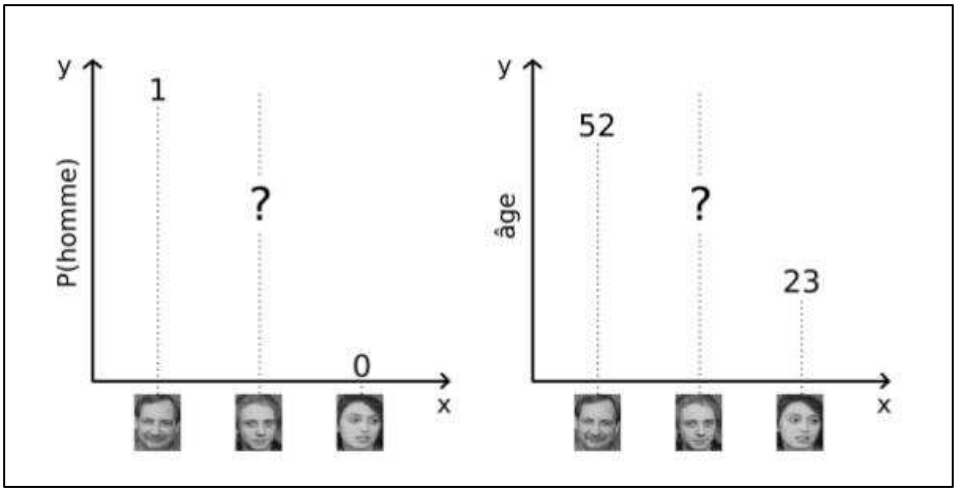


Figure1.1. Exemples d'apprentissage supervisé : classification (à gauche) pour prédire le sexe, et régression (à droite) pour prédire l'âge, Les photos aux deux extrémités de l'axe des x sont des exemples d'entraînement pour lesquels l'étiquette est connue, tandis que la photo du milieu est celle pour laquelle on veut obtenir une prédiction [52] .

L'algorithme est *entraîné* sur un ensemble de données pré-collectées (l'ensemble d'entrainement), de la forme $D = \{z_1, \dots, z_n\}$, avec $z_i = (x_i, y_i)$. De nombreux algorithmes supposent que ces exemples sont tirés de manière indépendante et identiquement distribuée d'une distribution P, c'est-à-dire. $z_i \sim P(X, Y)$ (où la forme exacte de P n'est pas connue

d'avance). Selon la tâche à résoudre, la prédiction d'un algorithme d'apprentissage supervisé va généralement tenter d'approximer l'une des trois quantités suivantes [52] :

– $P(x|y)$: dans notre exemple de classification (trouver le sexe), il faudrait prédire la probabilité qu'une photo soit une photo d'un homme par un nombre p (la probabilité que ce soit la photo d'une femme étant alors 1−p). Dans notre exemple de régression (trouver l'âge), la prédiction serait une densité de probabilité sur l'intervalle $]0, +\infty[$.

– $argmax_y P(x|y)$: dans notre exemple de classification, il s'agirait d'une prédiction binaire du sexe le plus probable ("homme" ou "femme"). Dans notre exemple de régression, il s'agirait de l'âge le plus probable.

– $E_Y = [Y|x] = \int_y yP(y|x)\,dy$: cette quantité n'a de sens que pour la régression, et il s'agirait dans notre exemple de prédire l'âge moyen de la personne étant donnée sa photo.

Il faut noter que la prédiction de $P(y|x)$ est la tâche la plus générale (dans la mesure où la résoudre permet également d'accomplir les deux autres), mais tous les algorithmes d'apprentissage ne sont pas capables d'estimer une distribution de probabilité [52].

3.2. Apprentissage non supervisé

Dans l'apprentissage non supervisé, un exemple $z_i = x_i \sim P(x)$ ne contient pas d'étiquette explicite. Il existe plusieurs types de tâches d'apprentissage non supervisé, parmi lesquelles les plus souvent rencontrées sont [52] :

- L'estimation de densité: estimer P(x), comme illustré en figure 1.2.
- La génération de données: tirer de nouveaux exemples d'une distribution la plus proche possible de P(X).
- L'extraction de caractéristiques
- Le regroupement (clustering): partitionner les exemples en groupes $G_1, \ldots, G_k$ tels que tous les exemples dans un même groupe soient similaires

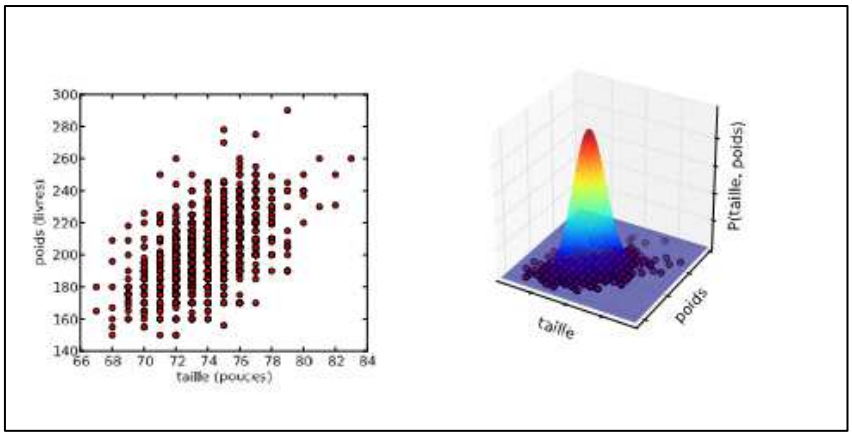


Figure1.2. Exemple d'apprentissage non supervisé : l'estimation de densité. À gauche, les données d'origine (taille et poids de 1033 joueurs de baseball aux Etats-Unis). À droite, l'estimation de la densité par une distribution Gaussienne [52].

3.3. Apprentissage semi-supervisé

Comme le nom l'indique, l'apprentissage semi-supervisé est à mi-chemin entre le supervisé et le non supervisé : certains exemples $z_i = (x_i, y_i)(1 \leq i \leq l)$ ont une étiquette, tandis que d'autres exemples $z_i = x_i(l + 1 \leq i \leq n)$ n'en ont pas (on dit qu'ils ne sont pas "étiquetés"). Les algorithmes d'apprentissage semi-supervisé tentent généralement de résoudre des problèmes d'apprentissage supervisé, mais en utilisant les exemples non étiquetés pour améliorer leur prédiction. La distribution des entrées $P(X)$ peut nous donner de l'information sur $P(Y|X)$ même en l'absence d'étiquettes. Un exemple typique où c'est le cas, pour une tâche de classification, est lorsque les exemples de chaque classe forment des groupes distincts dans l'espace des entrées, séparés par des zones de faible densité $P(x)$. La figure 1.3 montre ainsi deux classes dont la forme en croissant est révélée par les exemples non étiquetés. Un algorithme purement supervisé donc ignorant ces exemples ne pourrait pas identifier correctement ces deux classes [52].

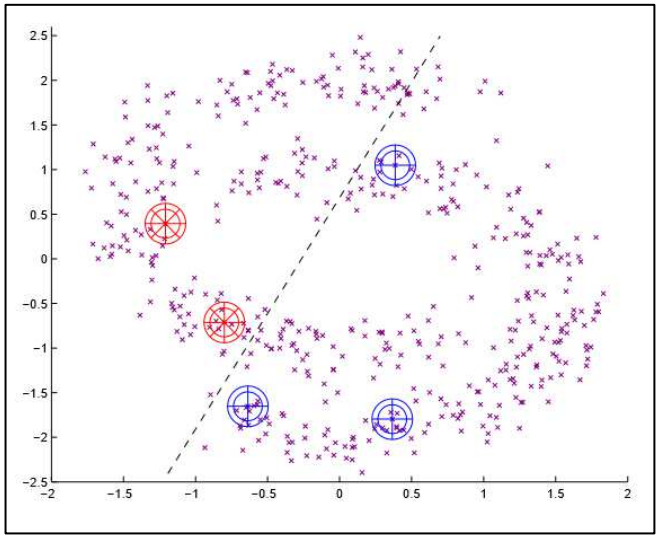


Figure1.3. Apprentissage semi-supervisé : ici seuls 5 exemples sont étiquetés (deux de la classe * en rouge, et trois de la classe + en bleu). Un algorithme n'utilisant pas les exemples non étiquetés (petits x mauves) séparerait les deux classes par exemple selon la ligne en pointillés, alors qu'un algorithme semi-supervisé (ou un humain) pourrait séparer les deux "croissants de lune" correspondant à chaque classe [52].

Dans le cadre de l'apprentissage supervisé décrit précédemment, l'application d'un algorithme se fait typiquement en deux étapes [52] :

1. L'algorithme va d'abord apprendre une tâche (phase d'entraînement) sur un ensemble $D = \{(x_1, y_1), \cdots, (x_n, y_n)\}$.
2. L'algorithme doit ensuite faire ses prédictions (phase de test) sur un ensemble T dans on ne fournit pas les étiquettes.

Un algorithme semi-supervisé peut également suivre ces deux étapes (en n'oubliant pas que D peut contenir également des exemples non étiquetés). On dit alors que la phase de prédiction sur l'ensemble de test se fait par induction. Mais puisque l'algorithme peut utiliser des exemples non étiquetés au cours de l'apprentissage, on peut également inclure T dans D : on parle alors de *transduction*, et en général les performances seront meilleures qu'en induction puisque plus d'exemples sont disponibles pour l'apprentissage. Il existe malgré tous des applications où il n'est pas possible d'inclure T dans l'ensemble d'entraînement, par exemple lorsque les éléments de T sont générés en temps réel et que l'on a besoin de prédictions immédiates : si ré-entraîner l'algorithme avant chaque nouvelle prédiction s'avère trop lent, l'induction est alors la seule option possible [52].

3.4. Apprentissage par renforcement

L'apprentissage par renforcement est une forme d'apprentissage supervisé où l'algorithme n'observe pas une étiquette pour chacune de ses prédictions, mais plutôt une mesure de la qualité de ses prédictions (possiblement prenant en compte toute une séquence de prédictions) [52].

4. Généralisation

4.1. Sur-apprentissage

L'entraînement d'un algorithme d'apprentissage consiste à extraire, de manière explicite ou implicite, des caractéristiques de la distribution de probabilité P qui génère les données. Mais P étant inconnu, l'algorithme se base à la place sur un nombre fini d'exemples d'entraînement, c'est-à-dire. Sur la distribution discrète $\hat{P}$ des exemples disponibles dans D (appelée la distribution *empirique*). Lorsque certaines caractéristiques apprises sur P ne s'appliquent pas à P, on parle de *sur-apprentissage*, et on risque une mauvaise *généralisation*, c'est-à-dire. Que l'algorithme ne va pas obtenir une bonne performance sur de nouveaux exemples tirés de P. Prenons l'exemple de la classification, lorsqu'un modèle estime $P(y|x)$ par une fonction $q_x(y)$. Afin de mesurer la similarité entre q_x et $P(.|x)$, on peut par exemple utiliser la divergence de Kullback-Leibler D_{KL}[5], définie par [52] :

$$D_{\mathrm{KL}}(P(.|x)||q_x) = \sum_y P(y|x)\, ln \frac{P(y|x)}{q_x(y)} \tag{1}$$

Cette quantité est toujours supérieure ou égale à zéro, et est égale à zéro si et seulement si q_x est égale à$P(.|x)$. La minimisation de la divergence de Kullback-Leibler est donc un critère raisonnable pour obtenir une fonction q_x qui approxime P(·|x). Vu que le but est d'obtenir une bonne approximation pour toutes les valeurs de x qui pourraient être générées par P, il est logique de considérer la minimisation du critère

$$\begin{aligned} C(q) &= E_X[D_{KL}(P(.|x)||q_x)] \\ &= \int_x \sum_y P(x)P(y|x) \ln \frac{P(y|x)}{q_x(y)} dx \end{aligned} \tag{2}$$

$C(q)$ Est ici ce que l'on appelle l'erreur de généralisation, c'est-à-dire. L'erreur moyenne sur des exemples tirés de P. Puisque P est inconnue, on minimise en pratique un critère $\hat{C}$ défini de la même manière en remplaçant P par $\hat{P}$. C'est le principe de minimisation du risque empirique [10], et $\hat{C}$ s'écrit ici [52] :

$$\hat{C}(q) = \sum_{i=1}^{n} \frac{1}{n} \ln \frac{1}{q_{x_i}(y_i)} = -\frac{1}{n} \sum_{i=1}^{n} \frac{1}{n} \ln q_{x_i}(y_i) \tag{3}$$

Qui est appelée la *log-vraisemblance négative* (en anglais NLL, pour "Negative Log-Likelihood"). Ce critère est minimisé dès que $q_{x_i}(y_i) = 1$, pour tous les exemples d'entraînement $(x_i, y_i) \in$ D *et ce quelle que soit la valeur de* $q_x(y)$ *pour des valeurs de x non observées* dans D [52].

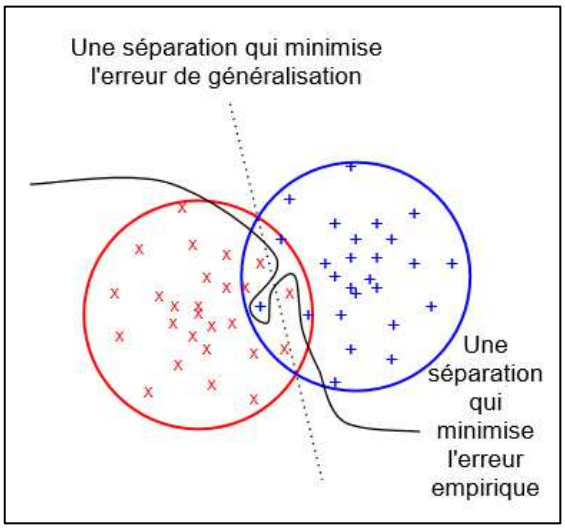


Figure1.4. Situation de sur-apprentissage : classification binaire (les classes sont les cercles rouge et bleu, avec respectivement les × et les + comme exemples d'entraînement). Une séparation idéale en termes d'erreur de généralisation serait la ligne en pointillés, mais un algorithme dont le but est uniquement de minimiser l'erreur empirique pourrait par exemple séparer les exemples selon la ligne pleine : la classification des exemples d'entraînement serait parfaite, mais l'erreur de généralisation serait plus élevée que celle de la ligne en pointillés [52].

Une fonction q peut donc minimiser $\hat{C}(q)$ sans nécessairement minimiser $C(q)$ Si c'est le cas, on est en situation de sur-apprentissage, illustrée en figure 1.4. Pour une distribution fixe des données, deux facteurs principaux augmentent le risque de sur-apprentissage [52] :

– Le manque d'exemples d'entraînement : moins il y a d'exemples, plus il existe de fonctions minimisant le critère $\hat{C}(q)$(éq. 3), parmi lesquelles seulement un petit nombre seront vraiment proches de la "vraie" solution au problème.

– Pas assez de contraintes dans la forme de la fonction q : moins la classe de fonctions à laquelle q appartient est restreinte, plus l'algorithme d'apprentissage risque de tirer parti de la fléxibilité de q pour apprendre des "détails" des exemples d'entraînement, qui ne se généralisent pas à la vraie distribution *P*. C'est le cas par exemple dans la figure 4 où la séparation tarabiscotée des exemples par la ligne pleine permet de

minimiser l'erreur empirique, mais va mener à une plus grande erreur de généralisation qu'une simple ligne droite [52].

4.2. Régularisation

Un moyen de lutter contre le sur-apprentissage est d'utiliser une technique dite de régularisation. Il existe plusieurs méthodes de régularisation, mais elles partagent le même principe : rajouter au processus d'apprentissage des contraintes qui, si elles sont appropriées, vont améliorer les capacités de généralisation de la solution obtenue.

Reprenons par exemple le cas de la classification, où l'on cherche à minimiser le critère $C(q)$(éq. 2), que l'on approxime par $\hat{C}(q)$(éq. 3). Comme nous venons de le voir, ce problème est mal défini car il existe une infinité de fonctions qui minimisent C, sans donner aucune garantie sur la valeur de C. Une première façon de régulariser le problème est donc de restreindre la forme de q : par exemple $x \in \mathbb{R}^d$et $y \in \{0,1\}$ on peut se limiter aux fonctions de la forme [52].

$$q_x(1) = \frac{1}{1+e^{-w^T x}} \tag{4}$$

Où $w \in \mathbb{R}^d$ est le vecteur de paramètres du modèle. Notons que si les x_i de l'ensemble d'entraînement sont linéairement indépendants, alors cette contrainte sur la forme de q n'est pas suffisante, puisqu'il est toujours possible que $\hat{C}(q)$soit arbitrairement proche de zéro sans pour autant avoir de garantie sur la valeur de $C(q)$. Une technique classique de régularisation consiste alors à rajouter au critère $\widehat{C}$ une mesure qui pénalise la *complexité* de la solution, suivant le principe du rasoir d'Occam qui dit que les hypothèses les plus simples sont les plus vraisemblables voir [2]. Une possibilité est de minimiser [52].

$$\tilde{C}(q) = \hat{C}(q) + \lambda||w||^2 \tag{5}$$

Au lieu de $\hat{C}$, pour q défini comme dans (l'éq. 4), afin d'empêcher le vecteur w de contenir des valeurs arbitrairement grandes (en valeur absolue). Le paramètre λ contrôle la force de cette contrainte (lorsque $\lambda \to +\infty$ la seule solution possible est la fonction constante $q_x(1) = q_x(0) = 0{,}5$, qui est la plus simple qu'on puisse imaginer). Le critère empirique $\hat{C}(q^*)$pour la fonction q^* qui minimise le critère régularisé $\tilde{C}$ pourrait ne pas être proche de zéro, mais on peut souvent ainsi – pour certaines valeurs de λ – obtenir des valeurs plus basses du critère de généralisation C (celui qui nous intéresse vraiment). C'est le principe de la minimisation du risque structurel [10] [52].

Dans cet exemple, nous avons utilisé $||w||^2$ pour mesurer la complexité de la fonction q définie à partir de w par (l'éq. 4). En général, il n'existe pas une seule mesure de complexité

universelle qui soit appropriée pour tous les algorithmes d'apprentissage, et le choix de la mesure de complexité à pénaliser joue un rôle très important. La complexité de Kolmogorov [8][4], est une mesure de complexité très générique qui est intéressante en théorie, même si en pratique elle est souvent impossible à utiliser directement. Elle consiste à dire que la complexité d'une fonction est la taille du plus petit programme qui l'implémente. Un premier obstacle à l'utilisation de cette complexité est le fait qu'il faille choisir un langage de programmation approprié : par exemple si le langage choisi contient une fonction primitive qui calcule le produit scalaire, alors, dans notre exemple ci-dessus la plupart des fonctions q définies par (l'éq. 4) ont la même complexité de Kolmogorov. Par contre, si le produit scalaire n'est pas une primitive du langage (et qu'il n'y a pas d'instruction de boucle), alors il faut l'écrire comme une somme de produits et q est d'autant plus complexe que w a d'éléments non nuls. Une autre difficulté est qu'il n'est en général pas possible d'optimiser la complexité de Kolmogorov de manière efficace, ce qui rend vaine son utilisation directe dans un processus d'optimisation. Elle a malgré tout de nombreuses applications, comme décrit dans le livre de [6] [52].

4.3. Malédiction de la dimensionnalité

On peut observer empiriquement – et dans certains cas justifier mathématiquement –que plus la dimension d de l'entrée x est élevée, plus les tâches d'apprentissage machine ont tendance à être difficiles à résoudre. C'est ce qu'on appelle *la malédiction de la dimensionalité* [1]. Il existe plusieurs manifestations de cette malédiction. La plus importante dans le contexte de cette thèse est le fait que le nombre de combinaisons possibles des entrées augmente exponentiellement avec la dimensionnalité d : en notant x_{ij} la valeur associée à la j-ème dimension de l'entrée x_i, si l'on suppose que ces entrées ne peuvent prendre qu'un nombre fini k de valeurs, alors le nombre de combinaisons possibles est égal à k^d. Un algorithme qui apprend "bêtement" à associer une valeur à chaque combinaison sans partager d'information entre les différentes combinaisons n'a aucune chance de fonctionner en haute dimension, car il ne pourra pas généraliser aux multiples combinaisons qui n'ont pas été vues dans l'ensemble d'entraînement. Dans le cas où x_{ij} n'est pas contraint dans un ensemble fini de valeurs, l'intuition reste la même pour certains algorithmes qui consistent à "partitionner" $\mathbb{R}^d$ en régions indépendantes (possiblement de manière implicite) : si le nombre de ces régions augmente exponentiellement avec d, alors un tel algorithme aura de la difficulté à généraliser pour de grandes valeurs de d. La figure 1.5 illustre ce phénomène en une et deux dimensions,

et il faut garder à l'esprit que la situation peut s'avérer encore bien pire lorsque l'on manipule des entrées à plusieurs centaines de dimensions [52].

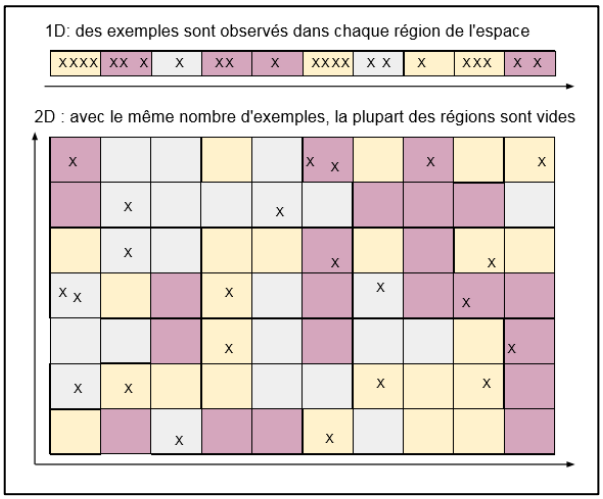


Figure1.5. Malédiction de la dimensionalité : si l'algorithme partitionne l'espace en régions indépendantes, le nombre d'exemples nécessaires pour remplir ces régions augmente de manière exponentielle avec la dimension. Ici, la couleur d'une région représente la classe majoritaire dans cette région, et un tel algorithme pourrait bien généraliser à partir de 23 exemples d'entraînement pour le problème du haut (1D), mais pas pour celui du bas (2D).

5. Différents types de modèles

5.1. Modèles paramétriques

En apprentissage machine, un modèle paramétrique est défini par un ensemble Θ de paramètres de dimension fini, et l'algorithme d'apprentissage associé consiste à trouver la meilleure valeur possible de Θ. La figure 1.6 montre un exemple de régression linéaire en une dimension, sans régularisation.

Les modèles paramétriques peuvent également être statistiquement inefficaces. S'il y a moins d'exemples d'apprentissage, alors le problème est sur-paramétré et on risque le sur-apprentissage : le modèle pourrait apprendre des paramètres taillés "sur mesure" pour les données d'entraînement, mais qui mèneront à une mauvaise généralisation. La conséquence de cette observation est qu'en général, un modèle avec un grand nombre de paramètres est statistiquement inefficace [52].

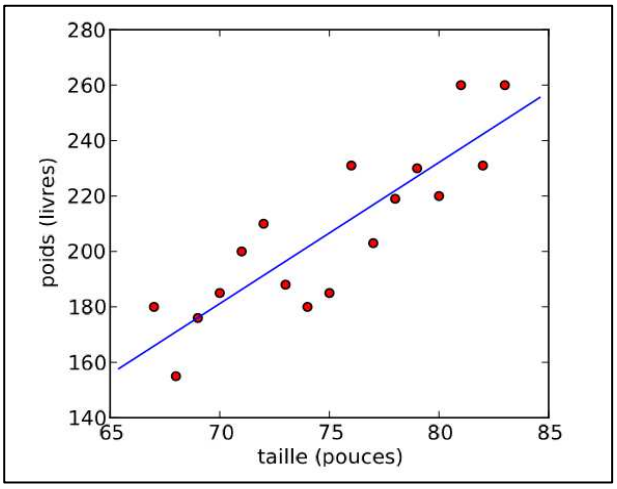


Figure1.6. Modèle paramétrique : la régression linéaire [52].

5.2. Modelés non paramétriques

Un modèle non paramétrique n'a au contraire pas d'ensemble exacte de paramètres : le nombre de variables utilisées par le modèle augmente généralement avec le nombre d'exemples dans l'ensemble d'entrainement. Un exemple de modèle non paramétrique pour résoudre le même problème de régression que celui décrit en 4.1 est l'algorithme des fenêtres de Parzen, aussi appelé régression de Nadaraya-Watson [7][11]. La figure 1.7 montre un exemple de régression par fenêtres de Parzen en une dimension [52].

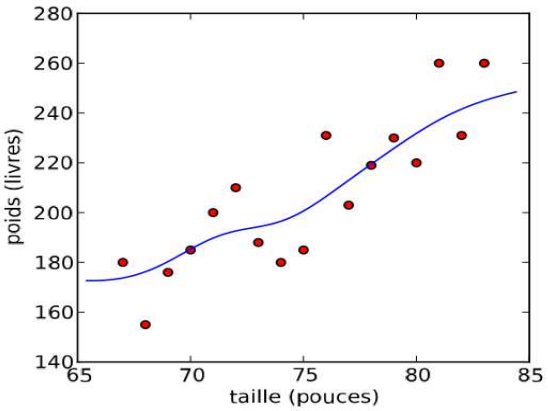


Figure1.7. Modèle non paramétrique : régression par fenêtres de Parzen. Les données sont les mêmes que dans l'exemple de la régression linéaire [52]

6. Algorithmes d'apprentissage

6.1. Réseaux de neurones

Comme leur nom l'indique, les réseaux de neurones sont inspirés de l'architecture du cerveau, étant organisés en couches de neurones connectées entre elles. La première couche, appelée la couche d'entrée, est de la même dimension que les entrées x et l'activation de son j-ème neurone (c'est-à-dire la valeur qu'il calcule) est égale à x_{ij} lorsque l'on calcule la prédiction du réseau sur l'exemple x_i. La dernière couche, appelée la couche de sortie, est de la même dimension que l'étiquette dans le cas d'une tâche supervisée. Par exemple, pour la classification, le j-ème neurone de la couche de sortie va calculer $P(Y = j|x_i)$ lorsque x_i est dans la couche d'entrée. Les couches intermédiaires sont appelées les couches cachées. Il existe de nombreuses variations dans les architectures de réseaux de neurones. L'architecture la plus connue est celle où il existe une unique couche intermédiaire h qui calcule une transformation non linéaire des entrées, de la forme [52] :

$$H\,(x) \;=\; \sigma\,(W^T x \;+\; b) \tag{7}$$

Où W est une matrice de poids, b est un vecteur de biais (de la même taille que le nombre de neurones dans la couche cachée), et σ est une transformation non linéaire élément par élément, dont l'opération sur chaque élément est typiquement la sigmoïde ou la tangente hyperbolique [52]

$$\text{Sigmoid(u)} = \frac{1}{1+e^{-u}} \tag{8}$$

$$\tanh(u) = \frac{e^{u}-e^{-u}}{e^{u}+e^{-u}} \tag{9}$$

La fonction donnant la sortie $\hat{y}$ en fonction de h dépend de la tâche ; en classification, on écrit souvent le j-ème neurone de $\hat{y}$, qui estime $P(Y = j|x)$,sous la forme :

$$\frac{e^{V_j^T h+c_j}}{\sum_k e^{V_k^T h+c_k}} \tag{10}$$

Avec V_j la j-ème rangée d'une matrice de poids V, et c le biais du j-ème neurone de sortie. La figure 1.8 montre une telle architecture à une couche cachée.

L'apprentissage dans un réseau de neurones consiste à optimiser les poids et biais de façon à minimiser un coût approprié à la tâche. Par exemple, pour la classification, on minimisera la log-vraisemblance négative empirique (éq. 3) qui, en notant Θ les paramètres à optimiser, et $f_j(x_i; \Theta)$ la valeur du j^{ème} neurone de sortie lorsque x_i est en entrée du réseau, se réécrit [52] :

$$\hat{C}(\Theta) = -\frac{1}{n}\sum_{i=1}^{n} \ln f_{y_i}(x_i; \Theta) \tag{11}$$

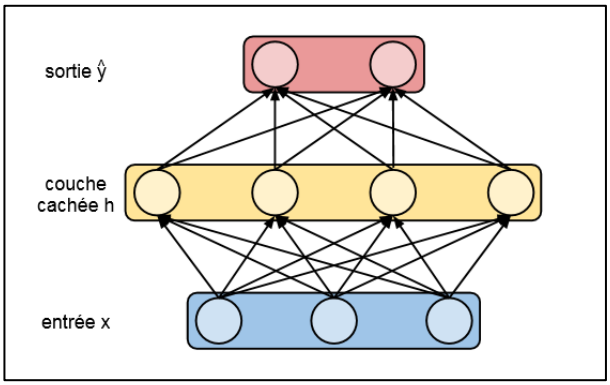


Figure1.8. Réseau de neurones à une couche cachée. Une flèche du neurone *i* vers le neurone *j* indique que l'activation de j dépend directement de celle de *i [54].*

Un autre exemple, en apprentissage non supervisé, est celui des réseaux de neurones auto-associateurs dont le but est d'extraire dans la couche cachée une représentation qui permet de reconstruire l'entrée x_i (donc l'étiquette y_i est en fait égale à x_i). Dans ce cas, le coût le plus fréquemment utilisé est l'erreur de reconstruction quadratique moyenne [52]

$$\hat{C}(\Theta) = \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{d}(f_i(x_i; \Theta) - x_{ij})^2 \tag{12}$$

Mais lorsque les entrées $x_{ij} \in \{0,1\}$, on peut également minimiser l'entropie croisée

$$\hat{C}(\Theta) = -\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{d}((x_{ij} ln f_i(x_i; \Theta) + (1 - x_{ij}) \ln(1 - f_i(x_i; \Theta) \tag{13}$$

Où le terme dans la somme peut s'interpréter comme la log-vraisemblance des données d'entraînement selon la distribution $P_\Theta(X_{ij} = 1|x_i) = f_i(x_{ij};\Theta)$.

L'apprentissage dans un réseau de neurones consiste à optimiser les poids et biais de façon à minimiser un coût approprié à la tâche. Les algorithmes d'optimisation les plus souvent utilisés pour minimiser $\hat{C}(\Theta)$ sont des algorithmes d'optimisation locale basés sur l'idée de *la*

descente de gradient : le gradient du coût $\hat{C}(\Theta)$ par rapport aux paramètres Θ, noté $\frac{\partial \hat{C}(\Theta)}{\partial \Theta}$ indique la direction dans laquelle le coût augmente le plus lorsque Θ varie. Descendre le gradient signifie déplacer les paramètres Θ dans la direction opposée, de manière à diminuer le coût. Le gradient par rapport à tous les paramètres du réseau peut se calculer de manière efficace grâce à l'algorithme de rétro-propagation du gradient [40] [52].

En apprentissage non supervisé, on parle des réseaux de neurones auto-associateurs dont le but est d'extraire dans la couche cachée une représentation qui permet de reconstruire l'entrée x_i (donc l'étiquette y_i est en fait égale à x_i). Dans ce cas, le coût le plus fréquemment utilisé est l'erreur de reconstruction quadratique moyenne. On trouve donc que les réseaux de neurones représentent l'algorithme le plus approprié pour notre projet et pour l'implémentation [52].

6.2. Machines de Boltzmann restreintes

Dans sa version la plus simple, une machine de Boltzmann restreinte (RBM)est un algorithme non supervisé qui modélise la distribution $P(X)$ où $x \in \{0,1\}^d$ comme la marginale d'une distribution jointe [52]

$$P(x,h) = \frac{e^{-\varepsilon(x,h)}}{\mathrm{Z}} \tag{14}$$

Où Z est *la fonction de partition* assurant que cette probabilité est bien normalisée :

$$Z = \sum_{x,h} e^{-\varepsilon(x,h)} \tag{15}$$

Le vecteur $h \in \{0,1\}^k$ représente la couche cachée, tandis que le vecteur *x* est appelée la couche visible. Les éléments d'une couche sont généralement appelés unités plutôt que neurones. La quantité $\varepsilon(x,h)$ est *l'énergie* de la paire (x,h), et l'équation 13 montre que plus l'énergie est faible, plus cette paire est probable. Une forme typique pour ε est [52]:

$$\varepsilon(x,h) = h^T W x - x^T b - h^T c \tag{16}$$

Où la matrice $W \in \mathbb{R}^{k \times d}$ et les vecteurs b $\in \mathbb{R}^d$ et c $\in \mathbb{R}^k$ sont les paramètres du modèle. Bien que ce modèle ait originellement été introduit sous un autre nom [44], il s'agit bien d'une forme particulière de machine de Boltzmann [28] [27]. Comparée à une machine de Boltzmann générique, l'énergie d'une RBM à la propriété que les probabilités conditionnelles

$P(x|h)$ et $P(h|x)$ se factorisent, ce qui rend l'entraînement (un peu) plus aisé. Cette propriété se visualise lorsque l'on représente une RBM sous la forme d'un modèle graphique non dirigé [48], comme dans la figure 1.9 : il y a des connexions entre les unités visibles et les unités

Cachées, mais aucune connexion de visible à visible, ni de cachée à cacher [52].

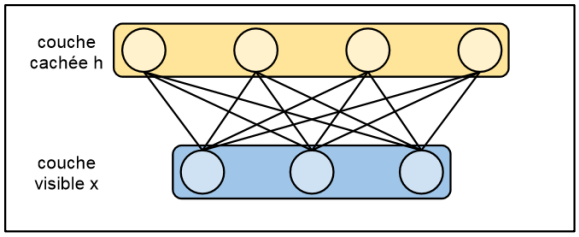


Figure1.9. Machine de Boltzmann restreinte

6.3. Architectures profondes (Deep Learning)

Le terme d'architecture profonde désigne toute une famille de modèles inspirés des réseaux de neurones, dont le point commun est la composition de transformations successives, permettant de calculer une fonction complexe de l'entrée. Par exemple, un réseau de neurones avec une seule couche intermédiaire peut être considéré comme une architecture profonde si l'on rajoute des couches cachées : chaque couche additionnelle augmente la profondeur du réseau, et déterminer la profondeur idéale en fonction des données fait partie de la problématique des algorithmes d'apprentissage pour architectures profondes. Les réseaux de ce type ont longtemps été ignorés, d'une part parce qu'ils se sont avérés beaucoup plus difficiles à optimiser que les réseaux à une seule couche cachée, d'autre part parce qu'il a été démontré que les réseaux à une seule couche sont des approximateurs universels [30] [52]. L'intérêt pour les réseaux profonds est récemment réapparu après avoir découvert qu'une initialisation non supervisée des poids du réseau peut mener à de bien meilleures performances que l'initialisation aléatoire utilisée jusqu'à présent.

6.4. K-plus proches voisins

L'algorithme des k plus proches voisins est un algorithme non paramétrique utilisé pour la régression et la classification. Etant donnée une mesure de distance dans l'espace d'entrée $\mathbb{R}^d$ (souvent prise comme la distance Euclidienne$||x_i - x_j||$) la prédiction du modèle sur un exemple de test $x \in T$ dépend uniquement des k plus proches voisins de x dans l'ensemble d'entraînement D. En notant $i_1(x), \ldots, i_k(x)$ les indices des k exemples de D les plus proches de x selon la distance choisie (ses "voisins"), la prédiction du modèle en régression est la moyenne des étiquettes observées chez ces k voisins [52] :

$$f(x) = \frac{1}{k}\sum_{j=1}^{k} y_{i_j(x)} \quad (17)$$

Et en classification il s'agit d'un vote parmi les k voisins :

$$f(x) = argmax_y \sum_{j=1}^{k} 1_{y=y_{i_j(x)}} \quad (18)$$

Où en cas d'égalité parmi les votes le modèle choisit aléatoirement l'une des classes majoritaires. La classification par les k plus proches voisins est illustrée en figure 1.11.

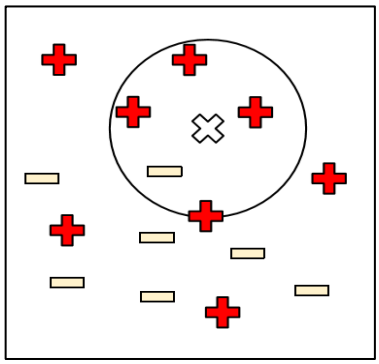


Figure1.11. K-plus proches voisins (k = 5, tâche de classification)

6.5. Fenêtres de Parzen

L'algorithme des fenêtres de Parzen a déjà été présenté dans le contexte de la régression non paramétrique, où on l'appelle parfois la régression à noyau ou la régression de Nadaraya et Watson [7] [11]. On peut également utiliser une approche similaire en apprentissage non supervisé pour l'estimation de densité [38] [36], en estimant la densité de probabilité au point *x* par [52] :

$$f(x) = \frac{1}{n}\sum_{i=1}^{n} K(x, x_i) \quad (19)$$

Ce qui correspond à placer une masse de probabilité "autour" de chaque exemple d'apprentissage x_i, dans un volume défini par le noyau K. Ici, K doit respecter les contraintes [52]

$$K(x, x_i) \geq 0$$

$$\int_x K(x, x_i)dx = 1 \quad (20)$$

De manière à ce que *f* soit une densité de probabilité valide. Le choix le plus répandu pour K est le noyau Gaussien, mais avec la normalisation appropriée [52]:

$$K(x, x_i) = \mathcal{N}(x_i; x_j, \sigma^2\mathbf{I}) = \frac{1}{(2\pi)^{d/2}\sigma d} e^{-\frac{||x_i - x_j||^2}{2\sigma^2}} \quad (21)$$

où l'on note $\mathcal{N}(\cdot; \mu, \Sigma)$ la densité de probabilité d'une Gaussienne de moyenne μ et covariance Σ. Un exemple d'estimation de densité par fenêtres de Parzen avec un noyau Gaussien est montré en figure 1.12 [52].

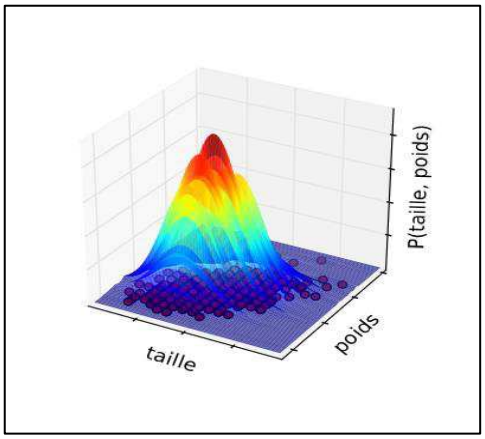


Figure1.12. Fenêtres de Parzen pour l'estimation de densité

6.6. Mélanges de Gaussiennes

Les mélanges de Gaussiennes généralisent les fenêtres de Parzen (avec noyau Gaussien) pour l'estimation de densité, en écrivant la densité comme une somme pondérée de Gaussiennes [52] :

$$f(x) = \sum_{j=1}^{N} \alpha_j \mathcal{N}(x; \mu_j, \Sigma_j) \quad (22)$$

Où N est le nombre de composantes du mélange, et α_j le poids de la j^{ème} composante (les poids sont tels que $\alpha_j \geq 0$ et$\Sigma_j \alpha_j = 1$). L'interprétation dite "générative'' de cette équation est que le modèle suppose que chaque exemple observé a été généré de la façon suivante [52] :

1.Une composante j est choisie aléatoirement, avec probabilité α_j.

2.Un exemple est généré par une distribution Gaussienne centrée en μ_j avec covariance Σj.

Si N = *n*, $\alpha_j = n^{-1}$, $\mu_j = x_i$ et $\Sigma_j = \sigma^2\mathbf{I}$ on retrouve les fenêtres de Parzen non paramétriques vues précédemment. Mais on peut également apprendre un modèle paramétrique de mélange en fixant N et en optimisant les poids α_j, les centres μ_j et les

covariances Σ_j de chaque Gaussienne. L'algorithme le plus populaire pour l'apprentissage d'un mélange de Gaussiennes est l'algorithme *Espérance-Maximisation* (EM) [21].

6.7. Méthodes à noyau

Plusieurs algorithmes mentionnés précédemment utilisent une fonction noyau $K(x_i, x_j)$ pour mesurer la similarité entre deux entrées x_i et x_j. Les méthodes dites "à noyau" incluent en particulier une catégorie d'algorithmes basés sur ce que l'on appelle "l'astuce du noyau", qui s'applique à tout algorithme qui peut s'exprimer uniquement à partir de produits scalaires de la forme $x_i^T x_j$. Cette astuce consiste à remplacer $x_i^T x_j$ dans l'algorithme d'origine par une fonction noyau $K(x_i, x_j)$, où K doit satisfaire certaines propriétés mathématiques (on dit que le noyau est défini positif – c'est le cas par exemple du noyau Gaussien que nous avons déjà utilisé). Cela revient à appliquer l'algorithme sur les données transformées implicitement et de manière non linéaire par une fonction ϕ telle que $\varphi(x_i)^T \varphi(x_j) = K(x_i, x_j)$ (une telle fonction ϕ existe automatiquement si K est défini positif, et en pratique on n'a pas besoin de la calculer explicitement). L'intérêt du noyau est principalement d'effectuer une extraction de caractéristiques non linéaire à partir des entrées, ce qui peut améliorer les performances de l'algorithme [52].

La plus célèbre méthode à noyau exploitant cette astuce est l'algorithme de la machine à vecteurs de support (SVM, pour "Support Vector Machine" en anglais). Il s'agit d'un algorithme de classification basé sur l'idée de marge : pour obtenir une meilleure généralisation, il est préférable de laisser une marge entre les exemples et la surface de décision [17][19][46]. Ce concept de marge est illustré par la figure 1.17 : dans le cas linéaire (c'est à dire. Sans utiliser l'astuce du noyau) la marge du classifieur est la distance entre l'hyper-plan séparant les deux classes et les exemples les plus proches. Dans le cas non linéaire, le même principe s'applique dans l'espace des exemples projetés implicitement par ϕ, ce qui se traduit dans l'espace des entrées par une séparation non linéaire des exemples de chaque classe. Depuis les SVMs, l'astuce du noyau et l'idée de marge ont inspiré un grand nombre de nouveaux algorithmes, ainsi que la "noyautisation" d'algorithmes existants [42] [52].

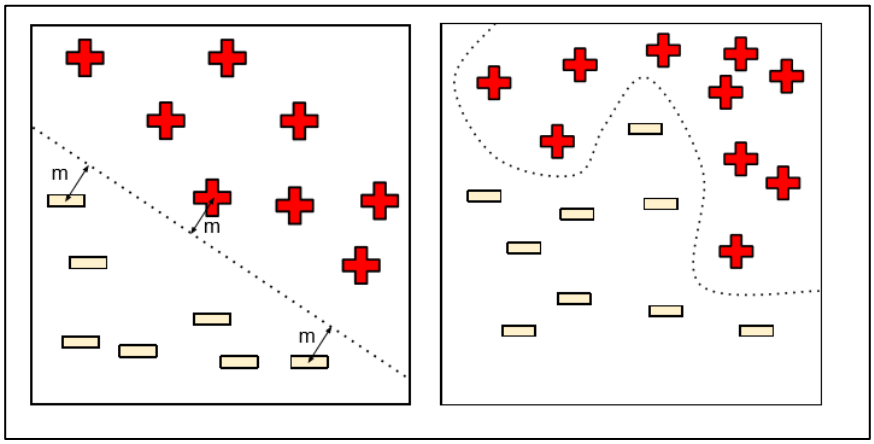


Figure1.13. Machine à vecteurs de support, dans le cas linéaire (à gauche) et non linéaire (à droite)

6.8. Arbres de décision

Les arbres de décision forment une famille d'algorithmes d'apprentissage utilisés pour la classification et la régression [18]. Un arbre de décision est un arbre où chaque nœud interne représente un test sur l'entrée x_i. L'exemple typique d'un arbre de décision est un arbre binaire où le test effectué à chaque nœud k est de la forme $x_{ij}<\theta_k$, c'est-à-dire. Qu'on compare la j^{ème} coordonnée de x_i à un seuil θ_k : si elle est plus petite, on continue de parcourir l'arbre en suivant la première branche du nœud, sinon on suit la seconde branche. Lorsqu'on atteint finalement une feuille de l'arbre (un nœud sans enfants), on dit que l'exemple x_i appartient à cette feuille. La figure 1.18 montre un tel arbre de décision [52].

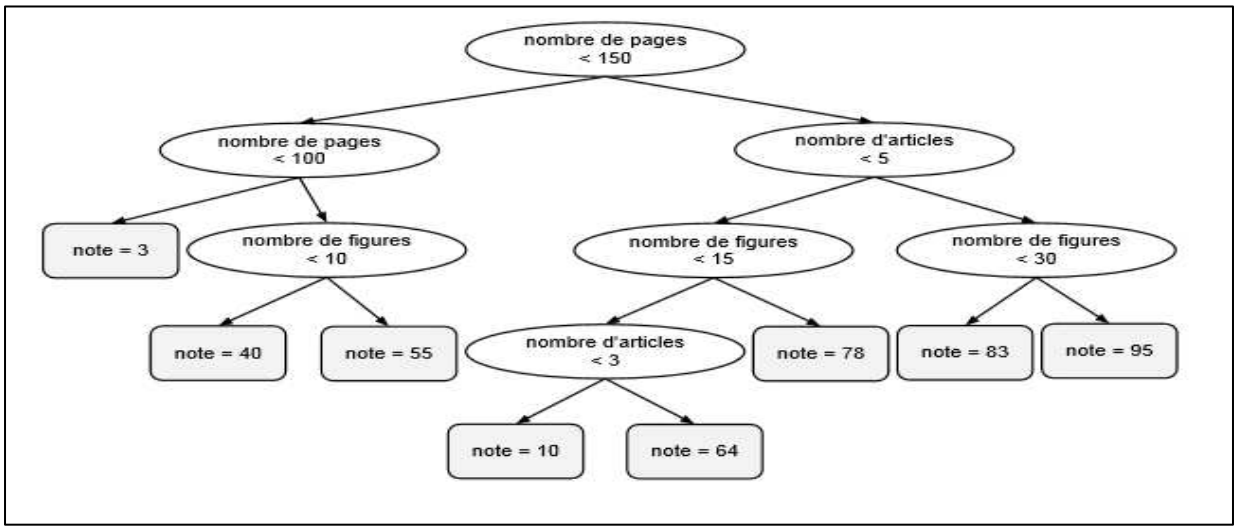


Figure1.14. Arbre de décision typique

L'apprentissage de ce type d'arbre de décision consiste à choisir les variables testées à chaque nœud, les seuils de comparaison, la profondeur de l'arbre, ainsi que la fonction de décision associée à chaque feuille. Il existe plusieurs algorithmes pour cela, mais leur principe de base est de faire en sorte que le résultat du test effectué à chaque nœud donne de l'information supplémentaire sur $P(Y|x)$. Idéalement, la distribution $P(Y|x)$pour un nouvel exemple *x* doit être bien approximée par la distribution empirique des étiquettes des exemples d'entraînement appartenant à la même feuille que *x* [52].

6.9. Méthodes Bayésiennes

Les méthodes Bayésiennes tirent leur nom de la célèbre règle de Bayes [52] :

$$P(\Theta|\mathcal{D}) = \frac{P(\Theta|\mathcal{D})P(\Theta)}{P(\mathcal{D})} \quad (28)$$

Qui est ici appliquée avec d'une part les paramètres Θ d'un modèle, et d'autre part les données d'entraînement D observées. La quantité $P(\Theta|D)$ est appelée la vraisemblance : c'est la probabilité d'observer les données D en supposant qu'elles ont été générées par notre modèle dont les paramètres sont Θ. P(Θ) est appelée la distribution a priori : c'est une distribution sur l'espace des paramètres qui reflète notre croyance sur les valeurs possibles des paramètres avant l'observation des données. P(Θ|D) est alors calculée comme étant proportionnelle au produit de la vraisemblance par la distribution a priori : elle est appelée la distribution a posteriori, c'est-à-dire. Qu'elle indique la probabilité des paramètres après avoir observé les données. Le terme P(D) est un terme de normalisation qui peut se calculer, si nécessaire, par $P(D) = \int_\Theta P(D|\Theta)P(\Theta)d\Theta$. Il suffira de garder à l'esprit qu'il s'agit de méthodes probabilistes, qui ont l'avantage en particulier de prendre en compte l'incertitude de manière naturelle : par exemple, la prédiction d'un modèle Bayésien supervisé sur une nouvelle entrée *x* peut s'écrire comme [52]

$$P(y|x, D) = \int_\Theta P(\Theta)P(\Theta|D)d\Theta \quad (29)$$

Et la variance de cette distribution peut être interprétée comme l'incertitude sur la prédiction de l'étiquette *y*. Notons qu'il n'est en général pas trivial de calculer une telle intégrale, et que plusieurs techniques ont été développées spécifiquement dans ce but [12] [52].

7. Conclusion

L'apprentissage automatique (Machine Learning) est un sujet vaste en évolution permanente. Les algorithmes qu'il met en œuvre ont des sources d'inspiration variées qui vont de la théorie des probabilités aux intuitions géométriques en passant par des approches heuristiques.

Dans le prochain chapitre, nous serons consacrés à la reconnaissance des formes et ses principaux techniques.

Chapitre02

Reconnaissance des formes

1. Introduction

Dans ce chapitre, nous définissons le domaine de reconnaissance de formes et ses caractéristiques principales et nous introduisons les différents types des formes (motifs, pattern) traité par cette branche.

2. Reconnaissance des formes (RF)

2.1. Définition

On désigne par **reconnaissance de formes** (ou parfois **reconnaissance de motifs**) un ensemble de techniques et méthodes visant à identifier des motifs à partir de données brutes afin de prendre une décision dépendant de la catégorie attribuée à ce motif. On considère que c'est une branche de l'intelligence artificielle qui fait largement appel aux techniques d'apprentissage automatique et aux statistiques [68].
Les formes à reconnaître peuvent être de nature très variée. Il peut s'agir de contenu visuel (code barre, visage, empreinte digitale…) ou sonore (reconnaissance de parole), d'images médicales (rayon X, EEG, IRM…) ou multi spectrales (images satellitaires) et bien d'autres [68].

Watanabe [53] a défini une forme comme : *« l'opposé du chaos ; c'est une entité vaguement définie, à laquelle on peut associer un nom* ». En des termes informatiques, une forme est un ensemble de valeurs, appelés attributs, auxquels est associé un nom (ou étiquette), qui est leur classe. Plusieurs formes peuvent avoir la même classe, on dit alors que ce sont les exemples ou réalisations de la classe [68].

Le problème que cherche à résoudre la reconnaissance des formes est d'associer une classe à une forme inconnue (qui n'a pas encore de classe associée). On considère souvent la Reconnaissance des formes comme un problème de classification : trouver la fonction qui affecte à toute forme inconnue sa classe la plus pertinente. Elle est partie intégrante de tout système intelligent destine à la prise de décision [54] [68].

2.2.Historique

Or que ce soit pour déchiffrer un texte dactylographié ou manuscrit, pour compter des chromosomes, reconnaître une tumeur, un char ou un avion de guerre, la compréhension de l'image, sa classification passe toujours par la reconnaissance d'une forme. « *Plusieurs approches théoriques ont été développées* », explique *Olivier Faugeras* [68].

« Les premières consistaient à faire des calculs à partir de l'image et construire des représentations symboliques de plus en plus complexes, d'abord en deux dimensions tel que sur l'image, puis tridimensionnelles, pour tenter de restituer une description proche de notre propre vision. » Un peu partout dans le monde, les chercheurs ont mis au point des méthodes mathématiques permettant de détecter les contours des objets à partir des changements rapides de contraste dans l'image, des ombres et des lumières, des régions homogènes en couleur, en intensité, en texture [68].

« Dès 1964, des chercheurs français, Georges Matheron (1930-2000) et Jean Serra, ont développé une autre approche théorique (baptisée morphologie mathématique) et un outil spécifique (l'analyseur de texture breveté en 1965, ndlr) d'abord pour analyser des microphotographies de terrain et évaluer des teneurs en minerai, puis pour d'autres applications comme la cytologie (caractérisation et comptage de cellules) » rappelle Olivier Faugeras. En 1968, ils créent le Centre de morphologie mathématique de l'Ecole des Mines de Fontainebleau. Leurs outils d'analyse et d'interprétation d'images sont longtemps restés franco-français, jusqu'à ce qu'un américain, Robert Haralick (Université du Kansas à cette époque, de Seattle actuellement), en fasse une large publicité dans les années 1980, en les adaptant à de nombreuses applications : industrielles comme l'inspection radiographique des ailes d'avions de Boeing, aériennes ou médicales [67] [68].

D'autres chercheurs, comme les américains Marvin Minsky et Seymour Papert du MIT (Massachussets Institute of Technology) ont considéré le problème dans l'autre sens, en cherchant à formaliser et à faire reproduire par l'ordinateur notre propre processus de reconnaissance d'images, donc notre propre vision. Cette démarche était dans l'air du temps, au cœur des promesses de « l'intelligence artificielle » qui devait permettre de mettre l'intelligence en équations et doter les ordinateurs de toutes les capacités humaines de raisonnement, mémoire, perception. Or la vision s'est révélée être un domaine particulièrement complexe à modéliser tant elle est basée sur une quantité phénoménale de connaissances à priori fondée sur notre intelligence et notre expérience [55] [68].

2. Méthodes

La reconnaissance de motifs peut être effectuée au moyen de divers algorithmes d'apprentissage automatique tels: un réseau de neurones. Les formes recherchées peuvent être des formes géométriques, descriptibles par une formule mathématique, telles que: cercle ou

ellipse, courbes de Bézier, splines, droite. Elles peuvent aussi être de nature plus complexe telles que : Lettre, chiffre, empreinte digitale.

Les algorithmes de reconnaissance peuvent travailler sur des images en noir et blanc, avec en blanc les contours des objets se trouvant dans l'image. Ces images sont le fruit d'algorithmes de détection de contours. Ils peuvent aussi travailler sur des zones de l'image prédéfinies issues de la segmentation de l'image [68].

3. la reconnaissance de plusieurs objets dans une image

Une seule image peut être constituée d'un ou plusieurs objets [56]. Dans le cas où on désire détecter plusieurs objets dans une même image, on peut utiliser le procédé de reconnaissance multi objets représenté sur la figure 2.1 ci- dessous [68].

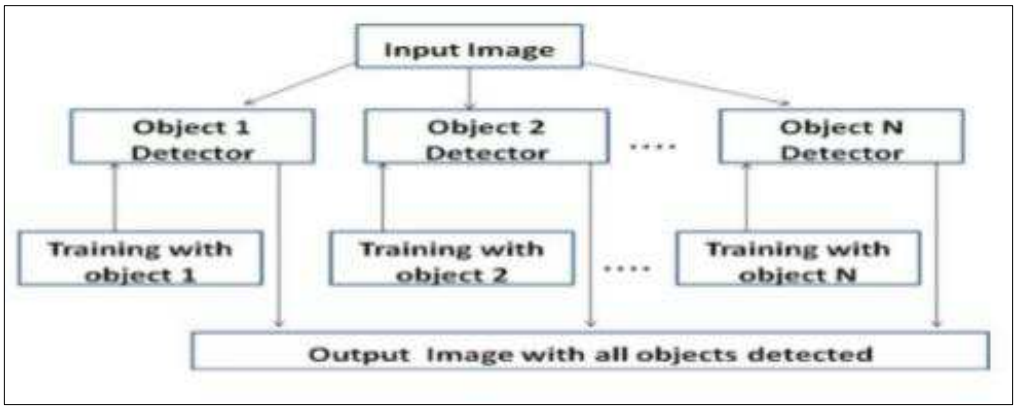


Figure 2.1 : La méthode de reconnaissance de multi objets dans une image

4. Application Typique de la reconnaissance des formes

4.1. Marketing

La reconnaissance des formes est souvent utilisée pour classer les consommateurs selon les produits qu'ils sont susceptibles d'acheter. Elle est aussi utilisée par les sociétés de vente pour classer les clients selon qu'ils soient de bons ou mauvais payeurs, ou encore selon qu'ils vont oui ou non passer à la concurrence [57] [68].

4.2. Finances

Les systèmes de reconnaissance des formes sont utilisés pour la détection de transactions bancaires frauduleuses ainsi que la prédiction des banqueroutes [58] [68].

4.3. Usinage

La qualité des produits dépend souvent de para métrisation correcte, et les relations exactes entre la qualité et les valeurs des paramètres n'est pas claire. Les systèmes de reconnaissance

des formes sont utilisés pour classer les paramètres selon la qualité des produits qu'ils sont susceptibles de générer. Ils permettent ainsi de réduire le nombre d'essais ce qui fait gagner du temps et de l'argent [59] [68].

4.4. Energie

Les systèmes de reconnaissance des formes sont utilisés pour prévoir la consommation électrique (réduite, normale, élevée), permettant ainsi aux clients de réduire si nécessaire leur consommation, et aux producteurs de mieux gérer leurs unités de production [60] [68].

4.5. Lecture automatisée

Les systèmes de reconnaissance des formes permettent de numériser les anciens documents ainsi que les archives, non pas sous la forme d'images, mais plutôt sous une forme textuelle [61] [68].

4.6. Sécurité

La reconnaissance vocale et rétinienne est un exemple d'applications typiques de la reconnaissance des formes pour l'authentification. La vérification des signatures est aussi très populaire [62] [68].

5. Schéma général d'un système de Reconnaissance des Formes

La majorité des systèmes de RF ont le schéma de fonctionnement suivant :

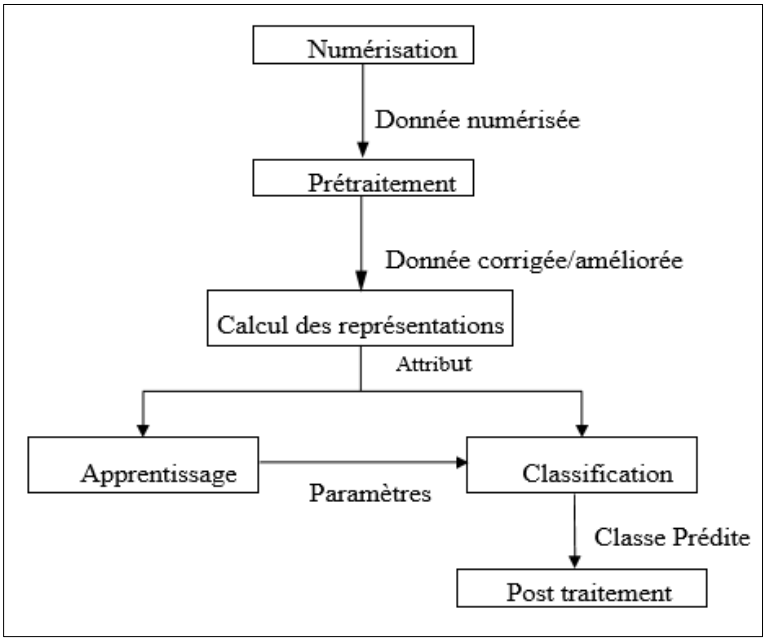


Figure 2.2 : Schéma général d'un système de reconnaissance des formes

5.1 Préparation des données

a. Numérisation: À partir des informations du monde physique, construire une représentation des données directement manipulable par la machine. Des capteurs (microphone, caméra, instruments de mesure) convertissent les signaux reçus du monde réel en une représentation numérique discrète. L'espace résultant, appelé *espace de représentation* a une dimension **r** très grande lui permettant de disposer du maximum d'informations sur les formes numérisées [68].

b. Prétraitement: Consiste à sélectionner dans l'espace de représentation l'information nécessaire au domaine d'application. Cette sélection passe souvent par l'élimination du bruit, la normalisation des données, ainsi que par la suppression de la redondance. Le nouvel espace de représentation a une dimension **r'** très inférieure à **r** mais demeure un espace de grande dimension et contient des informations encore assez primitives [68].

c. Calcul des représentations: Il s'agit de la phase finale de la préparation des données. Elle fournit un certain nombre de caractéristiques ou paramètres (les fameux attributs) en utilisant des algorithmes de sélection et/ou d'extraction d'attributs. Les attributs étant limités en nombre, *l'espace des paramètres* ainsi obtenu est de dimension p très petite par rapport à **r'** [68].

5.2 Apprentissage

L'apprentissage ou entraînement, est une partie importante du système de reconnaissance. Le classificateur étant généralement une fonction paramétrique, l'apprentissage va permettre d'optimiser les paramètres du classificateur pour le problème à résoudre, en utilisant des données d'entraînement. Lorsque les données d'entraînement sont préalablement classées, l'apprentissage est dit supervisé, sinon il est non supervisé [64] [68].

5.3 Classification

Cette phase est le noyau de la Reconnaissance des formes. En utilisant les modèles (paramètres) obtenus lors de l'apprentissage, le classificateur assigne à chaque forme inconnue sa ou ses formes les plus probables [68].

5.4 Post traitement

Cette phase a pour but de corriger les résultats de la classification en utilisant des outils spécifiques au domaine d'application. Par exemple pour un système de reconnaissance de textes manuscrits, le classificateur se charge de classer chaque caractère séparément, alors que

le post traitement appliqué un correcteur orthographique sur tout le texte pour valider et éventuellement corriger le résultat de la classification. Bien que facultative, cette phase permet d'améliorer considérablement la qualité de la reconnaissance [68].

6. Traitement d'image

L'image fournie par le capteur est transformée en un signal électrique. De ce signal il faut extraire les informations recherchées sur le contenu de la scène dont l'image a été captée. Les premières bases du traitement d'images sont directement issues du traitement du signal, phénomène normal puisque toute image, qu'elle soit continue ou numérique, peut être considérée comme un signal à 2 dimensions [68].

La vision n'est pas un domaine facile, car repérer un objet simple dans une image demande beaucoup d'opérations. Le traitement, souvent appelé prétraitement, regroupe toutes les techniques visant à améliorer la qualité d'une image. De ce fait, la donnée de départ est l'image initiale et le résultat est également une image [68].

La représentation la plus élémentaire correspond à l'image binaire pour laquelle chaque pixel ne peut prendre qu'une valeur parmi deux autres. Pour les images monochromes, chaque pixel peut prendre une valeur parmi N. N correspond généralement à une puissance de 2, ce qui facilite la représentation de l'image en machine. Par exemple, pour une image en niveau de gris chacun des pixels peut prendre une valeur parmi 256. Sa valeur est alors codée par un octet de donnée. Une image est constituée par une matrice X lignes et Y colonnes de pixels chacun codé par x bits [68]. Les données quantitatives liées à la représentation des images sont représentées dans le tableau suivant (Table 2.3) :

1 bit	2 couleurs (noir & blanc)
4 bits	16 couleurs
8 bits	256 couleurs ou niveaux de gris
16 bits	65536 couleurs
24 bits	16777216 couleurs (vraies couleurs)
32 bits	4 294 967 296 couleurs

Table 2.3 : Données quantitatives des images

6.1.Définition de l'image

Une image est constituée d'un ensemble de points pixels (Picture Element). Il représente le plus petit élément constituant d'une image numérique (on parle d'image numérique lorsque les quantités physiques qui caractérisent l'image sont converties par des valeurs numériques).

L'ensemble de ces pixels est contenu dans un tableau à deux dimensions constituant l'image. Les axes de l'image sont orientés de la façon suivante [68]:

- L'axe X est orienté à droite (largeur).
- L'axe Y est orienté de haut en bas (hauteur), contrairement aux notations conventionnelles en mathématiques, ou l'axe Y est orienté vers le haut.

Pour représenter informatiquement une image, il suffit donc de créer un tableau de pixels dont chaque case est codée par un certain nombre de débits déterminant la couleur ou l'intensité du pixel, la figure 2.4 présente un réseau de pixels (L'élément grisé encadré par la couleur rouge correspond aux coordonnées i, j) [68].

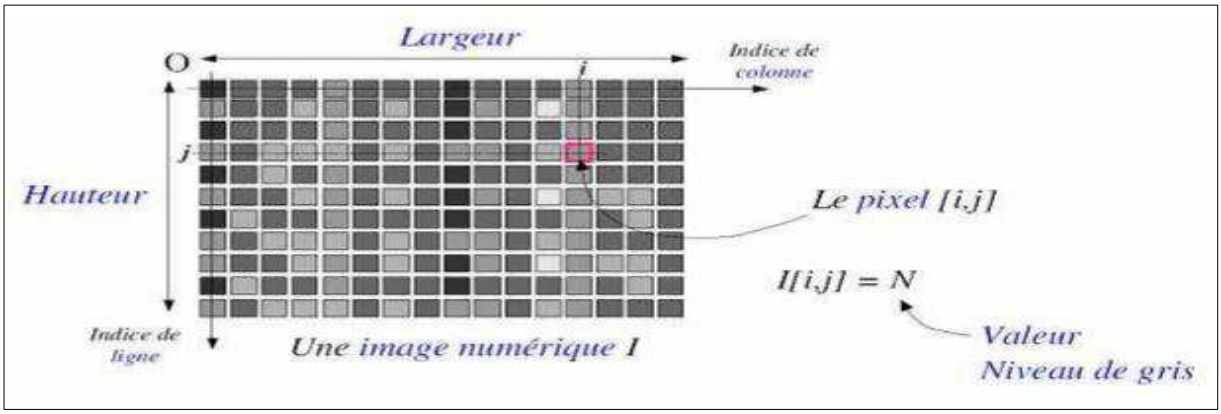


Figure 2.4 : Exemple de réseau de pixels [54]

Pixel: Le terme pixel est la contraction de l'anglais « Picture Element ». C'est le plus petit élément de l'image. Sortie de l'appareil photo une image est composée d'un nombre x de pixels. Un pixel a une couleur exprimée (codée) en langage binaire mathématique.

6.2. Acquisition d'une image

L'acquisition d'images constitue un des maillons essentiels de toute chaîne de conception et de production d'images. Pour pouvoir manipuler une image sur un système informatique, il est avant tout nécessaire de lui faire subir une transformation qui la rendra lisible et manipulable par ce système. Le passage de cet objet externe (l'image d'origine) à sa représentation interne (dans l'unité de traitement) se fait grâce à une procédure de numérisation. Ces systèmes de saisie, dénommés optiques, peuvent être classés en deux catégories principales:

- Les caméras numériques,
- Et les scanners.

A ce niveau, notons que le principe utilisé par le scanner est de plus en plus adapté aux domaines professionnels utilisant le traitement de l'image comme la télédétection, les arts graphiques, la médecine, etc. Le développement technologique a permis l'apparition de nouveaux périphériques d'acquisition appelés cartes d'acquisition, qui fonctionnent à l'instar des caméras vidéo, grâce à un capteur C.C.D. (Charge Coupled Device). La carte d'acquisition reçoit les images de la camera, de la T.V. ou du scanner afin de les convertir en informations binaires qui seront stockées dans un fichier.

6.3.Caractéristiques d'une image numérique

6.3.1. Dimension

C'est la taille de l'image. Cette dernière se présente sous forme de matrice dont les éléments sont des valeurs numériques représentatives des intensités lumineuses (pixels). Le nombre de lignes de cette matrice multipliée par le nombre de colonnes nous donne le nombre total de pixels dans une image.

6.3.2. Résolution

C'est la clarté ou la finesse de détails atteinte par un moniteur ou une imprimante dans la production d'images. Sur les moniteurs d'ordinateurs, la résolution est exprimée en nombre de pixels par unité de mesure (pouce ou centimètre). On utilise aussi le mot résolution pour désigner le nombre total de pixels affichables horizontalement ou verticalement sur un moniteur; plus grand est ce nombre, meilleure est la résolution.

6.3.3. Bruit

Un bruit dans une image est considéré comme un phénomène de brusque variation de l'intensité d'un pixel par rapport à ses voisins, il provient de l'éclairage des dispositifs optiques et électroniques du capteur.

6.3.4. Histogramme

L'histogramme des niveaux de gris ou des couleurs d'une image est une fonction qui donne la fréquence d'apparition de chaque niveau de gris (couleur) dans l'image. Il permet de donner un grand nombre d'information sur la distribution des niveaux de gris (couleur) et de voir entre quelles bornes est repartie la majorité des niveaux de gris (couleur) dans le cas d'une image trop claire ou d'une image trop foncée.

Il peut être utilisé pour améliorer la qualité d'une image (Rehaussement d'image) en introduisant quelques modifications, pour pouvoir extraire les informations utiles de celle-ci. Pour diminuer l'erreur de quantification, pour comparer deux images obtenues sous des éclairages différents, ou encore pour mesurer certaines propriétés sur une image, on modifie souvent l'histogramme correspondant.

6.3.5. Luminance

C'est le degré de luminosité des points de l'image. Elle est définie aussi comme étant le quotient de l'intensité lumineuse d'une surface par l'aire apparente de cette surface, pour un observateur lointain, le mot luminance est substitué au mot brillance, qui correspond à l'éclat d'un objet. Une bonne luminance se caractérise par:

- Des images lumineuses (brillantes);
- Un bon contraste : il faut éviter les images où la gamme de contraste tend vers le blanc ou le noir; ces images entraînent des pertes de détails dans les zones sombres ou lumineuses.
- L'absence de parasites.

6.3.6. Contraste

C'est l'opposition marquée entre deux régions d'une image, plus précisément entre les régions sombres et les régions claires de cette image. Le contraste est défini en fonction des luminances de deux zones d'images. Si L1 et L2 sont les degrés de luminosité respectivement de deux zones voisines A1 et A2 d'une image, le contraste C est défini par le rapport:

$$C = \frac{L_1 + L_2}{L_1 + L_2}$$

6.3.7. Images à niveaux de gris

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. Donc pour représenter les images à niveaux de gris, on peut attribuer à chaque pixel de l'image une valeur correspondant à la quantité de lumière renvoyée. Cette valeur peut être comprise par exemple entre 0 et 255. Chaque pixel n'est donc plus représenté par un bit, mais par un octet. Pour cela, il faut que le matériel utilisé pour afficher l'image soit capable de produire les différents niveaux de gris correspondant. Le nombre de niveaux de gris dépend

du nombre de bits utilisés pour décrire la " couleur " de chaque pixel de l'image. Plus ce nombre est important, plus les niveaux possibles sont nombreux.
Il existe plusieurs manières de convertir une image RGB en niveaux de gris. La plus simple est de faire :

$$gris = \frac{rouge + vert + bleu}{3}$$

Cela équivaut aussi à affecter la couleur en niveau de gris à chacune des trois composantes RGB. L'idéal est de faire ressortir la luminosité d'un pixel. Celle-ci vient principalement de la présence de la couleur verte. On emploie généralement les coefficients suivants [68] :

$$\text{Gris} = 0,299 \cdot \text{rouge} + 0,587 \cdot \text{vert} + 0,114 \cdot \text{bleu}$$

Dans ce cas on dispose d'une échelle de teintes de gris, et la plupart du temps on dispose de 256 niveaux de gris avec [68] :

0 ----> noir, ...127 ---> gris moyen, ..., 255 ---> blanc

Certaines images peuvent être codées sur deux octets ou plus (certaines images médicales, des images astronomiques...) ce qui peut poser des problèmes dans la mesure où les systèmes de traitement d'images courants supposent l'utilisation des pixels d'un octet.

6.3.8. Images en couleurs

Même s'il est parfois utile de pouvoir représenter des images en noir et blanc, les applications multimédias utilisent le plus souvent des images en couleurs. La représentation des couleurs s'effectue de la même manière que les images monochromes avec cependant quelques particularités. En effet, il faut tout d'abord choisir un modèle de représentation. On peut représenter les couleurs à l'aide de leurs composantes primaires. Les systèmes émettant de la lumière (écrans d'ordinateurs,) sont basés sur le principe de la synthèse additive : les couleurs sont composées d'un mélange de rouge, vert et bleu (modèle R.V.B.)

3.4 Système de traitement d'images

Un système de traitement numérique d'images est composé de:

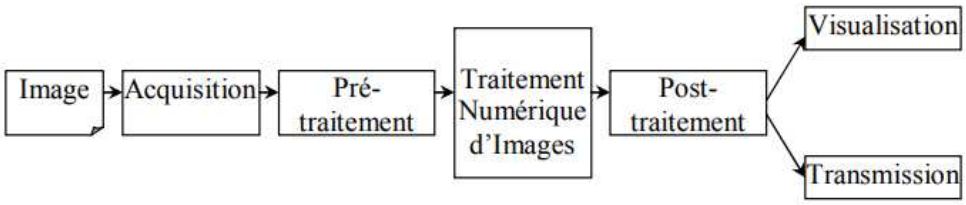


Figure 2.5. Schéma d'un système de traitement d'images

7. segmentation des images

La segmentation est une étape essentielle en traitement d'images et reste un problème complexe. La segmentation est un processus de la vision par ordinateur, généralement c'est la première étape de l'analyse d'image qui vient après le prétraitement. La segmentation est l'extraction de caractéristiques de l'objet, ce qui permet une distinction entre l'objet et le fond [68]. La segmentation aide à localiser et à délimiter les entités présentes dans l'image. Il existe une multitude de méthodes de segmentation dont l'efficacité reste difficile à évaluer. La segmentation des images est le procédé qui conduit à un découpage de l'image en un nombre fini de régions (ou segments) bien définies qui correspondent à des objets, des parties d'objets ou des groupes d'objets qui apparaissent dans une image. C'est une transformation très utile en vision artificielle [68].

Une erreur dans la segmentation de la forme à reconnaître augmente forcément le risque d'une mauvaise reconnaissance. Essentiellement, l'analyse de l'image fait appel à la segmentation où l'on va tenter d'associer à chaque pixel de l'image un label en s'appuyant sur l'information portée (niveaux de gris ou couleur), sa distribution spatiale sur le support image, des modèles simples (le plus souvent des modèles géométriques) [68].

7.1 Les principes de la segmentation

De nombreux travaux ont été réalisés sur ce sujet, dans des domaines aussi variés que le domaine médical, militaire, industriel, géophysique, etc… C'est toujours un sujet d'actualité et un problème qui reste ouvert où l'on retrouve de très nombreuses approches [68] :

- La segmentation par régions ;
- La segmentation par seuillage ;
- La segmentation par contours ;
- La Transformée de Hough ;
- La segmentation par étiquetage en composantes connexes ;

- La segmentation par LPE.

Toutes ces approches visent à l'extraction des caractéristiques. Après de nombreuses années passées à rechercher la méthode optimale, les chercheurs ont compris que la segmentation idéale n'existait pas. Il n'existe pas d'algorithme universel de segmentation à chaque type d'images correspond une approche spécifique. Une bonne méthode de segmentation sera donc celle qui permettra d'arriver à une bonne interprétation [68].

8. Reconnaissance de l'écriture manuscrite

La reconnaissance de l'écriture manuscrite est un traitement informatique qui a pour but de traduire un texte écrit en un texte codé numériquement. Il faut distinguer deux reconnaissances distinctes, avec des problématiques et des solutions différentes :

- La reconnaissance en ligne ;
- La reconnaissance hors-ligne.

La reconnaissance de l'écriture manuscrite fait appel à la reconnaissance de forme, mais également au traitement automatique du langage naturel. Cela veut dire que le système, tout comme le cerveau humain, reconnaît des mots et des phrases existant dans un langage connu plutôt qu'une succession de caractères. Ceci améliore grandement la robustesse

8.1. Reconnaissance hors-ligne

Le mode de reconnaissance hors linge est la reconnaissance des manuscrite écrit au préalable qui se présente au système de reconnaissance sous forme d'image discrète binaire en niveau de gris scanner ou bien pris à l'aide d'une caméra, L'écriture prend l'aspect d'un signal spatial bidimensionnel numérisé. C'est se mode qui va être traité dans ce projet.

8.2. Reconnaissance en-ligne

En cas en linge la reconnaissance est faite en fur et à mesure elle est effectuée en temp réel au moment où le caractère est tracé, par tracé en veut dire que les caractères sont transmis au système sous forme de coordonnés en fonction de temps saisi à l'aide d'un stylo et une tablette à numériser cette approche et appelée aussi approche « signal » [79]. 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 [80].

8.3. Travaux connexe

Cette partie est consacrée pour l'analyse des résultats de différents articles qui s'intéressent sur la reconnaissance de l'écriture manuscrite arabe et qui utilisent différentes méthodes de l'apprentissage automatique.

1.1.1. Le travail de M. Amrouche et al

Le travail de M. Amrouche et al. [69] utilise deux méthodes pour l'extraction des caractéristiques et fait une comparaison entre les deux pour définir laquelle est la plus performante, les deux méthodes sont les Réseaux des Neurones à Convolution (CNN) et Hnadcraft Features puis ils utilisent le Hidden Markov Modèles (HMM) pour la reconnaissance.

Dans l'expérimentation ils utilisent la base de données IFN/ENIT en suivant le scénario ''abc-d'' les sous-ensembles de données (abc) sont utilisés pour l'entrainement et la validation de l'approche, un autre sous-ensemble (d) utilisé pour la phase de test. La méthode de Handcraft Features donne un résultat de 87.93% et la méthode de CNN donne un résultat de 88.95% ce qui prouve que l'utilisation de CNN donne un meilleur résultat que Handcrafts Features avec un taux d'augmentation dans la précision est égale à 1.02%.

1.1.2. Le travail de M. Rabi et al

Dans les recherches de M. Rabi et al. [70] ils utilisent le modèle HMM avec l'algorithme de Baun Welch pour l'estimation des paramètres, l'application du modèle est sur la base de données IFN/ENIT aussi et en suivant le scénario ''abc-d'' le résultat obtenu du modèle proposé est à 87.93% dans le travail on trouvera une comparaison du système réalisé avec d'autres systèmes où le modèle proposé donne un taux de résultat élevé par rapport aux autres systèmes ce qui prouve que la reconnaissance a été améliorée pour une bonne performance.

1.1.3. Le travail de M. Abdullah et al

Dans le travail de M. Abdullah et al. [71] ils utilisent le modèle de reconnaissance des caractères optique (OCR) avec un classifieur Réseaux de Neurones Artificiel (RNA) pour l'extraction des caractères, en utilisant la base IFN/ENIT, mais seulement avec 100 mots choisis. Le taux de reconnaissance du système est égal à 70%, mais avec un test sur un ensemble de mots en dehors de la base le taux de reconnaissance a été amélioré a 90%. Ce qui prouve que le système proposé fait une bonne reconnaissance. Ce travail a fait une comparaison avec un autre travail qui utilise le modèle HMM qui donne un taux de 73% se qui prouve que le système proposé donne un meilleur résultat sur des nouvelles données différentes.

1.1.4. Le travail de H.A. Abed et al

Le travail de H.A. Abed et al. [72] fait une reconnaissance en utilisant l'algorithme de réseau de neurones artificiel de back-propagation d'erreur (EBPANN) sur 12 lettres isolées le résultat obtenu est à 93.608%. Durant la phase de validation quelques lettres donnent un taux de reconnaissance jusqu'à 100% et d'autre lettre donne un taux de 75.52% les résultats de la recherche prouvent que le système minimise le taux d'erreur durant la reconnaissance.

1.1.5. Le travail de A. Djeffal et al

Dans la recherche de A. Djeffal et al. [75] le système réalisé propose une reconnaissance des caractères en utilisant le classifieur support vecteur machine (SVM) pour la classification. Pour la validation du système, des classes de caractère ont été utilisées ce qui donne un taux de reconnaissance différent pour chaque caractère de chaque classe, il donne des résultats entre 100% et 75%. Malgré ces résultats, le système a trouvé des difficultés à reconnaitre des mots avec des caractères attachés et des mots avec des lignes discontinues.

1.1.6. Le travail de A. Mars et al

Dans la recherche de A. Mars et al. [73] il utilise une reconnaissance de l'écriture manuscrite arabe online, c'est la reconnaissance des formes écrites avec un stylo électronique. La base utilisée est un ensemble de 100 formes écrites de la part de différentes personnes. En utilisant les RNA avec le modèle CNN précisément Réseau de neurones à délai (TDNN), le système réalisé a été comparé avec le système de reconnaissance MyScript, il fait une évaluation sur les lettres indépendamment et une autre évaluation sur les mots où il donne un résultat de 98.5% pour les lettres et 96.9% pour les mots. Par contre le système MyScript donne un résultat meilleur que celui du système proposé. Ces résultats nous prouvent que le système est effectif.

1.1.7. Le travail de B. Alija et al

Le travail de B. Alija et al. [74] fait une reconnaissance sur l'écriture manuscrite arabe online où la collection de données à traité est écrite par des personnes avec une souris, le modèle proposé est d'utilisé quatre RNA un pour chaque étape. Le résultat obtenu est 99.1% durant la phase d'entrainement est 95.7% durant la phase de test. Le modèle donc donne une bonne précision durant la reconnaissance.

1.2. Comparaison entre les travaux réalisés

Les travaux cités dans la partie précédente concernant le domaine de la reconnaissance de l'écriture manuscrites arabes utilisent tous presque la même base de données IFN/ENIT, la comparaison entre eux est fait en fonction des résultats obtenus et le nombre d'échantillons

utilisées. Le but est de nous aider à proposer une approche en appliquant des nouvelles techniques sur la même base de données.

Le tableau suivant donne un résumé des résultats des recherches mentionnées :

Articles	Ensemble de données	Modèles	Résultats
M. Amrouche et al. [69]	Base de données IFN/ENIT (abc-d)	Handcraft Features	87.93%
		CNN	88.95
M. Rabi et al. [70]	Base de données IFN/ENIT (abc-d)	HMM	87.93%
M. Abdullah et al. [71]	IFN/ENIT (100 mots)	OCR(RNA)	70%
	Mots en dehors de la base	OCR(RNA)	90%
H.A. Abed et al. [72]	12 Lettres	EBPANN	93.608%
Djeffal et al. [76]	Classes de caractères	SVM	100%-75%
A. Mars et al. [74]	100 formes écrites (lettres)	CNN (TDNN)	98.5%
	100 formes écrites (mots)	CNN (TDNN)	96.9%
B. Alija et al. [75]	Données écrite online	RNA	95.7%

Tableu2.3. Lettres arabes avec des points diacritiques

Conclusion

On peut considérer les systèmes de reconnaissance des formes comme des systèmes à l'affectation d'une étiquette à une valeur d'entrée donnée, les méthodes de reconnaissance de formes visent généralement à fournir une réponse raisonnable pour toutes les entrées possibles et à effectuer une correspondance «la plus probable » des entrées, en tenant compte de leur variation statistique.

La Reconnaissance de Formes met en œuvre plusieurs étapes: segmentation des objets (analyse d'images), extraction de caractéristiques (géométrie, invariants, …), classification (supervisée ou non, méthodes probabilistes, statistiques, …). Les applications sont très variées et nécessitent des connaissances expertes du domaine d'application. Les méthodes sont nombreuses mais les principes de base sont assez stables.

Dans le prochain chapitre, nous allons décrire notre modèle proposé et son implémentation pour la reconnaissance de manuscrit arabe en utilisant deux méthodes pour la classification et la reconnaissance. (CNN et les réseaux de capsules).

Chapitre03

Conception et implémentation

III.1 Introduction

La reconnaissance de caractères a suscité une attention considérable au cours des dernières décennies. Les chercheurs ont fait des percées dans ce domaine avec le développement rapide d'algorithmes d'apprentissage profond et les applications ont connu une grandes réussite dans ce domaine. Jusqu' présent, ils utilisent des méthodes basées sur les réseaux de neurones. Les réseaux de neurones convolutifs (en angalais : Convolution Neural Netwrok CNN) ont connu un grand succès dans diverses applications de reconnaissance d'images, y compris celles de la reconnaissance de caractères manuscrits. Dans notre projet, nous nous sommes intéressés à la création d'un système de reconnaissance de caractères arabe manuscrit qui utilise des méthodes de traitement d'images basées sur l'apprentissage profond. Pour atteindre notre objectif, nous proposons dans ce chapitre une nouvelle architecture appliquée à la méthode des réseaux de neurones convolutifs (CNN) et les réseaux de capsules (Caps Net).

Ce dernier chapitre est consacré à la conception et la mise en place de notre projet qui permettra d'identifier les mots et les caracters de la langue arabe par reconnaissance des formes en utilisant deux méthodes appartiennent à la famille de Deep learning (apprentissage profond). Dans ce chapitre nous allons décrire également les différentes parties de notre système, les détails relatifs à chaque phase ainsi que l'explication et les details de chaque etape de ces phases.

III.2 Conception générale de notre système

Généralement, le processus de reconnaissance de caractères arabe a partir d'une image suit ces 5 étape essentielles :

1. Extraction du texte de l'image.
2. Prétraitement d'image.
3. Segmentation des caractères présents dans l'image.
4. Extraction de caractéristiques à partir de caractères segmentés.
5. Classification des caractères.

Notre syteme reçoit une image comme entrée , il va ensuite commencer la premiere étape "Extraction de text " qu'elle meme se compose d'une suite d'étape la plus importante est l'étape de prétraimtemnt , ou une suite d'instructions se projete sur l'image afin qu'on puisse identifier les zones de texes , ensuite commance la phase la plus importante la phase de segmentation ou nous allons segementer chaque texte identifier dans l'image a des suite d'imagettes et de caractere isoler, . ses dernières vont passer image par image à l'étape suivante reconnaissance

on doit noter que cette étape effectuera plusieurs taches : la construction de classifieur qui permet d'extraire les caractéristiques principales de l'image et reconnaitre la classe correcte du caractère dans image(classification).

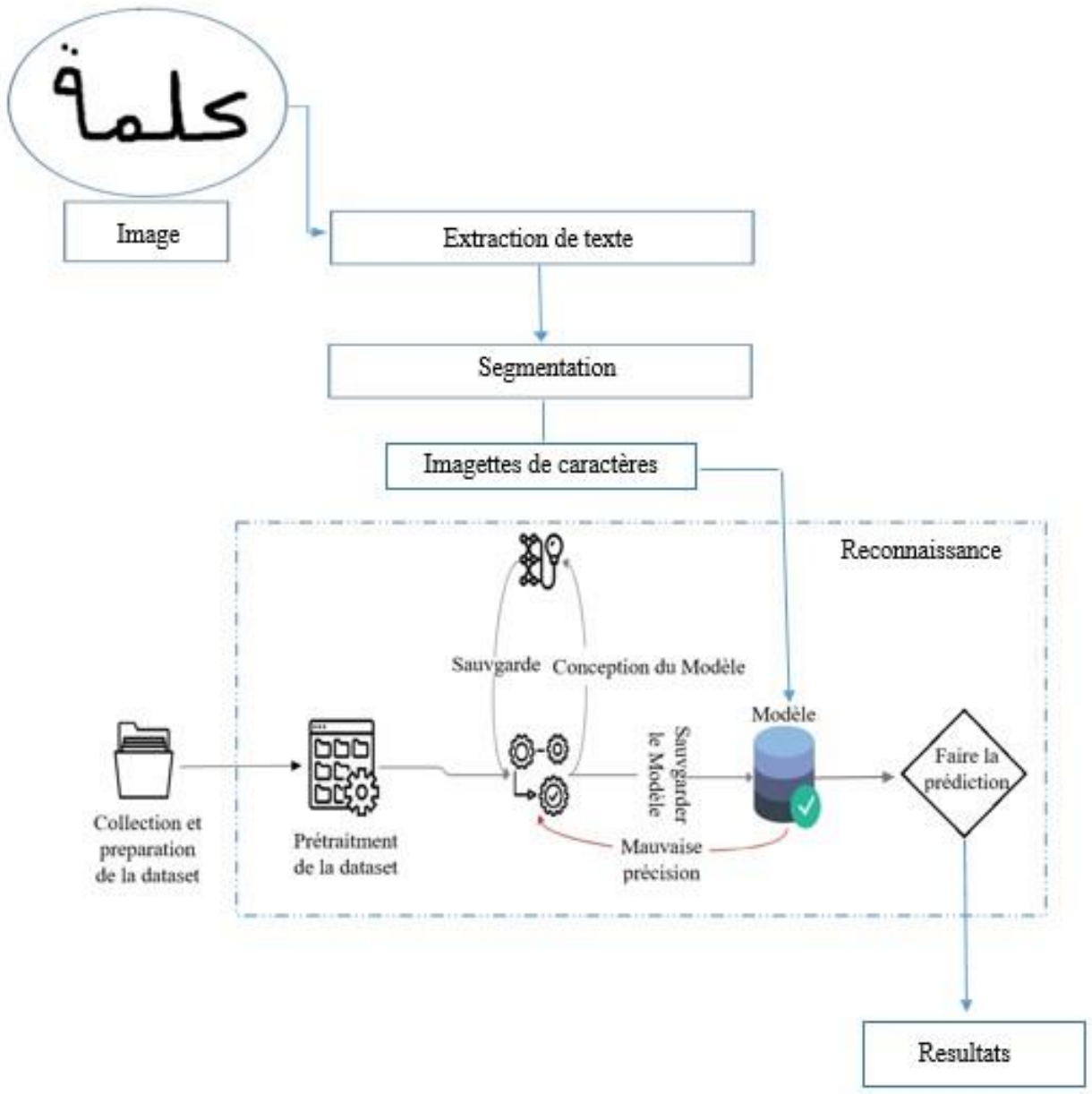


Figure III.1 – Architecture de notre système de reconnaissance

III.3 Conception détaillé

III.3.1 Extraction de text

Cette première phase consiste a extraire les text ou le mot ou bien le caractère dessiné sur l'image. pour cela il faut localiser toutes les zones de texte artificiel dans l'images. Il y a différentes approches qui traitent ce problème. Nous avons choisis et appliqué une méthode basée sur l'uniformité de la couleur pour la détection du texte, c'est une méthode utilsé on

général dans l'extraction des textes a partir d'une vidéo , mais comme une vidéo c'est une séquence de frame , donc l'application de la méthode directement sur une image va beaucoup nous faciliter la tache, dans le prétraitement on va expliquer on détail comment la méthode fonctionne du prétraitment jusqu'au résultat[69]

Figure III.2 – écriture arabe manuscrite dans une image

Figure III.3 – Détection de l'écriture arabe manuscrite dans une image

III.3.1.1 Prétraitement d'image

La méthode est basée sur l'hypothèse que les textes artificiels sont des régions caractérisées par une forte densité en contours verticaux. Elle est composée de 6 étapes[69]

1. Calculer l'image des gradients verticaux.
2. Binariser l'image des gradients.
3. Détecter les régions denses en contours verticaux.
4. Rendre les régions plus compactes.
5. Supprimer les fausses régions.
6. Localiser les régions trouvées dans des boîtes englobantes.

ci-dessous on va expliquer les étapes les plus importantes pour l'extraction de text d'une image.

III.3.1.2 Calculer l'image des gradients verticaux

L'image est convertie en niveaux de gris puisque la couleur n'est pas une caractéristique intrinsèque des textes. Puis, l'image est lissée avec un filtre gaussien pour atténuer le bruit avant de calculer le gradient vertical par l'utilisation d'opérateurs morphologiques : la différence entre la dilatation horizontale et l'érosion horizontale de l'image :

$$g(x, y) = \mathbb{V}(\mathbf{f}(x, y)) - \mathbb{V}(\mathbf{f}(x, y))$$

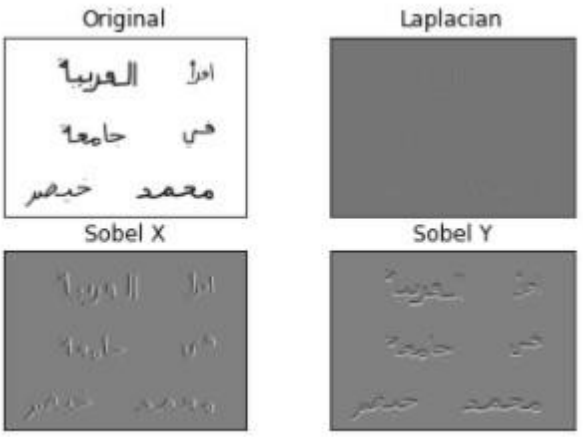


Figure III.4 – Image aprés l'application des gradients verticaux

III.3.1.3 Binariser l'image des gradients.

La binarisation est utilisée pour garder les structures verticales les plus contrastées. Le seuil est calculé à partir de la valeur d'entropie qui est bien adaptée à la détection de petites structures fines reposant sur un grand fond uni.

Pratiquement, la valeur de seuil est déterminée comme le niveau de gris qui maximise la quantité totale d'information fournie par le fond et les objets séparément. La quantité d'information est mesurée par l'entropie.

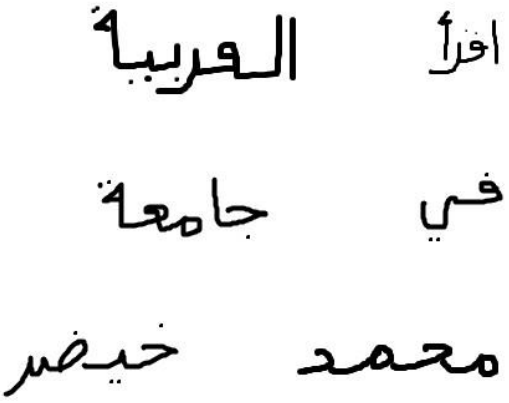


Figure III.5 – Avant la Binarisation par entropie

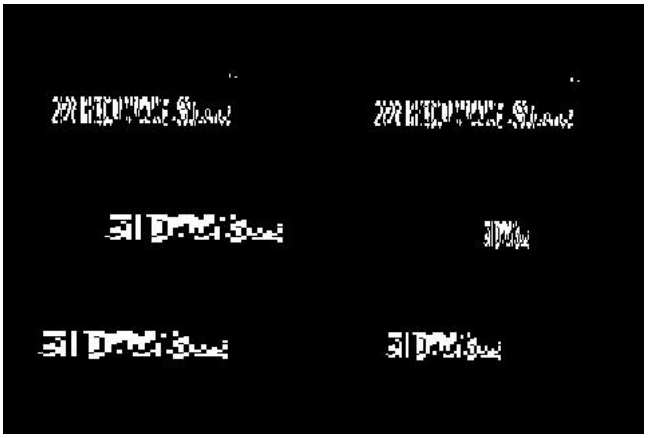


Figure III.6 – Après la Binarisation par entropie

III.3.1.4 Localiser les régions à l'intérieur de boîtes englobantes.

La localisation des régions est faite dans des boîtes englobantes qui sont ensuite dilatées de 2 pixels pour bien prendre englober tous les pixels des caractères.

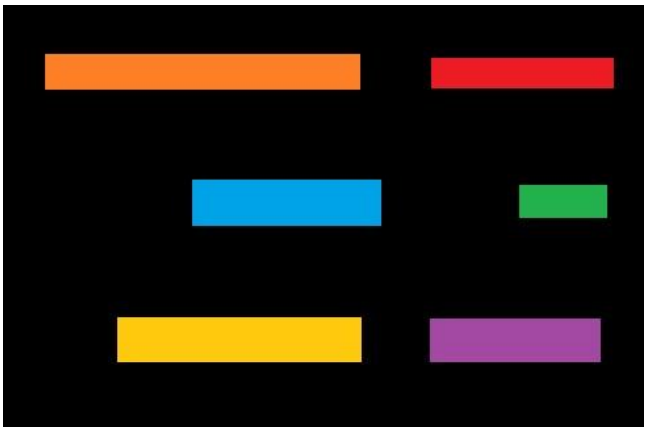


Figure III.7 – Les boîtes englobantes résultantes.

•Superposer le résultat à l'image initiale Les frontières des boîtes englobantes sont superposées à l'image initiale pour une meilleure visualisation des résultats.

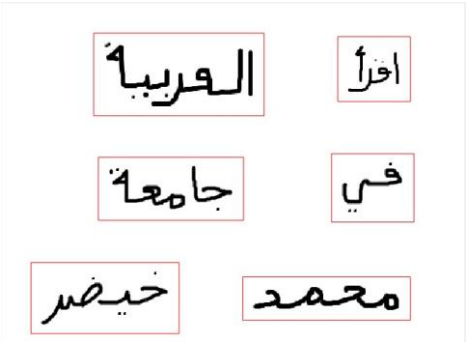


Figure III.8 – le resultat.

La figure au dessus montre le résultat aprés la superposision des boite englobante sur l'image initial.

Aprés cette étape importante , on a reussi a extraire le text d'une image ,on va entamer la procedure suivante qu'elle est plus importante,l'etape de segmentation, qui concerne a segmenter chaque mot caractère par caractère qu'on a reussit a extraire de la premiére étape (extraction de text)

III.3.2 Segmentation

La segmentation de texte est le processus de division du texte écrit en unités significatives, telles que des mots, des phrases ou des sujets ou bien des caractères . Le terme s'applique à la fois aux processus mentaux utilisés par les humains lors de la lecture de texte et aux processus artificiels mis en œuvre dans les ordinateurs comee notre cas aujourd'hui, qui font l'objet d' un traitement du langage naturel . Le problème n'est pas trivial, car si certaines langues écrites ont des bornes de mots explicites, telles que les espaces de mots de l'anglais écrit et les formes distinctives des lettres initiale, médiane et finale de l' arabe , ces signaux sont parfois ambigus et ne sont pas présents dans tous les écrits.

III.3.2.1 problèmes de segmentation des scripts arabes

De nombreux travaux de recherche ont été publiés dans le domaine de la segmentation et la reconnaissance de l'écriture arabe manuscrite Jusqu'à présent, ces efforts n'ont pas donné de

bons résultats car il n'y a pas assez de recherches sur ce sujet et il est plus difficile que celle de l'écriture latine pour les raisons suivantes :

• Les mots arabes se chevauchent et sont toujours écrits de manière cursive (plusieurs caractères peuvent être écrits en relation les uns avec les autres).

•L'arabe utilise de nombreux types d'objets externes, tels que les "points", "Hamza", "Madda" et les objets diacritiques.Les caractères arabes peuvent avoir plus d'une forme selon leur position : initiale, centrale, final, ou autonome comme le montre la figure III.9 .

Isoler	Final	Milieu	Début
ع	ـع	ـعـ	عـ

Figure III.9 – diffirenent écriture d'un caractere selon sa position

•L'arabe utilise de nombreuses ligatures, notamment dans les textes manuscrits. Les ligatures, illustrées dans la figure III.10 , sont des caractères qui occupent un espace horizontal commun, créant ainsi un chevauchement vertical des caractères reliés ou déconnecté.

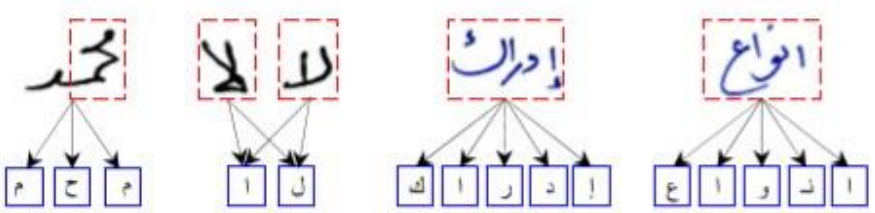


Figure III.10 – Ligature arabe

•Des écrivains différents et le même écrivain dans des conditions différentes écriront certains caractères arabes de manière complètement différente , comme le montre la figure III.11

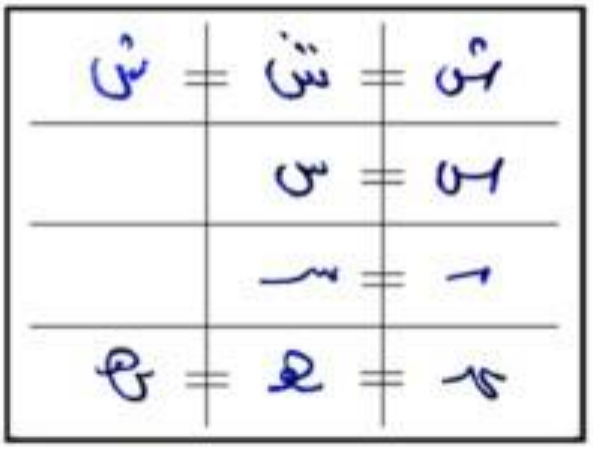


Figure III.11 – 4 caractères écrit différemment

•Les caractères comme Sin ou Shin complique également la segmentation et la reconnaissance lorsqu'elle se produit au milieu d'un mot comme le montre la figure III.12

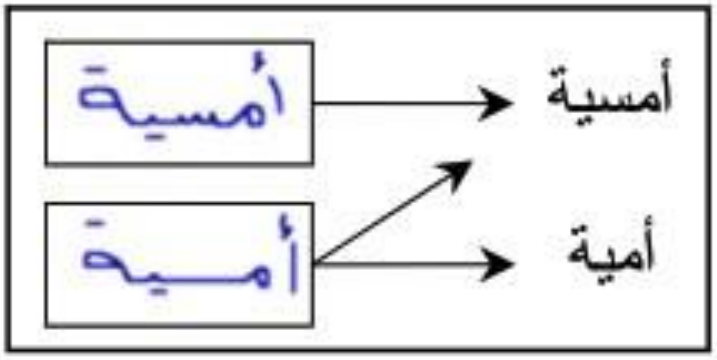


Figure III.12 – des lettres perdu quand ils sont au mileu du mot

Dans la suite on va essayer d'adopter une méthode de segmentation de texte manuscrit latin en modifiant son architecture et ses paramètres pour qu'elle puisse segmenter des texte manuscrit arabe.

III.3.2.1.1 Les techniques et méthodologies.

La segmentation est une étape cruciale dans le traitement des images pour la reconnaissance des caractères . À ce jour, il existe de nombreuses méthodes de segmentation, Dans notre travail

on va se focaliser sur c'est deux méthodes :

•Segmentation avec largeur moyenne des caractères.

•Segmentation basé sur la projection verticale et horizontal d'images.

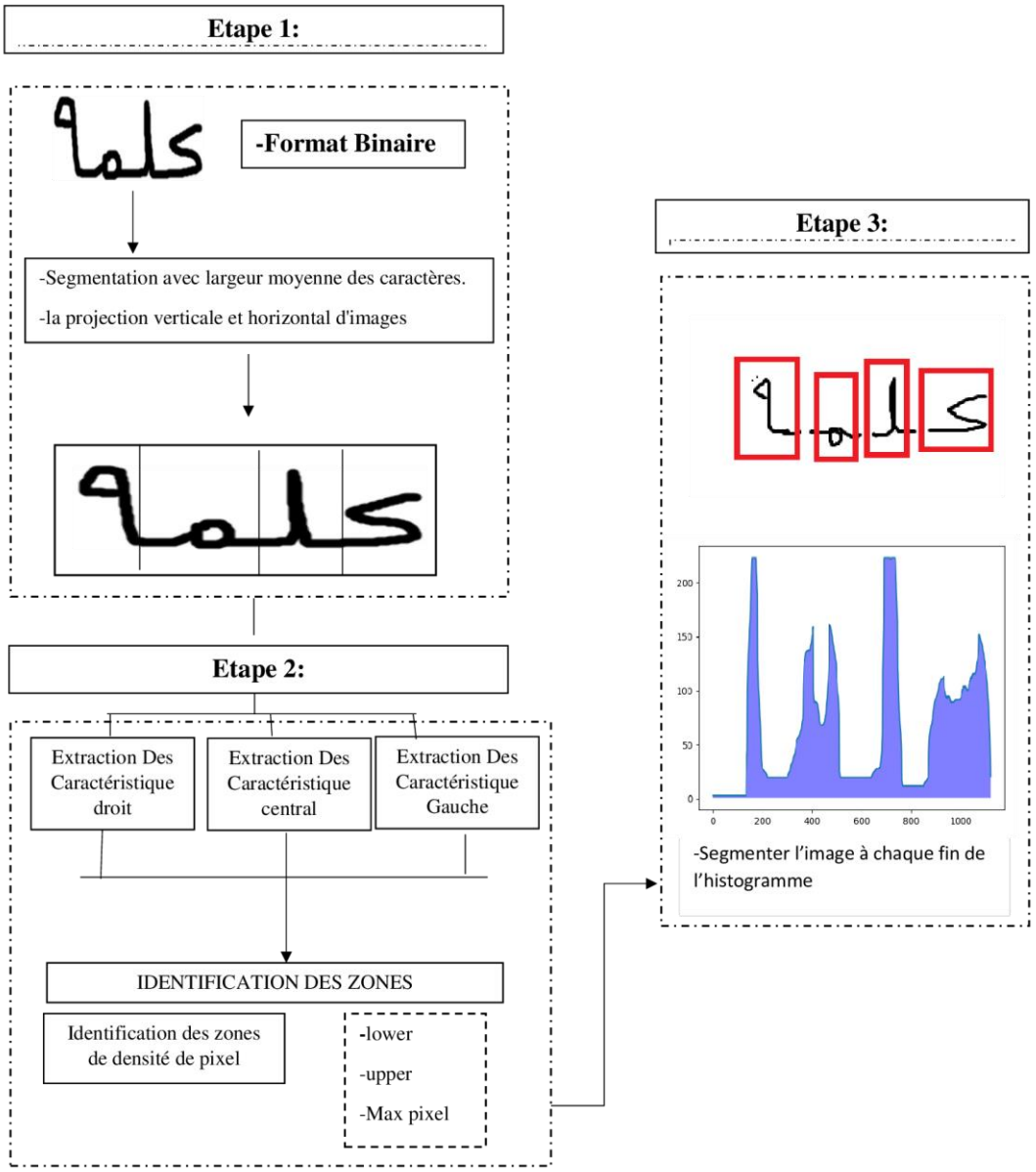


Figure III.13 – Composants de notre technique complète de segmentation de l'écriture.

Notre méthode se resume en general dans c'est étapes :

•conversion de l'image RGB en niveaux de gris en éliminant les informations de teinte et de

saturation tout en conservant l'illumination, puis la conversion de l'image en niveaux de gris en binaire format (de la matrice). L'image binaire de sortie a des valeurs de 0 comme pixel de premier plan (noir) pour tous les pixels de l'image d'entrée et 1 comme pixel d'arrière-plan (blanc) pour tous les autres pixels.

• La suppression du bruit : Le but de cette technique est de supprimer le bruit ainsi que les petits objets de premier plan qui ne faisaient pas partie de l'écriture. Une fois que le composant de l'image comme matrice ont été identifié, il était possible d'effectuer diverses opérations utiles.

• Suppression des signes de ponctuation (points) : La technique a été concue pour travailler sur des mots qui contient des signes de ponctuation. un probleme soulevée par cette procédure de traitement d'image est la suppression des points, comme montré plus tard, lors de la localisation du point de segmentation prospective en utilisant la technique de l'histogramme vertical , les signes de ponctuation provoqueront des point de segment incorrect car la plupart des signes de ponctuation sont placé sur un autre caractère ou au-dessus ou sous la ligature. En général, la forme de densité de n'importe quel point est plus petite que n'importe quel autre caractère dans le Scripts arabes ; par conséquent, nous pouvons le marquer et ensuite le supprimer, Lorsque les coordonnées x et y de chaque composant connecté (points) dans la matrice est identifiés.

• Projection de l'histogramme vertical : Aprés avoir détécter la region du milieu et la largeur du mot comme expliqué précédemment , la prochaine étape consiste à localiser les points de segmentation potentiels ,l'approche utilise l'analyse de l'histogramme vertical (densité). L'analyse est basée sur la distribution verticale du premier plan (Pixel). L'histogramme est dessiné en comptant le nombre total de pixels de premier plan (pixels noirs) dans chaque colonne du mot (matrice), l'histogramme est examiné pour les zones qui ont une densité de pixels de zéro. Ces zones sont marquées comme des points de segmentation définis (espace entre les caractères). Les zones à faible densité de pixels sont alors identifiés comme des points de segmentation.

la figure III.14 resume tout les étapes de notre segmentation en detail.

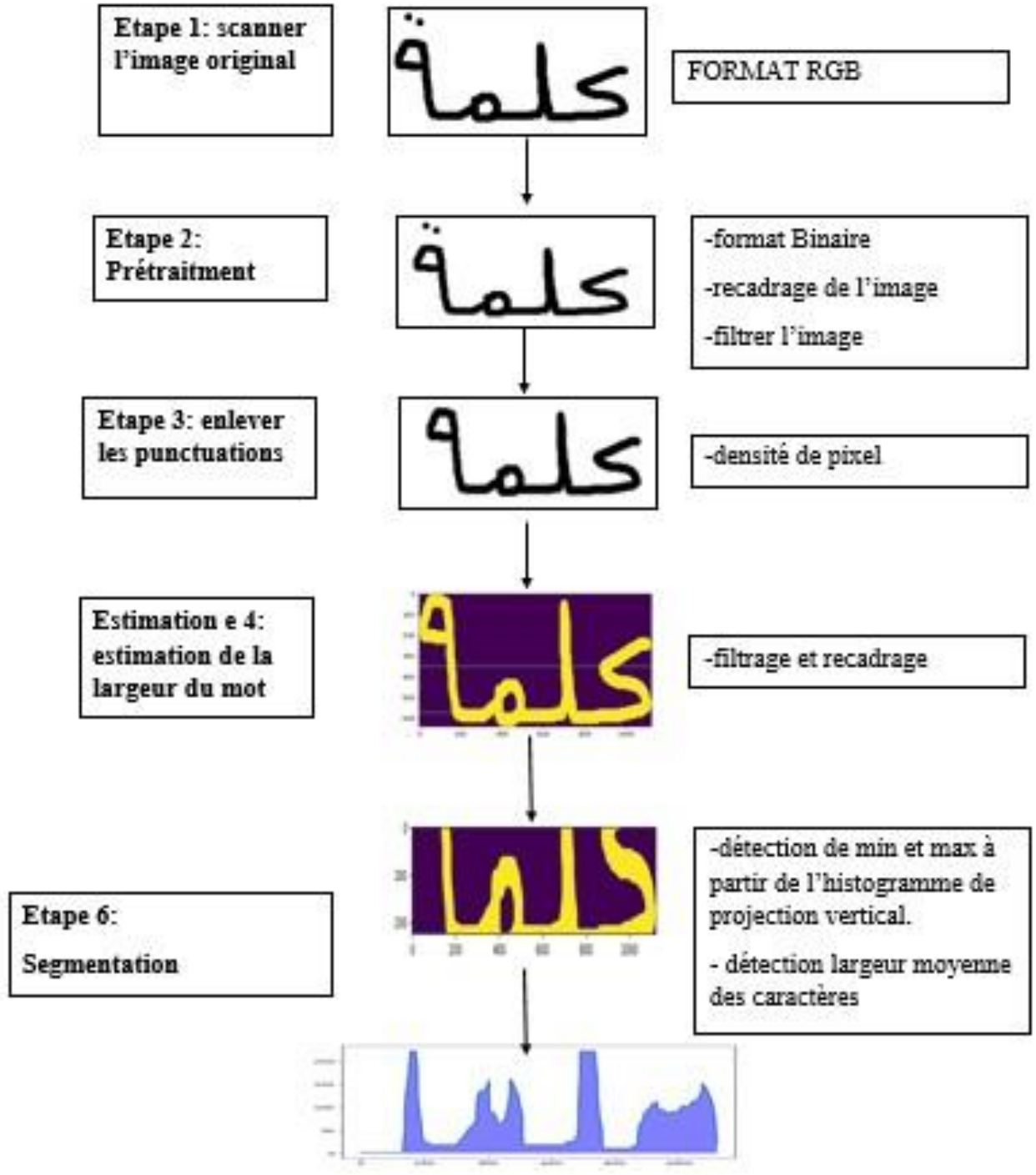


Figure III.14 – Apercu de notre technique de segmentation

A l'issue de l'étape de segmentation, l'image originelle est décomposée en une multitude d'imagettes correspondant aux caractères, prête pour la phase suivante qui consiste a la reconnaissance de c'est caractères.

III.3.3 Reconnaissance de caractères

Le processus de reconnaissance de caractères est le plus complexe. Généralement, il comporte deux phases : l'extraction de caractéristiques et la classification. L'extraction de caractéristiques permet de déterminer un vecteur dont les composantes caractérisent chaque type de caractère. La classification va permettre de déterminer la classe d'appartenance du caractère à l'aide de ce vecteur de caractéristiques. Pour cette approche on va utiliser les réseaux de neurones convolutionnels (CNN).

III.3.3.1 Système de reconnaissance à base de réseau neuronal convolutif

III.3.3.2 Architecture d'un CNN

En apprentissage automatique, un réseau de neurones convolutifs ou réseau de neurones à convolution (en anglais CNN ou ConvNet pour Convolutional Neural Networks) est un type de réseau de neurones artificiels acycliques (feed-forward), dans lequel le motif de connexion entre les neurones est inspiré par le cortex visuel des humains. Les neurones de cette région du cerveau sont arrangés de sorte qu'ils correspondent à des régions qui se chevauchent lors du pavage du champ visuel1. Leur fonctionnement est inspiré par les processus biologiques, ils consistent en un empilage multicouche de perceptrons, dont le but est de prétraiter de petites quantités d'informations. Les réseaux neuronaux convolutifs ont de larges applications dans la reconnaissance d'image et vidéo, les systèmes de recommandation4 et le traitement du langage naturel.

La partie qu'elle a fait parler de lui c'est bien la premiere partie de l'extraction des caracteristiques automatique,une image est passée à travers une succession de filtres, ou noyaux de convolution, créant de nouvelles images appelées cartes de convolutions. Certains filtres intermédiaires réduisent la résolution de l'image par une opération de maximum local ou bien moyenne local. Au final, les cartes de convolutions sont mises à plat et concaténées en un vecteur de caractéristiques, appelé code CNN.

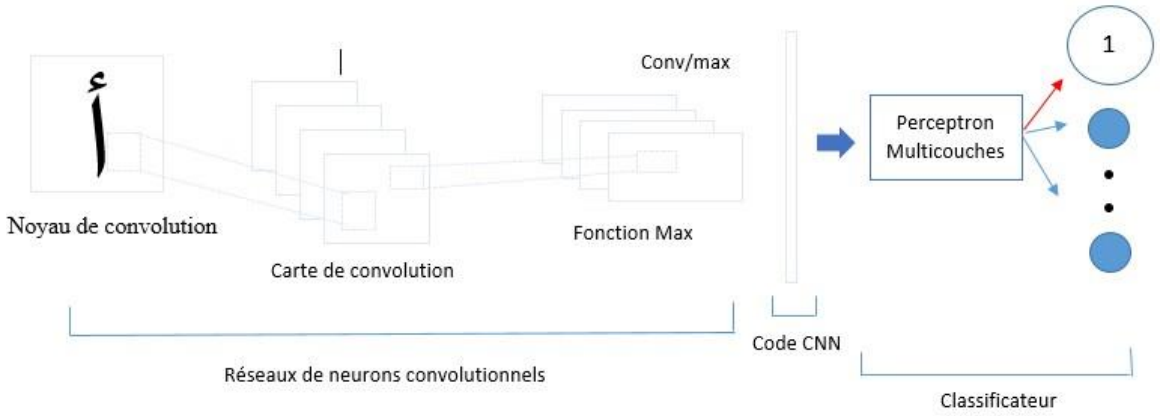


Figure III.15 – Architecture standard d'un réseau de neurones convolutionnel..

on général un réseau de neurone convolutionnel se compose de c'est couches essenteille :

— couche Convolutions (Conv).

— Couche de mise en commun (Pooling).

— La couche de correction ReLU.

— couche entiérement connectées (fully connected)

III.3.3.3 Couche Convolutions (Conv)

La couche de convolution est la composante clé des réseaux de neurones convolutifs, et constitue toujours au moins leur première couche.

Son but est de repérer la présence d'un ensemble de features dans les images reçues en entrée. Pour cela, on réalise un filtrage par convolution : le principe est de faire "glisser" une fenêtre représentant la feature sur l'image, et de calculer le produit de convolution entre la feature et chaque portion de l'image balayée. Une feature est alors vue comme un filtre : les deux termes sont équivalents dans ce contexte.

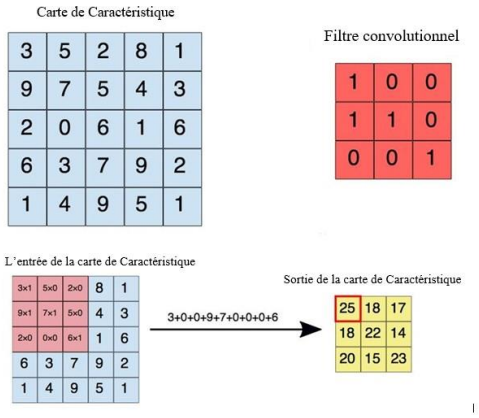


Figure III.16 – La couche convolutionnelle.

La couche de convolution reçoit donc en entrée plusieurs images, et calcule la convolution de chacune d'entre elles avec chaque filtre. Les filtres correspondent exactement aux features que l'on souhaite retrouver dans les images.

On obtient pour chaque paire (image, filtre) une carte d'activation, ou feature map, qui nous indique où se situent les features dans l'image : plus la valeur est élevée, plus l'endroit correspondant dans l'image ressemble à la feature.

III.3.3.4 Couche de mise en commun (Pooling).

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

L'opération de pooling consiste à réduire la taille des images, tout en préservant leurs caractéristiques importantes.

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

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

	Max pooling	Average pooling
But	Chaque opération de pooling sélectionne la valeur maximale de la surface.	Chaque opération de pooling sélectionne la valeur moyenne de la surface.
Commentaires	- Garde les caractéristiques détectées. - Plus communément utilisé.	- Sous-échantillonne la feature map. - Utilisé dans LeNet.

Figure III.17 – les deux types de la couche mise en commun.

III.3.3.5 La couche de correction ReLU.

ReLU (Rectified Linear Units) désigne la fonction réelle non-linéaire définie par :

$$\mathrm{ReLU}(x) = \max(0, x).$$

La couche de correction ReLU remplace donc toutes les valeurs négatives recues en entrées par des zéros. Elle joue le role de fonction d'activation.

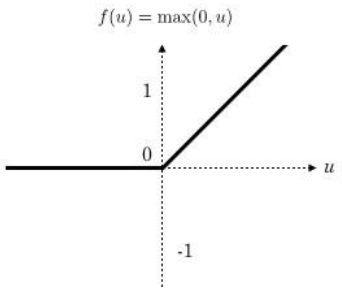


Figure III.18 – Allure de la fonction ReLU.

III.3.3.6 La couche entiérement connectées (fully connected).

La couche fully-connected constitue toujours la dernière couche d'un réseau de neurones.

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

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

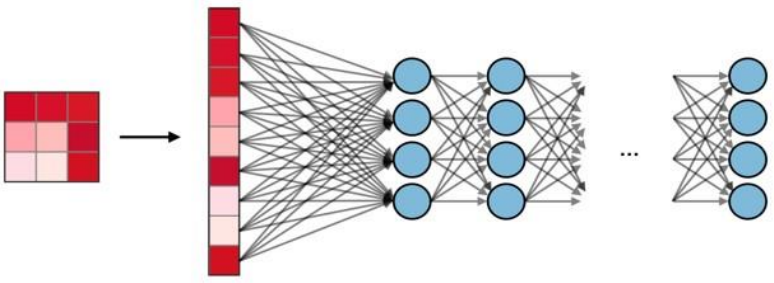


Figure III.19 – La couche entièrement connecté.

•Softmax

En mathématiques, la fonction softmax, ou fonction exponentielle normalisée, est une généralisation de la fonction logistique qui prend en entrée un vecteur Z = (z1,....,zk) de K nombres réels et qui en sort un vecteur σ(z) de K nombres réels strictement positifs et de somme 1.
La fonction est définie par :

$$\sigma(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \text{ pour tout } j \in \{1, \dots, K\}.$$

Figure III.20 – équation mathématique de la fonction softmax.

C'est-a-dire que la composante j du vecteur σ(z) est égale à l'exponentielle de la composante j du vecteur z divisée par la somme des exponentielles de toutes les composantes de z.
En théorie des probabilités, la sortie de la fonction softmax peut être utilisée pour représenter une loi catégorielle – c'est-à-dire une loi de probabilité sur K différents résultats possibles.La fonction softmax est également connue pour être utilisée dans diverses méthodes de classification en classes multiples, par exemple dans notre cas de réseau neuronal convolutif.

III.3.3.7 Pourquoi CNN ?

Si les réseaux neuronaux et autres méthodes de détection de motifs existent depuis 50 ans, il y a Dans un passé récent, les réseaux neuronaux convolutifs ont connu un développement important. Cette section couvre les avantages de l'utilisation de CNN pour la reconnaissance d'images.

— Robustesse face aux décalages et à la distorsion de l'image

La détection par CNN est résistante aux distorsions telles que le changement de forme dû à l'objectif de la caméra, les différentes conditions d'éclairage, les différentes poses, la présence d'occlusions partielles, les décalages horizontaux et verticaux.

— Moins d'exigences en matière de mémoire

Dans notre cas nous utilisons des couches entièrement connectée pour extraire les caractéristiques, l'image d'entrée de de taille 32x32 et une couche cachée de plus de 1000 caractéristiques nécessiteront des commandes d'une centaine de coefficients, et une mémoire énorme . Dans

la couche convolutionnelle, les mêmes coefficients sont utilisés à différents endroits, de sorte que le besoin de mémoire est réduit de façon drastique.

— un apprentissage plus facile et de meilleure qualité.(auto-extraction de caractéristiques)

En utilisant à nouveau le réseau neuronal standard qui serait équivalent à un CNN, car le nombre de paramètres serait beaucoup plus élevé, le temps de l'apprentissage augmenterait également proportionnellement. Dans un CNN, puisque le nombre de paramètres est réduit de façon drastique, le temps de l'apprentissage est réduit proportionnellement. De plus, on peut aussi concevoir un réseau neuronal standard dont les performances seraient identiques à celles d'un CNN. Mais en formation pratique,un réseau neuronal standard équivalent à CNN aurait plus de paramètres, ce qui entraînerait plus de bruit.

III.3.3.8 Base de donnée utilisé

Dans mon projet j'ai esseyaer plusieurs datasets et j'ai comparé les resultats surtout dans la phase de reconnaissance de caractere séparer et j'ai fini par garder et continuer le travail avec les datasets de AHCD, AHDD (pour la reconnaissance de caractere et les chiffres) et iesk (pour la reconnaissance des mots (limité) special mentionné dans la base de donnée iesk. .

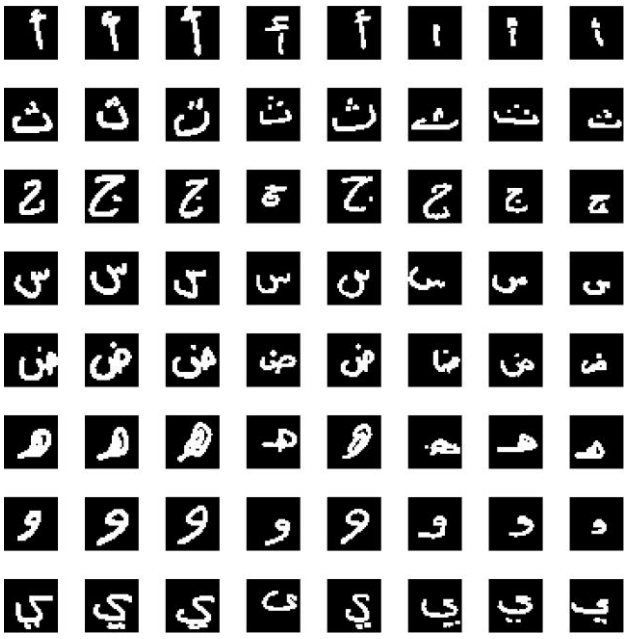


Figure III.21 – Visualisation des caractères arabe de AHCD

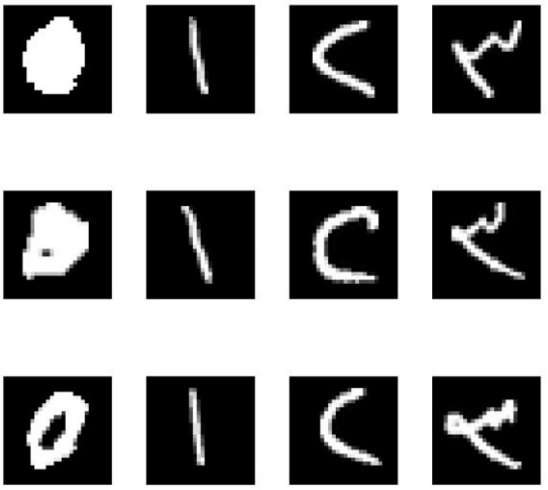


Figure III.22 – Visualisation des numéros arabe de AHDDl

L'ensemble de données est composé de 16 800 caractères écrits par 60 participants, la tranche d'age est comprise entre 19 et 40 ans, et 90 pour cent des participants sont droitier . Chaque

participant a écrit chaque caractère (de 'alef' à 'yeh' , 1 a 9) dix fois sur deux formes comme le montre les fig III.21 et III.22. Les formulaires ont été scannés à la résolution de 300 dpi. Chaque bloc est segmenté automatiquement pour déterminer les coordonnées de chaque bloc. La base de données est divisée en deux ensembles : un ensemble d'apprentissage (13 440 caractères à 480 images par classe) et un ensemble de test (3 360 caractères à 120 images par classe).

•IESK est une base de données manuscrite hors ligne. Il contient 285 pages de manuscrits historiques du 14ème siècle, plus de 6000 images de mots manuscrits et 8000 images de caractères segmentés. Le vocabulaire de la base de données de mots couvre la plupart de la partie arabe des noms de discours, des verbes, des noms de pays / villes, des termes de sécurité et des mots utilisés pour écrire des montants bancaires.

Mais le plus grand problème des base de donnée des mots arabes c'est que ils sont tous privés et conserver par les droit de l'auteur et pas disponible on ligne ,iesk offre une version beta un ficher de 300 mots, un seul example pour chaque mot.

III.3.3.9 Pré-traitement de la Base de donnée utilisé

Prétraitement des données est une technique qui contribue à améliorer la qualité des données pour promouvoir l'extraction des caractéristiques significatives à partir des données. De manière simple, le prétraitement des données dans L'apprentissage automatique est une technique d'exploration de données qui transforme les données brutes en un format compréhensible et lisible.

Pourquoi avons-nous besoin de prétraitement des données ?

Lorsqu'il s'agit de créer un modèle D'apprentissage automatique, le prétraitement des données est la première étape marquant le début du processus. Généralement, les données du monde réel sont incomplètes, incohérentes, inexactes (contiennent des erreurs ou des valeurs aberrantes) et manquent souvent de valeurs/tendances d'attribut spécifiques. C'est là que le prétraitement des données entre dans le scénario ,il aide à nettoyer, formater et organiser les données brutes, les rendant ainsi prêtes à l'emploi pour les modèles D'apprentissage automatique.

Il y a un pipeline à suivre dans le prétraitement des données comme le montre la figure III.23

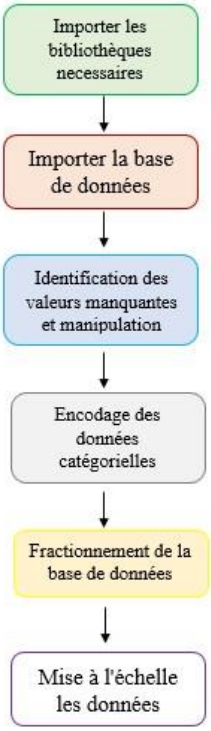


Figure III.23 – Pipeline de pré-traitement.

•Importer les bibliothèques necessaires

C'est la première étape à franchir dans le prétraitement des données. Les bibliothèques sont particulièrement utiles pour stocker les routines fréquemment utilisées.

•Importer la base de données

Nous pouvons importer la base de donnée savec plusieurs méthodes, il dépend du format des fichiers de données tels que (data.py, données.csv ...) chaque format de jeu de données peut être chargé avec une bibliothèque spécifique. Et il est utile si nous mettons notre ensemble de données dans le même répertoire de travail pour rendre l'accès plus rapide, pour notre cas on a un ensemble de données charger dans des fichiers CSV et des images .PNG

•Identification des valeurs manquantes et manipulation

Dans le prétraitement des données, il est essentiel d'identifier et de gérer correctement les valeurs manquantes, si nous manquons cette étape, nous pouvons nous retrouver avec un modèle inexacts.

•Encodage des données catégorielles

Les données catégoriques font référence aux informations qui ont des catégories spécifiques dans l'ensemble de données. les modèles d'apprentissage automatique sont basés sur des équations mathématiques et vous pouvez intuitivement comprendre que cela poserait un problème si nous

pouvions garder les données catégoriques dans les équations parce que nous ne voudrions que des nombres dans les équations. pour cele on va utiliser la méthode ”One-hot encoding”

Label	Entier	Encoding
أ	1	[1,0,0,0]
ب	2	[0,1,0,0]
ت	3	[0,0,1,0]

Figure III.24 – Encodage des données catégorielles

•Fractionnement de la base de données

Chaque jeu de données pour le modèle d'apprentissage automatique doit être divisé en un ensemble d'apprentissage / test. L'ensemble d'entraînement est utilisé pour l'apprentissage de notre modèle. Ici, le modèle sait la sortie. Un jeu de test est utilisé pour tester le modèle pour vérifier la précision du modèle. Le modèle ML utilise l'ensemble de tests pour prédire les résultats. Habituellement, nous prenons 70% ou 80% des données pour l'apprentissage du modèle tout en laissant de coté le reste 30% ou 20%.

Pour notre travail, nous utilisons un fractionnement de 80 : 20% .

•Mise a l'echelle les données

La mise à l'échelle des données marque la fin du prétraitement des données dans L'apprentissage automatique. C'est une méthode pour standardiser les variables indépendantes d'un ensemble de données dans une plage spécifique. En d'autres termes, la mise à l'échelle des entités limite la plage de variables afin que vous puissiez les comparer sur des bases communes. Il existe des méthodes pour effectuer l'étape de mise à l'échelle des entités en fonction des données telles que la normalisation, et la normalisation moyenne. Nous travaillons avec des images donc nous avons utilisé la méthode de normalisation qui est également appelé re-scaling qui s'incarne dans la formule suivante :

$$X^{0} = \frac{X - Min(X)}{Max(X) - Min(X)}$$

III.3.3.10 architecture de notre modèle CNN

Apres avoir préparer nos données ,et après plusieus manupilation dans la création de notre modèle pour avoir les meilleurs resulats, j'ai pu constater que l'architecture que je vais expliquer

et détailler en bas est la plus optimale .

On a développé 3 modèles pour les 3 types de classifcations.

— Un modèle pour la classification des lettres.

— Un modèle pour la classification des chiffres.

— Un modèle pour la classification des mots.

•Pour la la classification des lettres nous avons mis en place une architecture composée de 13 couches :

— 4 convolutions.

— 1 global average pooling.

— 2 dropout.

— 4 batch normalization.

— 2 couches entièrement connectées.

nous avons également utilisé l'optimiseur Adams pour calculer les valeurs de perte et la precision durant l'apprentissage.

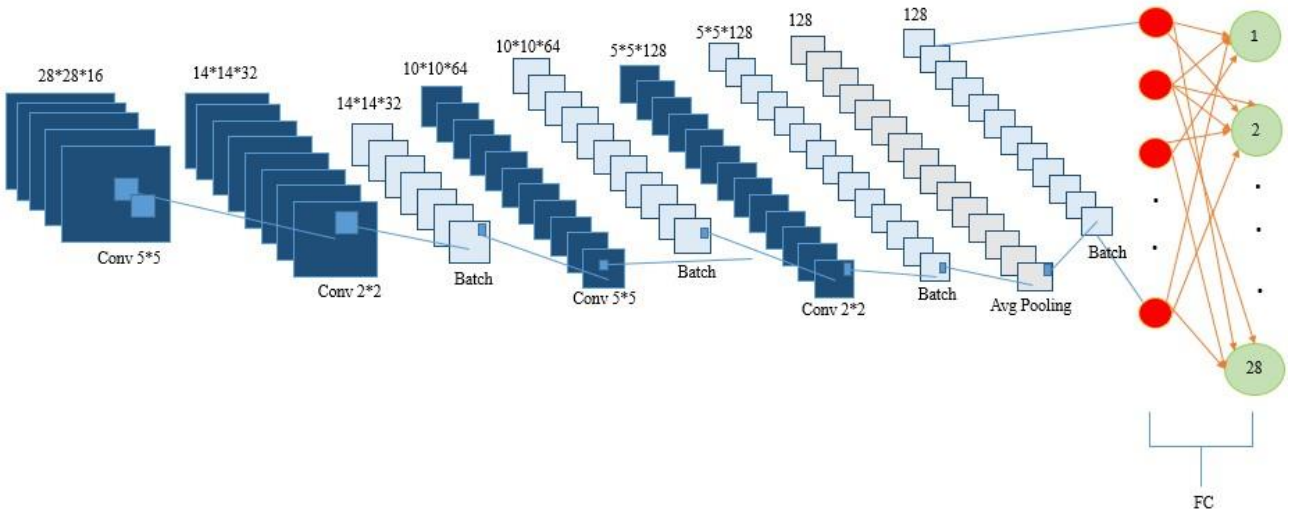


Figure III.25 – L'architecture de modèle CNN utilisé dans notre système pour la reconnaissance de caractères arabe

•Pour la la classification des chiffres nous avons mis en place une architecture composée de 13 couches :

— 4 convolutions.

— 1 global average pooling.

— 2 dropout.

— 4 batch normalization.

— 2 couches entièrement connectées.

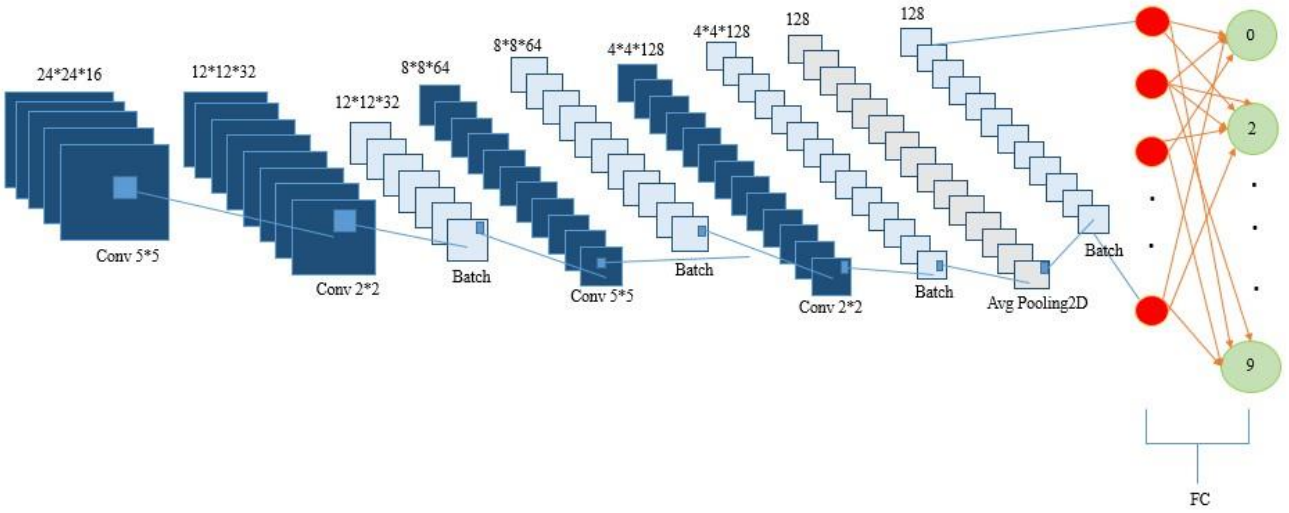
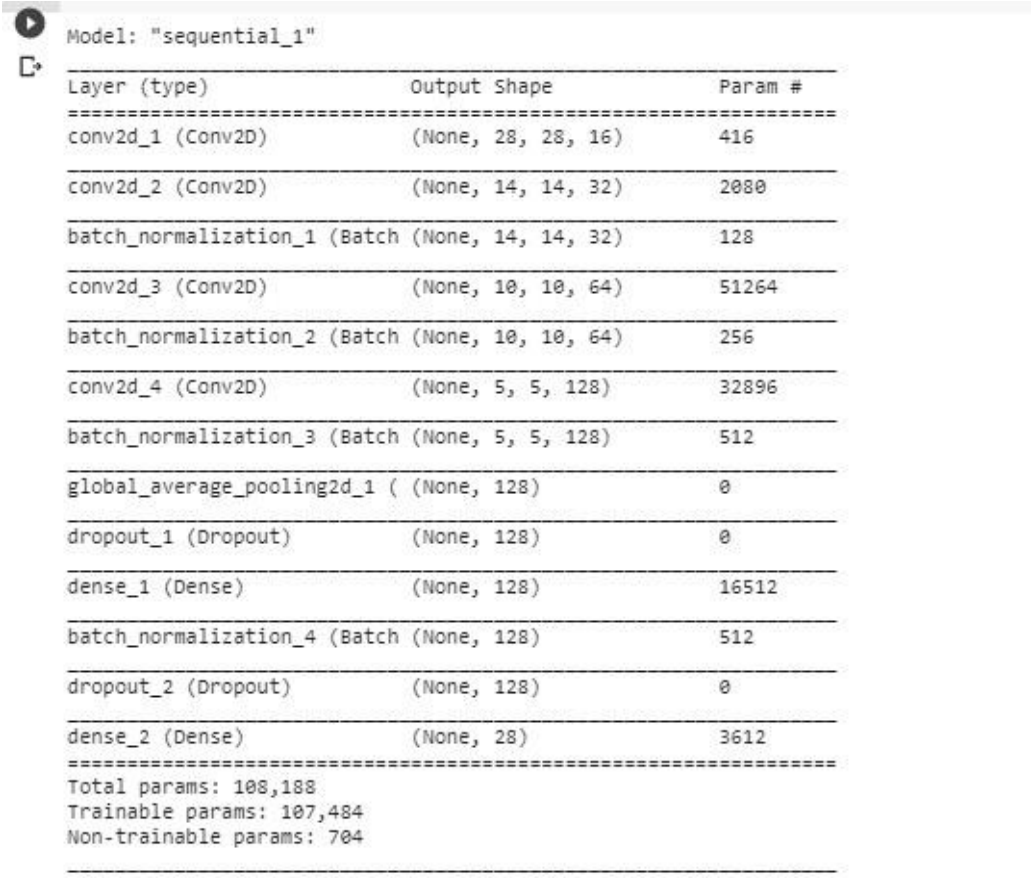


Figure III.26 – architecture de modèle CNN utilisé dans notre système pour la reconnaissance de numéros arabe

Nous avons effectué plusieurs tests afin de retenir les meilleures architecture de CNN conduisant aux meilleurs taux de classification. Nous avons varié le nombre de couches, le nombre et la taille des filtres de convolution. Nous avons retenu les modèle des figure III.25 et III.26. Les modèles sont ainsi développés est composé de 4 couches de convolution et une couche de Avrege-Pooling et 4 couches de batch normalization et deux FC, pour les deux modèles concernat les caractéres et les nombres arabes.



```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 28, 28, 16)        416
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 14, 14, 32)        2080
_________________________________________________________________
batch_normalization_1 (Batch (None, 14, 14, 32)        128
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 10, 10, 64)        51264
_________________________________________________________________
batch_normalization_2 (Batch (None, 10, 10, 64)        256
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 5, 5, 128)         32896
_________________________________________________________________
batch_normalization_3 (Batch (None, 5, 5, 128)         512
_________________________________________________________________
global_average_pooling2d_1 ( (None, 128)               0
_________________________________________________________________
dropout_1 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_1 (Dense)              (None, 128)               16512
_________________________________________________________________
batch_normalization_4 (Batch (None, 128)               512
_________________________________________________________________
dropout_2 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_2 (Dense)              (None, 28)                3612
=================================================================
Total params: 108,188
Trainable params: 107,484
Non-trainable params: 704
_________________________________________________________________
```

Figure III.27 – Architecture détaillé du CNN pour les caractéres arabes

L'image en entrée est de taille $32 * 32$, l'image passe d'abord à la première couche de convolution. Cette couche est composée de 16 filtres de taille $5 * 5$, la fonction d'activation ReLU est utilisée pour forcer les neurones à retourner des valeurs positives.

Cette convolution produit 16 features maps de taille 28x28 chacune. Ensuite, les 16 feature maps sont présentées en entrée à la deuxième couche de convolution qui est composée aussi de 32 filtres de taille $2 * 2$.

La fonction d'activation ReLU est appliquée sur les couches de convolutions.apres deux couche de convolutions (64et 128) seron appliqués de suite avec une dimension de $5 * 5$ et $2 * 2$ successivement, avant que la couche batch normalization sera appliqué qui sert a rendre les réseaux de neurones artificiels plus rapides et plus stables grâce à la normalisation de la couche d'entrée par recentrage et redimensionnement avec un axe de 3 et 1 .

finalment deux couche FC sont appliqués dotsé d'une fonction softamx pour la probabilté des veteus de classe plus sur.

III.4 Langage et outils utilisés dans l'implémentation

Pour mettre en œuvre cette application, les matériaux ayant les caractéristiques suivantes ont été utilisés :

— Processeur : Intel(R) CoreTM i5-7200U CPU @2.50 GHZ

— RAM : 12.00 GB

— SE : Windows 10

III.4.1 Paquets requis et bibliothèques

Pour développer mon application, j'ai utilisé un environnements de programmation , et des outils et des librairie différentes pour la programmation en arrière-plan (back-end), exécutable sur plusieur IDE (environment)

Python

Figure III.28 – Python logo

Python est un langage de programmation interprété, multi-paradigme et multiplateformes. Il favorise la programmation impérative structurée, fonctionnelle et orientée objet. C'est est un langage qui peut s'utiliser dans de nombreux contextes et s'adapter à tout type d'utilisation grace à des bibliothèques spécialisées. Il est cependant particulièrement utilisé comme langage de script pour automatiser des taches simples mais fastidieuses.

Tensorflow

Figure III.29 – Tenserflow Logo

TensorFlow est un outil open source d'apprentissage automatique développé par Google. Le code source a été ouvert le 9 novembre 2015 par Google et publié sous licence Apache. initiée par Google en 2011, et est doté d'une interface pour Python et Julia. TensorFlow est l'un des outils les plus utilisés en IA dans le domaine de l'apprentissage machine.

Keras

Figure III.30 – Keras Logo

La bibliothèque Keras permet d'interagir avec les algorithmes de réseaux de neurones profonds et d'apprentissage automatique, notamment Tensorflow,Concue pour permettre une expérimentation rapide avec les réseaux de neurones profonds, elle se concentre sur son ergonomie, sa modularité et ses capacites d'extension. Elle a été développée dans le cadre du projet ONEIROS (Open-ended Neuro-Electronic Intelligent Robot Operating System). Elle a été initialement écrite par Francois Chollet.

Numpy

Figure III.31 – Numpy Logo

NumPy est une extension du langage de programmation Python, destinée à manipuler des matrices ou tableaux multidimensionnels ainsi que des fonctions mathématiques opérant sur ces tableaux.

Matplotlib

Figure III.32 – matplotlib logo

Matplotlib est une bibliothèque du langage de programmation Python destinée à tracer et visualiser des données sous formes de graphiques. Elle peut être combinée avec les bibliothèques python de calcul scientifique NumPy et SciPy.
Plusieurs points rendent cette bibliothèque intéressante :

— Export possible en de nombreux formats matriciels (PNG, JPEG...) et vectoriels (PDF, SVG...).
— Documentation en ligne en quantité, nombreux exemples disponibles sur internet.

— Forte communauté très active.

Pandas

Figure III.33 – Pandas Logo

Pandas est une bibliothèque écrite pour le langage de programmation Python permettant la manipulation et l'analyse des données. Elle propose en particulier des structures de données et des opérations de manipulation de tableaux numériques et de séries temporelles. Pandas est un logiciel libre sous licence BSD.

•pour ce qui concerne l'environnement de developpement j'ai utilsé L'IDE :

Pycharme

Figure III.34 – Pycharme Logo

PyCharm est un environnement de développement intégré utilisé pour programmer en Python. Il permet l'analyse de code et contient un débogueur graphique. Il permet également

la gestion des tests unitaires, l'intégration de logiciel de gestion de versions, et supporte le développement web avec Django. Développé par l'entreprise tchèque JetBrains, c'est un logiciel multi-plateforme qui fonctionne sous Windows, Mac OS X et Linux. Il est décliné en édition professionnelle, diffusé sous licence propriétaire, et en édition communautaire diffusé sous li cence Apache..

Google colaboratory

Figure III.35 – Google colab Logo

Google colaboratory est une application web utilisée pour programmer dans plusieurs languages de programmation, dont Python,il m'a permis aussi d'utilser les ressources disposés par google dont NVIDIA Tesla K80 GPU pour la compilation et l'excusion du code en si peu de temps.

III.5 Résultat obtenu pour le modèle de CNN

Dans ce que suit nous allons presenter les résultats obtenu pour la reconnaissance des lettres et chiffres arabe en utilisant un système de reconnaissance a base CNN.

•présentation graphique de précision du modèle CNN pour les lettres arabes

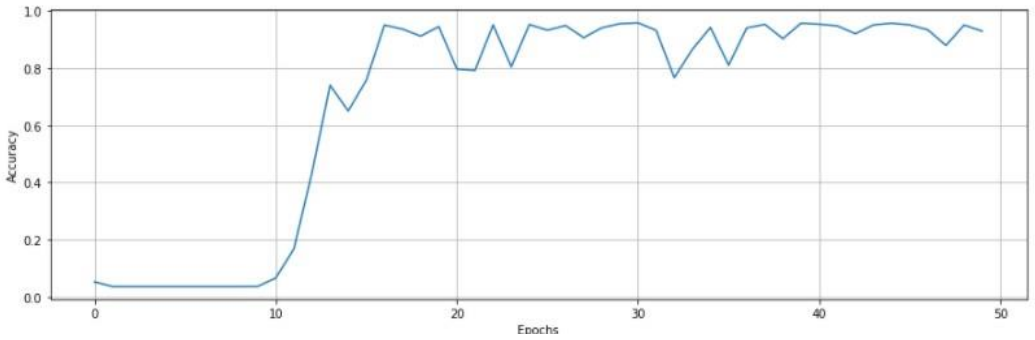


Figure III.36 – présentation graphique de précision du modèle CNN pour les lettres arabes

•présentation graphique de l'erreur du modèle CNN pour les lettres arabes

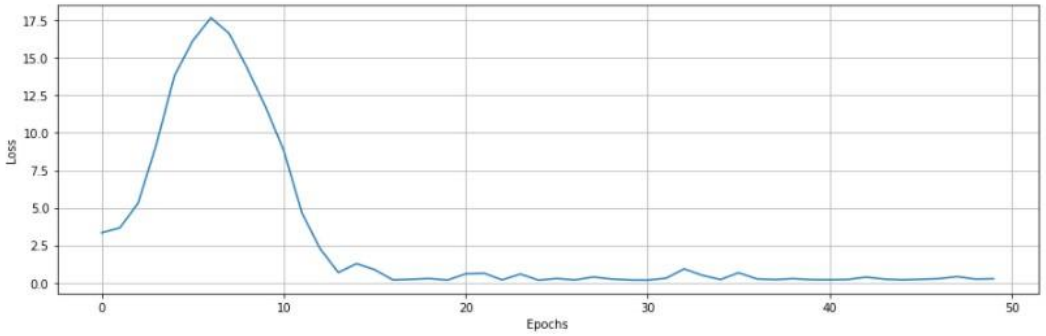


Figure III.37 – présentation graphique de l'erreur du modèle CNN pour les lettres arabes

•présentation graphique de précision du modèle CNN pour les numéros arabes

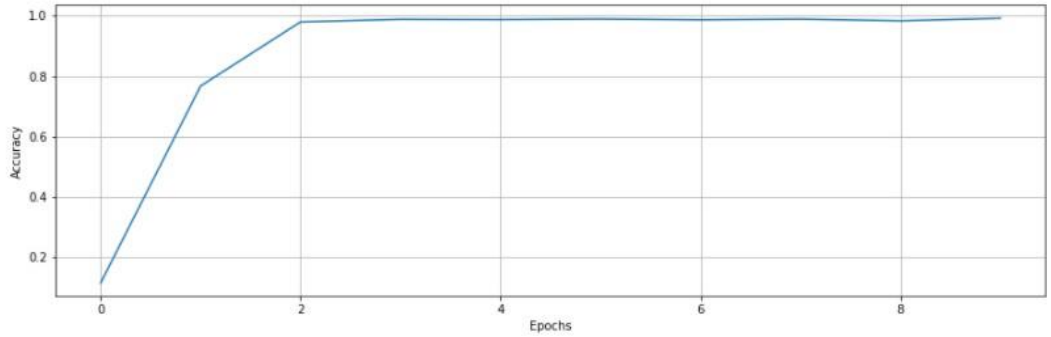


Figure III.38 – présentation graphique de précision du modèle CNN pour les numéros arabes

•présentation graphique de l'erreur du modèle CNN pour les numéros arabes

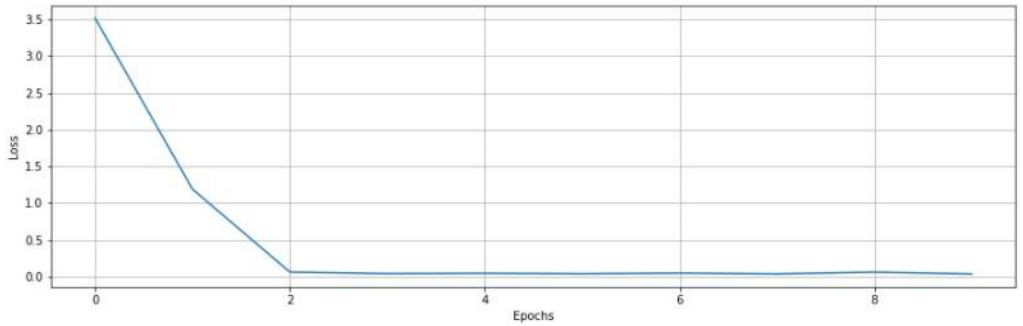


Figure III.39 – présentation graphique de l'erreur du modèle CNN pour les numéros arabes

•Après l'analyse des résultats obtenus, Nous avons constaté les remarques suivantes :

— D'après la Figure III.40 La précision de l'apprentissage et de test augmente avec le nombre d'époque, ceci reflète qu' chaque époque le modèle apprenne plus d'informations.

— Si la précision est diminuée alors on aura besoin de plus d'information pour faire apprendre notre modèle et par conséquent on doit augmenter le nombre d'époque.

— De même, l'erreur d'apprentissage et de la validation diminue avec le nombre d'époque.

	epoch	precision	errors
Caractères arabes	50	94%	20%
Numéros arabes	10	98%	3%

Figure III.40 – Tableaux de comparaison

III.5.1 Utilisation du modèle aprés la phase d'appretnissage

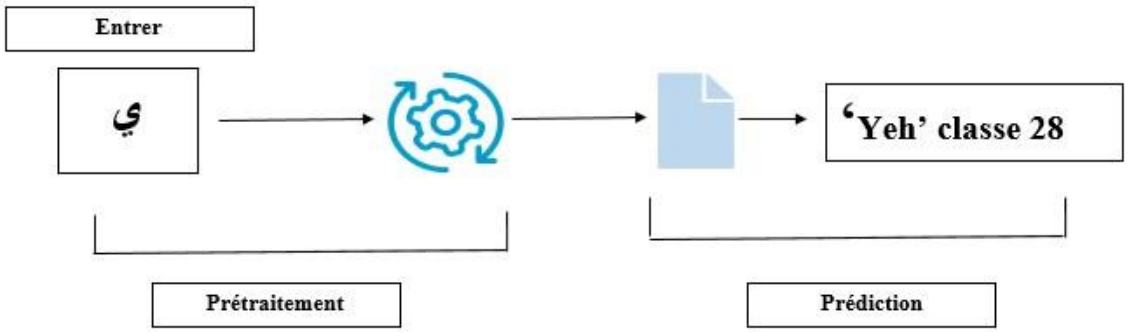


Figure III.41 – Exemple comment tester notre modèle

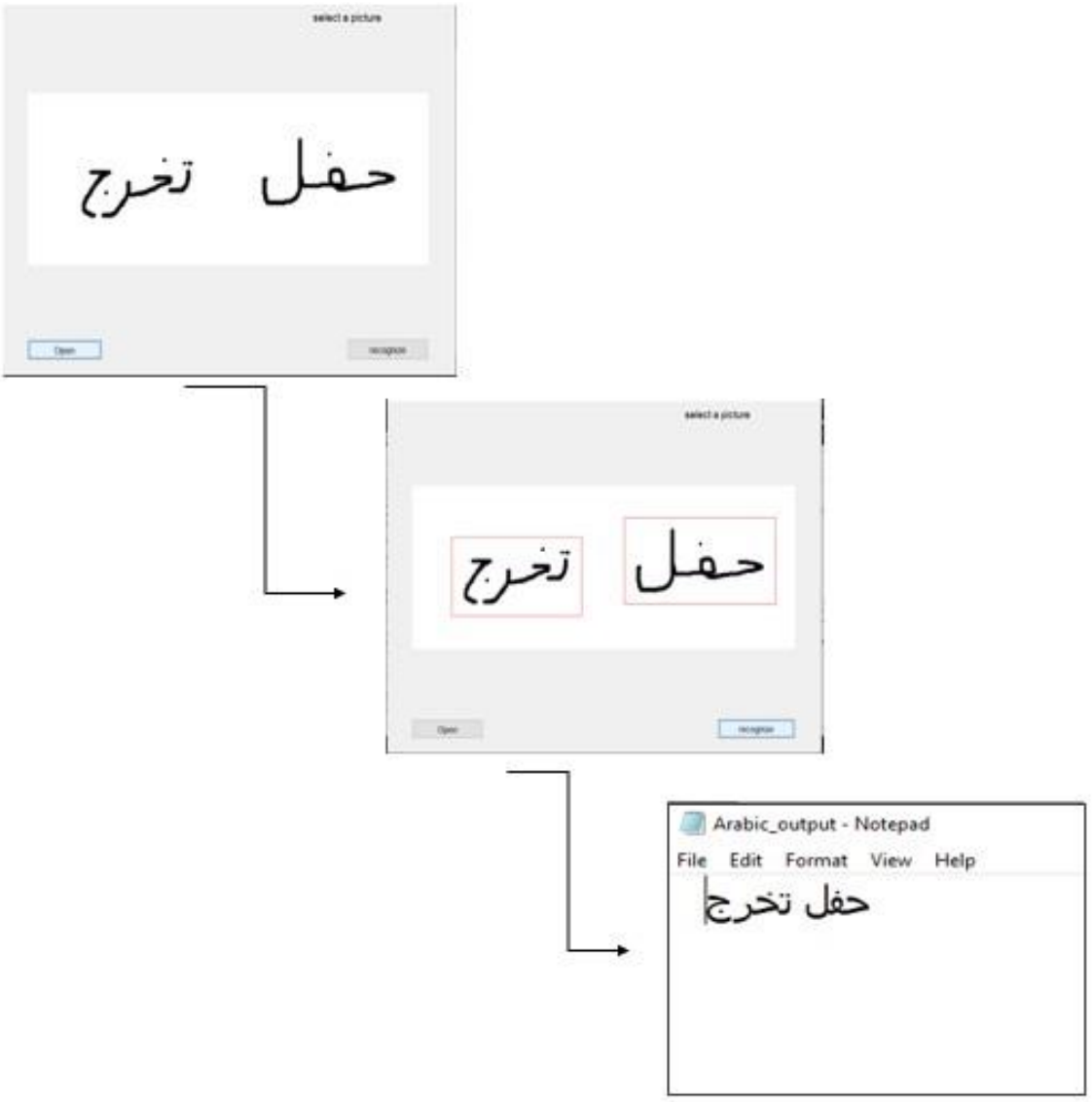


Figure III.42 – Exemple sur un texte

III.5.1.1 Les limites de réseau de neuronal convolutif

Malgré que les CNN (réseaux de neurones convolutifs) sont géniaux, ils ont leur limites et ils présentent des inconvénients fondamentaux. Prenons un exemple très simple et non technique. Imaginez un visage. Quels sont les composants d'un visage ? Nous avons le visage ovale, deux yeux, un nez et une bouche. Pour un CNN, la simple présence de ces objets peut être un indicateur très fort pour considérer qu'il y a un visage dans l'image. Les relations spatiales d'orientation et relatives entre ces composants ne sont pas très importantes pour un CNN.

La relation spatiale entre caractéristiques détectées par les phases successives de convolution est partiellement assurée par l'introduction de couches de neurones, les couches de Pooling, permettant de compresser l'information en réduisant la taille de l'image intermédiaire et ainsi d'élargir le champ de vision du réseau. Concrètement deux yeux et une bouche apparaissant sur le même champ de vision grace à l'opération de Pooling seront alors associés à un objet de plus haut niveau, un visage. Pour balayer ces inconvénients, nous avons recrutés à l'utilisation d'une autre approche d'apprentissage profond (DP) qui est l'un des extensions de CNN qui permet d'évaluer les performances et de corriger les défauts de CNN qui est L'architecture des réseaux de capsules (en anglais : Capsule neural networks CapsNet).

III.6 système de reconnaissance à base de réseau de capsule

Un CapsNet (ou réseau neuronal à capsule) est un système de machine learning de la famille des réseaux neuronaux artificiels (RNA) pouvant être utilisé pour mieux modéliser des relations hiérarchiques. Cette approche est une tentative de reproduire plus fidèlement une organisation neuronale biologique.

L'idée est d'ajouter des structures appelées capsules à un réseau de neurones à convolution (CNN) et de réutiliser les sorties de plusieurs de ces capsules pour former des représentations plus stables (par rapport à diverses perturbations) de capsules d'ordre supérieur. La sortie est un vecteur consistant en la probabilité d'une observation et une pose pour cette observation . Ce vecteur est similaire à ce qui est fait par exemple lors de la classification avec localisation dans CNN.

Parmi les autres avantages, les capsnets répondent au "problème de Picasso" dans la reconnaissance d'image : les images qui ont toutes les parties droites mais qui ne sont pas dans la

relation spatiale correcte (par exemple, dans un ”visage”, les positions de la bouche et de l’œil sont commuté). Pour la reconnaissance d’image, les capsnets exploitent le fait que, bien que les changements de point de vue aient des effets non linéaires au niveau des pixels, ils ont des effets linéaires au niveau de la pièce / de l’objet. Cela peut être comparé à l’inversion du rendu d’un objet à plusieurs parties.

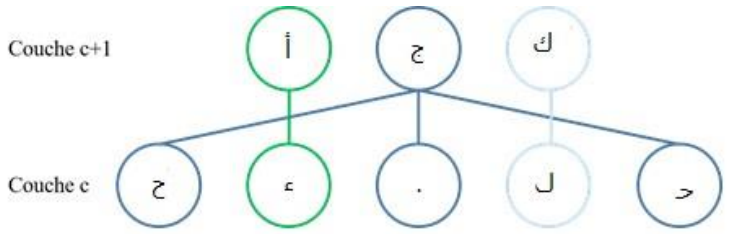


Figure III.43 – Exemple d’encapsulation

III.6.1 C’est quoi une capsule ?

Une capsule est un ensemble de neurones qui activent individuellement pour différentes propriétés d’un type d’objet, telles que la position, la taille et la teinte. Formellement, une capsule est un ensemble de neurones qui produisent collectivement un vecteur d’activité avec un élément pour chaque neurone pour maintenir la valeur d’instanciation de ce neurone (par exemple, la teinte).

Les neurones artificiels produisent un seul scalaire. De plus, les CNN utilisent des couches convolutives qui, pour chaque noyau, répliquent les poids de ce même noyau sur tout le volume d’entrée, puis produisent une matrice 2D, où chaque nombre est la sortie de la convolution de ce noyau avec une partie du volume d’entrée. Nous pouvons donc regarder cette matrice 2D en tant que sortie du détecteur de caractéristiques répliquées. Ensuite, toutes les matrices 2D du noyau sont empilées les unes sur les autres pour produire la sortie d’une couche convolutionnelle.

Ensuite, nous essayons d’obtenir une invariance des points de vue dans les activités des neurones. Nous faisons cela au moyen de la mise en commun maximale qui examine consécutivement les régions dans la matrice 2D décrite ci-dessus et sélectionne le plus grand nombre dans chaque région. En conséquence, nous obtenons ce que nous voulions : l’invariance des activités. L’invariance signifie qu’en modifiant un peu l’entrée, la sortie reste la même. Et l’activité n’est

que le signal de sortie d'un neurone. En d'autres termes, lorsque dans l'image d'entrée, nous décalons un peu l'objet que nous voulons détecter, les activités du réseau (sorties des neurones) ne changeront pas en raison du pooling maximal et le réseau détectera toujours l'objet.

Le mécanisme décrit ci-dessus n'est pas très bon, car la mise en commun maximale perd des informations précieuses et n'encode pas non plus les relations spatiales relatives entre les entités. Nous devrions plutot utiliser des capsules, car elles encapsuleront toutes les informations importantes sur l'état des caractéristiques qu'elles détectent sous la forme d'un vecteur (par opposition à un scalaire produit par un neurone). (Les capsules encapsulent toutes les informations importantes sur l'état de la caractéristique qu'elles détectent sous forme vectorielle.) Les capsules codent la probabilité de détection d'une entité comme la longueur de leur vecteur de sortie. Et l'état de la caractéristique détectée est codé comme la direction dans laquelle ce vecteur pointe (paramètres d'instanciation). Ainsi, lorsque la fonction détectée se déplace autour de l'image ou que son état change d'une manière ou d'une autre, la probabilité reste la même (la longueur du vecteur ne change pas), mais son orientation change.

III.6.1.1 Comment fonctionne une capsule ?

Comparons les capsules aux neurones artificiels. Le tableau ci-dessous résume les différences entre la capsule et le neurone :

		Capsule	**Neurone traditionnel**
Entrée de bas niveau		Vecteur ($\boldsymbol{u}_i$)	Scalaire ($\boldsymbol{x}_i$)
Opération	**Transformation affine**	$\hat{u}_{j\|i} = \boldsymbol{W}_{ij}\boldsymbol{u}_i$	—
	Pondération / Somme	$\boldsymbol{s}_j = \sum_i c_{ij}\hat{u}_{j\|i}$	$a_j = \sum_i W_i \boldsymbol{x}_i + b$
	Activation non linéaire	$\boldsymbol{v}_j = \frac{\\|\boldsymbol{s}_j\\|^2}{1+\\|\boldsymbol{s}_j\\|^2}\frac{\boldsymbol{s}_j}{\\|\boldsymbol{s}_j\\|}$	$h_i = f(a_j)$
Sortie		Vecteur ($\boldsymbol{v}_j$)	Scalaire (h_i)

Figure III.44 – Différences importantes entre les capsules et les neurone

Rappelez-vous qu'un neurone reçoit des scalaires d'entrée d'autres neurones, puis les multiplie par des poids et des sommes scalaires. Cette somme est ensuite transmise à l'une des nombreuses fonctions d'activation non linéaires possibles, qui prennent le scalaire d'entrée et produisent un scalaire en fonction de la fonction. Ce scalaire sera la sortie du neurone qui ira comme entrée à d'autres neurones. Le résumé de ce processus peut être vu sur le tableau et le diagramme ci-dessous sur le coté droit. En substance, le neurone artificiel peut être décrit en 3 étapes :

— pondération scalaire des scalaires d'entrée.
— somme des scalaires d'entrée pondérés.
— non-linéarité scalaire à scalaire.

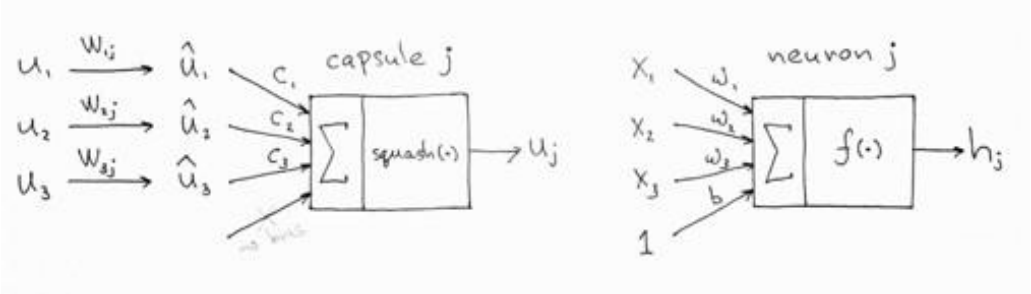


Figure III.45 – Gauche : diagramme de capsule ; à droite : neurone artificiel.

D'autre part, la capsule a des formes vectorielles des 3 étapes ci-dessus en plus de la nouvelle étape, transformée affine d'entrée :

— multiplication matricielle des vecteurs d'entrée.
— pondération scalaire des vecteurs d'entrée.
— somme des vecteurs d'entrée pondérés.
— non-linéarité de vecteur à vecteur.

III.6.2 Les étapes de calcules

Les étapes de calcul qui se déroulent à l'intérieur de la capsule sont :

III.6.2.1 Multiplication matricielle des vecteurs d'entrée

Les vecteurs d'entrée que reçoit notre capsule (u1, u2et u3 dans le diagramme) proviennent de 3 autres capsules de la couche inférieur. Les longueurs de ces vecteurs codent les probabilités que les capsules de niveau inférieur ont détecté leurs objets correspondants et les directions des vecteurs codent un état interne des objets détectés. Les vecteurs sont ensuite multipliés par des

matrices de poids W correspondantes qui codent des relations spatiales et autres importantes entre des caractéristiques de niveau inférieur et une caractéristique de niveau supérieur.

III.6.2.2 Pondération scalaire des vecteurs d'entrée

À première vue, cette étape semble très familière à celle où le neurone artificiel pondère ses entrées avant de les additionner. Dans le cas des neurones, ces poids sont appris lors de la rétropropagation, mais dans le cas de la capsule, ils sont déterminés en utilisant le routage dynamique , qui est une nouvelle façon de déterminer où va la sortie de chaque capsule.

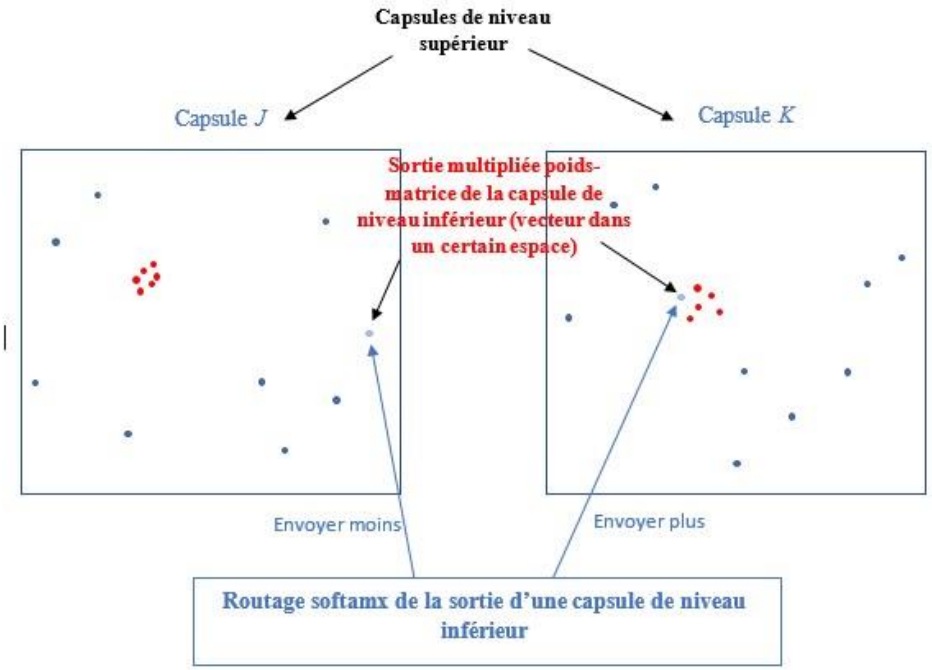


Figure III.46 – La capsule de niveau inférieur enverra son entrée à la capsule de niveau supérieur qui est d'accord avec son entrée. C'est l'essence de l'algorithme de routage dynamique.

Dans l'image ci-dessus, nous avons une capsule de niveau inférieur qui doit décider à quelle capsule de niveau supérieur elle enverra sa sortie. Il prendra sa décision en ajustant les poids C qui multipliera la sortie de cette capsule avant de l'envoyer vers des capsules de niveau supérieur gauche ou droite J et K.

Maintenant, les capsules de niveau supérieur ont déj reçu de nombreux vecteurs d'entrée d'autres capsules de niveau inférieur. Toutes ces entrées sont représentées par des points rouges et bleus. Lorsque ces points se regroupent, cela signifie que les prédictions des capsules de niveau

inférieur sont proches les unes des autres. C'est pourquoi, à titre d'exemple, il y a un groupe de points rouges dans les deux capsules J et K.
Alors, où notre capsule de niveau inférieur devrait-elle envoyer sa sortie : à capsule J ou à capsule K ? La réponse à cette question est l'essence de l'algorithme de routage dynamique. La sortie de la capsule inférieure, lorsqu'elle est multipliée par la matrice correspondanteW, atterrit loin du groupe rouge des prédictions correctes dans la capsule J. En revanche, il atterrira très près des vraies prédictions en rouge dans la capsule de droiteK. La capsule de niveau inférieur a un mécanisme de mesure de quelle capsule de niveau supérieur s'adapte le mieux à ses résultats et ajuste automatiquement son poids de telle manière que le poids C correspondant à capsule K sera élevé et le poids C correspondant à capsule J sera faible.

III.6.2.3 Somme des vecteurs d'entrée pondérés

Cette étape est similaire au neurone artificiel régulier et représente une combinaison d'entrées. Je ne pense pas qu'il y ait rien de spécial à propos de cette étape (sauf que c'est une somme de vecteurs et non une somme de scalaires). Nous pouvons donc passer à l'étape suivante.

III.6.2.4 fonction d'activation non-linéarité de vecteur à vecteur

Une autre innovation introduite par CapsNet est la nouvelle fonction d'activation non linéaire qui prend un vecteur, puis le écrase pour qu'il ait une longueur ne dépassant pas 1, mais ne change pas de direction.

$$v_j = \frac{\|\mathbf{s_j}\|^2}{1 + \|\mathbf{s_j}\|^2} \frac{s_j}{\|\mathbf{s_j}\|}$$

Le côté droit de l'équation $\frac{\mathbf{s_j}}{\mathbf{k}\mathbf{s_j}\mathbf{k}}$ met à l'échelle le vecteur d'entrée afin qu'il ait une longueur unitaire et le coté gauche effectue une mise à l'échelle supplémentaire. N'oubliez pas que la longueur du vecteur de sortie peut être interprétée comme la probabilité qu'une caractéristique donnée soit détectée par la capsule.

III.6.2.5 Routage dynamique entre les capsules

Dans cette partie on va expliquer les étapes de l'algorithme de routage , qu'on a connu d'apres les titre précedent que la capsule de niveau inférieur enverra son entrée à la capsule de niveau supérieur qui est d'accord avec son entrée. C'est l'essence de l'algorithme de routage dynamique. Maintenant que nous avons cela à l'esprit, parcourons l'algorithme ligne par ligne.

•L'algorithme de routage est le suivant :

Algorithm 1: Algorithme de routage

1 :procedure de routage($\hat{u}_{j|i}$,r,l)

2 :Pour tous les capsules i dans la couche l et j dans (l+1) : $b_{ij} \leftarrow 0$

3 : pour tous les itirations r faire

4 : pour tous les capsules i dans la couche l : $c_i \leftarrow softmax(b_i)$

5 : pour tous les capsules j dans la couche l+1 : $s_j \leftarrow \Sigma_i c_{ij} \hat{u}_{j|i}$

6 : pour tous les capsules j dans la couche l+1 : $v_j \leftarrow squash(s_j)$

7 : pour tous les capsules i dans la couche l et j dans l+ :

$$b_{ij} \leftarrow b_{ij} + \hat{u}_{j|i}.v_j$$

retourner v_j

La première ligne indique que cette procédure prend toutes les capsules à un niveau inférieur l et leurs sorties u, ainsi que le nombre d'itérations de routage r. La toute dernière ligne vous indique que l'algorithme produira la sortie d'une capsule de niveau supérieur v_j. Essentiellement, cet algorithme nous indique comment calculer le passage avant du réseau.

Dans la deuxième ligne, il y a un nouveau coefficient b_{ij} . Ce coefficient est simplement une valeur temporaire qui sera mise à jour de manière itérative et, une fois la procédure terminée, sa valeur sera stockée dans c_ij. Au début de l'apprentissage, la valeur de b_{ij} est initialisé à zéro. La ligne 3 indique que les étapes 4 à 7 seront répétées r fois (le nombre d'itérations de routage). L'étape de la ligne 4 calcule la valeur du vecteur c_i qui sont tous les poids d'acheminement pour une capsule de niveau inférieur i. Ceci est fait pour toutes les capsules de niveau inférieur. Pourquoi softmax ? Softmax s'assurera que chaque poids c_{ij} est un nombre non négatif et leur somme est égale à un. Essentiellement, softmax applique la nature probabiliste des coefficients c_{ij} .

À la première itération, la valeur de tous les coefficients c_{ij} sera égal, car sur la ligne deux tous b_{ij} sont mis à zéro. Par exemple, si nous avons 3 capsules de niveau inférieur et 2 capsules de niveau supérieur, alors tout c_{ij} sera égal à 0,5. L'état de tous c_{ij} être égal à l'initialisation de l'algorithme représente l'état de confusion et d'incertitude maximales : les capsules de niveau inférieur n'ont aucune idée des capsules de niveau supérieur qui conviendront le mieux à leur sortie. Bien sûr, à mesure que le processus se répète, ces distributions uniformes changeront.

Après tous les poids c_ij ont été calculés pour toutes les capsules de niveau inférieur, nous

pouvons passer à la ligne 5, où nous examinons les capsules de niveau supérieur. Cette étape calcule une combinaison linéaire de vecteurs d'entrée, pondérée par des coefficients de routage c_ij, déterminé à l'étape précédente. Intuitivement, cela signifie réduire les vecteurs d'entrée et les ajouter ensemble, ce qui produit un vecteur de sorties s_i. Ceci est fait pour toutes les capsules de niveau supérieur.

Ensuite, à la ligne 6, les vecteurs de la dernière étape sont passés à travers la non-linéarité de squash (fonction d'activation), qui garantit que la direction du vecteur est préservée, mais que sa longueur ne doit pas dépasser 1. Cette étape produit le vecteur de sortie v_j pour toutes les capsules de niveau supérieur.

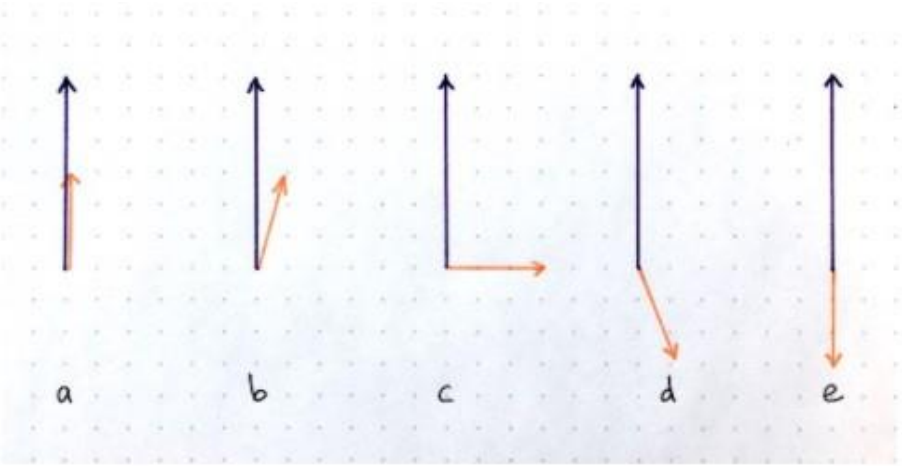


Figure III.47 – Exemple do produit scalaire.

la figure III.47 représente plusieurs scénarios possibles pour les deux vecteurs de longueurs données mais d'orientations relatives différentes,l'opération qui prend 2 vecteurs et produit un scalaire.

(a) valeurs positives la plus grande possibles.

(b) produit de point positif.

(c) produit de point zéro .

(d) produit de point négatif .

(e) produit de point négatif les plus grands possible .

III.6.2.1.1 L'architecture des réseaux de capsules

Dans cette partie et pour ne pas etre long , on va expliquer l'architecture des réseaux de capsule implimenter , donc on a deux reseaux de capsuler implimenter une pour les numéros arabe et l'autre pour les caractéres arabe.

Le CapsNet se compose de 2 parties : encodeur et décodeur. Les 3 premières couches sont codeur, et les 3 secondes sont décodeuses :

— Couche convolutionnelle.

— Couche PrimaryCaps.

— Couche DataCaps.

— 3 couches FC

•Partie 1 : Encodeur

Architecture de partie encodeur de CapsNet que nous avons appliqué sur la base de données est illustré dans la figure ci-dessous(base de donnée des lettres et chiffre arabes).

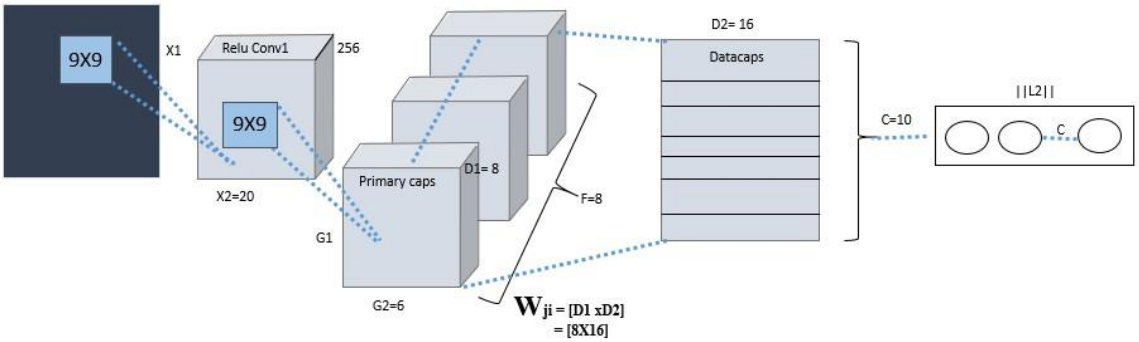


Figure III.48 – Architecture de partie encodeur de CapsNet.

La partie encodeur du réseau prend en entrée une image de 28 par 28 chiffres et apprend à l'encoder dans un vecteur à 16 dimensions de paramètres d'instanciation, c'est l que les capsules font leur travail. La sortie du réseau pendant la prédiction est un vecteur à 10 dimensions de longueurs des sorties de DataCaps. Le décodeur a 3 couches : deux d'entre elles sont convolutives et la dernière est entièrement connectée.

— Couche convolutionnelle :

Le travail de la couche convolutionnelle est de détecter les caractéristiques de base dans l'image. Dans le CapsNet, la couche convolutionnelle a 256 noyaux avec une taille de 9x9x1 et stride 1, suivis de l'activation ReLU. Pour calculer le nombre de paramètres.

— Couche PrimaryCaps :

Cette couche a 8 capsules primaires dont le travail est de prendre les caractéristiques de base détectées par la couche convolutionnelle et de produire des combinaisons des caractéristiques.

— Couche DataCaps :

Cette couche a 10 capsules de chiffres, une pour chaque classe. Chaque capsule prend en entrée un tenseur 6x6x8x8. Vous pouvez le considérer comme des vecteurs à 8 dimensions 6x6x8,

soit 288 vecteurs d'entrée au total. Selon le fonctionnement interne de la capsule, chacun de ces vecteurs d'entrée obtient sa propre matrice de poids 8x16 qui mappe l'espace d'entrée à 8 dimensions vers l'espace de sortie de la capsule à 16 dimensions. Ainsi, il y a 288 matrices pour chaque capsule, ainsi que 288 coefficients c et 288 b coefficients utilisés dans le routage dynamique. En multipliant : 288 x 8 x 16, nous obtenons 36864 paramètres entraînables par capsule, puis nous multiplions par 10 pour obtenir le nombre final de paramètres pour cette couche.

•Partie 2 : Décodeur

Le décodeur prend un vecteur à 16 dimensions du DataCaps correct et apprend à le décoder en une image d'un chiffre . Le décodeur est utilisé comme régulariseur, il prend la sortie du DataCaps correct comme entrée et apprend à recréer une image de 28 par 28 pixels, la fonction de perte étant la distance euclidienne entre l'image reconstruite et l'image d'entrée. Le décodeur force les capsules à apprendre les fonctionnalités utiles pour reconstruire l'image d'origine. Plus l'image reconstruite est proche de l'image d'entrée.

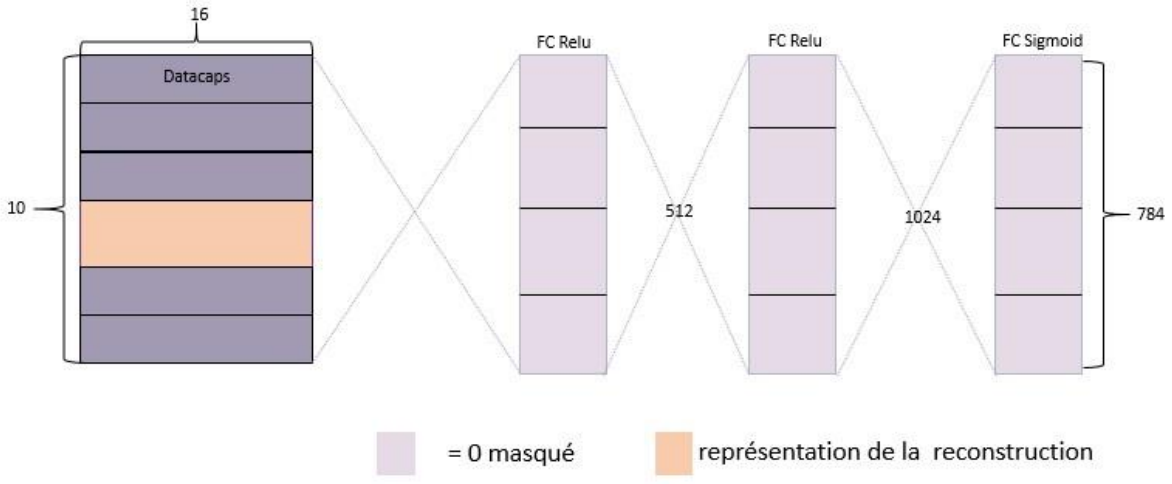
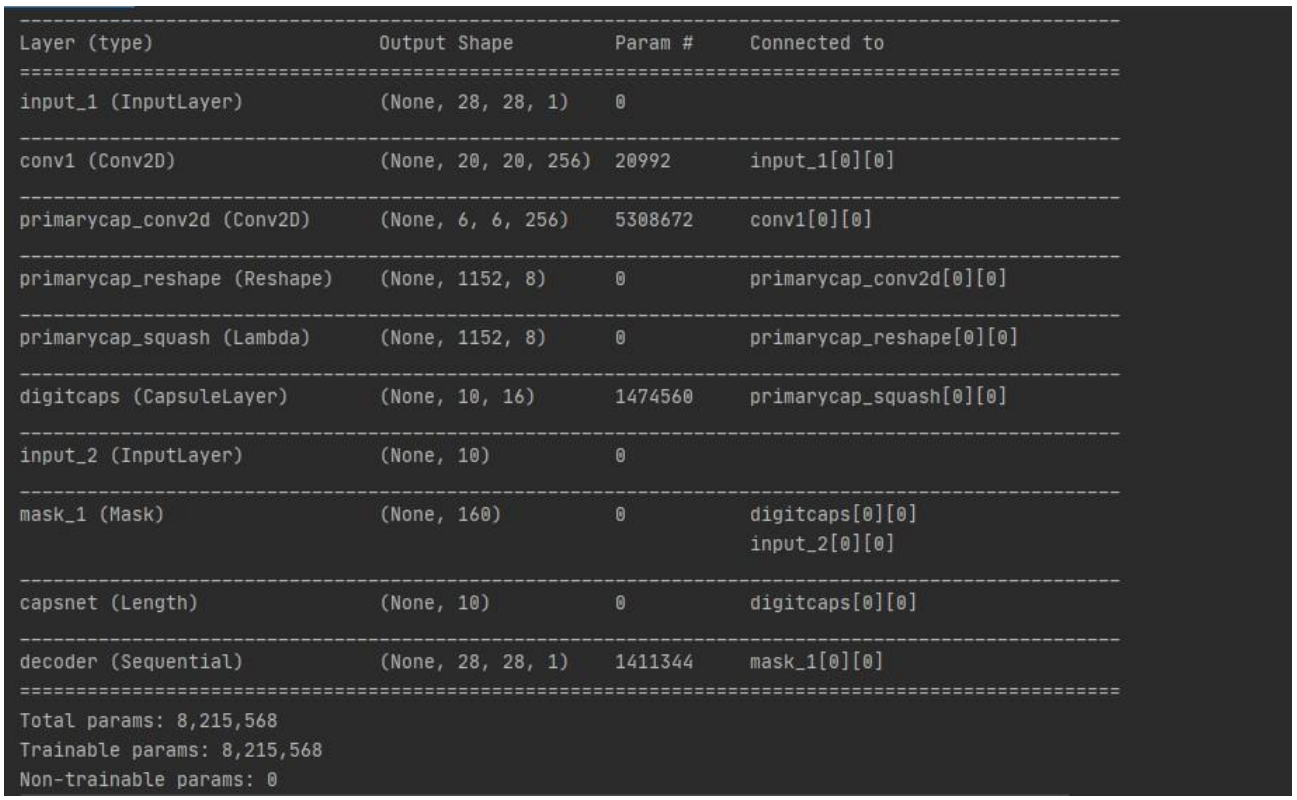


Figure III.49 – A : Architecture de partie Décodeur de CapsNet des numéros arabes

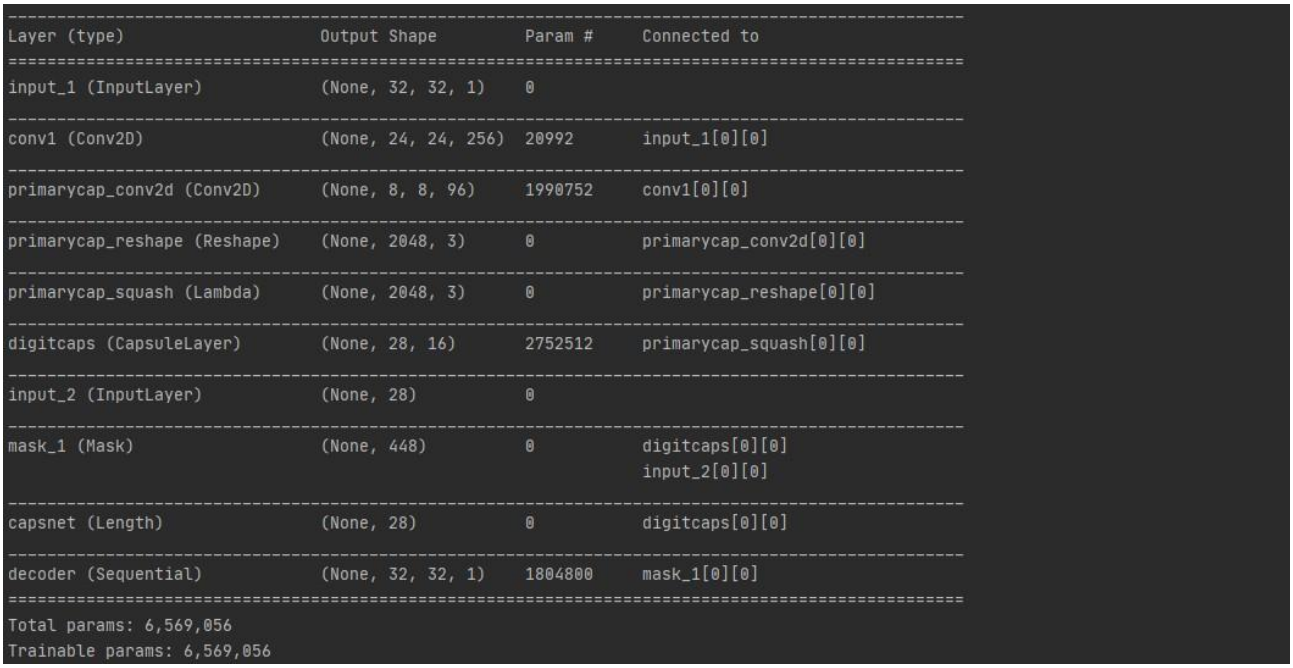


Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	(None, 28, 28, 1)	0	
conv1 (Conv2D)	(None, 20, 20, 256)	20992	input_1[0][0]
primarycap_conv2d (Conv2D)	(None, 6, 6, 256)	5308672	conv1[0][0]
primarycap_reshape (Reshape)	(None, 1152, 8)	0	primarycap_conv2d[0][0]
primarycap_squash (Lambda)	(None, 1152, 8)	0	primarycap_reshape[0][0]
digitcaps (CapsuleLayer)	(None, 10, 16)	1474560	primarycap_squash[0][0]
input_2 (InputLayer)	(None, 10)	0	
mask_1 (Mask)	(None, 160)	0	digitcaps[0][0] input_2[0][0]
capsnet (Length)	(None, 10)	0	digitcaps[0][0]
decoder (Sequential)	(None, 28, 28, 1)	1411344	mask_1[0][0]

Total params: 8,215,568
Trainable params: 8,215,568
Non-trainable params: 0

Figure III.50 – Architecture de notre modèle capsnet sur les numéros arabe

la figure III.50 et la visualisation détaillé de notre modèle avec tous les nombre paramètres utilisé. Pour le modèle de note architecture pour les lettres arabes nous avant utiliser le meme modèle en jouant juste sur les paramétres et le nombre de vecteur de la derniere couche entierment connecté de 10 a 28 sorties correspoend a le nombre des caractéres arabe et les dimenssion des images a 32*32px ,la figure en bas resume les detail de l'archeticture utiliser.



Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	(None, 32, 32, 1)	0	
conv1 (Conv2D)	(None, 24, 24, 256)	20992	input_1[0][0]
primarycap_conv2d (Conv2D)	(None, 8, 8, 96)	1990752	conv1[0][0]
primarycap_reshape (Reshape)	(None, 2048, 3)	0	primarycap_conv2d[0][0]
primarycap_squash (Lambda)	(None, 2048, 3)	0	primarycap_reshape[0][0]
digitcaps (CapsuleLayer)	(None, 28, 16)	2752512	primarycap_squash[0][0]
input_2 (InputLayer)	(None, 28)	0	
mask_1 (Mask)	(None, 448)	0	digitcaps[0][0] input_2[0][0]
capsnet (Length)	(None, 28)	0	digitcaps[0][0]
decoder (Sequential)	(None, 32, 32, 1)	1804800	mask_1[0][0]

Total params: 6,569,056
Trainable params: 6,569,056

Figure III.51 – Architecture de notre modèle capsnet sur les caractéres arabe

III.7 Résultats

Dans ce que suit nous allons presenter les résultats obtenu pour la reconnaissance des lettre et chiffres arabe en utilisant un système de reconnaissance a base de reseau de capsule

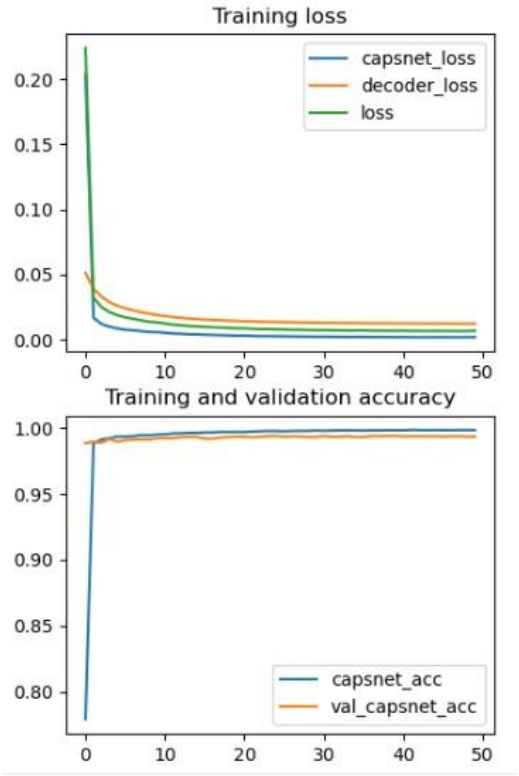


Figure III.52 – présentation Graphique de la précision et érreur pour le modele de numéros arabe

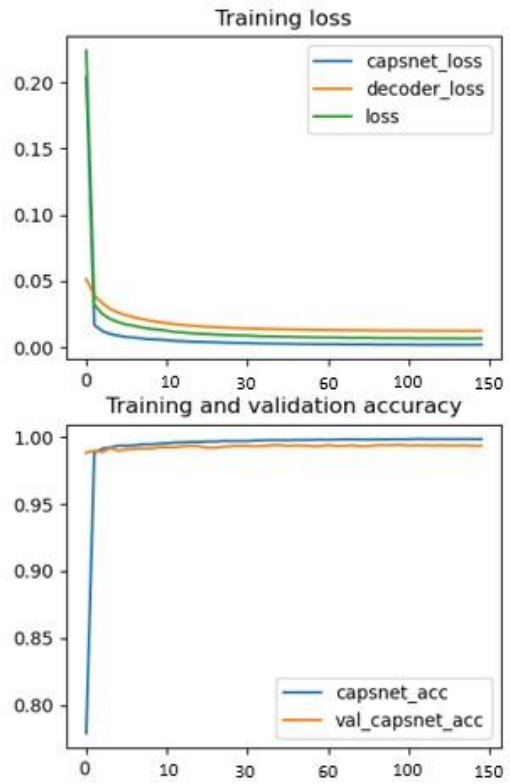


Figure III.53 – présentation Graphique de la précision et érreur pour le modele de caractéres arabe

Modèle Technique de réseau neuronal	Architecture utilisé			Nombre d'époque	Précision obtenue sur la base d'apprentissage	Précision obtenue sur la base de validation	Erreur	Temps d'exécution (En utilisons GPU)
	Couche de convolution	Couche de Pooling	Couche de Fully connected *(FC)*					
Modèle de CNN (Caractères)	04	01	02	50	94%	92%	26%	(8 heures)
Modèle de CNN (Numéros)	04	01	02	10	98%	99%	3%	(3heures)
Modèle de CapsNet (Caractères)	03		04	150	99%	96%	03%	10 jours
Modèle de CapsNet Numéros)	03	L'architecture de CapsNet ne contient pas la couche de *Pooling*	04	50	99%	92%	04%	(4jours)

Figure III.54 – différents résultats obtenus sur les deux modèles des deux techniques.

Le tableau montre la différence entre les deux modèles. Il nous affiche les résultats en terme d'apprentissage, validation et test, on a pu constater d'après notre travail que l'approche CNN et beaucoup plus rapide et moins couteuse en terme du temps et matériels, malgré que l'approche CapsNet approuve une précision considérable par rapport à CNN, Ceci revient à la grande dimension de la base de données ce qui nécessite l'utilisation d'un GPU au lieu d'un CPU.

— • L'approche CNN par rapport avec les réseaux de capsules n'a pas besoin d'un grand nombre d'époque, d'après le tableau on peut constater que pour l'apprentissage des lettres arabes et les nombres on a utilisé 50 et 10 époque respectivement, et on a obtenu un taux d'apprentissage très élevé [94%, 98%].

— • L'approche CNN comparée à CapsNet en terme de temps d'exécution pour le même nombre d' époque CNN et plus rapide mais CapsNet est plus précis (pour les caractères arabes et pour le même nombre d' époque CNN a pris 8h dans l'apprentissage au niveau des serveurs GOOGLE COLAB ce qui nous permet d'obtenir un taux de précision égal à 94%, autrement CapsNet a fallu du 10 jours pour le même apprentissage et le même

nombre d'époque mais on a obtenu 99% de précision pour les chiffres et les caractères).

— • L'approche CapsNet approuve une très grande évaluation en terme de perte de donnée, ça veut dire que l'algorithme capsnet ne perte pas beaucoup de données par rapport à CNN (CNN taux de perte 26% pour les caractères arabes, pour 03% de capsnet).

III.8 Conclusion

Nous avons présenté dans ce chapitre la conception et l'implémentation général et détailler de notre projet, qui consiste à la reconnaissance de caractères et numéros et les mots arabes basée sur le deep learning en utilisant l'approche CNN.

Dans ce projet on s'est focaliser sur l'approche des réseaux de capsules pour la reconnaissance de caractères et numéros arabe et on a utilisé les résultats obtenus par l'approche CNN (13 couches) pour comparer nos résultats finaux.

Après plusieurs tests on a pu déduire que l'approche CapsNet donne des meilleurs résultats on terme de reconnaissance par rapport à CNN mais elle plus couteuse dans le matérielles et le temps.

La comparaison des résultats a montré que le modèle CapsNet est bien meilleur que le modèle CNN, et que le nombre d'époque, la taille de la base de données sont des facteurs importants pour l'obtention de meilleurs résultats.

Conclusion Générale

Les développements technologiques de ces vingt dernières années ont permis aux systèmes numériques d'envahir notre vie quotidienne. Il y a plus à démontrer que le numérique occupe aujourd'hui une place de choix dans le monde. Parmi les composantes majeures des systèmes numériques, une grande importance est accordée à l'image. La représentation et le traitement des images numériques font l'objet de recherches très actives à l'heure actuelle. Le traitement des images est un très vaste domaine qui a connu, et connaît encore, une évolution importante depuis quelques décennies

Dans ce projet de master, nous avons concentré sur la reconnaissance de caractères et les chiffre et mot arabes en écriture manuscrite en utilisant un apprentissage approfondi (deep learning) afin d'améliorer la précision. Pour cela nous avons proposé une nouvelle architecture dans la méthode CNN et les réseaux de capsules (CapsNet).

Ce travail de recherche avait pour but de concevoir et implémenter un modèle de reconnaissance de l'écriture arabes manuscrite. Le procédé commence par prétraiter les images, détecter les zones de textes puis les extraire, puis déterminer les segments constituant le mot. Ensuite les segments sont analysés pour détecter les caractères et les classifier. Nos résultats sont très encourageant par rapport a des autres travaux dans le même thème. Nous avons créé deux modèles différents basés sur deux approches de reconnaissance très puissantes (CNN et CapsNet) et les appliqués sur les bases de données AHCD, AHDD, IESK, on a pu accomplir notre objectif avec un minimum d'erreur et un taux de précision très élevé.

Les résultats ont montré que l'approche des réseaux de capsules est bien meilleure que l'approche de réseau neuronal convolutif, on a pu obtenir une précision de 99% pour les caractères et les numéros avec l'approche CapsNet, et une précision de 94% et 96% pour les caractères et numéros arabes successivement avec l'approche CNN

Notre mémoire et organisé en 3 chapitres avec une introduction générale et une conclusion générale, dans le premier chapitre nous avons défini les notions de l'apprentissage automatique et

ses caractéristiques, dans le deuxième chapitre on a parlé de la reconnaissance des formes , et le troisième la conception et la modélisation de notre système de reconnaissance.

En ce qui concerne nos perspectives pour les travaux futurs, notre idée et notre objectif seront d'améliorer l'ensemble du système de reconnaissance de l'écriture arabe avec quelques ajouts importants :

- Rendre notre système capable de reconnaître des images plus difficile
- Améliorer notre base de donnée avec d'autres polices et tailles, et plus de mot pour la base de données des mots.
- Aujouter un mode en-ligne de reconnaissance (reconnaissance en temps reel)
- Ajouter plus de fonctionnalités (version mobile, site web).

Bibliography

[1] Bellman, R. 1961, Adaptive Control Processes : A Guided Tour, Princeton University Press, New Jersey.

[2] Blumer, A., A. Ehrenfeucht, D. Haussler et M. Warmuth. 1987, “Occam’srazor”, Inf. Proc. Let., vol. 24, p. 377–380.

[3] Fei-Fei, L., R. Fergus et P. Perona. 2006,“One-shot learning of object categories”, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 28, no 4, p. 594–611.

[4] Kolmogorov, A. N. 1965, “Three approaches to the quantitative definition of information”, Problems of Information and Transmission, vol. 1, no 1, p. 1–7.

[5] Kullback, S. 1959, Information Theory and Statistics, Wiley, New York.

[6] Li, M. et P. Vit´anyi. 2008, An Introduction to Kolmogorov Complexity and Its Applications, 3e ´ed., Springer, New York, NY.

[7] Nadaraya, E. A. 1964,“On estimating regression”, Theory of Probability and its Applications, vol. 9, p. 141–142.

[8] Solomonoff, R. J. 1964,“A formaltheoryof inductive inference”, Information and Control, vol. 7, p. 1–22, 224–254. Sutton, R. et A. Barto. 1998, Reinforcement Learning : An Introduction, MIT Press.

[9] Vapnik, V. 1998, Statistical Learning Theory, Wiley, Lecture Notes in Economics and Mathematical Systems, volume 454.

[10] Watson, G. S. 1964, “Smooth regression analysis”, Sankhya - The Indian Journal of Statistics, vol. 26, p. 359–372.

[11] Weizenbaum, J. 1966,“ELIZA-a computer program for the study of natural language communication between man and machine”, Commun. ACM, vol. 9, no 1, p. 36–45.

[12] Barber, D. 2011, Bayesian Reasoning and Machine Learning, Cambridge University Press.

[13] Bengio, Y. 2009, “Learning deep architectures for AI”, Foundations and Trends in Machine Learning, vol. 2, no 1, p. 1–127. Alsopublished as a book. Now Publishers, 2009.

[14] Bengio, Y., P. Lamblin, D. Popovici et H. Larochelle. 2007,“Greedy layerwise training of deep networks”, dans Advances in Neural Information

[15] Processing Systems 19 (NIPS’06), ´edit´e par B. Sch¨olkopf, J. Platt et T. Hoffman, MIT Press, p. 153–160.

[16] Bishop, C. M. 2006, Pattern Recognition and Machine Learning, Springer.

[17] Boser, B. E., I. M. Guyon et V. N. Vapnik. 1992, “A training algorithm for optimal margin classifiers”, dans COLT ’92 : Proceedings of the fifth annual workshop on Computational learning theory,ACM,NewYork,NY, USA, p. 144–152.

[18] Breiman,L.,J.H.Friedman,R.A.OlshenetC.J.Stone.1984,Classification and Regression Trees, Wadsworth International Group, Belmont, CA.

[19] Cortes,C.etV.Vapnik.1995,“Supportvectornetworks”,Machine Learning, vol. 20, p. 273–297.

[20] Cover, T. M. et P. E. Hart. 1967,“Nearest neighbor pattern classification”, IEEE Transactions on Information Theory, vol. 13, no 1, p. 21–27.

[21] Dempster, A. P., N. M. Laird et D. B. Rubin. 1977, “Maximum-likelihood from incomplete data via the EM algorithm”, Journal of Royal Statistical Society B, vol. 39, p. 1–38.

[22] Erhan, D., Y. Bengio, A. Courville, P.-A. Manzagol, P. Vincent et S. Bengio. 2010,“Why does unsupervised pre-training help deep learning?”, Journal of Machine Learning Research, vol. 11, p. 625–660.

[23] Haykin, S. 2008, Neural Networks and Learning Machines (3rd Edition), 3e ´ed., Prentice Hall.

[24] Hinton, G. E. 2002,“Training products of experts by minimizing contrastive divergence”, Neural Computation, vol. 14, p. 1771–1800.

[25] E., S. Osindero et Y. Teh. 2006a,“A fast learning algorithm for deep belief nets”, Neural Computation, vol. 18, p. 1527–1554.

[26] Hinton, G. E. et R. Salakhutdinov. 2006b,“Reducing the dimensionality of data with neural networks”, Science, vol. 313, no 5786, p. 504–507.

[27] Hinton, G. E. et T. J. Sejnowski. 1986, “Learning and relearning in Boltzmann machines”, dans Parallel Distributed Processing : Explorations in the Microstructure of Cognition. Volume 1 : Foundations, ´edit´e par D. E. Rumelhart et J. L. McClelland, MIT Press, Cambridge, MA, p. 282–317.

[28] Hinton, G. E., T. J. Sejnowski et D. H. Ackley. 1984,“Boltzmann machines : Constraint satisfaction networks thatlearn”, rapport technique TR-CMUCS-84-119, Carnegie-Mellon University, Dept. of Computer Science.

[29] Hoerl, A. et R. Kennard. 1970, “Ridge regression : biased estimation for non-orthogonal problems”, Technometrics, vol. 12, p. 55–67.

[30] Hornik, K., M. Stinchcombe et H. White. 1989, “Multilayer feedforward networks are universal approximators”, Neural Networks, vol. 2, p. 359– 366.

[31] Le Roux, N. et Y. Bengio. 2008,“Representational power of restricted Boltzmann machines and deep belief networks”, Neural Computation, vol. 20, no 6, p. 1631–1649.

[32] Ma, Y. et Y. Fu, ´ed..2011, Manifold Learning Theory and Applications, Taylor and Francis.

[33] Nadaraya, E. A. 1964,“On estimating regression”, Theory of Probability and its Applications, vol. 9, p. 141–142.

[34] Nocedal, J. et S. Wright. 2006, NumericalOptimization, Springer.

[35] O’Hagan, A. et J. F. C. Kingman. 1978,“Curve fitting and optimal design for prediction”, Journal of the Royal Statistical Society. Series B (Methodological), vol. 40, no 1, p. 1–42.

[36] Parzen, E. 1962, “On the estimation of a probability density function and mode”, Annals of Mathematical Statistics, vol. 33, p. 1064–1076.

[37] Rasmussen, C. E. et C. Williams. 2006, Gaussian Processes for Machine Learning, MIT Press.

[38] Rosenblatt, M. 1956,“Remarks on some nonparametric estimates of a density function”, The Annals of Mathematical Statistics, vol. 27, no 3, p. 832–837.

[39] Roweis,S.etL.K.Saul.2000,“Nonlineardimensionalityreductionbylocally linear embedding”, Science, vol. 290, no 5500, p. 2323–2326.

[40] Rumelhart, D. E., G. E. Hinton et R. J. Williams. 1986, “Learning representations by back-propagating errors”, Nature, vol. 323, p. 533–536.

[41] Salakhutdinov, R. et H. Larochelle. 2010,“Efficient learning of deep Boltzmann machines”, dans Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics (AISTATS 2010), JMLR W&CP, vol. 9, p. 693–700.

[42] Scholkopf, B. et A. J. Smola. 2002, Learning with Kernels : Support Vector Machines, Regularization, Optimization and Beyond, MIT Press, Cambridge, MA.

[43] Shakhnarovich, G., T. Darrell et P. Indyk. 2006, Nearest-Neighbor Methods in Learning and Vision : Theory and Practice (Neural Information Processing), MIT Press.

[44] Smolensky, P. 1986, “Information processing in dynamical systems : Foundations of harmony theory”, dans Parallel Distributed Processing, vol. 1, ′edit′e par D. E. Rumelhart et J. L. McClelland, chap. 6, MIT Press, Cambridge, p. 194–281.

[45] Tenenbaum, J., V. de Silva et J. C. Langford. 2000, “A global geometric framework for nonlinear dimensionality reduction”, Science, vol. 290, no 5500, p. 2319–2323.

[46] Vapnik, V. 1998, Statistical Learning Theory, Wiley, Lecture Notes in Economics and Mathematical Systems, volume 454.

[47] Vincent,P.,H.Larochelle,I.Lajoie,Y.BengioetP.-A.Manzagol.2010,“Stacked denoising autoencoders : Learning useful representations in a deep network with a local denoising criterion”, Journal of Machine Learning Research, vol. 11, no 3371–3408.

[48] Wainwright, M. J. et M. I. Jordan. 2008, “Graphical models, exponential families, and variational inference”, Foundations and Trends in Machine Learning, vol. 1, no 1-2, p. 1–305.

[49] Watson, G. S. 1964, “Smooth regression analysis”, Sankhya - The Indian Journal of Statistics, vol. 26, p. 359–372.

[50] Williams, C. K. I. et C. E. Rasmussen. 1996, “Gaussianprocesses for regression”, dans Advances in Neural Information ProcessingSystems 8 (NIPS’95), ´edit´e par D. Touretzky, M. Mozer et M. Hasselmo, MIT Press, Cambridge, MA, p. 514–520.

[51] Chapelle, O., B. Sch¨olkopf et A. Zien, ´ed.. 2006, Semi-Supervised Learning, MIT Press, Cambridge, MA.

[52] ‘’Apprentissage machine efficace : théorie et pratique’’, par Olivier Delalleau , D´epartement d’informatique et de recherche op´erationnelle Facult´e des arts et des sciences

[53] Watanabe, S. Watanabe, “Pattern recognition: Human and Mechanical”, New York Wiley, 1985

[54] Theodoridis et Koutroumbas, S. Theodoridis, K. Koutroumbas, “Pattern Recognition, Second Edition”, Academic Press, Elsevier, 2003.

[55] https://interstices.info/jcms/p_88807/marvin-minsky-un-des-cerveaux-de-l-intelligenceartificielle

[56] V. Bjorn, “One Finger at a Time: Best Practices for Biometric Security,” Banking Information Source (Document ID: 1697301411), April, 2009 .

[57] Chung et al, K.W. Cheung, J.T. Kwok, M.H. Law, K.C. Tsui,"Mining customer product ratings for personalized marketing", Decision Support Systems, vol. 35, issue 2, pp. 231-243, 2003

[58] Kaastra et Boyd, I. Kaastra, M. Boyd, "Designing a neural network for forecasting financial and economic time series", Neurocomputing. Vol. 10, no. 3, pp. 215-236. 1996.

[59] Gupta et al, S.K. Gupta, W.C. Regli, D.S. Nau, "Manufacturing Feature Instances: Which Ones to Recognize?", Symposium on Solid Modeling and Applications, pp. 141152, 1995.

[60] Hippert et al, H.S. Hippert, C.E. Pedreira, R.C. Souza, "Neural networks for shortterm load forecasting: a review and evaluation", IEEE Transactions on Power Systems, vol. 16, Part 1, pp. 44-55, 2001.

[61] Munich et Perona, M.E. Munich, P. Perona, "Visual Input for Pen-Based Computers", IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 24, issue 3, 2002.

[62] Yang et al, M.H. Yang, D.J. Kriegman, N. Ahuja, "Detecting Faces in Images: A

51 Survey", IEEE Transactions On Pattern Analysis And Machine Intelligence PAMI, vol.24; part 1, pp. 34-58, 2002.

[63] Woznica et Menal, P. Woznica, M. Mennal, "Reconnaissance des formes", 2003.

[64] Theodoridis et Koutroumbas, S. Theodoridis, K. Koutroumbas, “Pattern Recognition, Second Edition”, Academic Press, Elsevier, 2003.

[65] J.P. Cocquerez et S. Philipp, Analyse d’images : filtrage et segmentation Ed. Masson, Année 1995.

[66] Phan Viet Anhe, Développement d’un module de segmentation pour un système de reconnaissance biométrique basé sur l’iris, novembre 2008.

[67] https://www.interstices.info/jcms/c_5952/histoire-du-traitement-d-images

[68] ‘’Mise au Point d’une Application de Reconnaissance de Formes’’, Mémoire de fin d’études pour l’obtention du diplôme de Master en Informatique, DJABEUR DJEZZAR Mohammed Rafik, BENKADA Feth-a.

[69] Christian wolf. Détection de textes dans des images issues d’un flux vidéo pour L’indexation sémantique 2003.