République Algérienne Démocratique et Populaire

Université A. MIRA de Béjaia

Faculté des Sciences Exactes
Département de Recherche Opérationnelle

Mémoire présenté pour l'obtention du diplôme de master

جامعة بجاية
Tasdawit n Bgayet
Université de Béjaïa

Spécialité : Sciences de Données et Aide à la Décision

Théme

Traitement Automatique Des Caractères Latins en Tamazight

Présenté par :

BOUAMAMA Melissa
BERREGUIA Rahma

Sous la direction de : Dr. L. ASLI & Mr. A. ZAIDI

Défendu le 02/07/2024, devant le jury composé de :

M^{r} K. ABBAS	Professeur	Président de jury	UAMB - Bejaia
M^{r} A. LAOUAR	M.A.A	Examinateur	UAMB - Bejaia
M^{me} A. BOUDRIOUA	Docteur	Examinatrice	UAMB - Bejaia

Année Universitaire 2023 – 2024

Remerciments

Louange à Allah, Seigneur de l'univers, qui nous a accordé la santé, la patience et la persévérance nécessaires pour mener à bien ce travail. C'est grâce à sa guidance et à sa bénédiction que l'achèvement de ce mémoire a été possible.

Qu'Allah répande ses bénédictions sur notre prophète Mohammed, sur sa famille et sur ses compagnons.

Tout d'abord, nous tenons à exprimer notre sincère gratitude à notre encadrant, le Dr. ASLI Larbi, le chef de notre spécialité et de notre département, pour son orientation et ses précieux conseils qui ont grandement contribué à la réalisation de notre travail.

Nous souhaitons également remercier le centre de recherche en langue et culture amazighes (CRLCA) pour nous avoir accueillis et permis de réaliser notre stage au sein de leur institution. Leur soutien nous a offert l'opportunité de travailler sur un thème original et important, apportant ainsi une contribution à la réussite de notre projet.

Nous exprimons notre profonde gratitude à Mr. ZAIDI Ali, chef de département informatiques et audio-visuel au CRLCA, pour son co-encadrement, le temps et les ressources qu'il nous a généreusement accordés. Son soutien inestimable a été essentiel à la réussite de ce travail.

Nous remercions vivement le président du jury, le Pr. ABBAS Karim, Vice-Doyen de notre faculté, pour l'honneur qu'il nous a fait en présidant le jury.

Merci également à Mr. LAOUAR Abdelhak et Mme. BOUDRIOUA Asma d'avoir accepté de faire partie du jury et d'avoir accepté d'évaluer ce travail.

Enfin, un grand merci à nos familles pour leur soutien et leur encouragement au cours de ces années des études, et à tous ceux qui ont contribué de près ou de loin à la réalisation de ce travail.

Dédicace

Merci Allah de m'avoir donné la capacité d'écrire et de réfléchir, la force d'y croire, la patience d'aller jusqu'au bout du rêve.

À ma chère mère,

Qui m'a toujours entourée de son amour et a tout sacrifié pour mon bonheur. Que Dieu veille sur elle et la comble de ses bénédictions.

À la mémoire de mon défunt père,

Qui nous a quittés mais demeure toujours vivant dans mon cœur. J'espère être toujours à la hauteur de ses espérances.

À mon frère et ma soeur,

Qui ont toujours été à mes côtés et m'ont offert un soutien inébranlable. Leur amour est ma force quotidienne. Avec eux, chaque moment est précieux.

À ma binôme,

Pour leur soutien, son énergie et son bonne humeur tout au long de notre parcours. Sa collaboration a été essentielle pour la réalisation de ce travail. Ensemble, nous avons surmonté tous les défis, et je suis fière de ce que nous avons accompli.

À mes chers amis,

Pour leur encouragement, leur aide et leur soutien dans les moments difficiles. Pour leur conseil et les souvenirs des bons moments passés ensembles.

À toute ma famille,

À tous ceux que j'aime et qui m'aiment sans exception,

À moi-même.

Melissa

Dédicace

Avant tout, je tiens à exprimer ma plus profonde gratitude à Dieu.

Je dédie ce travail à :

Ma chère mère et ma grand-mère,

Votre amour, votre patience et vos sacrifices ont été ma plus grande source d'inspiration. Sans votre soutien indéfectible, ce projet n'aurait pas été possible.

À la mémoire de mon cher père,

Bien que de nombreuses années se soient écoulées depuis ton départ, ton souvenir reste vivant dans mon cœur. Ta sagesse, ton amour et tes enseignements continuent de me guider chaque jour. Ce mémoire est dédié à toi, en reconnaissance de l'influence indélébile que tu as eu sur ma vie et sur ce parcours académique, avec tout mon amour et mon respect.

Mes frères, Taki Eddin et Chams Eddine,

Votre soutien constant et vos encouragements m'ont toujours poussée à donner le meilleur de moi-même. Vous êtes pour moi une source inépuisable d'inspiration et de force.

Ma binôme,

Merci pour ton soutien, ton énergie et ta bonne humeur tout au long de notre parcours. Ta collaboration a été essentielle pour la réalisation de ce travail. Ensemble, nous avons surmonté tous les défis, et je suis fière de ce que nous avons accompli.

Mes chers amis,

Votre amitié est un trésor que je chéris profondément. Merci pour votre présence, votre soutien et les moments de joie partagés. Vous avez rendu ce voyage beaucoup plus agréable et motivant.

Avec toute mon affection et ma gratitude,
Rahma

Table des matières

Liste des figures

Liste des algorithmes

Liste des tableaux

Liste d'abriviations

OCR	Optical Character Recognition
CRLCA	Centre de Recherche en Langue et Culture Amazighes
LMDT	Langue Métalangue et Didactique de Tamazight
LAPA	Littérature Art et Patrimoine Amazighs
TAL	Traitement Automatique des Langues
PDA	Personal Digital Assistant
ML	Machine Learning
KNN	K Nearest Neighbor
NN	Neural Network
SVM	Support Vector Machine
HMM	Hidden Markov Model
IA	Intelligence Artificielle
CNN	Convolutional Neural Network
RNN	Recurrent Neural Network
LSTM	Long Short Term Memory
DL	Deep Learning
RNA	Réseaux de Neurones Artificiels
MLP	Multi-Layer Perceptron
Tanh	Tangente Hyperbolique
RELU	REctified Linear Unit
FC	Fully Connected
ResNet	Residual Network
VGG	Visual Geometry Group
DenseNet	Dense Convolutional Network
BiLSTM	Bidirectional Long Short Term Memory
CAKL	Corpus Amazighe Kabyle Latin

Introduction Générale

Contexte et Problématique

Dans le contexte technologique actuel, l'évolution rapide des technologies de l'information a profondément transformé notre manière de gérer les données et les documents. L'intelligence artificielle joue un rôle important aujourd'hui, grâce à la création de machines capables d'effectuer des tâches similaires à celles de l'intelligence humaine.

Parmi ces technologies, nous trouvons la reconnaissance optique de caractères (OCR), qui est essentielle pour convertir des documents papier en formats numériques. Cette technologie facilite diverses tâches pour l'humain, telles que l'accès, la recherche, la traduction, l'archivage et la préservation des informations.

Les systèmes OCR sont généralement conçus pour les langues largement utilisées et soutenues par de vastes bases de données. Les langues peu reconnues ou régionales, quant à elles, sont souvent négligées, ce qui entraîne une absence d'outils efficaces pour leur numérisation.

La langue amazighe, également connue sous le nom de tamazight, illustre parfaitement cette problématique. En dépit de sa reconnaissance officielle en Algérie en 2002, il n'existe pas encore de système dédié à la transcription automatique des caractères latins utilisés pour cette langue.

La nécessité de créer un système de reconnaissance automatique pour cette langue sera donc impérieux. Ce besoin sera particulièrement pressant pour le centre de recherche en langue et culture amazighes (CRLCA), où nous allons effectuer notre stage.

Parmi les activités du CRLCA figureront l'analyse et le traitement automatiques de la langue amazighe. Un système de reconnaissance permettra de numériser efficacement leurs documents, et facilitant ainsi leur accès et leur préservation. Cette numérisation deviendra indispensable.

La problématique de ce mémoire se définira donc : « Concevoir un système de reconnaissance optique de caractères efficace pour la langue amazighe transcrite en latin, capable de surmonter les défis afin de répondre aux besoins de numérisation du centre de recherche en langue et culture amazighes ».

Objectif et Solution

Dans ce mémoire, nous visons à créer un système de reconnaissance optique de caractères adapté pour les caractères de la langue amazighe. Nous allons utiliser les techniques d'apprentissage profond, car ces dernières ont démontré des résultats remarquables dans le domaine de la reconnaissance d'images.

Notre objectif principal consiste à élaborer un modèle de deep learning, capable d'identifier avec précision les caractères amazighs imprimés, dans diverses tailles et polices.

Par la suite, nous avons l'intention de développer ce modèle pour qu'il puisse reconnaître également des mots et des phrases, non seulement des caractères individuels, ce qui permettra une utilisation plus pratique de la reconnaissance amazighe.

Présentation de l'organisme d'accueil

Le centre de recherche en langue et culture amazighes (CRLCA), est un établissement public de recherche à caractère scientifique et technologique à vocation sectorielle, créé par le décret exécutif n° 17-95 du 29 Joumada El Oula 1438 correspondant au 26 février 2017 [11]. Il est localisé dans le campus Aboudaou de l'université de Bejaia.

FIGURE 1 – Centre de recherche en langue et culture amazighes

Il est placé sous la tutelle de la Direction Générale de la Recherche Scientifique et Développement Technologique du Ministère de l'Enseignement Supérieur et de la Recherche Scientifique.

Le centre a pour mission la réalisation des programmes de recherche scientifique dans les différents domaines, et de promouvoir la langue et la culture amazighes dans toutes ses formes [32]. Parmi ces missions, nous citons :

En matière de langue amazighe :

- Mettre en œuvre des projets de recherche dans les sciences et techniques linguistiques appliquées à la langue amazighe dans toutes ses variétés.
- Réaliser des travaux de recensement, d'adaptation et de production de la terminologie scientifique et technique.
- Développer des méthodes et techniques de traduction pour répondre aux besoins du système éducatif, de formation et de recherche.

En matière de culture amazighe :

- Étudier et documenter l'évolution de la culture amazighe à travers l'histoire.
- Transcrire et valoriser les expressions de la culture amazighe.
- Recenser et valoriser les pratiques culturelles et coutumes amazighes.

Le CRLCA est composé de trois départements techniques, trois services administratifs et quatre divisions de recherche. Chacun de ces composants a ses propres missions et tâches.

Les divisions de recherche sont constituées de quatre divisions :

- La division « Langue, Métalangue et Didactique de Tamazight (LMDT) ».
- La division « Civilisation Amazighe ».
- La division « Littérature, Art et Patrimoine Amazighs (LAPA) ».
- La division « Terminologie, Traduction et Traitement Automatique des Langues (TAL) ».

Chaque division est chargée de mener des études et des recherches approfondies dans son domaine pour promouvoir la langue et la culture amazighes sous toutes leurs formes. À l'avenir, ces divisions deviendront des sources essentielles de ressources, assurant la continuité et la pérennité du centre [32].

Afin de réaliser notre objectif, nous avons l'opportunité de travailler au sein de la dernière division, celle de TAL. Parmi ses tâches, nous trouvons :

- La terminologie amazighe.
- Les traductions déjà réalisées de et vers tamazight.
- Les bases de données de corpus amazighs.
- Le correcteur électronique de la langue amazighe.
- Les programmes de reconnaissance du caractère amazigh pour les applications en OCR.

Voici un organigramme qui explique en détail la structure du CRLCA [11] :

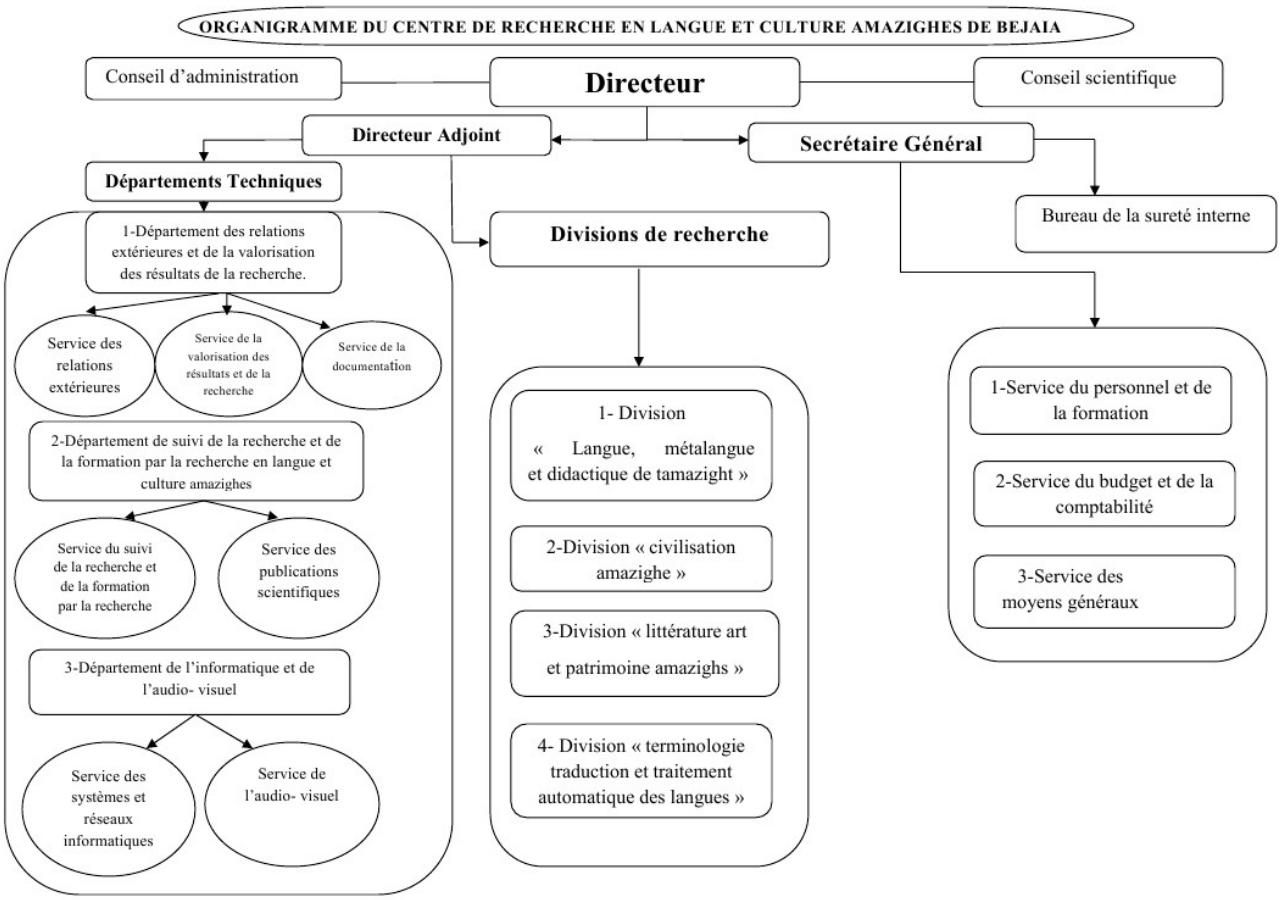


FIGURE 2 – Organigramme du centre de recherche en langue et culture amazighes

Organisation du mémoire

Ce mémoire est structuré en cinq chapitres, chacun abordant un aspect spécifique du traitement automatique de la langue amazighe. Nous avons organisé notre travail de la manière suivante :

Le premier chapitre présente les concepts généraux liés à la reconnaissance automatique de l'écriture, ainsi que les étapes et les méthodes utilisées à cet effet. Quelques travaux déjà réalisés dans ce domaine y sont également présentés.

Le deuxième chapitre donne un aperçu général du deep learning et explore les réseaux de neurones artificiels, qui constituent le modèle utilisé pour notre travail.

Le troisième chapitre est consacré à la langue amazighe et à son alphabet latin. Nous y décrivons la base de données que nous avons proposée pour le traitement de cette langue imprimée en multi-polices. Nous présentons des détails sur la méthodologie suivie pour la collecte et le nettoyage de ces données.

Le quatrième chapitre décrit les différentes phases de notre système, en mettant l'accent sur nos contributions à la reconnaissance hors ligne de l'écriture amazighe imprimée. Cela inclut les prétraitements effectués sur la base de données, les différentes étapes de segmentation et la procédure d'extraction des caractéristiques utilisées.

Enfin, le cinquième chapitre est consacré à la partie implémentation. Nous y présentons l'architecture proposée pour notre système, discutons des résultats obtenus sur la base de données construite localement, et faisons une comparaison entre les différents résultats.

Nous terminons notre travail par une conclusion générale, dans laquelle nous présentons un bref aperçu des points discutés dans ce mémoire, les difficultés rencontrées lors de la réalisation de ce travail, et proposons des perspectives pour les travaux futurs.

1 Reconnaissance de l'Écriture

À mesure que le temps progresse, la technologie évolue vers un univers numérique où tout se métamorphose en format électronique, notamment la transcription textuelle. Aujourd'hui, cette dernière est devenue essentielle et largement répandue dans divers secteurs, pour l'exécution de nombreuses tâches.

Elle permet de convertir des fichiers audio, vidéo et des documents en texte éditable électroniquement, cela permet de faciliter de multiples opérations pour l'humanité, telles que la numérisation, l'archivage des documents, la traduction automatique et la reconnaissance.

Ce chapitre est dédié à l'exploration de quelques notions fondamentales. Pour mieux appréhender les définitions, nous préciserons d'abord le champ du traitement d'images numériques. Ensuite, nous allons approfondir la compréhension du système de reconnaissance d'écriture, en détaillant ses multiples aspects et son architecture, puis nous décrirons ses différentes phases. Enfin, nous mettrons en lumière quelques travaux significatifs réalisés dans ce domaine et présenterons quelques défis de l'extraction du texte à partir des images.

Sommaire

1.1 Notions et préliminaires

Voici quelques notions fondamentales à connaître pour mieux appréhender les définitions :

Définition* 1.1.1.** ***Segmentation : *un processus qui permet de diviser une image contenant du texte, en différentes régions ou composants, tels que des lignes, des mots et des caractères.*

Définition* 1.1.2.** ***Transcription textuelle : *un processus qui consiste à convertir un audio, une vidéo, une image ou tout autre support visuel en texte écrit.*

Définition* 1.1.3.** ***Graphème : *la plus petite unité distinctive d'un système d'écriture, qui permet de représenter les sons de l'alphabet d'une langue graphiquement.*

Définition* 1.1.4.** ***Extraction des primitives : *un processus de détection et d'identification à partir d'une image les caractéristiques de base d'un texte.*

Définition* 1.1.5.** ***Une étiquette : *une annotation ou une marque, associée à une image qui contient du texte, indique les caractères ou les mots présents dans l'image.*

Définition* 1.1.6.** ***Seuil : *une valeur spécifique utilisée pour séparer les pixels d'une image en deux groupes distincts.*

Définition* 1.1.7.** ***Histogramme de niveau : *une représentation graphique de la distribution des niveaux de gris dans une image numérique.*

Définition* 1.1.8.** ***La fonte : *un ensemble de caractères d'un même style et de la même taille, elle décrit les différents styles d'écriture comme les polices de caractères.*

Définition* 1.1.9.** ***Classificateur : *un algorithme ou un modèle, qui est entraîné pour classer des données dans différentes catégories, en fonction de quelques caractéristiques des données d'entrée.*

Définition* 1.1.10.** ***Profil de projection des densités de pixels : *une technique utilisée dans le prétraitement d'images afin extraire des caractéristiques pertinentes.*

Définition* 1.1.11.** ***Histogramme directionnel : *une représentation de la distribution des gradients d'intensité dans différentes directions dans une image.*

Définition* 1.1.12.** ***Histogramme des transitions : *une mesure de la fréquence des changements de niveaux de gris ou de couleur le long des contours dans une image.*

1.2 Traitements d'images numériques

Le traitement d'images numériques est un domaine vaste, qui englobe un ensemble de techniques et méthodes dédiées à la manipulation, à l'analyse, à l'interprétation et à la modification d'images, afin d'améliorer leur qualité, d'extraire des informations précises et de les adapter à divers besoins et applications.

1.2.1 Caractéristiques d'une image numérique

***Définition* 1.2.1.** *Une image peut être définie comme une fonction bidimensionnelle, f(x,y), où x et y sont des coordonnées spatiales, et l'amplitude de f à n'importe quelle paire de coordonnées (x, y) est appelée l'intensité ou le niveau de gris de l'image à ce point [10].*

***Définition* 1.2.2.** *L'image est un ensemble structuré d'informations, qui sont caractérisé par différents paramètres [10].*

Parmi ces paramètres, nous pouvons citer :

Dimension : la taille de l'image sous forme d'une matrice de pixels.
Bruit : un phénomène de brusque variation de l'intensité d'un pixel par rapport à ses voisins [22].
Contraste : c'est l'opposition marquée entre les régions sombres et les régions claires d'une image [22].
Luminance : le degré de luminosité des points de l'image.
Images à niveaux de gris : la couleur du pixel peut prendre des valeurs, allant du noir au blanc (entre 0 et 255), en passant par un nombre fini de niveaux intermédiaires [22].
Images en couleurs : elles utilisent plusieurs valeurs de luminosité pour chaque pixel, les couleurs sont représentées à l'aide des couleurs primaires (le rouge, le vert et le bleu).
Profondeur : le nombre de bits par pixel.
Le poids de l'image : la dimension de l'image multipliée par sa profondeur.
Contours et Textures : représentent les frontières entre les objets de l'image, ou la limite entre deux pixels, son extraction permet d'identifier les points qui séparent deux textures dans une image. Les textures décrivent la structure des contours [10].

1.3 Applications de traitements d'images numériques

Les applications du traitement d'images numériques sont diverses, et englobent nombreux domaines, tels que :

- L'imagerie médicale (diagnostic médical, tomographie).
- La sécurité et la surveillance (détection d'intrusion, reconnaissance faciale).
- La recherche et les sciences (microscopie et imagerie, astronomie et imagerie spatiale).
- La robotique en industrie (reconnaissance de pièces, véhicule autonome).
- Les applications militaires (reconnaissance aérienne et sous-marine).
- La reconnaissance de l'écriture (les adresses de courriers postaux, les documents classiques et modernes).

1.4 Reconnaissance de l'écriture

La reconnaissance de l'écriture ou la reconnaissance optique de caractères, également appelée en anglais Optical Character Recognition (OCR), est un processus informatique qui permet de reconnaître les lettres dans un fichier image de texte, et de le convertir en un fichier texte [17]. En d'autres termes, il permet de convertir des documents papier ou imprimés en fichiers éditables contenant le même texte, en appliquant plusieurs traitements à travers des modules constituant le système.

La tâche d'un OCR consiste à analyser l'image et la segmenter en mots et caractères, puis effectuer la reconnaissance. Il résulte une transcription textuelle de l'image par laquelle des traitements automatiques sont possibles comme la recherche de mots, de noms, résumé [8].

1.4.1 Domaine d'application de l'OCR

En raison de la capacité de l'OCR à convertir des images contenant du texte en texte éditable, l'OCR est utilisé dans différents secteurs et dans plusieurs domaines d'activité, parmi les secteurs nous citons :

- Les services financiers.
- L'administration publique.
- Les soins de santé.
- La banque.
- Le commerce.
- Les voyages.
- Le secteur juridique.
- Etc.

Et parmi les domaines d'activité, nous citons quelques-uns :

L'analyse financière : permet d'analyser des textes, automatiser la création de rapports financiers, la gestion et l'archivage électronique des documents.

Numérisation de documents : permet de stocker, rechercher et partager des documents de manière électronique.

Reconnaissance de formulaires : extraire des données à partir de formulaires remplis manuellement, tels que des formulaires médicaux.

La gestion automatique des chèques : la lecture automatique du montant en chiffres et en lettres.

Gestion des stocks : lire et traiter les factures, les bons de commande et de livraison, cela permet de faciliter la gestion automatisée des stocks et des approvisionnements.

Vérification de passeports : extraire les informations contenues dans les passeports des voyageurs, comme le nom, prenom, la nationalité, le numéro de passeport.

Traduction automatique : traduire automatiquement des documents imprimés ou des panneaux dans différentes langues.

Reconnaissance de plaques d'immatriculation : lire et reconnaître les caractères des plaques d'immatriculation des véhicules et surveiller leur mouvement.

1.5 Différents aspects de l'OCR

En raison d'absence d'un système d'OCR générique capable de traiter tous les types de documents et de textes de manière efficace, la structuration de l'OCR est organisée selon trois principaux critères : l'outil d'acquisition, les approches de reconnaissance utilisées et les techniques d'extraction des caractéristiques [49].

Dans le premier critère, il existe deux systèmes de reconnaissance importants : la reconnaissance en-ligne et la reconnaissance hors-ligne. Le second critère se compose de deux approches principales : l'approche globale et l'approche analytique. Le dernier critère, nous trouvons des méthodes statistiques, structurelles et stochastiques.

Cependant, avant de définir ces critères, nous devons d'abord parler de la différence entre la reconnaissance de l'imprimé et celle du manuscrit.

En ce qui concerne le dernier critère (les techniques d'extraction des caractéristiques), nous le définirons en détail dans la prochaine section.

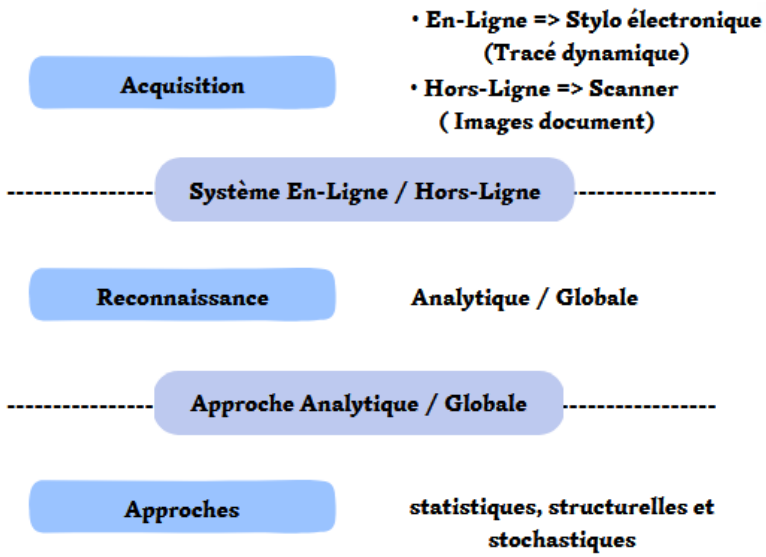


FIGURE 1.1 – Différents aspects de l'OCR

1.5.1 Reconnaissance de l'imprimé et du manuscrit

L'OCR peut être subdivisé en reconnaissance de l'écriture manuscrite et de l'écriture imprimée [19]. Néanmoins, la méthode de la reconnaissance de l'imprimé et du manuscrit n'est pas la même.

La reconnaissance de l'imprimé concerne l'analyse et la conversion de texte imprimé, tel que celui trouvé dans des livres, des magazines, des journaux par exemple.

La reconnaissance du manuscrit représente l'analyse et la conversion de texte écrit à la main, tels que les documents manuscrits et les formulaires remplis à la main.

La reconnaissance de manuscrits est plus difficile à appliquer que celle de l'imprimés, cette dernière est facile à réaliser, car le texte est plus uniforme, possède des caractères clairement définis et des polices standardisées, ce qui simplifie la phase de lecture.

Dans le cas du manuscrit, nous trouvons plusieurs styles d'écriture humaine, ces styles peuvent changer d'une personne à l'autre, et même chez la même personne dans des circonstances différentes. Les systèmes OCR pour la reconnaissance du manuscrit doivent être capables de reconnaître et d'interpréter une grande variété de styles d'écriture, ce qui rend cette tâche plus difficile.

FIGURE 1.2 – Texte en tamazight transcrit en latin manuscrit et en imprimé

1.5.2 Système en-ligne et hors-ligne

Il s'agit de deux systèmes différents, classés en fonction du mode d'acquisition, chacun possède ses propres outils et ses propres algorithmes pour la reconnaissance de caractères.

Les systèmes en-ligne (on-line) prennent l'information chronologique des mouvements du bras du scripteur [33], l'utilisateur écrit sur une tablette spéciale, puis le système reconnaît l'écriture et envoie le résultat, les symboles étant reconnus au fur et à mesure qu'ils sont écrits à la main.

Ce système est généralement réservé à l'écriture manuscrite. En ce qui concerne les équipements électroniques utilisés pour écrire, nous trouvons par exemple les PDA (Personal Digital Assistant) et les Pocket PC.

Parmi les avantages de la reconnaissance en ligne :

- Le taux de reconnaissance élevé, le fait que l'utilisateur écrive sur une tablette spéciale réduit le bruit.
- La possibilité d'apporter des modifications et des corrections à l'écriture de manière interactive et disponible.

Les systèmes hors-ligne (off-line) prennent comme entrée une image numérisée d'un document déjà rédigé, qu'il soit imprimé ou manuscrit, par exemple un formulaire, un livre. La reconnaissance hors ligne est plus difficile que celle en ligne, elle est souvent confrontée à des difficultés telles que les anciens documents, ils peuvent présenter une écriture manuscrite irrégulière ou des polices de caractères difficiles à reconnaître.

De plus, lors du processus de numérisation des documents, des imperfections telles que des taches, des plis ou des dégradations du papier peuvent survenir et entraîne un taux de bruit élevé dans l'image. Pour surmonter ces défis, les systèmes hors ligne doivent utiliser des techniques avancées de prétraitement pour améliorer la qualité de l'image. Nous aborderons ces techniques dans ce chapitre.

Malgré ces difficultés, la reconnaissance hors ligne est largement utilisée dans de nombreux domaines, tels que l'archivage de documents et le traitement de formulaire, car elle offre une bonne solution pour convertir des documents papier en format numérique en facilitant leur gestion et leur traitement.

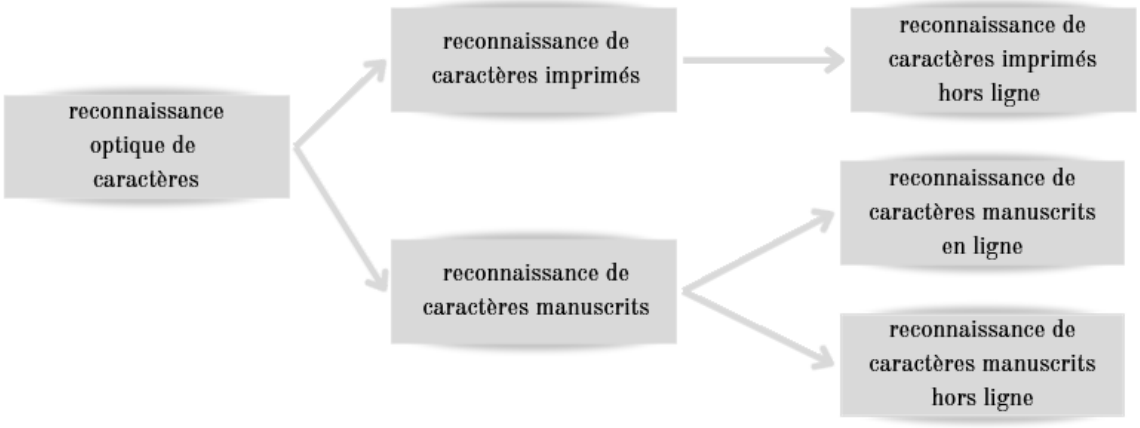


FIGURE 1.3 – Reconnaisance on-line et off-line

1.5.3 Approche globale et analytique

Il existe deux approches pour la reconnaissance de mots : l'approche globale et l'approche analytique.

L'approche globale ou approche holistique, consiste à apprendre et reconnaître un mot dans son ensemble, elle considère le mot comme une seule entité et le décrit indépendamment des caractères qui le constituent [23].

L'avantage de cette approche, est qu'elle permet de garder le caractère dans son contexte avoisinant, ce qui facilite une modélisation plus efficace des variations de l'écriture, et des dégradations qu'elle peut subir.

Le défaut de cette approche est qu'elle n'est applicable qu'à de petits vocabulaires, au-delà de quelques dizaines de mots, cette approche devient non fiable et coûteuse, elle est pénalisante en termes de complexité du traitement, de taille mémoire et de temps de calcul.

L'approche analytique consiste à segmenter le mot en sous-parties, telles que des caractères ou des graphèmes, qui sont ensuite modélisés, les mots sont reconstruits en assemblant ces modèles, ils permettent l'utilisation de lexiques libres basés sur l'alphabet ou les graphèmes appris [23].

Contrairement à l'approche globale, l'approche analytique offre l'avantage de pouvoir reconnaître un vocabulaire sans limite prédéfinie, car le nombre de caractères est naturellement fini.

L'extraction des primitives est plus simple sur un caractère que sur une chaîne de caractères, néanmoins reconnaître les entités segmentées puis tendre vers une reconnaissance du mot, peut être une tâche délicate susceptible de générer divers types d'erreurs.

1.6 Étapes du système de reconnaissance

Les étapes des systèmes de l'OCR peuvent varier légèrement, en fonction de la mise en œuvre spécifique et de la complexité du système.

La figure 1.4 représente globalement le processus général des systèmes de l'OCR.

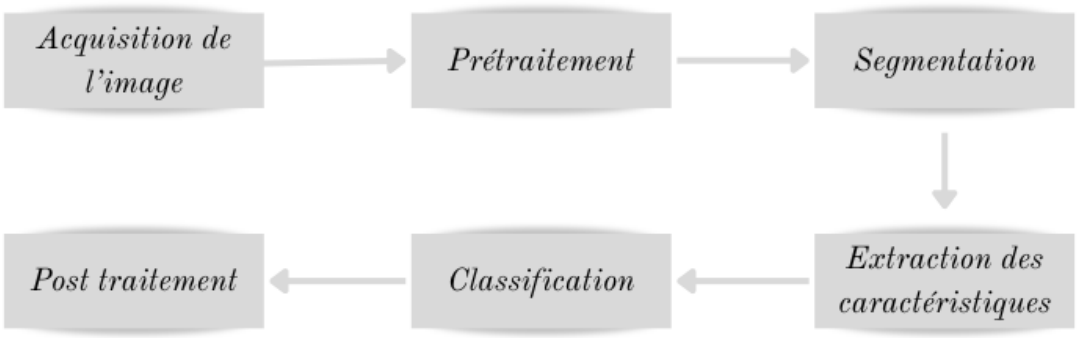


FIGURE 1.4 – Étapes générales des systèmes OCR

Un système OCR prend une image de texte en entrée en utilisant un dispositif d'acquisition, cette image est ensuite soumise à un prétraitement pour améliorer sa qualité [49].

Par la suite, elle subit une étape de segmentation pour extraire les segments de base du système, puis une extraction des caractéristiques est nécessaire pour obtenir une description de ces éléments. Ces descriptions sont fournies à un classificateur qui assigne une étiquette à chaque description [49].

Enfin, des opérations de post-traitement peuvent être faites pour évaluer la classification, ou pour reconstruire les éléments du document, par exemple les mots ou les lignes de texte [49].

Remarque 1.6.1. *Les étapes de prétraitement, segmentation et post-traitement ne sont pas nécessairement exécutées par tous les systèmes OCR [33].*

1.6.1 Acquisition de l'image

La phase d'acquisition consiste à capturer l'image d'un texte écrit sur papier, en utilisant des capteurs physiques tels qu'un scanner ou une caméra, et de la convertir en grandeur numérique adaptée au système de traitement, à condition que cette opération soit réalisée avec le minimum de dégradation possible. Une image en niveau de gris est alors obtenue.

1.6.2 Prétraitement

Le prétraitement est une étape importante avant la reconnaissance du texte dans des documents numérisés. Il vise à préparer les données (image d'entrée) pour la phase d'analyse suivante (extraction des caractéristiques).

Généralement, le prétraitement n'est pas spécifique à la reconnaissance de texte, mais constitue une étape classique en traitement d'image, il s'agit d'un ensemble de traitements appliqués à l'image, afin d'améliorer sa qualité, réduire le bruit superposé aux données, et essayer de ne conserver que l'information significative de la forme représentée.

Parmi les opérations de prétraitement généralement utilisées, nous avons :

- La binarisation.
- La réduction du bruit.
- Le redressement de l'écriture.
- La squelettisation.
- L'extraction de la ligne de base.
- La normalisation.

1.6.2.1 Binarisation

La binarisation consiste à transformer une image en couleur, ou composée de plusieurs niveaux de gris en une image binaire. Cette dernière est constituée de deux valeurs 0 et 1, permettant de distinguer l'image du support papier en blanc des traits des gravures et des caractères en noir [19].

Une étude comparative de différentes méthodes a été donnée par Trier et al. ([Trie95a] [47], [Trie95b] [46]). Aussi, Sezgin et al. [Sezg04] [41] proposent un tour d'horizon complet des différentes méthodes de binarisation, en les décrivant et en les évaluant. La plupart des méthodes sont basées sur un calcul de seuil afin de délimiter les deux classes. Celui-ci peut être global ou local [19].

L'approche globale : elle est calculée à partir d'une mesure globale sur l'ensemble de l'image [17]. Ce seuil est déterminé sur l'image entière et est ensuite appliqué à tous les pixels de l'image pour segmenter l'histogramme des niveaux de gris en deux classes correspondant au fond et à la forme.

L'approche locale : elle se caractérise par le fait que la classification d'un pixel dépend à la fois de ce pixel lui-même, et des informations de ses voisins locaux [17].

1.6.2.2 Réduction du bruit

Il est possible lors de la première phase qu'un bruit s'introduise dans l'image. Le bruit est une erreur aléatoire dans la valeur de pixel, généralement causée par la reproduction, la numérisation ou la transmission de l'image originale.

Plusieurs études ont été menées pour éliminer tout type de bruit présent dans une image, comme les lignes invisibles, le flou et les arrière-plans perturbés.

En OCR, Farahmand et al (2013) ont regroupé les différents types de bruit pouvant apparaitre dans les documents scannés et ont discuté quelques méthodes pour éliminer chaque type [1].

1.6.2.3 Redressement de l'écriture

L'un des problèmes rencontrés en OCR est le redressement de l'écriture. Ce redressement peut être dû à une mauvaise positionnement du document sur le scanner, si le document a été placé en biais ou être intrinsèque au texte lui-même, il est nécessaire de corriger cette inclinaison afin de rétablir la structure des lignes horizontales de l'image du texte, cela se fait en détectant l'angle d'inclinaison [23].

Si α représente l'angle d'inclinaison pour redresser l'image, une rotation isométrique d'angle $-\alpha$ est effectuée grâce à cette transformation linéaire [23] :

$$x' = x\cos(\alpha) + y\sin(\alpha) \tag{1.1}$$
$$y' = y\cos(\alpha) + x\sin(\alpha) \tag{1.2}$$

1.6.2.4 Squelettisation

La squelettisation est l'une des techniques utilisées lors de la phase de prétraitement, elle vise à réduire l'information utile, en ne gardant que le squelette de la forme, cela permet de simplifier la forme des caractères dans une image, à un ensemble de lignes plus faciles à traiter.

1.6.2.5 Normalisation

La normalisation est une opération de prétraitement très importante pour la reconnaissance de caractères, elle vise à mettre les caractères d'une image à des tailles standard, cette taille peut varier d'une écriture à l'autre, d'une fonte à l'autre, et même au sein d'une même fonte après agrandissement ou réduction, ce qui peut causer une instabilité des paramètres [19].

L'objectif de la normalisation est de réduire la variation intra-classe de la forme des caractères, afin de faciliter le processus d'extraction de caractéristiques, et améliorer la précision de la classification.

Voici un exemple explicatif des effets de certaines opérations de prétraitement appliquées à une image écrite en langue amazighe :

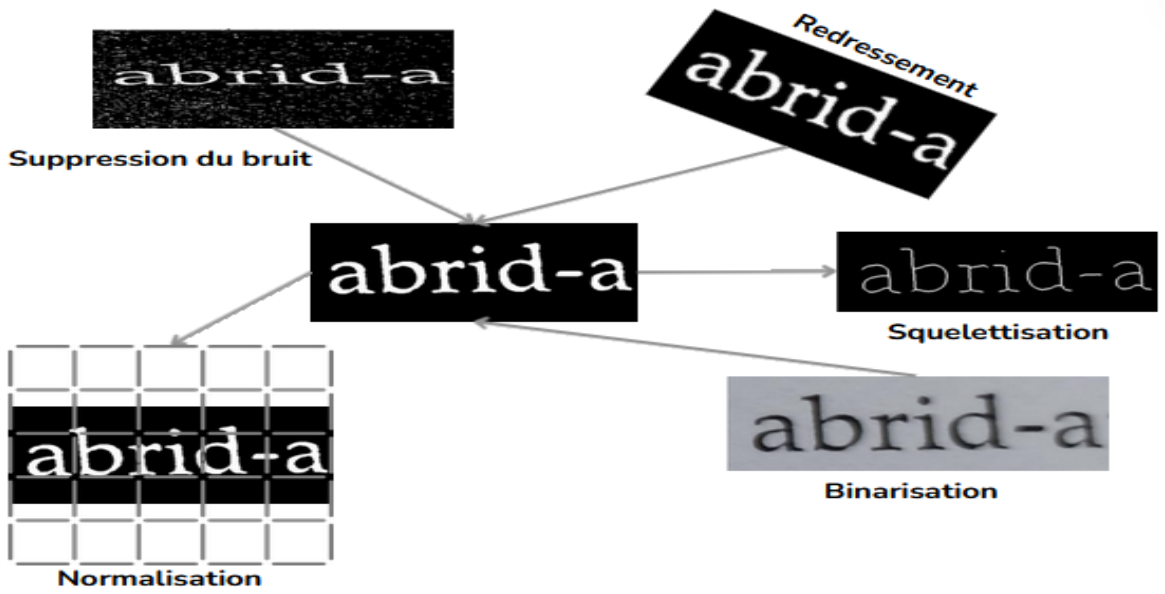


FIGURE 1.5 – Exemple des étapes générales de la phase de prétraitement

Dans la phase de prétraitement d'un document, une ou plusieurs techniques peuvent être mises en œuvre pour préparer les données avant leur analyse, le choix de prétraitement dépend de plusieurs facteurs :

Le type d'écriture du document : un document manuscrit nécessitera plus de techniques qu'un document imprimé.

La qualité du document à traiter : un document de haute qualité peut nécessiter moins de corrections et d'ajustements que celui de qualité inférieure.

La méthode d'analyse adoptée : pour étudier un document, la méthode d'analyse joue un rôle important dans le choix des techniques de prétraitement à utiliser, certaines méthodes peuvent être plus sensibles aux imperfections des données, qui nécessitent des techniques de prétraitement plus rigoureuses pour garantir des résultats fiables.

1.6.3 Segmentation

La phase de segmentation permet de séparer les blocs de texte des blocs graphiques, et d'extraire de chaque bloc de texte des lignes, puis des mots, des caractères et des pseudo-caractères si nécessaire, cela à partir des lignes extraites [17].

Cette opération donne un résultat sous forme d'un caractère ou autre chose isolé issu d'une image. Les méthodes de segmentation dépendent de plusieurs critères : la langue traitée, les fontes utilisées, etc.

On distingue quatre niveaux de segmentations :

Segmentation de la page : permet de localiser dans chaque page, les zones d'information conformément à leur apparence physique, elle est associée généralement à l'étiquetage qui consiste à

déterminer la nature du media représenté dans chaque zone, que ce soit texte, graphique ou photographie. Cette classification permet ensuite d'orienter la reconnaissance, vers des systèmes spécialisés dans l'analyse de chaque type de media [31].

Segmentation de texte en lignes : les lignes sont extraites en utilisant la méthode de la projection horizontal, cette méthode calcule l'histogramme horizontal de l'image, si le nombre de lignes blanches successives rencontrées du haut vers le bas est supérieur à un seuil fixé, le nombre de ligne est augmenté, puis le système extrait le début et la fin de chaque ligne [9].

Segmentation d'une ligne en mots : il s'agit d'analyser la ligne pour la découper en mots ou pseudo-mots, cette étape est réalisée en déterminant l'histogramme des projections verticales des lignes, afin de détecter les espaces entre les mots et pouvoir les séparer [9].

Segmentation d'un mot en caractères : c'est l'étape la plus décisive pour tout système OCR, le choix de l'algorithme est le facteur clé pour décider la précision du système. Un point de segmentation se trouve dans la ligne de base qui ne contient aucune information, il est situé entre deux caractères. La segmentation consiste à localiser deux points de segmentations, puis extraire le caractère entre eux [9].

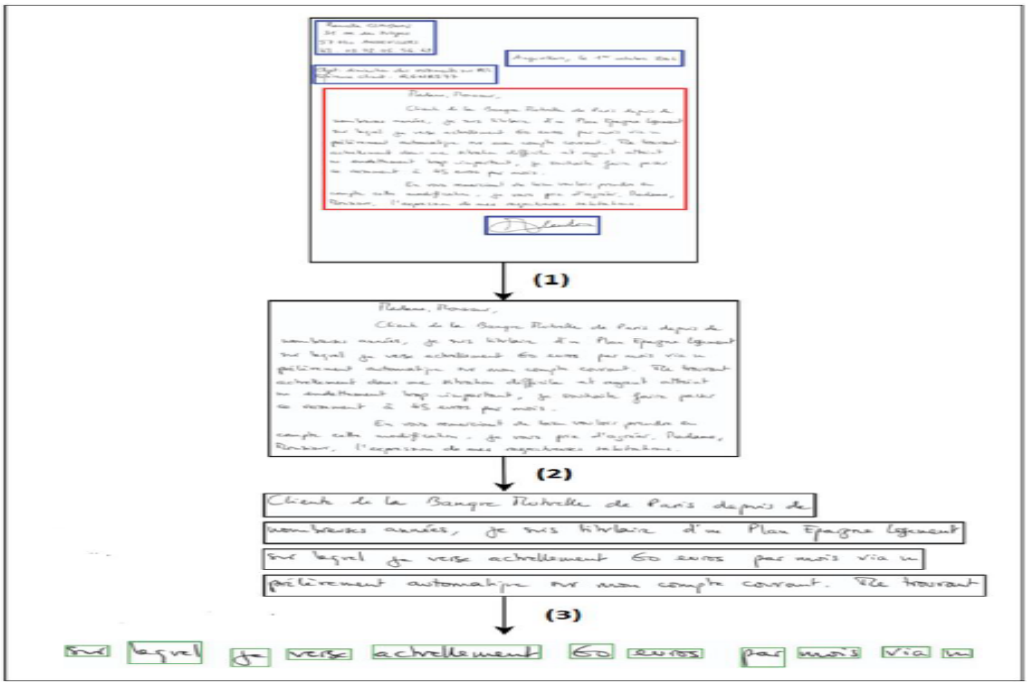


FIGURE 1.6 – Différents niveaux de segmentation

1.6.4 Extraction de caractéristiques

L'extraction de caractéristiques est l'une des phases les plus importantes du système, elle décrit diverses caractéristiques des éléments segmentés. La reconnaissance d'un caractère passe d'abord par l'analyse de sa forme et l'extraction de ses traits primitifs qui seront exploités pour son identification.

On distingue quatre approches essentielles pour classer les types de caractéristique :

1.6.4.1 Caractéristiques structurelles

Elles décrivent une forme en termes de sa topologie et de sa géométrie, en fournissant des informations sur ses propriétés globales et locales [31].

Parmi ces caractéristiques, nous trouvons les voyellations, la hauteur et la largeur du caractère, le nombre de points diacritiques, la position par rapport à la ligne de base, etc.

1.6.4.2 Caractéristiques statistiques

Elles fournissent des mesures quantitatives sur une forme, et offrent des informations locales sur son contenu, elles incluent : le profil de projection des densités de pixels, l'histogramme directionnel, l'histogramme des transitions, les directions des contours dans une fenêtre locale et les moments invariants [19].

Ces caractéristiques sont essentielles car elles sont invariantes en translation, taille et rotation, fournissant ainsi une représentation robuste de la forme pour la reconnaissance [19].

1.6.4.3 Transformations globales

Elle consiste à convertir la représentation basée sur les pixels en une représentation plus abstraite, et permet de réduire la dimension des caractères tout en préservant un maximum d'informations sur la forme à reconnaître [4].

1.6.4.4 Superposition des modèles (template matching)

La méthode de template matching est applicable aux images binaires en niveaux de gris, elle se base sur l'utilisation de l'image de la forme comme vecteur de caractéristiques, pour être ensuite comparée à un modèle pixel par pixel durant la phase de reconnaissance.

À la suite de cette comparaison, une mesure de similarité est calculée pour évaluer la proximité entre l'image analysée et le modèle de référence, il permet de déterminer le degré de correspondance entre les deux.

1.6.5 Classification

La classification dans un système OCR comprend deux tâches importantes : l'apprentissage et la reconnaissance et la prise de décision. Lors de cette phase, on utilise les caractéristiques de l'étape précédente pour identifier un segment de texte et le relier à un modèle de référence.

1.6.5.1 Apprentissage

L'apprentissage est la capacité du système à apprendre à partir d'exemples, afin d'identifier et de classifier efficacement de nouveaux caractères ou mots, cette tâche implique l'utilisation d'algorithmes de machine learning (ML), qui analysent les caractéristiques extraites des caractères ou des mots, pour créer un modèle de classification, qui sera ensuite utilisé pour attribuer une catégorie aux caractères ou mots analysés, cela permet de faciliter leur reconnaissance dans de nouvelles images.

On distingue quatres types d'apprentissage :

L'apprentissage supervisé : consiste à utiliser un ensemble de données étiquetées, pour apprendre aux modèles à faire des prédictions, permettant de classer les données ou de prédire des résultats avec précision.

L'apprentissage non supervisé : est utilisé lorsque l'information servant à entraîner un modèle n'est ni classifiée ni étiquetée, il faut la modéliser pour la comprendre.

L'apprentissage par renforcement : est une méthode d'optimisation itérative, d'un algorithme basée uniquement sur ses actions, et la réponse de l'environnement associé dans lequel il évolue.

L'apprentissage semi-supervisé : est une approche qui permet à une machine de résoudre un problème, en lui donnant à la fois des données étiquetées et non étiquetées pour l'entraînement.

1.6.5.2 Reconnaissance et décision

La reconnaissance est le processus par lequel une forme ou un caractère est identifié, et classifié en fonction de son modèle discriminant, elle implique le calcul des probabilités d'émission de la forme par différents modèles supposés a priori équiprobables, la forme à reconnaître est ensuite affectée à la classe dont le modèle fournit la probabilité la plus élevée, qui permet de déterminer le caractère ou la forme présente dans l'image avec précision.

Les approches de reconnaissance peuvent être regroupées en quatre groupes à savoir :

Approche statistique : basée sur l'analyse statistique des caractéristiques des formes à identifier, cette analyse caractérise les classes en utilisant des méthodes basées sur des lois de probabilité, comme la théorie de la décision bayésienne et les méthodes de classification non supervisées.

Approche structurelle : elle se base sur la configuration physique des caractères, elle vise à identifier les composants élémentaires ou fondamentaux et à détailler leurs interactions, ces composants peuvent être des caractéristiques topologiques comme une boucle, un arc, etc [40].

Approche stochastique : elle utilise un modèle pour la reconnaissance, prenant en compte la grande variabilité de la forme. La distance utilisée dans les techniques de comparaison est remplacée par des probabilités calculées de manière plus fine par apprentissage [40].

Approche hybride : est une méthode qui combine entre l'approche statistique et structurelle, son objectif est d'améliorer les performances de reconnaissance, en opérant sur deux niveaux distincts, le premier extrait et reconnaît les primitives des formes et le second utilise les caractéristiques des formes pour reconstruire des graphes pour chaque forme [5].

1.6.6 Post traitement

Afin d'améliorer le taux de reconnaissance, une étape de vérification des résultats est recommandée, appelée le post traitement. Cette phase est facultative et souvent implémentée sous forme d'un ensemble d'outils liés à la fréquence des caractères dans une chaîne, au vocabulaire et à d'autres informations.

1.7 Méthodes de classification

Voici les classifieurs les plus couramment utilisés en reconnaissance de l'écriture :

L'approche bayésienne : elle consiste à sélectionner parmi un ensemble de caractères celui pour lequel la série des primitives extraites a la plus haute probabilité a posteriori par rapport aux caractères précédemment appris [17].

Méthode des k plus proches voisins (KNN) : l'algorithme compare la forme inconnue avec les formes stockées dans une classe de référence appelée prototype et attribue la classe à la plus proche [17].

Réseaux de neurones (NN) : ils sont composés d'éléments connectés simples ou de neurones, ces éléments sont fortement inspirés du système nerveux biologique [17].

Machines à vecteurs de support (SVM) : les SVM sont un groupe de méthodes d'apprentissage supervisé, la classification utilise des ensembles de données d'entraînement et de test. Le classificateur SVM standard produit un modèle basé sur les données d'entraînement puis prend l'ensemble de données de test et prédit de les classer dans l'une des deux classes distinctes uniquement [17].

Méthodes de test : elles consistent à appliquer des tests sur chaque caractère, concernant la présence ou l'absence d'éléments ou de primitives simples, pour déterminer sa classe [17].

L'approche syntaxique : chaque caractère est représenté par une phrase dans un langage ou le vocabulaire est constitué de primitive, les formes qui appartiennent à la même classe présente une structure commune qui peut être représentées par une même grammaire [5].

Comparaison de lignes : les caractères sont représentés par des chaînes de primitives, la comparaison du caractère traité avec le modèle de référence, consiste à mesurer la ressemblance entre les deux chaînes et à se prononcer sur celui-ci. Cette mesure peut se faire par le calcul de distance ou par examen de l'inclusion de toute ou une partie d'une chaîne dans l'autre [17].

Modèle de Markov caché (HMM) : une méthode probabiliste dont le modèle est composé d'un ensemble d'états, de probabilités de transition entre ces états et d'observations faites par le système sur une image. Ces observations sont représentées par des variables aléatoires, dont la distribution dépend de l'état [17].

1.8 Revue de la littérature sur les systèmes OCR

Voici une Revue de la littérature sur les systèmes OCR, résumant les travaux réalisés dans quelques articles et thèses :

K. Karthick, K. B. Ravindrakumar, R. Francis et S. Ilankannan (2019) [28] examinent les étapes de la reconnaissance de texte, en mettant l'accent sur les avancées récentes en OCR pour les langues arabes et chinoises manuscrites hors ligne.

B. Kiessling (2021) [30] présente des recherches sur les caractéristiques des textes arabes, notamment la transcription de textes dans des documents historiques manuscrits et imprimés par machine. Il a développé un système pour l'OCR multilingue, faisant référence à une technologie de l'OCR capable de reconnaître et de traiter plusieurs scripts ou langues à la fois.

Yingying Zhu, Cong Yao et Xiang Bai (2014) [51] entreprennent la détection et la reconnaissance de texte dans des scènes naturelles pour la langue anglaise.

H. Hu, Q. Wang, K. Huang, M. Wen et F. Coenen (2022) [24] décrivent l'utilisation des modèles de langage basés sur la récupération pour la reconnaissance hors ligne de textes chinois manuscrits.

Après lecture de ces articles, il ressort que certaines méthodes représentent une avancée notable dans le domaine de la reconnaissance de texte. De plus, les méthodes actuelles de reconnaissance de texte pour les langues non latines se heurtent toujours à des obstacles liés à la complexité des caractères, aux variations de style d'écriture et aux caractéristiques complexes.

1.9 Défis de l'extraction du texte à partir des images

La complexité des environnements, les modes d'acquisition d'images et la variation du contenu posent différents défis lors de l'extraction du texte à partir des images capturées. En effet, les documents texte capturés par des caméras, posent des problèmes photométriques et géométriques [1].

Les problèmes photométriques incluent les variations des conditions d'éclairage, qui peuvent altérer la lisibilité et la clarté du texte. Par exemple, un éclairage insuffisant ou trop intense peuvent compliquer l'analyse et la reconnaissance du texte dans cette image.

Le mode d'acquisition des images joue aussi un rôle, une caméra fixe dans des bonnes conditions donne une qualité d'image différente par rapport à une caméra mobile qui peuvent causer une dégradation de la qualité de l'image, ce qui influence sur la netteté et la fidélité des documents capturés.

Les défis géométriques sont associés à la forme des images textuelles, les documents capturés peuvent présenter des distorsions causées par l'angle de prise de vue ou la courbure du support. Par exemple, un livre ouvert ou une feuille pliée peut produire des lignes de texte courbes ou inclinées, rendant difficile la reconnaissance automatique du texte.

Voici une figure qui montre quelques problèmes rencontrés pour les documents texte capturés par des caméras [1] :

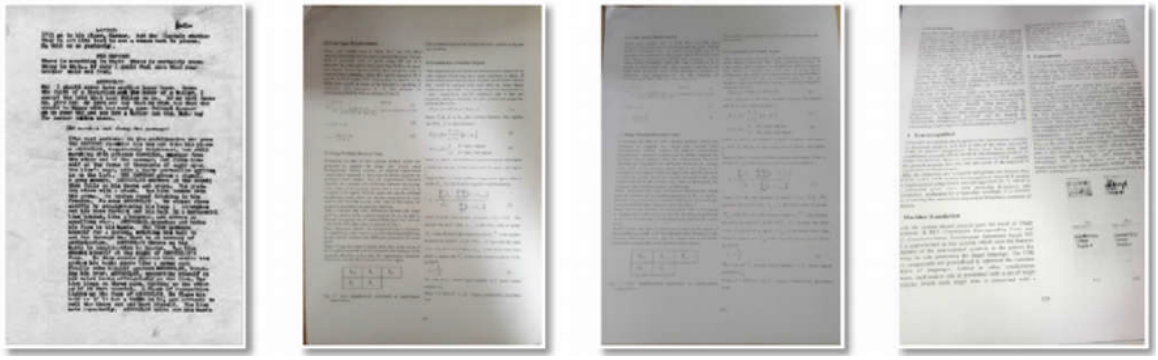


FIGURE 1.7 – Problèmes rencontrés pour les documents texte capturés par des caméras

Conclusion

Dans ce chapitre, nous avons introduit succinctement le domaine du traitement d'images numériques, et exposé les caractéristiques inhérentes à une image numérique ainsi que les diverses applications de ce champ. Ensuite, nous avons procédé à une définition du système OCR, cela en détaillant son architecture qui se compose de plusieurs étapes : acquisition, prétraitement, segmentation, extraction des caractéristiques, classification et post-traitement, tout en examinant les approches pertinentes pour chacune de ces phases. Enfin, une revue de la littérature a été entreprise abordant quelques articles préexistants portant sur l'OCR, ainsi que quelques défis de l'extraction du texte à partir des images sont présentés.

Dans le prochain chapitre, nous présenterons la méthode que nous avons choisie pour la phase de classification.

2

Réseaux de Neurones Artificiels

L'apprentissage automatique, également appelé machine learning (ML) en anglais, est une technologie de l'intelligence artificielle (IA), qui permet aux ordinateurs d'apprendre de se développer, et de s'améliorer de façon autonome à partir d'un ensemble de données, sans avoir à être programmés explicitement.

Cependant, lorsque on dispose d'un grand ensemble de données, la reconnaissance optique de caractères devient une tâche complexe, elle nécessite une approche plus robuste que l'utilisation exclusive des algorithmes de machine learning. Plusieurs approches ont été mises en œuvre pour résoudre ce problème, comme l'utilisation des réseaux de neurones.

Dans ce chapitre, nous présenterons un aperçu sur les réseaux de neurones. Nous commencerons par définir le domaine du deep learning et décrire son importance. Ensuite, nous aborderons les réseaux de neurones artificiels et biologiques. Nous expliquerons le perceptron et le perceptron multicouche. Enfin, nous explorerons quelques types de réseaux de neurones, notamment les réseaux de neurones convolutifs, les réseaux de neurones récurrents et les réseaux de longue mémoire à court terme.

Sommaire

2.1 Deep learning

Le deep learning (DL) ou apprentissage profond, est un type d'intelligence artificielle dérivé du machine learning, où la machine est capable d'apprendre par elle-même. Contrairement à la programmation, où elle se contente d'exécuter à la lettre des règles prédéterminées [50].

Les applications basées sur l'apprentissage profond sont capables d'apprendre sans intervention humaine, en s'appuyant sur des données à la fois non structurées et non étiquetées [14].

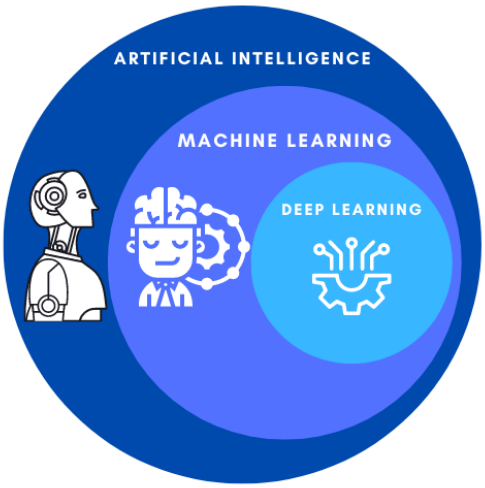


FIGURE 2.1 – Relation entre l'intelligence artificielle, machine learning et deep learning

Cette approche a été très réussie dans divers domaines, tels que :

- Reconnaissance d'image.
- Reconnaissance vocale.
- Cyber sécurité.
- Conduite autonome.
- Rédaction de textes.
- Traitement automatique des langues.
- Vision par ordinateur.

2.2 Importance de deep learning

Le deep learning est une approche essentielle de l'IA. Plus les recherches dans ce domaine progressent, plus le deep learning devient important et indispensable :

- Le deep learning fonctionne avec des données structurées et non structurées.
- Les algorithmes de deep learning peuvent effectuer des opérations complexes d'une façon efficace.
- Il reçoit une grande quantité de données en entrée et analyse ces derniers afin d'extraire les caractéristiques d'un objet.
- Il permet de créer des modèles capables d'apprendre des représentations hiérarchiques et abstraites des données.
- Les modèles de deep learning ont tendance à bien fonctionner avec une grande quantité de données, ce qui en fait la solution idéale pour maintenir les performances du modèle. En revanche, les modèles de machine learning plus classiques montrent des signes de saturation

une fois qu'ils atteignent un certain point, ce qui se traduit par des performances de plus en plus inefficaces.

2.3 Réseaux de neurones artificiels - RNA

Le deep Learning s'appuie sur un réseau de neurones artificiels (RNA) s'inspirant du cerveau humain. En 1943, le premier modèle de «neurone formel » a été créé, proposé par Warren McCulloch et Walter Pitts. Il s'agit d'un neurone binaire, constituant ainsi la première représentation informatique de cerveau humain. Puis, en 1957, c'est avec le «perceptron» que l'on commence à parler de « réseau de neurones artificiels ».

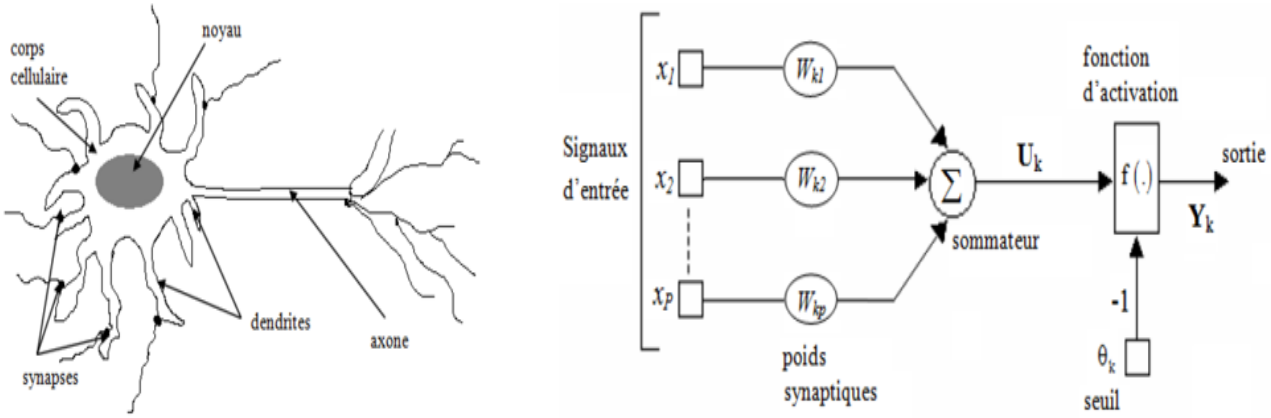


FIGURE 2.2 – Schéma d'un neurone biologique vs le modèle de neurone formel

Un réseau neuronal artificiel est un groupe d'algorithmes et de méthodes mises en places pour tenter d'approcher le fonctionnement du cerveau humain par le biais d'unités de calculs, tentant de s'approcher le plus possible des neurones biologiques.

Le neurone formel rappelle beaucoup le neurone biologique, c'est pour cette raison que le vocabulaire utilisé dans la littérature pour décrire un neurone formel est largement emprunté à la biologie [19].

Nous constatons quatre éléments de base :

- Un ensemble des caractéristiques qui sont introduites dans le modèle pour le processus d'apprentissage.
- Un ensemble de synapses ou connexions, chaque connexion est caractérisée par un poids W_{ij}.
- Un sommateur qui réalise une sommation des signaux d'entrée pondérés par les poids synaptiques relatifs. L'opération constitue une combinaison linéaire des signaux d'entrée :

$$U_k = \sum_{j=1}^{p} W_{kj} x_j$$

- Une fonction d'activation qui limite l'amplitude de la sortie du neurone, de cette façon l'amplitude de la sortie sera normalisée.

2.3.1 Fonctions d'activation

Dans un réseau de neurones, les entrées sont introduites dans les neurones du réseau. Chaque neurone possède un poids, les entrées qui sont des valeurs réelles, sont multipliées par ces poids, puis

ils seront passées à travers une fonction d'activation.

Une fonction d'activation est une équation mathématique qui détermine la sortie de chaque élément dans le réseau neuronal. Elle permet au modèle de représenter et d'apprendre des relations complexes dans les données, d'où la nécessité d'être non linéaire, sinon elle ne serait qu'une combinaison linéaire de ses entrées, ce qui limite sa capacité à modéliser des données complexes.

On distingue différents types de fonctions d'activation, en voici quelques-unes présentées dans cette figure :

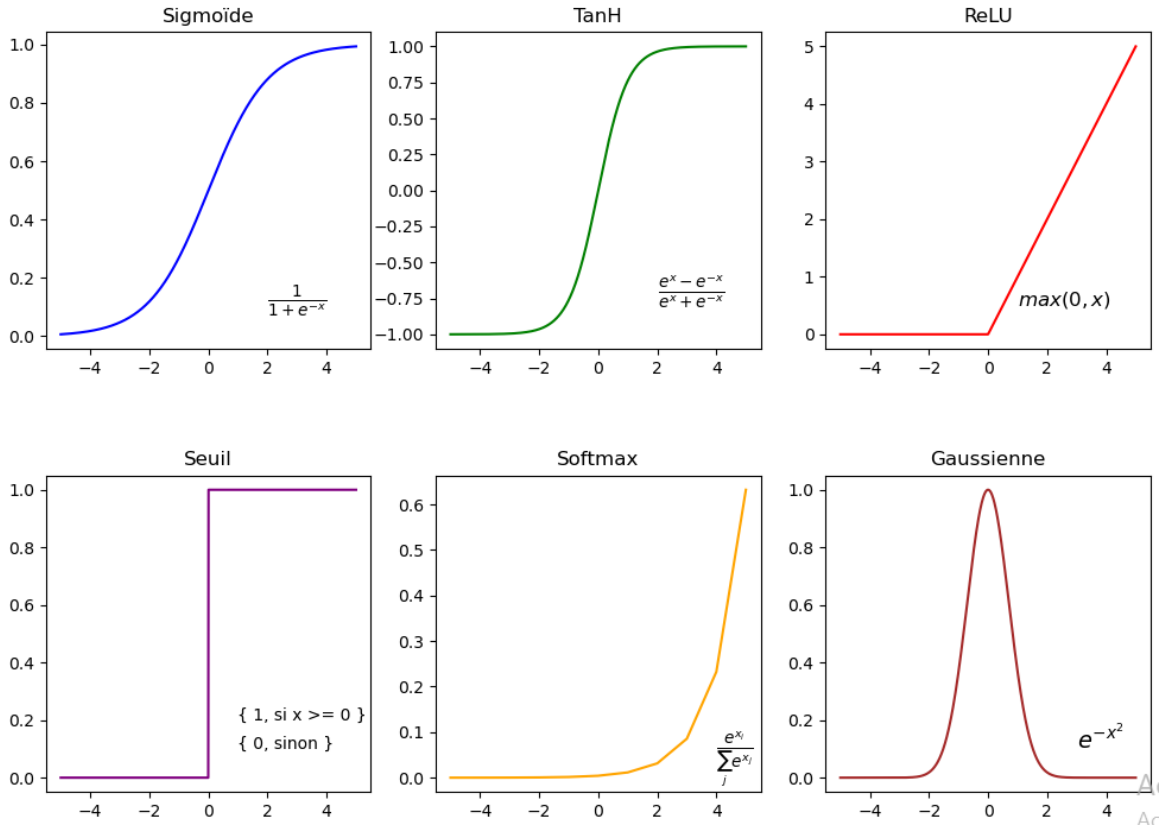


FIGURE 2.3 – Fonctions d'activation

Voici le rôle de quelques-unes des fonctions d'activation :

Sigmoïde : elle est utile pour les problèmes de classification binaire, elle transforme les valeurs en une plage comprise entre zéro et un.

Tangente Hyperbolique (Tanh) : elle est similaire à la sigmoïde mais elle génère des valeurs comprises entre -1 et 1.

Rectified Linear Unit (ReLU) : l'une des fonctions les plus populaires, elle laisse les valeurs positives inchangées, et transforme toutes les valeurs négatives en zéro.

Seuil : elle attribue une valeur de sortie de 1 si l'entrée est supérieure à la valeur de seuil, sinon 0. Elle est aussi utilisée dans les problèmes de classification binaire.

Softmax : elle normalise les sorties pour chaque classe entre 0 et 1, où la somme de toutes les valeurs est égale à 1. Puis, elle renvoie la probabilité que l'entrée appartient à une classe. Elle

est utile pour les problèmes de classification multiclasse.

2.4 Réseaux de neurones multicouches

2.4.1 Perceptron

Le perceptron est l'un des modèles de réseaux de neurones, développé par Frank Rosenblatt en 1958, et représente le réseau neuronal le plus simple. Il est conçu pour effectuer la classification dans un problème à deux classes seulement. Il consiste en un seul neurone qui possède un seuil ainsi qu'un vecteur de poids synaptiques ajustables [19].

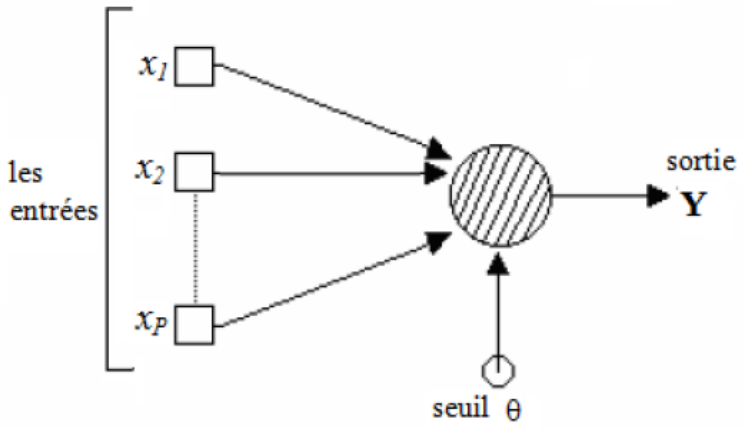


FIGURE 2.4 – Perceptron

2.4.2 Perceptrons multicouches

Un perceptron multicouche, souvent abrégé en MLP (Multi-Layer Perceptron), est une structure composée de plusieurs couches de neurones. Ils sont donc une extension des perceptrons simples, mais avec une architecture en couches.

Les perceptrons multicouches ont une architecture de base car chaque neurone d'une couche est liée à tous les neurones de la couche suivante, mais n'a aucun lien avec les neurones de la même couche [7].

Un MLP est constitué d'une couche d'entrée, une couche de sortie et une ou plusieurs couches cachées, comme le démontre cette figure :

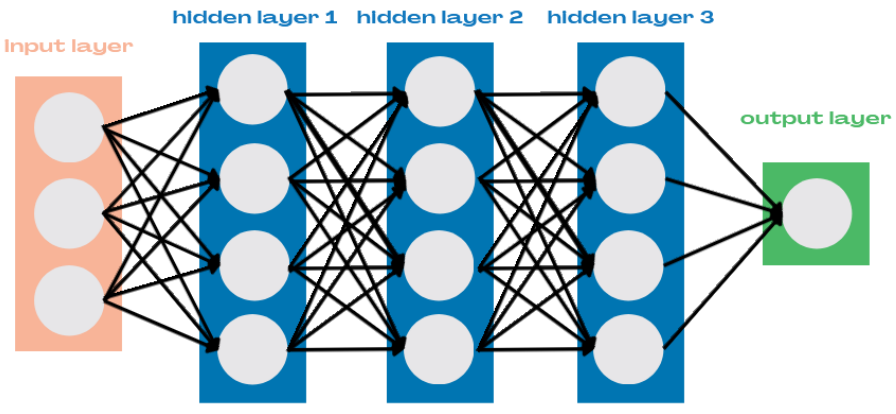


FIGURE 2.5 – Perceptron multicouche

D'où :

La couche d'entrée (input layer) : prend en compte les données d'entrée et les transmet à la première couche cachée.

Les couches cachées (hidden layers) : réalisent des calculs mathématiques sur les entrées. L'un des enjeux de la création de réseaux de neurones réside dans le choix du nombre de couches cachées et du nombre de neurones pour chaque couche.

La couche de sortie (output layer) : génère les données de sortie, nous pouvons appliquer différentes fonctions d'activation aux couches cachées selon le type de problèmes à résoudre.

Par exemple, dans le cas de régression, nous n'appliquons aucune fonction d'activation sur la couche de sortie. Pour la classification binaire, la sortie donne une prédiction de P(Y = 1/X), la fonction d'activation sigmoïde est généralement utilisée. Pour la classification multiclasses, la couche de sortie contient un neurone par classe i, donnant une prédiction de P(Y = i/X). La somme de toutes ces valeurs doit être égale à 1. La fonction multidimensionnelle softmax est généralement utilisée [7].

Les MLP ont été utilisés pour résoudre des problèmes difficiles de classification par apprentissage supervisé à partir d'un algorithme connu sous le nom d'algorithme de rétropropagation du gradient qui se base sur la règle d'apprentissage par correction d'erreur [19].

2.4.3 Types de réseaux de neurones

Les RNA sont utilisés pour des tâches variées telles que la classification, la reconnaissance d'image et la prédiction. Il existe plusieurs types de réseaux de neurones, chacun ayant une architecture et des applications spécifiques.

Voici quelques-uns des types de réseaux de neurones les plus couramment utilisés :

- Réseaux de Neurones Feedforward.
- Réseaux de Neurones Récurrents.
- Réseaux de Neurones Convolutifs.
- Long Short-Term Memory.
- Réseaux de Neurones Génératifs Adversaires.
- Réseaux de Neurones Transformateurs.
- Réseaux de Neurones à Convolution Tridimensionnelle.

Dans ce qui suit, nous allons vous présenter les types de réseaux de neurones que nous allons utiliser afin de modéliser notre système OCR.

2.5 Réseaux de neurones convolutifs (CNN)

2.5.1 Introduction

L'une des capacités essentielles de l'humain est d'analyser son environnement via la reconnaissance visuelle. Cependant, cette tâche était très difficile pour les ordinateurs jusqu'en 2012, lorsque Alex Krizhevsky a développé les réseaux de neurones convolutifs [44].

Les réseaux de neurones convolutifs ont révolutionné de nombreux domaines comme l'imagerie médicale, la reconnaissance faciale et la reconnaissance de textes. Dans le domaine du l'OCR, les réseaux de neurones convolutifs ont révolutionné la manière dont les machines peuvent interpréter et transcrire les documents écrits.

Traditionnellement, la reconnaissance de caractères était un défi majeur pour les systèmes informatiques, qui exige des algorithmes complexes et spécifiques à chaque langue ou style d'écriture. Avec les réseaux de neurones convolutifs, il est devenu possible de développer des modèles plus robustes et polyvalents capables de s'adapter à diverses polices et configurations textuelles.

2.5.2 Définition

Les réseaux de neurones convolutifs (en anglais Convolutional Neural Networks - CNN), également connu sous le nom de ConvNets, est une sous-catégorie de réseaux de neurones artificiels, conçu pour traiter des données structurées, comme les images. Ils présentent les modèles les plus performants pour la reconnaissance des images.

Convolution signifie une opération mathématique qui combine deux fonctions, pour produire une autre fonction qui exprime le changement en raison de l'opération effectuée [39].

Le CNN compare les images fragment par fragment. Ces fragments sont appelés caractéristiques, sont les éléments qu'il recherche [48].

Leur fonctionnement est simple, l'utilisateur fournit une image sous forme de matrice de pixels, puis celle-ci dispose de 3 dimensions : deux dimensions pour une image en niveaux de gris, une troisième dimension de profondeur 3 pour représenter les couleurs fondamentales RVB (Rouge, Vert, Bleu) [48].

2.5.3 Architecture d'un réseau de neurones convolutif

L'architecture d'un réseau de neurones convolutif se compose de plusieurs couches : des couches convolutives (Convolutional Layers), des couches de mutualisation (Pooling Layers), des couches de Rectification (ReLU Layers) et des couches entièrement connectées (Fully Connected Layers).

La Figure suivante montre la structure générale d'un réseau CNN.

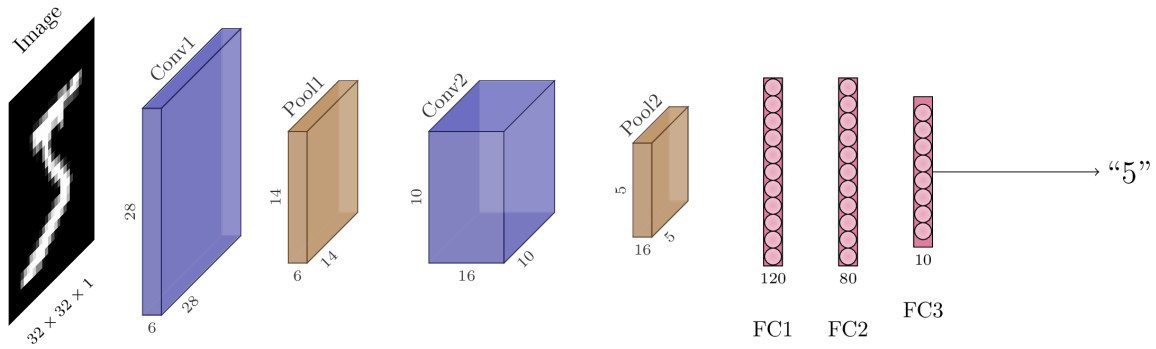


FIGURE 2.6 – Schéma de l'architecture d'un CNN

2.5.3.1 Couche de convolution (CONV)

La convolution est la première couche à extraire des entités d'une image d'entrée, elle préserve la relation entre les pixels en apprenant les caractéristiques de l'image à l'aide de petits carrés de données

d'entrée [14].

La convolution est une opération mathématique simple généralement utilisée pour le traitement et la reconnaissance d'images [48]. Elle est composée de deux éléments principaux : les filtres (ou noyaux) et l'opération de convolution.

Filtres (noyaux) : les couches convolutives appliquent des filtres sur les données d'entrée, chaque filtre détecte des caractéristiques spécifiques comme les bords, les textures, etc.

Convolution : le filtre parcourt l'image et effectue une multiplication point par point suivie d'une somme pour produire une nouvelle matrice appelée « feature map ».

D'abord, la taille de la fenêtre qui va se déplacer à travers toute l'image sera définie. Ensuite, cette fenêtre sera positionnée en haut à gauche de l'image, elle se déplacera d'un certain nombre de cases vers la droite, ce que l'on appelle le pas. Lorsqu'elle atteindra le bord de l'image, elle se déplacera d'un pas vers le bas et continuera ainsi jusqu'à ce que le filtre ait parcouru la totalité de l'image [12], comme le montre cette figure :

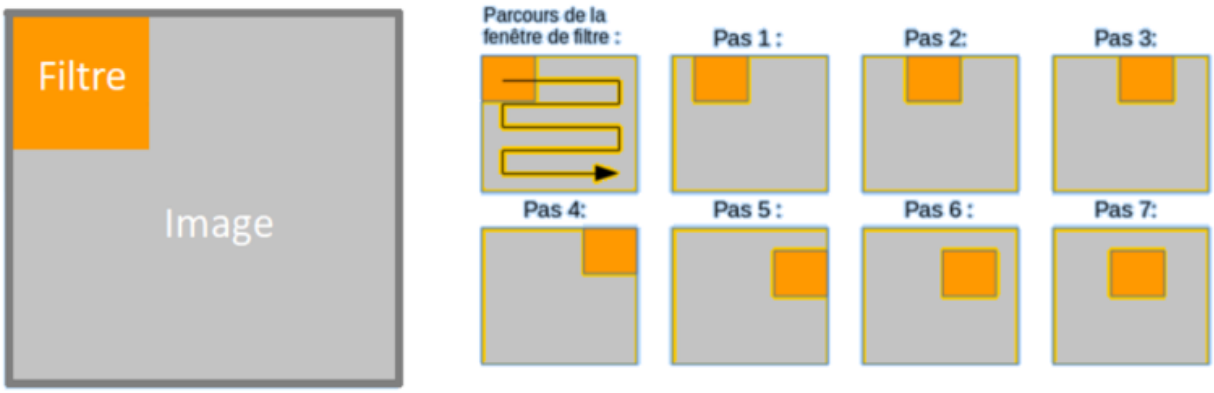


FIGURE 2.7 – Schéma du parcours de la fenêtre de filtre sur l'image

2.5.3.2 Couche de mutualisation (Pooling)

La couche de pooling, connu sous le nom de sous-échantillonnage, elle est généralement placé entre deux couches de convolution, elle reçoit en entrée plusieurs feature maps, et applique à chacune d'entre elles l'opération de pooling [12].

L'opération de pooling permet de réduire la taille des images, tout en gardant les caractéristiques importantes. Pour cela, il existe deux grands types de pooling :

Max pooling : c'est une technique qui consiste à réduire la dimension des feature maps en ne conservant que la valeur maximale dans chaque région d'une taille prédéfinie.

Average pooling : la moyenne des valeurs dans chaque région est prise. Le pooling aide à réduire la taille des données, à diminuer le nombre de paramètres et à contrôler le surapprentissage (overfitting).

Voici une figure qui explique comment trouver la valeur de chaque région dans les deux types de pooling [37] :

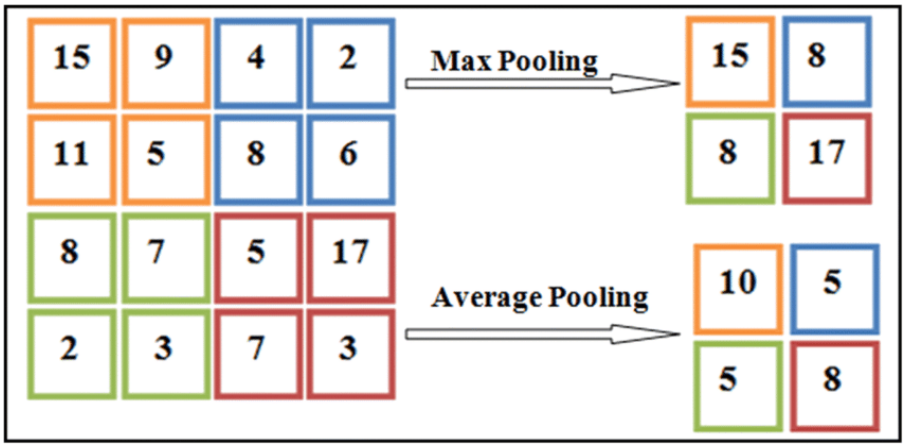


FIGURE 2.8 – Max pooling et average pooling

L'image est découpée en cellules régulières, et au sein de chaque cellule, on conserve une valeur selon le type de pooling choisi. Pour ne pas perdre trop d'informations, des cellules carrées de petite taille sont utilisées, il est conseillé d'utiliser des cellules adjacentes de 2 x 2 pixels qui ne se chevauchent pas, ou des cellules de 3 x 3 pixels, distantes les unes des autres d'un pas de 2 pixels, mais entraîne un chevauchement [12].

En sortie, on obtient le même nombre de feature maps qu'en entrée, mais avec une réduction significative de leur dimension [12].

Bien que de nombreuses informations soient perdues dans la couche de mise en commun, cela présente également un certain nombre d'avantages pour le CNN. Ils contribuent à réduire la complexité, à améliorer l'efficacité et à limiter le risque de surajustement [48].

Définition* 2.5.1.** ***Le surajustement (overfitting) : *est un comportement indésirable d'apprentissage automatique qui se produit lorsque le modèle fournit des prédictions précises pour les données d'entraînement mais pas pour les nouvelles données.*

2.5.3.3 Couche de correction ReLU (REctified Linear Unit)

ReLU est une fonction d'activation très utilisée dans les CNN. Elle est souvent appliquée en sortie d'une couche de convolution. ReLU introduit la non-linéarité en remplaçant toutes les valeurs négatives en entrée par des zéros.

Il existe d'autres fonctions d'activation qui peuvent aussi être utilisées dans les couches convolutives, comme la fonction Sigmoïde et la Tanh, mais elles sont moins courantes que ReLU.

ReLU est définie par **ReLU(x) = max(0, x)**. Voici sa représentation graphique :

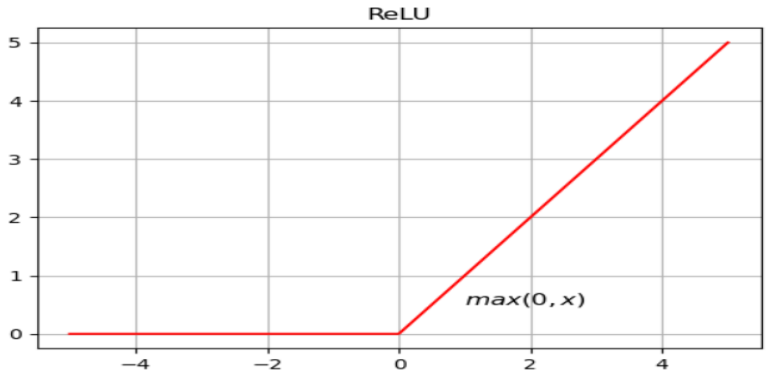


FIGURE 2.9 – Représentation de la fonction ReLU

2.5.3.4 Couche entièrement connectée (Fully-Connected - FC)

Après plusieurs couches convolutives et de pooling, les données sont aplaties en un vecteur et passent par une ou plusieurs couches entièrement connectées. La couche entièrement connectée, également appelée couche dense ou fully connected (FC) en anglais, est souvent la couche finale dans une architecture CNN.

La couche FC est similaire au réseau entièrement connecté dans les modèles conventionnels [14], les neurones dans cette couche ont des connexions vers toutes les sorties de la couche précédente [44].

Elle permet de classifier l'image en entrée du réseau et renvoie un vecteur de taille N, où N est le nombre de classes dans un problème de classification d'images. Chaque élément du vecteur indique la probabilité pour l'image en entrée d'appartenir à une classe [12].

Afin de calculer cette probabilité, la couche FC multiplie chaque élément par un poids, ensuite il fait la somme, puis il applique la fonction d'activation softmax si $N > 2$.

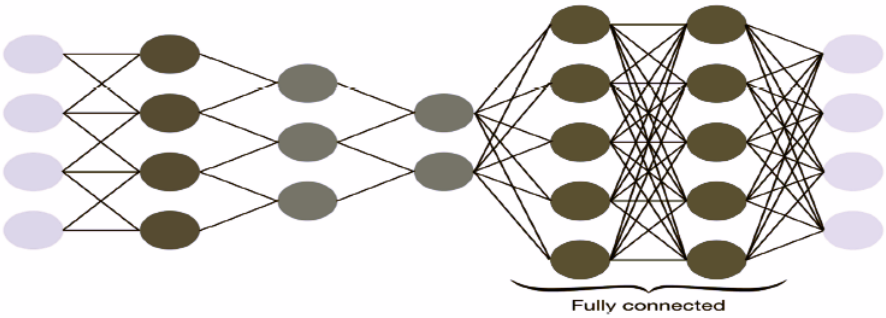


FIGURE 2.10 – Représentation de la couche fully connected

Voici une liste de certains des réseaux convolutifs les plus célèbres :

- **LeNet :** est une des premières architectures de CNN développée, proposée par Yann LeCun en 1998, elle se compose de 7 couches et utilisée dans de nombreuses tâches de vision par ordinateur, y compris la classification d'images, la détection d'objets et la reconnaissance de chiffres manuscrits.

- **AlexNet :** proposé par Alex Krizhevsky, Ilya Sutskever, et Geoffrey Hinton en 2012, composé de 8 couches. Le réseau avait une architecture très similaire à LeNet, mais était plus profond,

plus grand et comportait des couches convolutives empilées les unes sur les autres [44].

- **VGGNet :** construit par Visual Geometry Group (VGG) à l'université d'Oxford en 2014, composé de 16 (VGG-16) ou 19 (VGG-19) couches.

- **ResNet :** residual network développé par Kaiming He et al. A été le vainqueur de ILSVRC en 2015 [44]. Introduit les connexions résiduelles qui facilitent l'entraînement de réseaux très profonds, pouvant être composés de jusqu'à 152 couches.

- **DenseNet :** introduit par Gao Huang et al en 2016, il utilise des connections denses où chaque couche reçoit les entrées de toutes les couches précédentes, favorise la réutilisation des caractéristiques et améliore le flux d'information à travers le réseau.

- **Etc**.

2.6 Réseaux de neurones récurrents (RNN)

2.6.1 Introduction

Les réseaux profonds et MLP ne sont pas adaptés au traitement des problèmes séquentiels, car ils n'ont pas de mémoire et ne prennent pas en compte de contexte qu'il soit temporel ou spatial [45]. Les données séquentielles ne possèdent pas de longueur fixe, ce qui complique la tâche à ces réseaux, car ils sont généralement conçus pour manipuler des données de taille constante. D'où le besoin de créer des réseaux de neurones récurrents, qui résout ce problème.

2.6.2 Définition

Les réseaux de neurones récurrents, ou Recurrent Neural Networks en anglais (RNN), sont une classe de réseaux de neurones utilisés pour modéliser et traiter des séquences de données. Ils sont dérivés des réseaux de neurones traditionnels (feedforward).

Dans les réseaux neuronaux traditionnels, toutes les entrées et sorties sont indépendantes les unes des autres. Cependant, dans des cas comme la prédiction du mot suivant dans une phrase, les mots précédents sont nécessaires et doivent donc être mémorisés [7].

C'est là qu'interviennent les RNN, qui ont résolu ce problème grâce à une couche cachée. La caractéristique primordiale des RNN est l'état caché, qui mémorise certaines informations spécifiques sur une séquence [7].

Ils sont appelés récurrents car ils effectuent la même tâche pour chaque élément d'une séquence, la sortie dépendant des calculs précédents. Les RNN ont une mémoire qui capture des informations sur ce qui a été calculé jusqu'à présent [44].

Cela les rend particulièrement utiles pour des tâches telles que la reconnaissance de la parole, la reconnaissance de textes manuscrits, la description des images, la traduction automatique, et la modélisation de séries temporelles.

Voici une figure qui explique la différence entre un RNN et un réseau neuronal traditionnel :

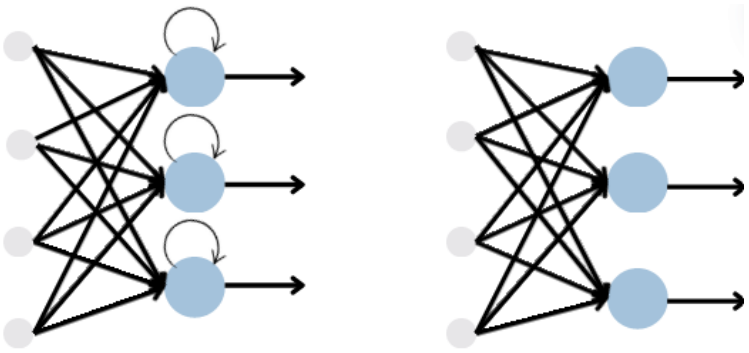


FIGURE 2.11 – Recurrent neural network vs feed-forward neural network

2.7 Réseaux de neurones à mémoire court terme et long terme (LSTM)

2.7.1 Définition

Les réseaux à mémoire à long et court terme, long short-term memory networks en anglais (LSTM), est un type spécial de RNN, introduit en 1997 par Hochreiter et Schmidhuber.

Les LSTM ont été introduits avec un double objectif : d'une part, pour résoudre le problème de disparition du gradient lors de l'apprentissage d'un réseau récurrent, d'autre part, pour permettre de garder la mémoire sur le long terme [48].

2.7.2 Structure des LSTM

Le modèle LSTM est constitué de différents blocs de mémoire appelés cellules. Chaque cellule LSTM contient plusieurs composants essentiels :

2.7.2.1 États de la cellule et caché

L'état de la cellule C_t stocke les informations à long terme. L'état caché h_t est utilisé pour produire la sortie à chaque étape temporelle et pour les calculs internes.

2.7.2.2 Trois portes

Porte d'oubli f_t : détermine quelles informations de l'état de la cellule précédente C_{t-1} doivent être oubliées. Elle prend en entrée l'état caché précédent h_{t-1} et l'entrée actuelle X_t, et produit un vecteur de valeurs entre 0 et 1 (0 signifie oublier et 1 signifie garder). Sa formule est :

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{2.1}$$

où :

- f_t est le vecteur de la porte d'oubli à l'instant t.
- σ est la fonction sigmoïde.
- W_f est la matrice de poids pour la porte d'oubli.
- h_{t-1} est l'état caché de la couche précédente à l'instant $t-1$.

- x_t est l'entrée à l'instant t.
- b_f est le biais pour la porte d'oubli.

Remarque 2.7.1. *Les descriptions ci-dessus s'appliquent de manière similaire aux portes d'entrée et de sortie, en remplaçant f_t, W_f et b_f respectivement par les éléments correspondants des portes d'entrée et de sortie.*

Porte d'entrée i_t : contrôle quelles nouvelles informations doivent être ajoutées à l'état de la cellule. Elle utilise l'état caché précédent et l'entrée actuelle. Sa formule est :

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{2.2}$$

Porte de sortie o_t : décide quelles informations de l'état de la cellule doivent être utilisées pour produire la sortie actuelle. Sa formule est :

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{2.3}$$

2.7.2.3 Nouveaux contenus candidats ($\tilde{C}_t$)

Ils représentent les nouvelles informations candidates pour être ajoutées à l'état de la cellule. Ce vecteur est calculé en utilisant la fonction d'activation tangente hyperbolique (tanh) appliquée à une combinaison linéaire de l'état caché précédent h_{t-1} et de l'entrée actuelle x_t.

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \tag{2.4}$$

2.7.3 Fonctionnement des LSTM

La construction d'un réseau LSTM commence par l'identification et l'exclusion des informations inutiles, décidées par la fonction sigmoïde. Cette fonction utilise la sortie de l'unité LSTM précédente (h_{t-1}) et l'entrée courante (x_t) pour déterminer quelles parties de l'ancienne sortie doivent être oubliées, tâche accomplie par la porte d'oubli [14].

Ensuite, la nouvelle entrée x_t est intégrée et stockée dans l'état de la cellule, mettant à jour cet état avec cette formule :

$$C_t = f_t \circ C_{t-1} + i_t \circ \tilde{C}_t \tag{2.5}$$

Cette étape combine une couche sigmoïde, qui décide de l'intégration des nouvelles informations, et une couche *tanh*, qui ajuste les valeurs en leur attribuant un poids entre -1 et 1. Les valeurs combinées mettent à jour l'état de la cellule [14].

Enfin, la sortie h_t est une version filtrée de l'état de la cellule o_t. La couche sigmoïde sélectionne les parties de l'état de la cellule pour la sortie, et ces valeurs sont multipliées par celles générées par la couche *tanh* appliquée à l'état de la cellule C_t. La formule de l'état caché est [14] :

$$h_t = o_t \circ \tanh(C_t) \tag{2.6}$$

Voici une figure qui explique le fonctionnement d'un LSTM [36] :

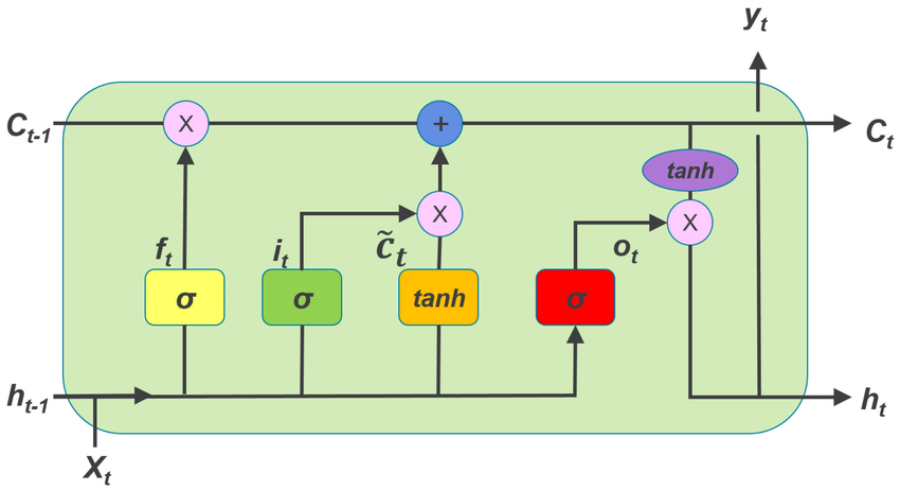


FIGURE 2.12 – Schéma d'un réseau de neurones à mémoire court terme et long terme

2.7.4 LSTM bidirectionnel (BiLSTM)

Les LSTM bidirectionnels (BiLSTM) sont une variante avancée des réseaux de neurones à mémoire court terme et long terme, conçus pour traiter des séquences de données. En prenant l'exemple d'une séquence de texte, le parcours peut se faire de la gauche vers la droite ou inversement. Il n'est possible de parcourir une séquence de plusieurs manières [45].

La mémoire ne modélise pas les mêmes éléments selon le sens de parcours retenu. Il est donc intéressant de pouvoir modéliser l'information selon différents parcours et ainsi de mettre en œuvre plusieurs réseaux à mémoires LSTM. C'est pour cette raison que les RNN bidirectionnels ont été introduits en 1997 par Schuster et Paliwal, et que l'idée a été reprise pour les RNN LSTM en 2005 par Graves et Schmidhuber [45].

L'architecture bidirectionnelle peut extraire l'information contextuelle dans les deux directions en même temps avec des couches cachées avant et arrière. Donc, un modèle BiLSTM comprend deux couches LSTM distinctes :

LSTM avant : traite la séquence d'entrée dans l'ordre chronologique (de t = 1 à t = T).
LSTM arrière : traite une copie inversée de la séquence d'entrée (de t = T à t = 1).

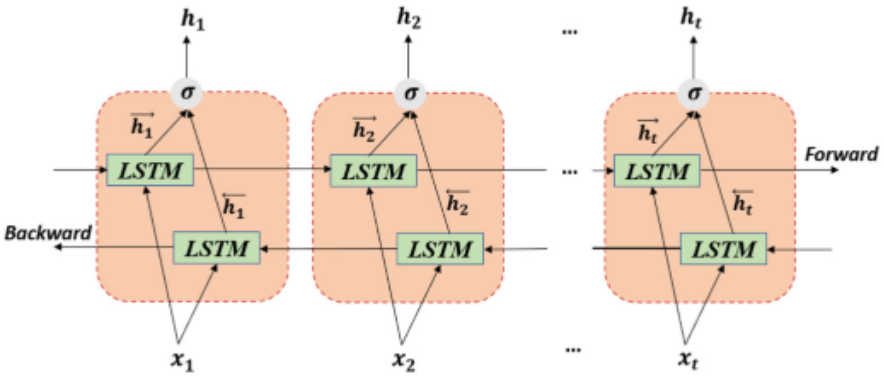


FIGURE 2.13 – Architecture de BiLSTM

Dans cette figure, $h_t^{\rightarrow}$ et $h_t^{\leftarrow}$ représentent respectivement les sorties des couches cachées avant et arrière. Ces sorties sont calculées itérativement, la couche avant de 1 à t et la couche arrière de t à 1, en utilisant le LSTM standard. La couche BiLSTM produit un vecteur de sortie Y, où chaque élément est calculé selon une fonction σ qui couple $h_t^{\rightarrow}$ et $h_t^{\leftarrow}$. En utilisant cette formule [42] :

$$y_t = \sigma(h_t^{\rightarrow}, h_t^{\leftarrow}) \tag{2.7}$$

σ peut être une fonction de sommation, multiplication, concaténation ou moyenne. La sortie finale est un vecteur Y, où chaque y_t, est calculé selon la fonction de couplage choisie.

Conclusion

Plusieurs méthodes de classification existent pour la reconnaissance de l'écriture. Le choix d'utiliser les réseaux de neurones n'est pas fait aléatoirement.

Parmi les avantages de cette méthode, on trouve que les réseaux de neurones sont efficaces pour traiter des données complexes et variées telles que les images de caractères. Ils gèrent de grandes quantités de données d'entraînement et tirent profit des gros volumes de données pour améliorer leurs performances.

Dans ce chapitre, nous avons donné une définition générale des réseaux de neurones, en commençant par expliquer le deep learning. Ensuite, nous avons présenté les types de réseaux de neurones que nous allons utiliser pour former et entraîner notre modèle : les CNN, les RNN et les LSTM.

Dans le chapitre suivant, nous offrirons un aperçu de la langue amazighe. Ensuite, nous présenterons la base de données que nous utiliserons pour notre système OCR.

3

Langue Amazighe et Reconnaissance Optique de Caractères

Les systèmes OCR visent à effectuer une conversion complète d'une image de document en un fichier texte modifiable. Cependant, cette tâche devient complexe lorsque la langue du texte n'est pas reconnue par ces systèmes, ce qui entraîne une mauvaise reconnaissance.

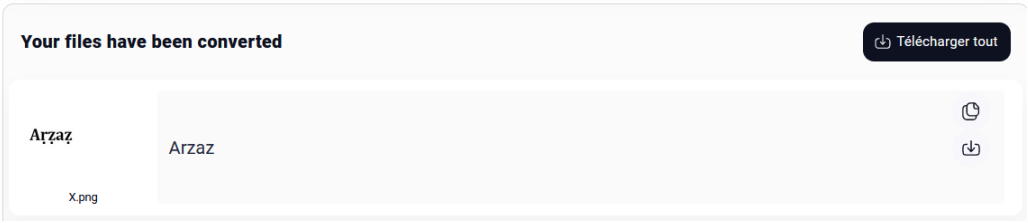


FIGURE 3.1 – Exemple d'une mauvaise reconnaissance d'une image écrite en amazighe

De nombreuses langues à travers le monde sont parlées uniquement par un nombre restreint de personnes, et ne bénéficient pas de la même reconnaissance que des langues plus répandues, telles que l'anglais et le français, par exemple la langue amazighe.

Dans ce chapitre, nous aborderons la langue amazighe, et explorer ses différentes variantes, notamment celles parlées en Algérie. Ensuite, nous décrirons la relation entre cette langue et l'OCR, en détaillant la collecte des données que nous avons entreprise. Enfin, nous présenterons certains articles portant sur des travaux déjà réalisés dans ce contexte.

Sommaire

3.1 Langue amazighe

La langue est un système de signes vocaux spécifique à une communauté donnée **[F. de Saussure]**, utilisé pour échanger des idées, des informations et des émotions. Elle est structurée par des règles grammaticales qui organisent le discours de manière cohérente.

Chaque langue possède son propre ensemble de règles, de vocabulaire et de conventions sociales qui reflètent la culture et l'histoire des communautés qui la parlent. D'un point de vue sociolinguistique, la langue représente un symbole d'identité et d'appartenance culturelle. **[F. de Saussure]**

Il existe plus de cinq mille langues parlées dans le monde aujourd'hui. Les langues les plus parlées sont l'anglais, le chinois, le hindi, l'espagnol, l'arabe, etc.

La langue amazighe, également connue sous le nom de berbère ou tamazight, est l'une des langues les plus anciennes de l'humanité. Elle s'étend maintenant de la mer Rouge aux îles Canaries et du Niger dans le Sahara à la mer Méditerranée, incluant la section nord de l'Afrique [34].

En outre, elle est parlée par une douzaine de pays. Parmi ces pays, nous citons : l'Algérie, le Maroc, la Tunisie, la Libye, l'Égypte, le Niger, le Mali, le Burkina Faso, la Mauritanie, etc. Cependant, l'Algérie et le Maroc sont les deux pays ayant la plus grande population amazighe. Il n'existe pas de données officielles sur le nombre de locuteurs de l'amazighe, mais on peut estimer leur nombre à environ trente à quarante millions de personnes.

On distingue trois systèmes d'écriture pour transcrire la langue amazighe :

Le tifinaghe : qui est un alphabet authentique attesté dans les inscriptions libyennes depuis l'antiquité [15].
L'alphabet arabe : il est utilisé depuis l'arrivée des arabes à la fin du 6ème siècle [15].
L'alphabet latin : développé par des chercheurs coloniaux à la fin du 19ème siècle, et utilisé plus tard par les chercheurs nationaux [15].

Dans notre travail, nous nous intéressons à la langue amazighe transcrite en latin.

3.2 Variétés de la langue amazighe

3.2.1 Berbère en Algérie

Le tamazight est une langue parlée en Afrique du Nord par les berbères, dans la population algérienne, elle est parlée par environ un tiers de personnes, et est considérée comme un élément majeur de l'identité culturelle et linguistique du notre pays [27].

Les variantes, également appelées « dialectes », sont constituées de sous-variantes liées au niveau régional. Les travaux descriptifs et comparatifs sur la structure linguistique des dialectes au sein des dialectes mettent en évidence des différences significatives à chaque niveau de langue.

Par exemple, le kabyle apparaît aujourd'hui comme des dialectes locaux ou régionaux, chaque énoncé se distingue des autres par des caractéristiques linguistiques différentes : phonétique, morpho-syntaxe, vocabulaire et parfois sémantique. Cette variation rend très difficile pour les développeurs la conception d'une langue kabyle standard.

La diversité géographique des locuteurs du tamazight a conduit au développement de variantes telles que :

3.2.1.1 Kabyle

Le kabyle est la première variété amazighe en nombre de locuteurs en Algérie qui se situe dans le nord du l'Algérie. Elle est parlée presque dans dizaines de wilayas comme Bejaia, Tizi-Ouzou, Bouira, Jijel, massif du Djurjura, Bordj Bouarreridj, Sétif et Boumerdas.

Selon M.A. Haddadou : « **C'est le dialecte algérien le plus important, aussi bien par le nombre de locuteurs que par l'abondance et la qualité de la documentation réunie depuis près d'un siècle et demi. Le kabyle est également à la pointe de la revendication berbère et c'est dans les départements kabyles (Tizi-Ouzou et Bejaïa) et partiellement kabyles (Bouira, Boumerdes, Sétif) que la demande sociale, en matière d'enseignement et de production culturelle est la plus forte.**»

Selon Salem Chaker : « **En Algérie, la principale région berbérophone est la kabyle.** »

3.2.1.2 Touareg

C'est un dialecte de la langue berbère parlé par les Touaregs qui présente un groupe ethnique nomade du Sahel et du Sahara, il s'étend sur plusieurs pays d'Afrique de l'ouest et du nord [27]. Ce dialecte est employé au sud d'Algérie par les touarègues qui sont connus sous l'appellation de « les hommes bleu » qui se trouve dans le Hoggar, Tassili, Aïr et Adrar des Iforas. Il a plusieurs variantes notamment le Tamahaq, le Targui, le Tamachaq, le Tawellammat, le Taheggart et le Tassili [27].

Selon M.A. Haddadou « **Parce qu'il est isolé des autres dialectes et, par conséquent moins affecté par l'emprunt arabe, le touareg est souvent considéré comme le dialecte le plus « pur » du berbère.**»

Selon Salem Chaker : « **L'aire du touareg est très vaste : elle recouvre des milliers de kilomètres carrés et s'étend sur plusieurs pays : Algérie, Libye, Niger, Mali, Haute Volta, Nigeria, mais le nombre de locuteurs est très réduit puisqu'il ne dépasserait pas, selon les estimations, le million de personne.**»

3.2.1.3 Mozabite

C'est un dialecte de la langue berbère parlé par les Mozabites dans le nord du Sahara algérien et Ghardaïa.

Selon M.A. Haddadou : « **L'appellation mozabite regroupe les parlers des sept villes de la vallée du Mzab (sud algérien) : Ghardaïa, considérée comme la capitale de l'ensemble, Melika, Beni Isguen, Bou-Noura, el Atteuf, à l'intérieur de la vallée, et, à l'extérieur, Berriane et, la plus éloignée, Guerrara, à 100 km de Ghardaïa.**»

3.2.1.4 Chaouia

Appelé aussi l'aurasien, il est pratiqué par les chaouis du Grand Aurès, situé dans les régions de l'est et du centre-est du pays. Ces régions sont : Batna, Khenchela, Biskra, Oum El Bouaghi, Souk Ahras, Tébessa et Aïn M'Lila.

3.2.1.5 Chenoui

Un dialecte berbère zénète parlé par les chenaoua, parmi les régions qui utilisent ce dialecte, nous citons Tipaza et Chlef.

Voici une carte géographique illustrant la localisation de chaque dialecte du tamazight en Algérie [27] :

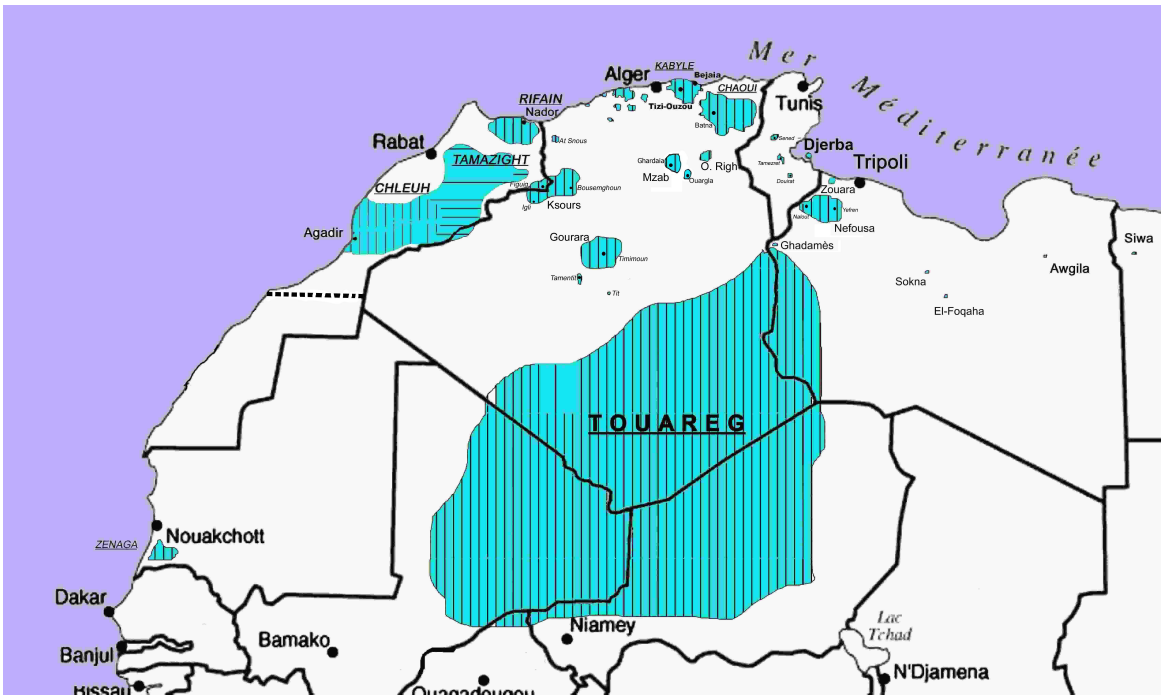


FIGURE 3.2 – Carte géographique du monde berbère

3.2.2 Berbère en d'autre pays

Le berbère est une langue ancienne parlée au Maroc et présente dans plusieurs régions, notamment le Rif, les montagnes de l'Atlas, le Souss et le Sahara. On y trouve les variantes suivantes : le tamazight du Moyen Atlas, le rifain, le chleuh et le soussi [27].

Cette langue est également parlée depuis des milliers d'années en Libye, notamment à Ghadamès et Nefousa, et est la langue maternelle de nombreux Libyens [27].

Les Berbères sont présents en Tunisie depuis des millénaires, et ont constitué une part importante de la population jusqu'à l'arrivée des Arabes au VIIe siècle [27].

Ils étaient également présents dans l'Égypte ancienne, notamment dans les oasis du désert occidental. Des preuves de leur présence peuvent être trouvées à travers des inscriptions en langue berbère découvertes dans des temples égyptiens, et des poteries ornées de motifs typiques de l'art berbère, comme la variante le siwi [27].

On retrouve également leur présence dans d'autres pays tels que le Sénégal, la Mauritanie et les Îles Canaries.

Dans le cadre de notre travail, nous nous intéressons au kabyle qui se situe au nord de l'Algérie.

3.3 Alphabet kabyle

***Définition* 3.3.1.** *L'alphabet est un système de symboles utilisés pour représenter les sons spécifiques d'une langue. Il est constitué d'un ensemble fini de lettres, chacune représentant un ou plusieurs phonèmes, c'est-à-dire les unités sonores d'une langue.*

***Définition* 3.3.2.** *Le mot alphabet est d'origine latine « alphabétum », formé avec les deux premières lettres de l'alphabet grec « alpha et bêta ». Il désigne « l'ensemble des lettres utilisées pour la représentation graphique des unités phoniques d'une langue, et disposées dans un cadre conventionnel ». C'est-à-dire l'alphabet est un ensemble d'éléments graphiques appelés « lettre » [21].*

Il existe de nombreux alphabets différents à travers le monde, chaque langue possède son propre système alphabétique qui représente ses différents sons. L'alphabet permet de transcrire les mots et les phrases de manière écrite, ce qui offre un moyen de communication et de préservation de la langue.

L'alphabet kabyle utilise l'alphabet latin adapté pour la variation phonologique de la langue, il existe 33 lettres pour l'alphabet kabyle illusté dans la figure suivante :

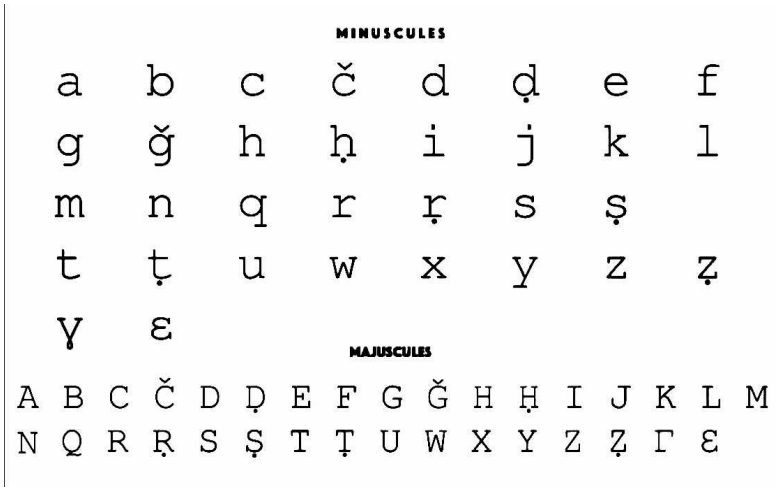


FIGURE 3.3 – Alphabet kabyle latin

Ces caractères sont composés de l'alphabet latin et de diacritiques qui représentent un ensemble de marques accompagnant une lettre ou un graphème. Les diacritiques placés au-dessus sont appelés diacritiques en exposant, tandis que ceux placés en dessous sont appelés diacritiques en indice. Dont 23 lettres sont latines, 8 diacritiques et 2 grecques.

- **Lettres latines :** a, b, c ,d ,e ,f ,g, h, i, j, k, l, m , n, q, r, s, t, u, w, x, y, z
- **Lettres grecques :** ɛ et ɣ
- **Lettres diacritiques :** č, ḍ, ǧ, ḥ, ṛ, ṣ, ṭ, ẓ

3.3.1 Ponctuation, Chiffres et Unicode

En ce qui concerne la ponctuation, il n'existe pas de signes de ponctuation spécifiques à l'écriture amazighe. Les signes de ponctuation sont présents dans la transcription étudiée comme dans toutes les autres langues. On peut trouver : les guillemets « », les points « . », les virgules « , », les points-virgules « ; », les deux-points « : », les points d'interrogation « ? », les points d'exclamation « ! », etc.

Mais les signes de ponctuation les plus fréquents qui peuvent être détectés au milieu d'un mot sont le tiret « - » et « ˇ ».

Exemple 3.3.1.

- *Tesslef-as (Elle lui a prêté).*
- *Yeččur (Il a rempli).*
- *Yeğğa (Il a quitté).*

De plus, les chiffres arabes occidentaux (0, 1, 2, 3, 4, 5, 6, 7, 8, 9) ainsi que tous les symboles logiques conventionnels (+, -, *, /, =, Ø, etc) sont utilisés. Les caractères avec des diacritiques sont considérés comme des caractères spéciaux. Comme de nombreux éditeurs de texte ne peuvent pas lire ce type de caractère, une solution consiste à utiliser l'Unicode.

L'Unicode est une norme qui permet de représenter et de stocker des données textuelles dans différentes langues et écritures, ce qui donne une meilleure compatibilité entre les éditeurs de texte. Les données peuvent être traitées et affichées correctement et indépendamment de la plateforme utilisée.

3.3.2 Voyelle - Consonne

Une voyelle est un son de la parole produit sans obstruction significative du flux d'air par les organes vocaux. En revanche, une consonne est un son de la parole produit avec une obstruction du flux d'air par les organes vocaux, les lettres de l'alphabet qui ne sont pas des voyelles sont considérées comme des consonnes. Les Voyelles en kabyle comporte trois voyelles, plus une voyelle de lecture le «e».

- Le «a» est moins ouvert qu'en français entre le «a» et le «e».
- Le «i» se prononce entre le entre le «i» et le «é».
- Le «u» se prononce comme un «ou».
- Le «e» qu'on nomme ilem, certains le considère comme une voyelle muette, d'autres le considère comme une voyelle à part entière.

3.3.3 Phonème - Graphème

Le phonème est la plus petite unité distinguée et caractérisée qui ne possède pas un sens, et qui peut être séparé par division dans une chaine parlée. Par contre, le graphème c'est la plus petite unité d'un système d'écriture alphabétique [29]. Voici un tableau qui présente les phonémes des lettres kabyle :

N°	*Maj*	*Min*	*phonème*	*phonème*	*N°*	*Maj*	*Min*	*phonème*	*phonème*
1	A	a	a	ya	18	L	l	l	yal
2	B	b	b b(p) b(v)	yab	19	T	t	t ṭ t̲	Yat yats yath
3	C	c	c	yac	20	N	n	n	yan
4	Č	č	č	yatch	21	Q	q	q	yaq
5	D	d	d d̲	yad yadh	22	Ḍ	ḍ	ḍ d̲	yaḍd yaddh
6	R	r	r	yar	23	M	m	m	yam
7	E	e	e	yé(ilem)	24	S	s	s	yas
8	ɛ	ɛ	ɛ	ya'	25	Ṣ	ṣ	ṣ	yass
9	F	f	f	yaf	26	Ṛ	ṛ	ṛ	yaṛr
10	G	g	g	yag	27	Ṭ	ṭ	ṭ	yatt
11	Ğ	ğ	ğ	yadj	28	U	u	u	yğu
12	ɣ / Γ	ɣ	ɣ	yagh	29	W	w	w	yaw
13	H	h	h	yah	30	X	x	x	yakh
14	Ḥ	ḥ	ḥ	yahh	31	Y	y	y	yay
15	I	i	i	yi	32	Z	z	z	yaz
16	J	j	j	yaj	33	Ẓ	ẓ	ẓ	yazz
17	K	k	k	yak					

TABLE 3.1 – Les phonémes de l'alphabet kabyle

3.4 OCR et la langue amazighe

La transcription automatique de documents imprimés en format électronique est un domaine dynamique de nos jours. En effet, pour sélectionner, former et entraîner un modèle OCR, il est indispensable de disposer d'un ensemble de données de qualité, le succès de la reconnaissance de caractères repose en grande partie sur la qualité des données utilisées.

Dans le cadre de notre travail, nous désignons ces données sous le terme de « corpus ». Les corpus sont devenus une étape très importante, ils sont une exigence essentielle pour le développement, et l'évaluation de diverses méthodes de reconnaissance de caractères.

À notre connaissance, il n'existe pas de corpus dédié à la langue amazighe transcrit en latin. Tous les travaux se concentrent sur le Tifinagh, et ont été testés sur des bases de données locales, qui contiennent un nombre restreint de caractères de l'alphabet amazighe, c'est pourquoi nous avons élaboré un corpus de données que nous avons collecté, nettoyé, segmenté, etc.

Dans cette partie, nous allons d'abord décrire ce qu'est un corpus, puis nous présenterons en détail notre propre corpus de données, que nous avons nommé **CAKL** (Corpus Amazighe Kabyle Latin).

3.4.1 Qu'est ce qu'un corpus

Avant de définir le terme « corpus », jetons un coup d'œil sur ses origines. Il provient du latin, et au fil du temps, il a eu plusieurs significations différentes, telles que le corps ou la chair (Cicéron Nat. 2,139), la personne (Salluste C.33.2), le cadavre (César G.2,10), et l'ensemble de l'État (Tite-Live 1,17, 2).

***Définition* 3.4.1.** *Le corpus est un recueil de pièces ou de documents qui concernent une même matière, discipline ou doctrine (Dalbera, 2003) [26].*

***Définition* 3.4.2.** *Les corpus permettent de multiples applications en didactique des langues, de la description linguistique à la programmation des contenus en passant par les ressources pédagogiques (dictionnaires, grammaire, manuels d'usage) (Boulton et Landure, 2010) [26].*

Un corpus est une collection de divers matériaux, rassemblés selon un ensemble de critères afin qu'il soit représentatif et balancé. Les matériaux composant un corpus peuvent être de genres variés : textes, audio, vidéo, images ou d'autres types de données, et les paramètres qu'il faut prendre en considération pour qu'un corpus soit balancé, nous citons à titre d'exemple, le genre, le domaine, la longueur, le registre linguistique, etc [35].

Les propriétés du corpus varient selon les applications pour lesquelles il sera utilisé, par exemple, si le corpus est assemblé pour un objectif, il n'est pas nécessaire d'avoir des exemples de divers genres, le plus important dans la collecte des corpus est d'inclure les spécificités de la langue étudiée [35].

Un corpus peut contenir des textes monolingues (une seule langue), ou bien multilingues (plusieurs langues), il peut être composé de corpus parallèles (avec des traductions), ou de corpus comparables (composé de textes dans des langues différentes qui partage une partie du vocabulaire employé et traite le même sujet, à la même époque et dans un même registre) [35].

Un corpus peut être annoté ou étiqueté. Un corpus annoté fournit des informations supplémentaires sur le texte qu'il contient. D'autre part, un corpus étiqueté associe des étiquettes à chaque élément afin d'identifier des caractéristiques spécifiques du texte.

3.4.2 Types de corpus

Il existe deux types de corpus, les corpus linguistiques et les corpus de reconnaissance. Un corpus linguistique se caractérise par sa grande taille qui peut atteindre des dizaines de millions, par contre un corpus de reconnaissance se caractérise par la variété des exemples en termes de style d'écriture et de police, aussi par une taille suffisante pour un bon entraînement [18].

Un corpus linguistique est utilisé souvent à des fins linguistiques. En revanche, l'objectif d'un corpus de reconnaissance est de former un système qui reconnaîtra un texte [18].

Dans notre travail, nous nous intéressons à ce dernier type de corpus. Un bon corpus implique un bon apprentissage et donc une bonne reconnaissance.

3.4.3 État de l'art

Avant d'expliquer la procédure de construction de notre corpus, nous décrivons ici quelques travaux représentatifs de l'état de l'art sur quelques corpus déjà réalisés en langues anglaise, française et arabe.

Un des premiers corpus représentatifs de l'anglais américain écrit est le Brown Corpus, compilé par W. Nelson Francis et Henry Kučera à l'université Brown, d'où son nom. Il a été assemblé en 1961, et contient environ un million de mots provenant de textes variés [35].

Un autre corpus de taille similaire a été compilé pour l'anglais britannique, nommé le Lancaster-Oslo- Bergen Corpus. Il a été compilé sous la direction de Geoffrey Leech de l'université de Lancaster, Stig Johansson de l'université d'Oslo, en collaboration avec Knut Hofland du centre norvégien d'informatique pour les sciences humaines de Bergen [35]. Ainsi que d'autres corpus plus larges existent.

Le premier corpus connu pour la langue française a été compilé dans les années soixante, nommé le Trésor de la Langue Française, également connu sous le sigle TLF. L'objectif de ce projet était de compiler un vaste corpus de mots et d'exemples linguistiques pour documenter et analyser la langue française sur une période allant du 1789 à 1960 [35].

Un autre corpus a été écrit et annoté, est la base de données FREEBANK. Il a été développé et maintenu par l'institut de recherche en informatique de Toulouse en 2004, en collaboration avec d'autres institutions de recherche. Cette base est utilisée dans le domaine de la recherche en linguistique et en traitement automatique du langage naturel [35].

Concernant la langue arabe, le premier corpus annoté est celui réalisé par Khoja et al en 2001, il est connu sous le nom d'Arabic Treebank, annoté avec les étiquettes suivantes : nom défini ou indéfini, verbe, particule, point de ponctuation ou numéro. Parmi les autres corpus annotés avec des données morphologiques et syntaxiques, on trouve le Penn Arabic Treebank en 2004 et le Prague Arabic Dependency Treebank en 2006 [35].

3.4.4 Construction de notre corpus de données

L'objectif de notre travail est de constituer un corpus pour la langue amazighe transcrite en latin. Nous avons crée un corpus composé de plus de trois cent mille fichiers images. Parmi celles-ci, 14 699 images ont été créées localement par nos soins, tandis que le reste a été collecté à partir de différentes sources via la plateforme Kaggle.

Dans notre travail, nous avons essayé de construire trois types de corpus :

- Corpus de caractères : celui sur lequel nous allons implémenter notre modèle.
- Corpus de mots et de phrases : nous avons commencé de collecter un petit ensemble de données. Bien que cela ne soit pas suffisant pour l'implémentation, c'est un début pour enrichir notre base avec plusieurs types d'images.

Corpus de caractères : il contient environ 309 748 images, qui comprennent des caractères écrits avec 70 polices (Amazigh Arial, Amazigh Times New Roman, Amazigh Muhdu, Cambria, Calibri, Amazigh Bangle, etc), sans oublier les variantes en gras et de l'italique pour chaque police. Les caractères sont stockés dans des dossiers, chaque dossier portant le nom de la lettre, comme le montre cette figure :

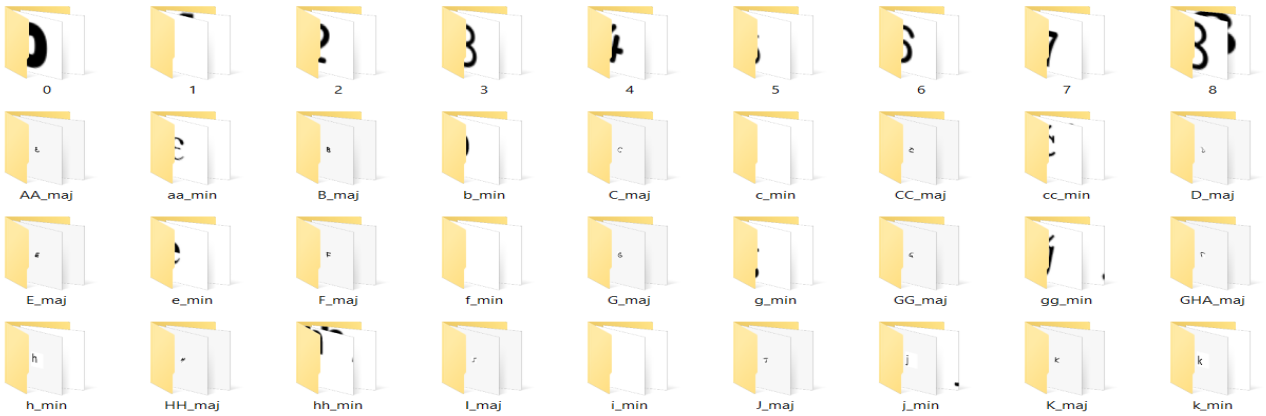


FIGURE 3.4 – Corpus de caractères

Nous avons distingué entre les lettres majuscules et minuscules, les lettres en majuscule sont suivies de « _maj », tandis que les lettres en minuscule sont suivies de « _min ». Concernant les lettres non latines, nous avons doublé la lettre qui lui ressemble en latin, pour qu'elle soit reconnue lors de la phase de traitement. Par exemple, pour la lettre « Č_maj », nous l'avons nommée « CC_maj ».

Corpus de mots : nous avons généré 5 800 mots écrits avec les polices suivantes : Cambria, Amazigh Arial et Amazigh Times New Roman. En général, notre corpus contient 601 mots, stockés dans des dossiers comme le montre cette figure :

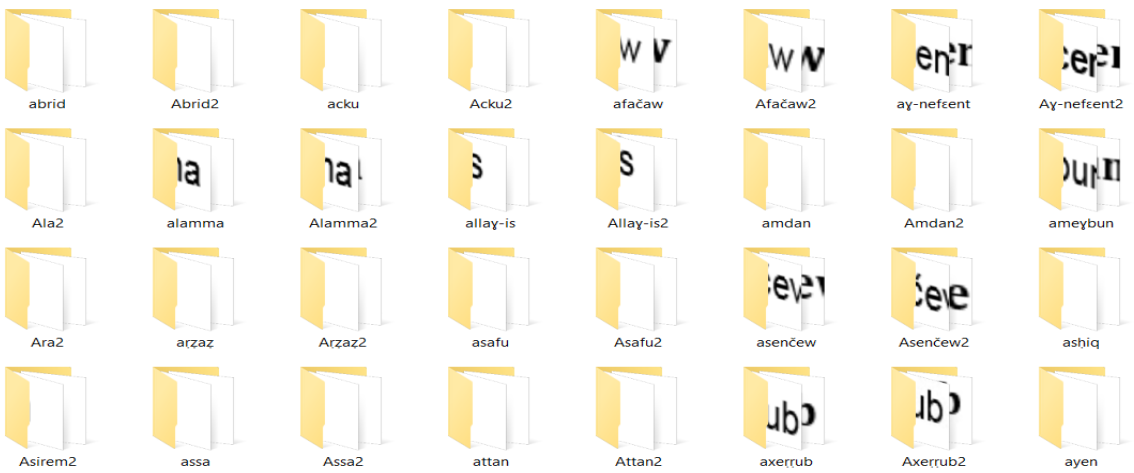


FIGURE 3.5 – Corpus de mots

Nous avons distingué entre les mots qui commencent par une majuscule et ceux qui commencent par une minuscule, en ajoutant aux dossiers des mots qui commencent par une majuscule le chiffre « 2 ». Cela permettra à notre modèle de distinguer entre la majuscule et la minuscule.

Corpus de phrases : nous avons 1 860 phrases écrites avec les mêmes polices que les mots, stockées également dans des dossiers avec ces différentes polices. En général, notre corpus contient 310 phrases.

Voici un tableau qui explique en détail le nombre d'occurrences de chaque caractère dans notre corpus, ainsi que le pourcentage de ce caractère par rapport au nombre total :

Num	Lettre	Nbr d'occ	Nom	Pourcentage	Num	Lettre	Nbr d'occ	Nom	Pourcentage
1	A	7319	A_maj	2.36%	40	e	3308	e_min	1.06%
2	B	7342	B_maj	2.37%	41	ɛ	483	aa_min	0.15%
3	C	7342	C_maj	2.37%	42	f	3175	f_min	1.02%
4	Č	5273	CC_maj	1.70%	43	g	3171	g_min	1.02%
5	D	7332	D_maj	1.87%	44	ǧ	213	gg_min	0.06%
6	Ḍ	4259	DD_maj	1.37%	45	ɣ	198	gha_min	0.06%
7	E	7318	E_maj	2.36%	46	h	3176	h_min	1.02%
8	Ɛ	4623	AA_maj	1.49%	47	ḥ	208	hh_min	0.06%
9	F	4489	F_maj	1.44%	48	i	3177	i_min	1.02%
10	G	5238	G_maj	1.69%	49	j	3170	j_min	1.02%
11	Ǧ	3388	GG_maj	1.09%	50	k	3268	k_min	1.05%
12	Ɣ	280	GHAA_maj	0.09%	51	l	3175	l_min	1.02%
13	H	6342	H_maj	2.04%	52	m	3195	m_min	1.03%
14	Ḥ	4190	HH_maj	1.35%	53	n	3158	n_min	1.01%
15	I	4396	I_maj	1.41%	54	q	3201	q_min	1.03%
16	J	6296	J_maj	2.03%	55	r	3172	r_min	1.02%
17	K	6333	K_maj	2.04%	56	ṛ	203	rr_min	0.06%
18	L	7336	L_maj	2.36%	57	s	3218	s_min	1.03%
19	M	7326	M_maj	2.36%	58	ṣ	213	ss_min	0.06%
20	N	7211	N_maj	2.32%	59	t	3178	t_min	1.02%
21	Q	6339	Q_maj	2.04%	60	ṭ	208	tt_min	0.06%
22	R	7333	R_maj	2.36%	61	u	3118	u_min	1.00%
23	Ṛ	6675	RR_maj	2.15%	62	w	3088	w_min	0.99%
24	S	7329	S_maj	2.36%	63	x	3248	x_min	1.04%
25	Ṣ	5251	SS_maj	1.69%	64	y	3249	y_min	1.04%
26	T	6334	T_maj	2.04%	65	z	3282	z_min	1.05%
27	Ṭ	5259	TT_maj	1.69%	66	ẓ	196	zz_min	0.06%
28	U	7333	U_maj	2.36%	67	Ɣ	5255	GHA_maj	1.69%
29	W	6334	W_maj	2.04%	68	0	3380	0	1.09%
30	X	6878	X_maj	2.22%	69	1	3366	1	1.08%
31	Y	6206	Y_maj	2.00%	70	2	3364	2	1.08%
32	Z	6319	Z_maj	2.04%	71	3	3366	3	1.08%
33	Ẓ	3700	ZZ_maj	1.19%	72	4	3362	4	1.08%
34	a	3186	a_min	1.02%	73	5	3362	5	1.08%
35	b	3184	b_min	1.02%	74	6	3368	6	1.08%
36	c	3283	c_min	1.05%	75	7	3358	7	1.08%
37	č	209	cc_min	0.06%	76	8	3364	8	1.08%
38	d	3200	d_min	1.03%	77	9	3360	9	1.08%
39	ḍ	209	dd_min	0.06%					

TABLE 3.2 – Tableau représentatif des statistiques des caractères de notre base

3.4.5 Élaboration méthodologique de notre corpus

Pour créer ce corpus, nous avons suivi une démarche simple et basique, que nous allons présenter dans cette partie.

La première étape consiste à installer le clavier Unicode de tamazight sur notre machine, ce qui nous permettra d'écrire facilement tous les caractères de cette langue avec le clavier, puis nous avons installé les polices tamazight pour les utiliser par la suite. La deuxième étape était la collecte des données, où nous avons collecté des mots et des phrases provenant de diverses sources telles que les livres, mémoires écrits en amazighe, romans, articles, etc. Nous avons tout écrit dans des fichiers Word avec différentes polices, pour fournir une version textuelle.

Voici un exemple des mots écrits et stockés textuellement dans un fichier Word :

uḥric	uḥric	**uḥric**	*uḥric*	***uḥric***	uḥric	**uḥric**	*uḥric*	***uḥric***
Uḥric	Uḥric	**Uḥric**	*Uḥric*	***Uḥric***	Uḥric	**Uḥric**	*Uḥric*	***Uḥric***
dacu	dacu	**dacu**	*dacu*	***dacu***	dacu	**dacu**	*dacu*	***dacu***
Dacu	Dacu	**Dacu**	*Dacu*	***Dacu***	Dacu	**Dacu**	*Dacu*	***Dacu***
kulci	kulci	**kulci**	*kulci*	***kulci***	kulci	**kulci**	*kulci*	***kulci***
Kulci	Kulci	**Kulci**	*Kulci*	***Kulci***	Kulci	**Kulci**	*Kulci*	***Kulci***
Acku	Acku	**Acku**	*Acku*	***Acku***	Acku	**Acku**	*Acku*	***Acku***
acku	acku	**acku**	*acku*	***acku***	acku	**acku**	*acku*	***acku***
Ticraḍ	Ticraḍ	**Ticraḍ**	*Ticraḍ*	***Ticraḍ***	Ticraḍ	**Ticraḍ**	*Ticraḍ*	***Ticraḍ***

FIGURE 3.6 – Exemple de quelques mots avec différentes polices

Enfin, nous avons capturé les caractères, mots et phrases de différentes tailles, nettoyé et supprimé les images inutiles, et trié les images dans des dossiers comme nous l'avons expliqué précédemment.

Voici une figure montrant un exemple de captures faites via le site de Marcel Délèze sur « Č » :

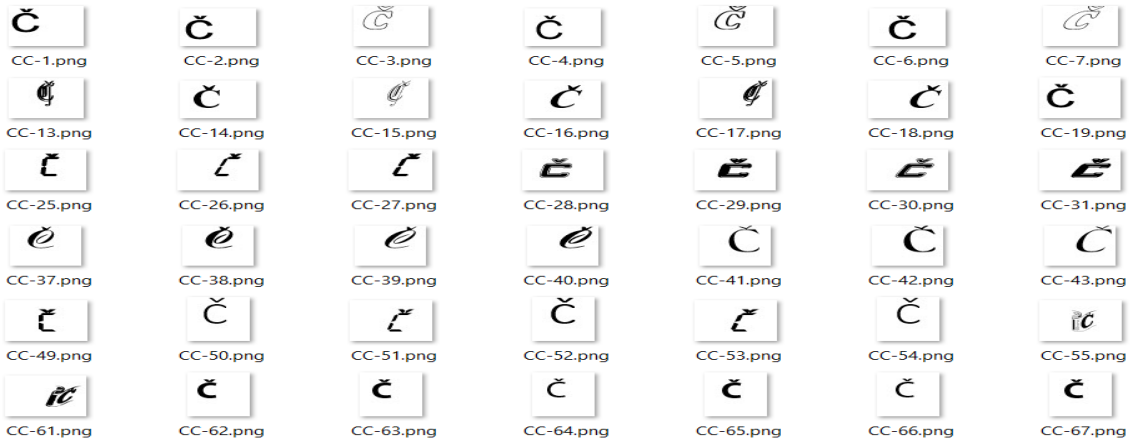


FIGURE 3.7 – Exemple du caractère « Č » dans différentes polices

3.5 Revue de la littérature sur la reconnaissance de la langue amazighe

Après avoir effectué plusieurs recherches dans l'espoir de trouver des articles portant sur la langue amazighe transcrite en latin, nous n'avons malheureusement réussi à trouver que quatre articles que

nous présenterons maintenant. Néanmoins, ces articles portent sur le latin du Maroc, dont l'alphabet diffère du notre.

Kh. EL GAJOUI et F. ATAA ALLAH (2014) [15] détaillent dans leur étude la conception d'un système de reconnaissance des caractères de la langue amazighe, ils ont exposé en détail le fonctionnement du système OCR, ainsi que la description de ses différents modules, en plus des approches élaborées pour chaque module.

Kh. EL Gajoui, F. Ataa Allah et M. Oumsis (2015) [16] ont proposé d'étudier cette langue reconnaissable par ses caractères diacritiques, ils ont suggéré l'utilisation d'un système basé sur les réseaux neuronaux, et ont examiné sa réactivité à ce type de caractères.

Ensuite, en 2016 [17] ils ont publié un deuxième article portant sur un système OCR, destiné au traitement de la langue amazighe y compris les marques diacritiques. Pour ce faire, ils ont élaboré un corpus spécifique à cette langue, puis ont employé une méthode de binarisation non linéaire lors de la phase de prétraitement, tout en adoptant une approche structurelle, reposant sur des caractéristiques d'approximation polygonale pour la phase de classification.

En 2018 [18] ils ont publié un troisième article portant sur la recherche en OCR de documents historiques et récents. Ils ont décrit la procédure de construction de ce corpus en trois niveaux : ligne, mot et caractère, ils ont mené par la suite une évaluation comparative du corpus, en utilisant un système basé sur l'approche Long Short Term Memory.

Conclusion

Dans ce chapitre, nous avons commencé par définir de manière générale la langue amazighe, et les systèmes d'écriture utilisés pour la transcrire. Ensuite, nous avons présenté les différentes variétés de cette langue en Algérie, ainsi que les variantes qui existent, en abordant brièvement les autre variantes présentes dans d'autres pays parlant cette langue.

Par la suite, nous avons abordé l'alphabet kabyle, et parler sur tous les éléments qui lui sont associés tels que la ponctuation, les chiffres, les voyelles et les consonnes, ainsi que les phonèmes et graphèmes correspondants.

Ensuite, nous avons expliquer en détail ce qu'est un corpus, ses différents types, et présenté un état de l'art des premiers corpus existants en arabe, français et anglais. Nous avons présenté aussi en détail notre propre corpus, nommé **CAKL**, et les étapes que nous avons suivies pour sa construction.

Enfin, nous avons introduit une revue de littérature portant sur certains articles réalisée, traitant le domaine de la reconnaissance de la langue amazighe transcrite en alphabet latin.

Maintenant que notre corpus est prêt à être utilisé, une étape de nettoyage est nécessaire avant la phase de traitement, d'où l'importance de la phase de prétraitement. Dans le chapitre suivant, nous présenterons les résultats du prétraitement, de la segmentation et de l'extraction de caractéristiques obtenus.

4

Contributions à la Reconnaissance Hors Ligne de l'Écriture Amazighe Imprimée

Il existe deux approches de reconnaissance : la reconnaissance en ligne et la reconnaissance hors ligne. La première traite uniquement les documents manuscrits, tandis que la seconde approche traite à la fois les documents imprimés et manuscrits.

Chaque cas de reconnaissance nécessite un système OCR différent pour le prétraitement, la segmentation et l'extraction des caractéristiques. Cela dépend du type de base de données utilisé et du modèle choisi pour la classification.

Notre travail se concentre sur la reconnaissance hors ligne de documents imprimés en langue amazighe, en utilisant des réseaux de neurones artificiels comme classifieurs.

Dans ce chapitre, nous allons présenter une partie de notre système OCR. Cette partie englobe les trois grandes phases suivantes : le prétraitement, la segmentation et l'extraction de caractéristiques. Chacune comporte un ensemble d'étapes, de méthodes et d'opérations à appliquer afin d'obtenir des images prêtes à être analysées.

D'abord nous définirons l'environnement de développement, le langage de programmation ainsi que les bibliothèques importantes utilisées. Ensuite, nous expliquerons de manière détaillée les différentes techniques utilisées.

Sommaire

4.1 Environnement de développement

4.1.1 Outils et langage

Les algorithmes que nous avons utilisés ont été implémentés dans l'environnement Jupyter, via le navigateur Anaconda, et nous avons opté pour le langage Python comme langage de programmation.

Le navigateur Anaconda est une interface utilisateur graphique de bureau, inclus dans la distribution Anaconda, qui nous permet de lancer des applications et de gérer facilement les packages, les environnements et les canaux conda sans utiliser des commandes de ligne de commande [38].

Pendant notre travail, nous avons utilisé le framework Anaconda pour diverses raisons, la raison la plus importante est que c'est une distribution open source et libre, qui regroupe plusieurs outils et bibliothèques utilisés dans les domaines de la science des données, de l'analyse de données et de l'apprentissage automatique. Anaconda est donc une distribution Python faite pour la data science.

Jupyter Notebook est une interface Web dans laquelle nous pouvons taper du code Python, l'exécuter et voir directement les résultats, y compris une visualisation à l'aide de graphiques [3].

Nous avons choisi de travailler sur Jupyter Notebook car il offre une interface flexible, qui nous a permis d'effectuer de multiples opérations, allant de l'écriture de code à la visualisation et au multimédia. Il nous a également permis d'écrire des textes, des titres et des commentaires directement dans le même document que le code. De plus, Jupyter Notebook offre la possibilité de sauvegarder le document et de le télécharger sous différentes extensions telles que html, pdf et ipynb.

Python est un langage de programmation open source, interprété, qui ne nécessite pas d'être compilé pour fonctionner. Ceci permet de voir rapidement les résultats d'un changement dans le code. Toutefois, ce langage de programmation s'est hissé parmi les plus utilisés dans le domaine du développement de logiciels, de gestion d'infrastructure et d'analyse de données [3].

4.1.2 Bibliothèques Python

Plusieurs bibliothèques ont été utilisées pour l'implémentation de notre code. Voici les bibliothèques les plus importantes :

4.1.2.1 TensorFlow

TensorFlow est une bibliothèque logicielle open source utilisé dans l'apprentissage automatique pour les réseaux neuronaux, elle est capable de fonctionner sur plusieurs CPU et GPU et peut être utilisée sur plusieurs plateformes [25]. Grâce à TensorFlow, nous avons la possibilité de créer et de

personnaliser des modèles d'OCR selon nos besoins spécifiques, ce qui simplifie notre travail dans le domaine de la reconnaissance de caractères.

4.1.2.2 NumPy

NumPy est une bibliothèque fondamentale pour les opérations mathématiques et les calculs numériques en Python. Elle permet d'effectuer des opérations mathématiques sur des données et des nombres. [2].

4.1.2.3 Matplotlib

Matplotlib est utile pour la visualisation de données et l'affichage d'images. Il peut être utilisé pour présenter les résultats et créer des représentations graphiques [2]. Cette bibliothèque offre une grande flexibilité et une vaste gamme de fonctionnalités pour produire des graphiques de qualité dans divers formats, tels que des graphiques en 2D et en 3D, des histogrammes, des diagrammes à barres, des nuages de points, etc.

4.1.2.4 OpenCV

OpenCV est une bibliothèque spécialisée dans le traitement d'images et de vidéos. Elle est utilisée pour des tâches telles que la lecture, l'édition, l'analyse d'images et l'extraction d'informations à partir d'elles [2].

4.1.2.5 Keras

Keras conçue par François Chollet, est une bibliothèque open source en Python utilisée pour faciliter la mise en œuvre des réseaux neuronaux. Cette API d'apprentissage profond de haut niveau offre une flexibilité en étant compatible avec des backends tels que TensorFlow et Theano.

4.2 Corpus de données

Puisque il n'existe pas de corpus préexistant bien établi pour la langue amazighe transcrit en latin kabyle contrairement à d'autres langues comme l'anglais, le français, l'arabe, etc. Nous avons donc opté pour la création de notre propre corpus, nommé **CAKL**, dans le chapitre précédent, nous avons expliqué en détails la méthodologie que nous avons employée pour sa conception.

Ce corpus est structuré en trois niveaux distincts, le premier niveau compte environ 309 mille images, contenant des caractères issus de différentes polices amazighes, le deuxième corpus est constitué d'environ 5 800 images de mots, chacun représenté en trois polices différentes, incluant l'italique et le gras, le dernier corpus comprend 1 860 phrases, qui se compose d'une seule ou de plusieurs lignes.

4.3 Prétraitement de l'image

Maintenant que le corpus est prêt à être utilisé, il arrive parfois que l'image d'entrée ne soit pas toujours prête pour le traitement et présente plusieurs lacunes. Il est alors nécessaire d'effectuer une série d'opérations sur ces images afin d'en améliorer la qualité, et préparer le texte de l'image pour la phase de reconnaissance, ce que l'on appelle le prétraitement.

Le prétraitement inclut toutes les fonctions effectuées avant l'extraction des primitives pour produire une version nettoyée de l'image d'origine [19].

Voici les opérations que nous avons effectuées sur les images de notre corpus afin de les préparer pour la prochaine phase :

4.3.1 Redimensionnement

Pour chaque catégorie de corpus (lettre, mot, phrase), après une série de tests, nous avons sélectionné des redimensionnements visant à réduire la taille de l'image tout en préservant la netteté des caractères et en évitant toute perte de détails.

Voici les dimensions choisies :

- Les lettres : 100x100 pixels.
- Les mots : 150x50 pixels.
- Les phrases d'une seule ligne : 700x30 pixels.
- Les phrases de plus d'une ligne : 700x190 pixels.

L'application de ces redimensionnements permet d'accélérer le traitement des images, ce qui revêt une importance lors de l'analyse d'un grand nombre d'images.

Voici le résultat obtenu après l'implémentation de cette étape en Python pour le caractère « ε » :

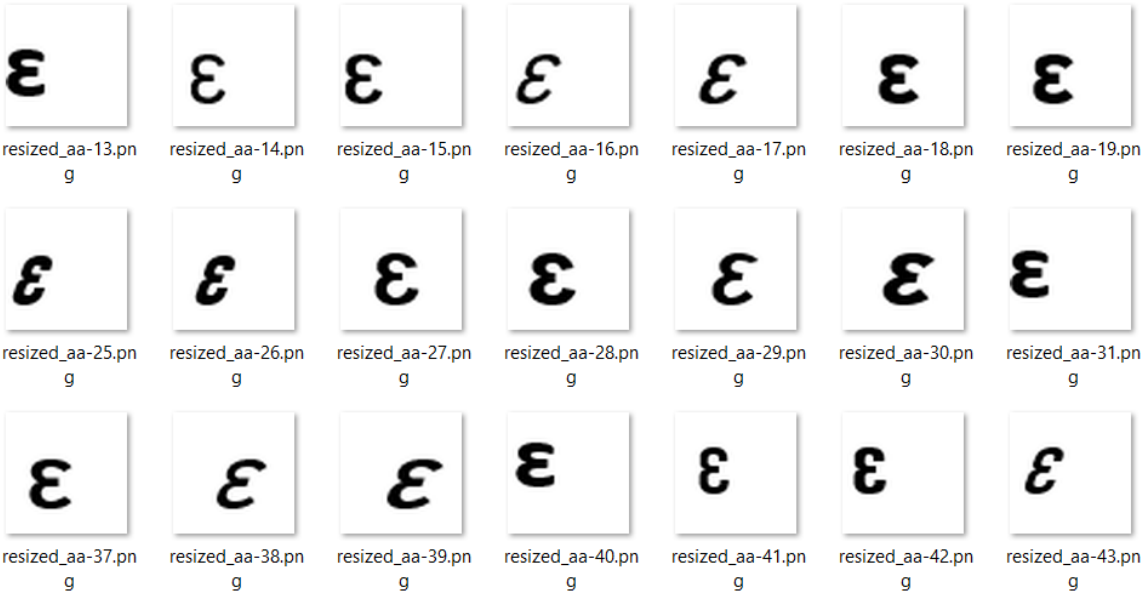


FIGURE 4.1 – Exemple d'un ensemble de caractères redimensionné avec la même taille

4.3.2 Chargement de l'image

L'image d'entrée est chargée puis elle est convertie en image en niveaux de gris [31]. Cette conversion spécifique est utilisée pour remplacer les pixels de différentes couleurs par nuances de gris, préservant ainsi les détails et les caractéristiques des objets ou du texte présents dans l'image.

Voici un exemple d'une image où une couleur a été convertie en niveaux de gris :

FIGURE 4.2 – Exemple d'une image convertie en niveaux de gris

Lors de la conversion en niveaux de gris, les couleurs noir et blanc ne changent pas. Chaque pixel est représenté par une valeur de luminosité allant de 0 à 255. Les couleurs autres que le noir et le blanc sont converties en plusieurs nuances de gris. Par exemple, les couleurs proches du blanc deviennent plus claires tandis que celles proches du noir deviennent plus sombres.

4.3.3 Binarisation

Il existe plusieurs techniques de binarisation couramment utilisées en traitement d'images.

Voici quatre types de binarisation populaires :

1. Binarisation globale avec un seuil fixe.
2. Binarisation globale avec un seuil inversé.
3. Binarisation locale adaptative.
4. Binarisation d'Otsu.

Nous avons testé ces quatre techniques sur plusieurs images pour déterminer laquelle est la meilleure pour nos images. Dans cette image avec un fond gris, nous remarquons que les trois techniques, à l'exception de celle adaptative, donnent de meilleurs résultats :

FIGURE 4.3 – Les techniques de binarisation

Mais pour cette image, par exemple, avec un fond blanc et une écriture grise, la technique d'Otsu et la technique adaptative donnent de meilleurs résultats par rapport aux autres techniques.

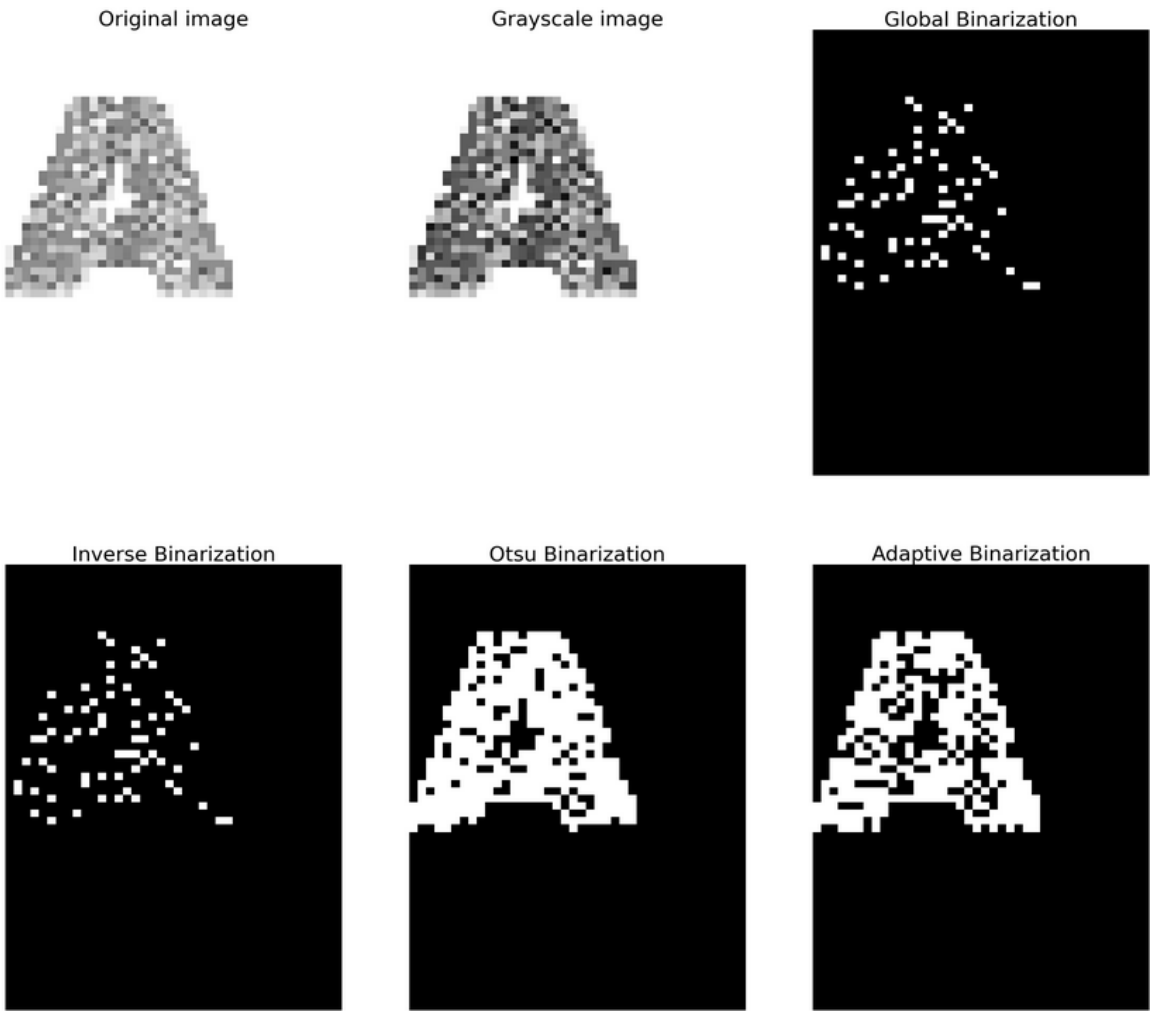


FIGURE 4.4 – Les techniques de binarisation appliquées à un caractère

Après avoir effectué plusieurs tests sur différentes images présentant diverses polices et couleurs, nous avons constaté que la technique de **binarisation d'Otsu** donne de meilleurs résultats, nous avons donc choisi cette méthode.

La méthode d'Otsu consiste en premier lieu à effectuer une analyse pour déterminer la valeur de seuil sur l'image, donc, elle appartient à la classe des approches globales. En seconde lieu, elle classe tous les pixels dont les valeurs sont au-dessus de ce seuil comme étant de pixels d'arrière plan, et tous les autres pixels comme étant de pixels de texte [6]. Elle est très souvent utilisé dans le domaine de l'OCR.

Le seuil optimal est calculé de manière à maximiser la variance inter-classe, et il peut varier d'une image à l'autre en fonction des caractéristiques de chaque image. Par exemple, pour la première image ci-dessus, la valeur du seuil est de 113, tandis que la valeur du seuil de la deuxième image est de 203.

Voici un algorithme qui explique comment fonctionne l'algorithme d'Otsu :

Algorithm 1 Algorithme de binarisation d'Otsu

Entrées: Image
Sorties: Image binarisée
1: Calculer l'histogramme de l'image
2: Initialiser la somme totale des pixels et la somme totale des intensités
3: **pour** chaque intensité de pixel *i* de 0 à 255 **faire**
4: Mettre à jour la somme totale des pixels et la somme totale des intensités
5: **fin pour**
6: Initialiser la variance maximale et le seuil optimal
7: **pour** chaque intensité de pixel *i* de 0 à 255 **faire**
8: Calculer la probabilité de classe avant et après *i*
9: Calculer la variance inter-classe
10: **si** variance inter-classe > variance maximale **alors**
11: Mettre à jour la variance maximale et le seuil optimal
12: **finsi**
13: **fin pour**
14: Binariser l'image en utilisant le seuil optimal

4.3.4 Réduction du bruit

L'élimination du bruit est une étape indispensable dans la phase de prétraitement. Nous pouvons dire que c'est un problème qui existe depuis longtemps et qui persiste encore aujourd'hui, il n'est pas facile d'éliminer un bruit dans une image car le type de bruit diffère d'une image à l'autre, cela a poussé les chercheurs à créer plusieurs techniques pour la suppression du bruit.

Dans le cadre de notre travail, nous avons testé quatre techniques sur les quatre méthodes de binarisation que nous avons présentées précédemment afin de déterminer quel couple de traitements est le plus efficace, ce qui donne 16 présentations différentes sur plusieurs images. Les techniques que nous avons utilisées sont :

- Le filtre gaussien.
- Le filtre médian.
- Le filtre bilatéral.
- Le filtre de moyennes non locales.

Voici un exemple d'une image testée sur les étapes de binarisation et de réduction de bruit de notre corpus :

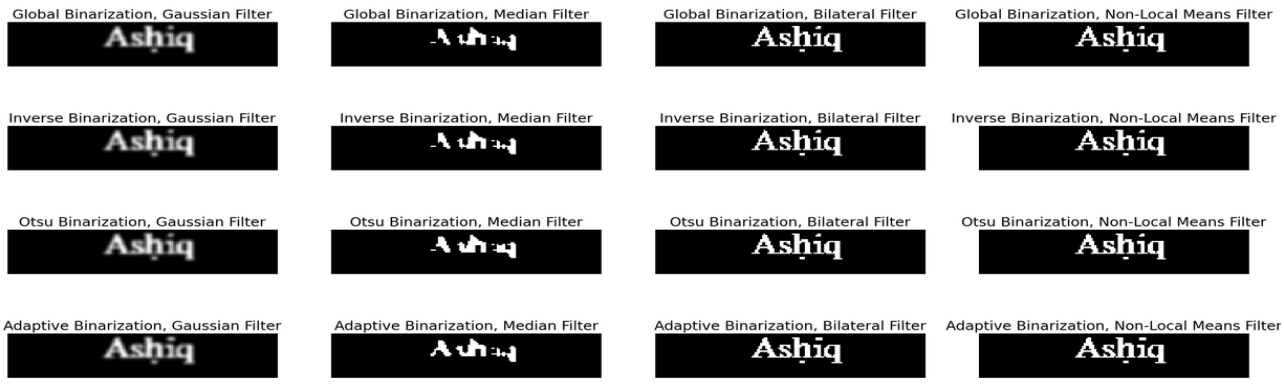


Figure 4.5 – Performance des techniques de binarisation et d'élimination de bruit

Après cette visualisation et l'utilisation de différentes images de fond blanc et gris, ainsi que d'écriture en noir et en gris, nous constatons que les méthodes de binarisation d'Otsu et de binarisation adaptative avec filtre bilatéral et filtre des moyennes non locales donnent les meilleurs résultats. Nous allons continuer notre travail en nous basant sur ces quatre méthodes.

4.3.5 Détection et correction d'inclinaison des lignes de texte

Les méthodes de correction d'inclinaison des lignes de texte sont utilisées pour redresser horizontalement les lignes d'écriture obliques. A cet effet, deux étapes sont appliquées. Premièrement, l'angle d'inclinaison est estimé, deuxièmement, l'image d'entrée est tournée par l'angle estimé [19].

Il existe de nombreuses méthodes de détection et correction de l'inclinaison horizontale du texte. Ces méthodes sont :

- La projection des profils.
- La transformée de Fourier.
- La transformée de Hough.
- La méthode basée sur la détection de contours.
- Etc.

Dans notre corpus de données, nous n'avons pas de photos mal redressées, mais si, à l'avenir, nous avons une photo mal redimensionnée en entrée, nous utiliserons la méthode de la transformée de Hough à cette étape.

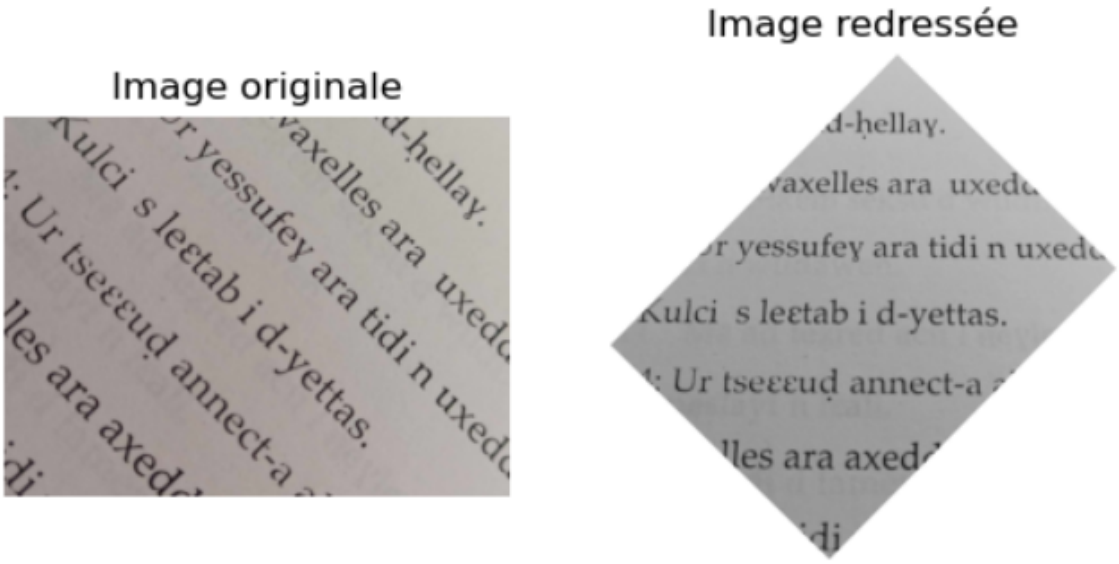


FIGURE 4.6 – Détection et correction d'inclinaison des lignes de texte

La première étape consiste à détecter les lignes sur l'image binaire en utilisant la transformée de Hough probabiliste. Ensuite, on calcule l'angle dominant en prenant la moyenne des angles de toutes les lignes détectées, en utilisant cet angle dominant, on procède au redressement de l'image.

Voici l'algorithme de transformée de Hough pour la détection et la correction d'inclinaison des lignes de texte :

Algorithm 2 Détection et correction d'inclinaison des lignes de texte

Entrées: Image binaire
Sorties: Image redressée

Détecter les lignes sur l'image binaire en utilisant la transformée de Hough probabiliste
Appliquer la transformée de Hough probabiliste à l'image binaire pour détecter les lignes
Stocker les coordonnées des lignes détectées
Calculer l'angle dominant des lignes détectées
Initialiser une liste vide pour les angles des lignes
pour chaque ligne détectée **faire**
    Extraire les coordonnées de début et de fin de la ligne $(x1, y1)$ et $(x2, y2)$
    Calculer l'angle de la ligne par rapport à l'horizontale en utilisant la formule :
        $\text{angle} = \arctan\left(\frac{y2-y1}{x2-x1}\right)$
    Convertir l'angle en degrés et l'ajouter à la liste des angles
fin pour
Calculer l'angle dominant en prenant la moyenne des angles de toutes les lignes détectées
Redresser l'image en utilisant l'angle dominant
Calculer l'angle de rotation nécessaire pour redresser l'image :
    angle_de_rotation = −angle_dominant
Appliquer une transformation de rotation à l'image binaire en utilisant l'angle de rotation calculé
Obtenir l'image redressée
Retourner l'image redressée

4.3.6 Squelettisation

La squelettisation est l'une des techniques les plus utilisées dans la reconnaissance de l'écriture, elle permet de simplifier l'image du caractère en une image plus facile à traiter en la réduisant à une forme avec un épaisseur de 1 pixel tout en conservant ses propriétés topologiques [6].

Il existe de nombreuses méthodes de squelettisation. Nous avons appliqué l'algorithme de Zhang-Suen, qui est reconnu pour sa robustesse et sa large utilisation.

Nous avons appliqué cet algorithme aux quatre méthodes déjà citées précédemment, et voici les résultats testés sur une image :

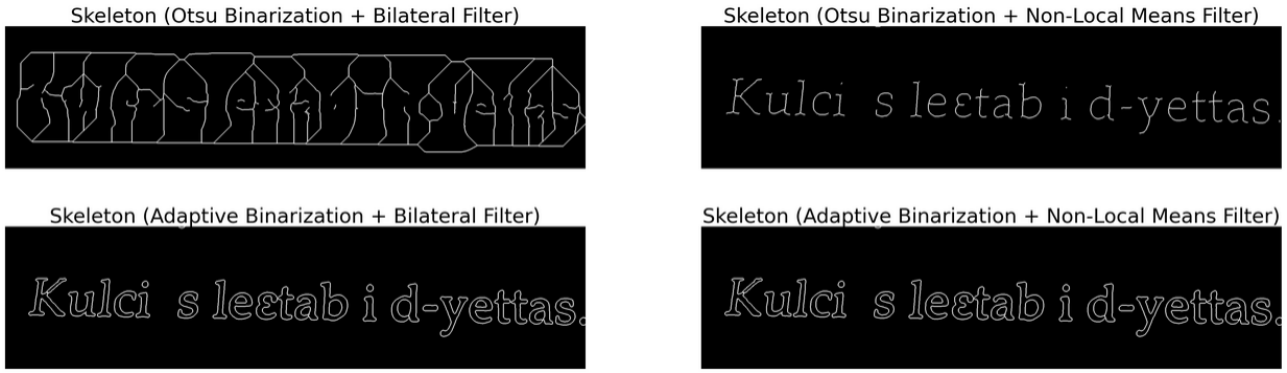


Figure 4.7 – Résultats de la squelettisation appliquée après la binarisation et l'élimination du bruit

Voici l'algorithme de Zhang-Suen expliqué pour la squelettisation de nos images, d'où :

- **N(x, y)** : Nombre de voisins noirs autour du pixel (x, y).
- **T(x, y)** :Nombre de transitions de blanc à noir autour du pixel (x, y).
- **P** : Les voisins spécifiques du pixel (x, y) :

P2	P3	P4
P7	P1	P5
P6	P8	P9

Algorithm 3 Algorithme de Zhang-Suen

Entrées: Image binaire
Sorties: Image squeletisée

Convertir l'image en une matrice binaire
répéter
  Sous-étape 1 : Marquer les pixels pour suppression
  pour chaque pixel noir P1 dans l'image **faire**
    Calculer le nombre de voisins noirs N(P1)
    Calculer le nombre de transitions blanc-noir T(P1)
    si $2 \leq N(P1) \leq 6$ et $T(P1) = 1$ **alors**
      si au moins un des voisins P2, P4, P6 est blanc **et** au moins un des voisins P4, P6, P8 est blanc **alors**
        Marquer le pixel P1 pour suppression
      finsi
    finsi
  fin pour
  Sous-étape 2 : Supprimer les pixels marqués
  pour chaque pixel marqué **faire**
    Supprimer le pixel (le rendre blanc)
  fin pour
  Sous-étape 3 : Marquer les pixels pour suppression (seconde phase)
  pour chaque pixel noir P1 dans l'image **faire**
    Calculer le nombre de voisins noirs N(P1)
    Calculer le nombre de transitions blanc-noir T(P1)
    si $2 \leq N(P1) \leq 6$ et $T(P1) = 1$ **alors**
      si au moins un des voisins P2, P4, P8 est blanc **et** au moins un des voisins P2, P6, P8 est blanc **alors**
        Marquer le pixel P1 pour suppression
      finsi
    finsi
  fin pour
  Sous-étape 4 : Supprimer les pixels marqués (seconde phase)
  pour chaque pixel marqué **faire**
    Supprimer le pixel (le rendre blanc)
  fin pour
jusqu'à l'image ne change plus
return Image squeletisée

Remarque 4.3.1. *Nous n'avons pas utilisé l'algorithme de Zhang-Suen directement, un post-traitement a été réalisé afin de conserver exclusivement les parties du squelette connectées aux lettres de l'écriture. Ce processus permet d'éliminer les artefacts indésirables et de ne garder que les éléments pertinents pour une analyse ultérieure, quel que soit le type d'image.*

4.3.7 Normalisation

Dans cette étape, nous allons comparer différentes méthodes de normalisation : Min-Max, Z-score et Logarithmique, nous utiliserons les résultats obtenus dans l'étape précédente, pour chaque méthode de normalisation, puis nous observerons les différences dans les squelettes avant et après le post-traitement.

FIGURE 4.8 – Différentes méthodes de normalisation

Nous avons choisi la méthode de normalisation par mise à l'échelle Min-Max pour l'appliquer à nos données.

Mise à l'échelle Min-Max : Cette méthode met à l'échelle les données dans une plage spécifiée, généralement entre 0 et 1, en utilisant la formule suivante :

$$X_{\text{norm}} = \frac{X - X_{\text{min}}}{X_{\text{max}} - X_{\text{min}}} \tag{4.1}$$

D'où :

X : est la valeur originale de la caractéristique à normaliser.
$\mathbf{X}_{\text{norm}}$: est la valeur normalisée de la caractéristique.
$\mathbf{X}_{\text{min}}$: est la valeur minimale de la caractéristique dans l'ensemble des données.
$\mathbf{X}_{\text{max}}$: est la valeur maximale de la caractéristique dans l'ensemble des données.

Remarque 4.3.2. *Il n'existe pas de méthode de prétraitement d'image qui convienne parfaitement à toutes les images dans tous les cas. Chaque image peut avoir des caractéristiques différentes en termes de luminosité, de contraste, de netteté, de bruit, etc. Par conséquent, il est souvent nécessaire d'adapter les méthodes de traitement en fonction des propriétés spécifiques de chaque image.*

Voici un exemple d'une image d'un caractère de notre corpus, sur lequel nous avons appliqué toutes les étapes de prétraitement expliquées précédemment :

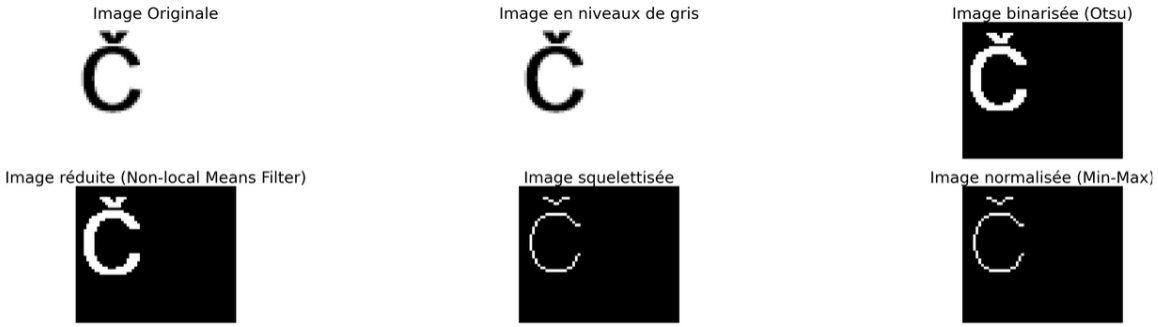


FIGURE 4.9 – Différentes étapes de prétraitement appliquées

4.4 Segmentation

Après la phase de prétraitement, la majorité des systèmes OCR isolent les caractères individuels avant de les reconnaître. Ceci est effectué durant la phase de segmentation [23].

C'est l'une des phases les plus difficiles et qui prend plus de temps, le processus de segmentation commence par séparer le bloc de texte en lignes, puis les lignes en mots, les mots en caractères et parfois même des unités plus petites.

Remarque 4.4.1. *Le but principal de notre travail est la reconnaissance des caractères. Dans cette partie, nous n'avons pas utilisé cette phase, mais nous l'avons testée sur nos mots et phrases de notre corpus.*

4.4.1 Segmentation de texte en lignes

Pour extraire les lignes de texte dans des images, on utilise généralement la méthode de projection horizontale, cette méthode consiste à projeter les pixels d'une image sur l'axe horizontal, créant ainsi un histogramme qui représente la somme des pixels sur chaque ligne horizontale.

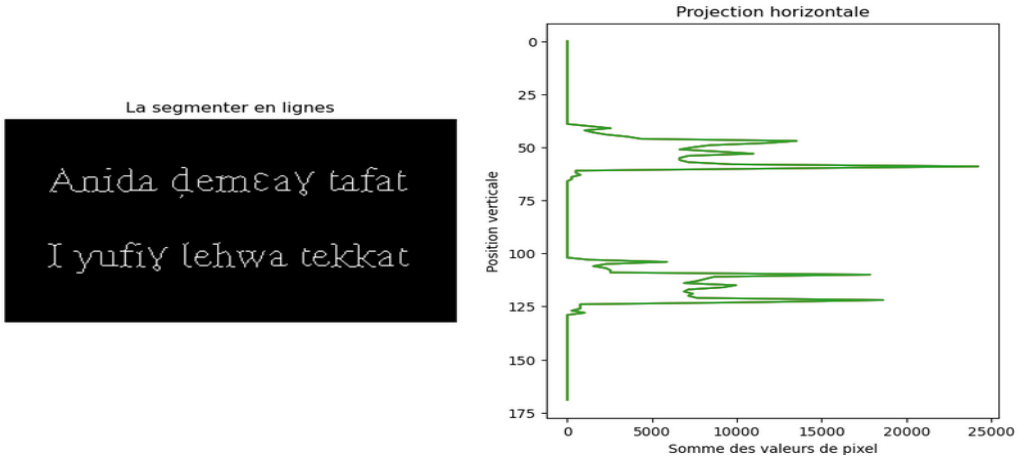


FIGURE 4.10 – Segmentation en lignes

4.4.2 Segmentation de ligne en mots et caractères

L'avantage de l'écriture amazighe imprimée est qu'elle n'est pas cursive, ce qui facilite l'opération de segmentation d'une ligne de texte en mots et caractères. Il existe plusieurs méthodes pour le processus de segmentation classées en deux catégories.

Segmentation sans classification : le système n'applique pas des méthodes de classification dans le processus de segmentation.

Segmentation avec classification : le système utilise des bases de connaissances et des classifieurs durant la segmentation.

4.4.2.1 Segmentation de ligne en mots

Avant de segmenter les caractères, il faut d'abord analyser la ligne pour la découper en mots, il existe plusieurs méthode pour le faire, nous avons utilisé la méthode de projection verticale. Cette méthode consiste à segmenter des blocs de texte en colonnes afin de détecter les espaces entre les mots et les caractères, les valeurs des pixels sont projetées le long de l'axe vertical.

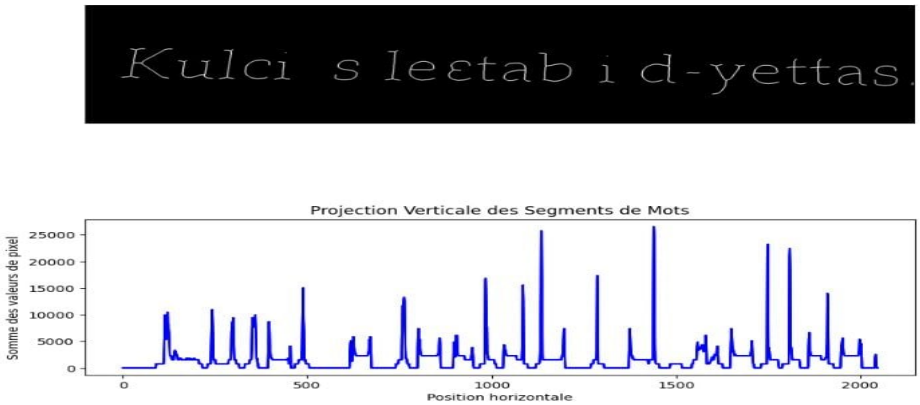


FIGURE 4.11 – Segmentation en mots

4.4.2.2 Segmentation de mot en caractères

Dans cette partie, nous avons utilisé la méthode de segmentation à partir du contour. Cette méthode détecte les contours en analysant les variations significatives dans les valeurs des pixels. Elle utilise des opérateurs tels que le détecteur de Canny ou la transformée de Hough pour générer un contour, puis identifie les points de liaison des caractères en balayant ce contour et en les associant en fonction de leur proximité.

FIGURE 4.12 – Segmentation en caractères

4.5 Extraction de caractéristiques

Afin de classer des images avec l'apprentissage automatique, le choix des caractéristiques et du classificateur doit être effectué manuellement. Cependant, avec l'apprentissage profond, la phase d'extraction et de classification sont réunies en une seule étape, comme le montre cette image :

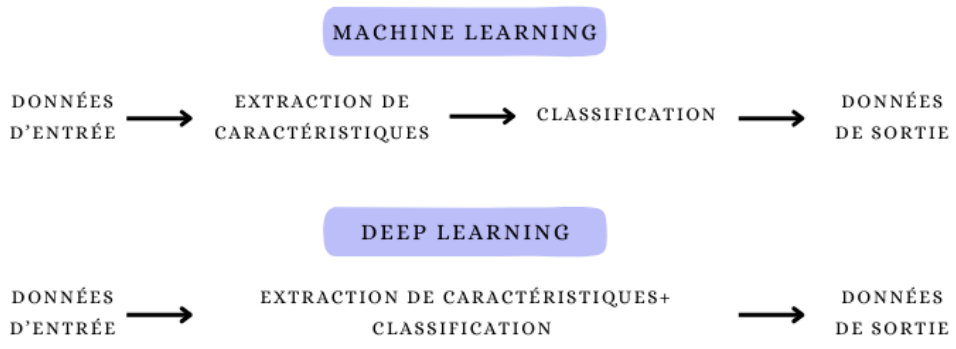


FIGURE 4.13 – Feature extraction between a traditional machine and deep learning

Dans notre travail, nous allons utiliser la méthode du deep learning, mais dans ce chapitre, nous allons expliquer brièvement comment cette phase se déroule en utilisant la méthode du machine learning.

Pour cet exemple, nous avons utilisé une approche structurelle, en mettant l'accent sur la forme, la taille et la connectivité des pixels de l'image, afin d'extraire des caractéristiques significatives pour la reconnaissance et la classification des caractères.

4.5.1 Étapes de l'extraction de caractéristiques

- **Mise à l'échelle :** l'objectif de cette étape est de s'assurer que le caractère remplit toute l'image pour une analyse uniforme.
- **Segmentation verticale :** l'objectif ici est de diviser l'image en segments verticaux égaux pour une analyse plus fine. Pour cela, l'image est découpée en N segments égaux [9].
- **Étiquetage et analyse des composants connectés :** cette étape vise à identifier les régions distinctes dans chaque segment en utilisant l'étiquetage des composants connectés [9].
- **Classification basée sur la taille :** l'objectif est de catégoriser les composants identifiés en fonction de leur taille. Pour ce faire, des seuils sont définis pour classer les composants en catégories comme «None», «Small», ou «Large». Une fonction de classification « c » est définie selon l'équation suivante :

$$c(s) = \begin{cases} \text{None} & \text{si } s = 0 \\ \text{Large} & \text{si } s > d \\ \text{Small} & \text{sinon} \end{cases}$$

  Cette fonction est appliquée à chaque élément du tuple (s_1, s_2, s_3) pour obtenir un tuple de classes (c_1, c_2, c_3) [9].
- **Cartographie vers des symboles d'observation :** l'objectif final est de convertir les classes de taille en symboles d'observation. Chaque combinaison de classes de taille est assignée à un symbole unique pour faciliter la reconnaissance [9].

Conclusion

À la fin de ce chapitre, nous pouvons conclure que nous disposons désormais d'un corpus de données nettoyé et prétraité, prêt à être entraîné et analysé.

Nous avons d'abord défini l'environnement de développement que nous avons utilisé pour la réalisation de ce travail. Ensuite, nous avons présenté les bibliothèques importantes et nécessaires pour l'analyse d'une image.

Nous avons ensuite détaillé toutes les étapes utilisées pour la phase de prétraitement, allant du redimensionnement, du chargement de l'image, de la binarisation, de la réduction du bruit, de la détection et correction de l'inclinaison des lignes de texte, de la squelettisation jusqu'à la normalisation, en discutant des différentes méthodes existantes à chaque étape.

Après cela, nous avons abordé la segmentation et ses diverses techniques, allant de la segmentation d'une ligne de texte jusqu'à un caractère. Enfin, nous avons expliqué la phase d'extraction des caractéristiques.

Dans le chapitre suivant, nous allons présenter les résultats de notre implémentation. Une discussion et une analyse des résultats seront effectuées.

5 Implémentation et Résultats

Dans le but d'élaborer un système OCR capable de répondre aux besoins de reconnaissance des caractères de la langue amazighe, nous proposons d'adopter l'approche des réseaux neuronaux pour la phase de classification.

Dans le chapitre précédent, nous avons présenté les différentes étapes de la phase de prétraitement, nous avons expliqué le fonctionnement de la segmentation des lignes, des mots et des caractères, ainsi que la phase d'extraction des caractéristiques.

Dans ce chapitre, nous allons développer notre système en proposant deux modèles. Le premier sera défini manuellement. Nous commencerons par présenter ce modèle et l'architecture globale du système proposé. Ensuite, nous testerons les résultats obtenus sur le corpus **CAKL**. Enfin, nous analyserons et discuterons les résultats obtenus.

Le second modèle est un modèle prédéfini existant. Nous allons suivre la même démarche que pour le premier modèle, et un bilan des résultats sera présenté. Pour finir, nous comparerons les deux modèles implémentés, puis nous décrirons le système final que nous avons développé.

Sommaire

5.1 Schéma final

Voici le schéma final sur lequel nous avons travaillé tout au long de ce mémoire, depuis la collecte des données jusqu'à la classification des caractères :

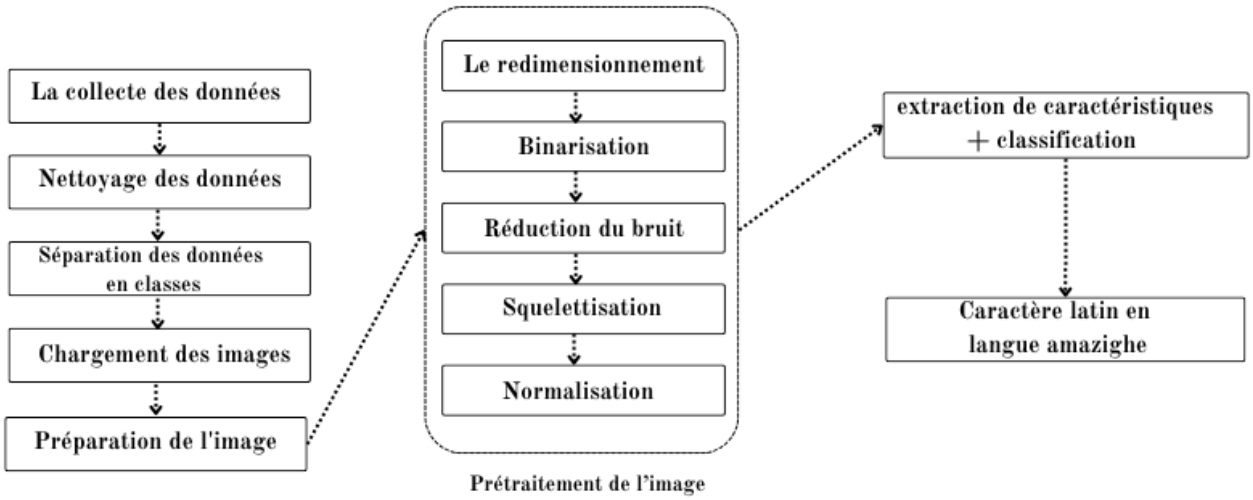


FIGURE 5.1 – Schéma de notre système OCR

5.2 Phase d'apprentissage

Dans cette phase, nous allons entraîner notre modèle avec notre corpus. Cette phase se divise en deux sous-étapes :

5.2.1 Préparation des données

Après le prétraitement des données, certaines images ont été mal prétraitées, ce qui nous a conduit à effectuer un nettoyage manuel pour chaque lettre. À la fin, nous avons obtenu 306 496 images. Nous avons considéré chaque lettre et chaque chiffre comme une classe, ce qui nous donne un total de 77 classes.

Après avoir préparé les données pour l'entraînement, il faut charger l'ensemble des images avec les étiquettes correspondantes et définir les dimensions des images. Ensuite, il faut encoder les étiquettes en valeurs numériques, les convertir en encodage one-hot, et transformer les listes d'images et d'étiquettes en tableaux NumPy.

5.2.2 Division des données

Les données sont prêtes, il est temps de les diviser en trois sous-ensembles. Dans notre cas, nous avons fait une division comme suit :

Entraînement : cet ensemble contient les données sur lesquelles notre modèle va apprendre les caractéristiques et les relations nécessaires pour faire la reconnaissance. Nous avons utilisé 70% des données du corpus.

Validation : cet ensemble est utilisé pour évaluer la performance du modèle pendant l'entraînement et pour ajuster les hyperparamètres. Ces hyperparamètres sont des paramètres définis avant le processus

d'entraînement. Nous avons utilisé 15% des données du corpus.

Test : cet ensemble est utilisé pour évaluer la performance finale du modèle après que l'entraînement soit terminé. Ces données ne sont jamais utilisé pendant l'entraînement ou la validation, ce qui permet d'obtenir une estimation objective de la performance du modèle. Nous avons utilisé aussi 15% des données du corpus.

Remarque 5.2.1. *Un tableau détaillant la répartition du corpus en ensembles d'entraînement, de test et de validation est présenté dans l'annexe 2.*

5.3 Architecture du modèle proposé

D'après les articles et les connaissances sur les fonctionnalités des différents réseaux neuronaux, ainsi que les tests effectués sur un petit nombre de classes, nous avons choisi cette architecture pour notre modèle. Celui-ci est composé de quatre couches convolutionnelles, suivies de quatre couches de maxpooling, deux couches RNN, et une couche dense :

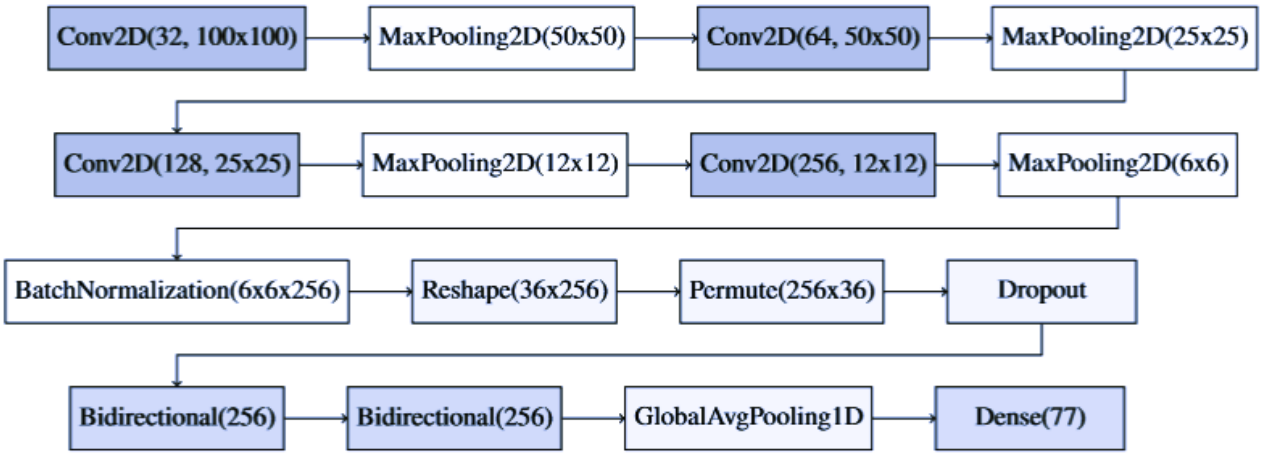


FIGURE 5.2 – Architecture de notre modèle

Notre image d'entrée est de dimension 100 x 100 pixels. L'image va d'abord passer par la première couche de convolution, composée de 32 filtres de taille 3 x 3.

On utilise l'architecture séquentielle qui est une pile linéaire de couches, parce qu'elle permet de définir facilement un modèle en empilant des couches les unes sur les autres, ce qui simplifie la création et la compréhension des réseaux neuronaux.

La première couche est **Conv2D (Convolution 2D)**. La forme de sortie de cette couche est (None, 100, 100, 32). D'où :

None : représente le lot (batch) d'images en entrée, qui peut varier est donc non spécifié.
100, 100 : Ce sont les dimensions spatiales de la sortie, c'est-à-dire la hauteur et la largeur de chaque carte de caractéristiques (feature map).
32 : représente le nombre de filtres utilisés par cette couche.

Après chaque couche de convolution, on applique la couche **MaxPooling2D** qui réduit la dimensionnalité spatiale des cartes de caractéristiques en préservant les caractéristiques les plus importantes.

Ensuite, la couche de **Batch Normalization** est appliquée, qui normalise les valeurs pour qu'elles aient une distribution plus uniforme et contrôlée.

La couche **reshape** change la forme des données qui sort de la couche précédente sans modifier le contenu, la couche **permute** réorganise les dimensions des tenseurs en entrée selon un ordre spécifié. Le **dropout** est une technique de régularisation utilisée pour réduire le surapprentissage.

La couche **bidirectionnelle** consiste à capturer les dépendances à long terme dans les séquences de données.

On utilise la couche **GlobalAveragePooling1D** pour calculer la moyenne des valeurs de sortie précédentes, elle permet de conserver les caractéristiques importantes.

Enfin, la couche **Dense**, entièrement connectée, utilise ces moyennes pour produire une sortie finale avec le nombre approprié de classes pour la classification.

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 100, 100, 32)	320
max_pooling2d (MaxPooling2D)	(None, 50, 50, 32)	0
conv2d_1 (Conv2D)	(None, 50, 50, 64)	18,496
max_pooling2d_1 (MaxPooling2D)	(None, 25, 25, 64)	0
conv2d_2 (Conv2D)	(None, 25, 25, 128)	73,856
max_pooling2d_2 (MaxPooling2D)	(None, 12, 12, 128)	0
conv2d_3 (Conv2D)	(None, 12, 12, 256)	295,168
max_pooling2d_3 (MaxPooling2D)	(None, 6, 6, 256)	0
batch_normalization (BatchNormalization)	(None, 6, 6, 256)	1,024
reshape (Reshape)	(None, 36, 256)	0
permute (Permute)	(None, 256, 36)	0
dropout (Dropout)	(None, 256, 36)	0
bidirectional (Bidirectional)	(None, 256, 256)	168,960
bidirectional_1 (Bidirectional)	(None, 256, 256)	394,240
global_average_pooling1d (GlobalAveragePooling1D)	(None, 256)	0
dense (Dense)	(None, 77)	19,789

Total params: 971,853 (3.71 MB)

FIGURE 5.3 – Configuration de notre modèle

Cette figure présente ces différentes couches ainsi que leurs paramètres, le calcul de quelques paramètres s'effectue comme suit :

Pour les couches de convolutions : **nombre de paramètres = nombre de filtres * (taille du filtre * nombre de canaux d'entrée + 1)**, pour conv2d par exemple, le nombre de paramètres est (32 * (3 * 3 * 1 + 1)) = 320.

Remarque 5.3.1. *Pour les couches suivantes, le nombre de canaux d'entrée correspond au nombre de filtres de la couche précédente.*

La couche max_pooling2d n'a pas de paramètres à apprendre, elle effectue simplement une opération qui prend la valeur maximale dans chaque région de la carte de caractéristiques.

Les couches reshape, permute, dropout et global_average_pooling1d ne retournent pas de paramètres.

Pour la couche dense : **nombre de paramètres = nombre de sorties × (nombre d'entrées + 1)**, donc 77+(256+1) = 19 789.

Total params : c'est la somme de tous les paramètres du modèle.

5.3.1 Méthode d'exploration des hyperparamètres avec la validation croisée

Avant d'entraîner le modèle final, nous l'avons testé sur un petit ensemble de données (environs 9000 images) avec différentes combinaisons de paramètres afin de trouver la combinaison optimale qui améliore les performances du modèle.

Voici quelques-uns des hyperparamètres que nous avons testés :

- Fonctions d'activation.
- Initialisateurs de noyau : déterminent les valeurs initiales des poids des couches du réseau.
- Optimiseurs : sont des algorithmes utilisés pour ajuster les poids du réseau pendant l'entraînement afin de minimiser la fonction de perte.

Après avoir entraîné ces paramètres, nous avons obtenu les meilleurs résultats avec une précision de 85%. Les paramètres choisis sont : «la fonction ReLU», «uniform» et « l'optimiseur Nadam».

5.3.2 Entraînement du modèle

L'entraînement de notre modèle a été effectué en utilisant les meilleurs hyperparamètres déterminés précédemment. Les résultats de l'entraînement sont présentés ci-dessous :

```
Epoch 1/50
6705/6705 [==========] - ETA: 0s - loss: 0.6146 - accuracy: 0.8299
Epoch 1: val_loss improved from inf to 0.35690, saving model to
   ocr_model_best_params.h5
6705/6705 [==========] - 3860s 575ms/step - loss: 0.6146 - accuracy:
   0.8299 - val_loss: 0.3569 - val_accuracy: 0.8986
...
Epoch 16/50
6705/6705 [==========] - ETA: 0s - loss: 0.1156 - accuracy: 0.9615
Epoch 16: val_loss improved from 0.19824 to 0.19589, saving model to
   ocr_model_best_params.h5
6705/6705 [==========] - 3841s 573ms/step - loss: 0.1156 - accuracy:
   0.9615 - val_loss: 0.1959 - val_accuracy: 0.9455
...
Epoch 26/50
6705/6705 [==========] - ETA: 0s - loss: 0.0982 - accuracy: 0.9668
Epoch 26: val_loss did not improve from 0.19589
```

```
6705/6705 [==========] - 3603s 537ms/step - loss: 0.0982 - accuracy:
   0.9668 - val_loss: 0.2089 - val_accuracy: 0.9426
Epoch 26: early stopping
```

Nous avons utilisé 50 époques dans ce modèle, chaque époque représente une itération complète sur l'ensemble des données d'entraînement. Cela signifie que le modèle a parcouru 6705 groupes de données, chaque groupe contient un certain nombre d'exemples.

Il est calculé sur un ensemble de validation, qui représente l'erreur entre les prédictions générées par le modèle et les valeurs réelles, en utilisant l'ensemble de validation. De même, val_accuracy est calculée sur cet ensemble et représente l'accuracy (précision) du modèle, c'est-à-dire sa capacité à correctement prédire les étiquettes.

En parallèle, le modèle calcule la perte (loss), qui mesure l'erreur entre ses prédictions et les valeurs réelles sur l'ensemble d'entraînement. De plus, l'accuracy (précision) est calculée et représente la capacité du modèle à correctement classifier les données d'entraînement.

Nous avons utilisé également l'arrêt prématuré (early stopping) pour arrêter l'entraînement du modèle lorsque les performances sur l'ensemble de validation ne s'amélioraient plus, cette technique contribue à éviter le surapprentissage. Dans notre cas, l'entraînement a été arrêté après 26 époques sans amélioration de la perte de validation (val_loss), avec une patience définie à 10 époques.

5.3.3 Résultat et interprétation

Notre modèle a été entraîné avec succès, avec un taux d'exactitude (accuracy) de 96% et une perte de 0.1. Voici les graphes qui présentent l'historique de l'entraînement, et illustrent les courbes de perte et d'exactitude pour les phases d'entraînement et de validation :

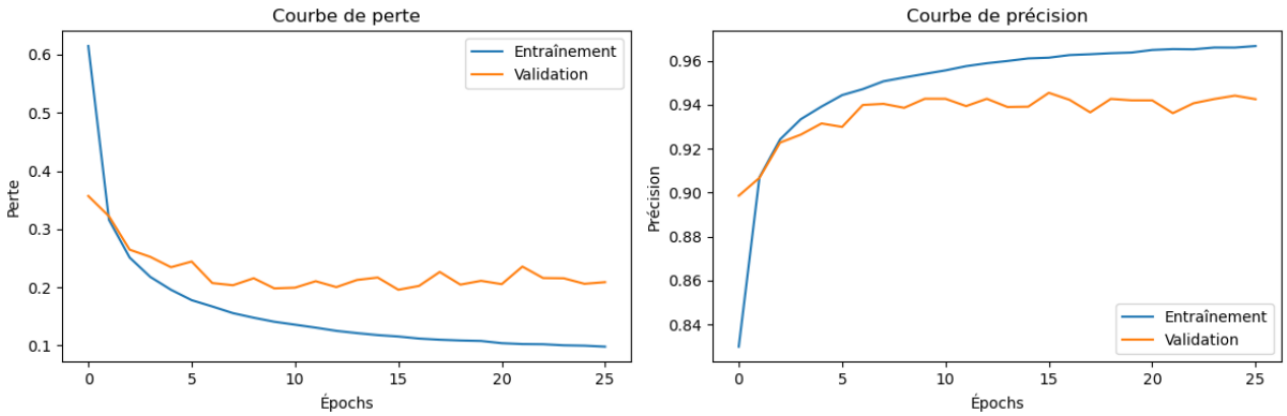


FIGURE 5.4 – L'historique de l'entraînement

5.3.3.1 Courbe de Perte

La courbe de perte d'entraînement (en bleu) atteint environ 0.1 après 25 époques, tandis que celle de validation (en orange) se stabilise autour de 0.2. Donc, on peut dire que la perte diminue au fur et à mesure que les époques augmentent, le modèle s'adapte mieux aux données d'entraînement.

La diminution rapide de la perte indique que le modèle apprend efficacement les caractéristiques de la base. Néanmoins, les fluctuations de la perte de validation indiquent que le modèle pourrait être sensible aux variations des données de validation, ce qui pourrait signaler un surajustement.

5.3.3.2 Courbe de Précision (accuracy)

L'accuracy augmente avec les époques, qui montre une amélioration continue du modèle. L'accuracy d'entraînement atteint alors environ 96% après 25 époques. L'accuracy de validation montre également une tendance à la hausse mais atteint environ 94%, avec des fluctuations après 10 époques. Alors, on peut dire que l'accuracy augmente au fur et à mesure que les époques augmentent.

5.3.3.3 Rapport de classification

Avant de discuter sur les résultats obtenus pour chaque caractère, il est important de définir quelques notions :

L'accuracy (exactitude) : elle représente la proportion de prédictions correctes parmi le nombre total de prédictions effectuées. Sa formule est :

$$\text{Accuracy} = \frac{\text{Vrai Positifs} + \text{Vrai Négatifs}}{\text{Vrai Positifs} + \text{Vrai Négatifs} + \text{Faux Positifs} + \text{Faux Négatifs}} \tag{5.1}$$

Précision : proportion des instances positives prédites correctement parmi toutes les instances prédites comme positives. Sa formule est :

$$\text{Précision} = \frac{\text{Vrai Positifs}}{\text{Vrai Positifs} + \text{Faux Positifs}} \tag{5.2}$$

Rappel : proportion des instances positives prédites correctement parmi toutes les instances réellement positives. Sa formule est :

$$\text{Rappel} = \frac{\text{Vrai Positifs}}{\text{Vrai Positifs} + \text{Faux Négatifs}} \tag{5.3}$$

F1-score : moyenne harmonique de la précision et du rappel. Sa formule est :

$$F1 = 2 \cdot \frac{\text{Précision} \cdot \text{Rappel}}{\text{Précision} + \text{Rappel}} \tag{5.4}$$

Sachant que :

- **Vrai Positifs :** les instances positives correctement classifiées par le modèle.
- **Faux Positifs :** les instances négatives incorrectement classifiées comme positives par le modèle.
- **Faux Négatifs :** les instances positives incorrectement classifiées comme négatives par le modèle.
- **Vrai Négatifs :** les instances négatives correctement classifiées comme négatives par le modèle.

Support : le nombre d'instances de chaque classe dans l'ensemble de données.

Moyenne Macro (macro-average) : représente la moyenne de la précision, du rappel et du F1-score sur l'ensemble des classes, elle calcule les moyennes non pondérées de ces mesures.

Moyenne Pondérée (weighted avg) : représente la moyenne de la précision, du rappel et du F1-score sur l'ensemble des classes, elle calcule les moyennes pondérées par le support de chaque classe.

Matrice de confusion : un tableau qui montre le nombre d'instances correctement et incorrectement classées. Elle présente les performances de classification de chaque classe sur les données de test. La diagonale principale montre le nombre de prédictions correctes, hors diagonale montre les erreurs de

classification. La diagonale principale représente les vrais positifs pour chaque classe, tandis que les cellules hors diagonale représentent les faux positifs et les faux négatifs.

Classe	Précision	Rappel	F1-score	Support	Classe	Précision	Rappel	F1-score	Support
A	0.94	0.96	0.95	482	K	0.97	0.96	0.96	1118
B	0.94	0.87	0.90	516	L	0.97	0.95	0.96	1112
C	0.93	0.97	0.95	526	M	0.94	0.94	0.94	1099
D	0.93	0.97	0.95	523	N	0.96	0.97	0.96	1101
E	0.98	0.94	0.96	496	Q	0.96	0.97	0.96	1077
F	0.95	0.92	0.93	513	R	0.97	0.95	0.96	648
G	0.99	0.92	0.96	515	S	0.96	0.93	0.95	779
H	0.95	0.96	0.95	479	T	0.96	0.91	0.93	959
I	0.94	0.96	0.95	486	U	0.77	0.84	0.80	609
J	0.95	0.93	0.94	483	W	0.95	0.93	0.94	896
X	0.94	0.96	0.95	924	Y	0.99	0.96	0.97	1068
Z	0.89	0.97	0.93	1131	Č	0.97	0.90	0.93	1034
Ɛ	0.97	0.97	0.97	972	Ɣ	0.94	0.96	0.95	1047
Ğ	0.94	0.95	0.94	1079	Ɣ	0.98	0.96	0.97	983
Ḍ	0.94	0.96	0.95	1079	Ḥ	0.94	0.89	0.92	940
Ṛ	0.96	0.91	0.94	1031	Ṣ	0.96	0.97	0.96	899
Ṭ	0.93	0.95	0.94	939	Ẓ	0.90	0.95	0.93	471
0	0.97	0.95	0.96	492	1	0.84	0.95	0.89	486
2	0.95	0.96	0.96	491	3	0.94	0.94	0.94	472
4	0.98	0.93	0.96	424	5	0.94	0.92	0.93	477
6	0.97	0.96	0.96	455	7	0.95	0.97	0.96	480
8	0.92	0.96	0.94	453	9	0.96	0.94	0.95	499
a	0.77	0.67	0.72	472	b	0.96	0.95	0.96	457
c	0.97	0.95	0.96	454	d	0.97	0.91	0.94	481
e	0.93	0.93	0.93	492	f	0.88	0.90	0.89	491
g	0.98	0.92	0.94	483	h	0.91	0.95	0.93	462
i	0.84	0.97	0.90	429	j	0.88	0.95	0.91	496
k	0.92	0.95	0.94	493	l	0.89	0.91	0.90	517
m	0.96	0.97	0.96	827	n	0.57	0.53	0.55	32
q	0.92	0.97	0.95	678	r	0.73	0.65	0.69	34
s	0.96	0.97	0.96	537	t	0.59	0.81	0.69	27
u	0.74	0.84	0.79	70	w	0.72	0.75	0.74	28
x	0.98	0.98	0.98	782	y	0.99	0.95	0.97	676
z	0.76	0.80	0.78	35	č	0.97	0.96	0.97	597
ğ	0.70	0.93	0.80	30	ɛ	0.97	0.97	0.97	982
ɣ	0.59	0.88	0.71	25	ḍ	0.97	0.98	0.98	767
ḥ	0.71	0.61	0.66	33	ṛ	0.99	0.97	0.98	730
ṣ	0.88	0.68	0.76	31	ṭ	0.95	0.97	0.96	554
ẓ	0.70	0.66	0.68	29					
accuracy								0.94	45974
macro avg						0.91	0.91	0.91	45974
weighted avg						0.94	0.94	0.94	45974

TABLE 5.1 – Tableau de précision, rappel, f1-score et support du premier modèle

En analysant les résultats montrés dans le tableau, nous avons remarqué que certains caractères présentent un taux de reconnaissance relativement bas par rapport aux autres classes comme le «a», «i», «u», «z», «ɣ», «ḥ», etc.

Certaines classes telles que «n», «r», «t», «u», «w», «z», «ǧ», «ɣ», «ḥ», «ṣ» et «ẓ» montrent des F1-scores faibles, ce qui signifie que le modèle rencontre des difficultés à prédire correctement ces classes. Par exemple, la classe «ɣ» a un F1-score de 0.71, ce qui est bas par rapport aux autres.

En revanche, certaines classes comme «A», «C», «D», «I», «X», «Y» et «ɛ» montrent des performances avec des F1-scores autour de 0.95 et plus, donc le modèle identifie et classe correctement la plupart des instaces réelles de ces classes.

Le support varie considérablement entre les classes, allant de 25 pour la classe «ɣ» à 1131 pour la classe «Z». Cela est dû à la dégradation du nombre d'instances entre chaque image, certaines images contiennent jusqu'à 7 000 instances, tandis que d'autres n'en contiennent que 200.

Compte tenu du nombre élevé de 77 classes, la taille de la matrice de confusion dépasse nos capacités d'affichage, ce qui la rend peu lisible. La matrice est disponible dans l'annexe 2.

Selon la matrice de confusion, les classes présentant une faible précision sont mal prédites. Par exemple, la classe «a» a été prédite comme classe «U» à 128 reprises, de même que la classe «U» a été prédite comme classe «a» à 48 reprises. Cela est dû à la ressemblance entre les deux lettres après le prétraitement, comme le montre cette figure :

FIGURE 5.5 – La ressemblance entre «U» et «a»

De plus, quelques classes ont été prédites comme d'autres classes entre 20 et 50 fois à cause du même problème, ce qui a réduit l'accuracy du modèle.

5.3.4 Phase de test

Une fois notre système exécuté, nous avons testé notre modèle sur un ensemble de test. Cet ensemble a subi les mêmes prétraitements que l'ensemble d'entraînement, mais il n'avait jamais été vu par notre modèle auparavant. Voici les résultats pour quelques images :

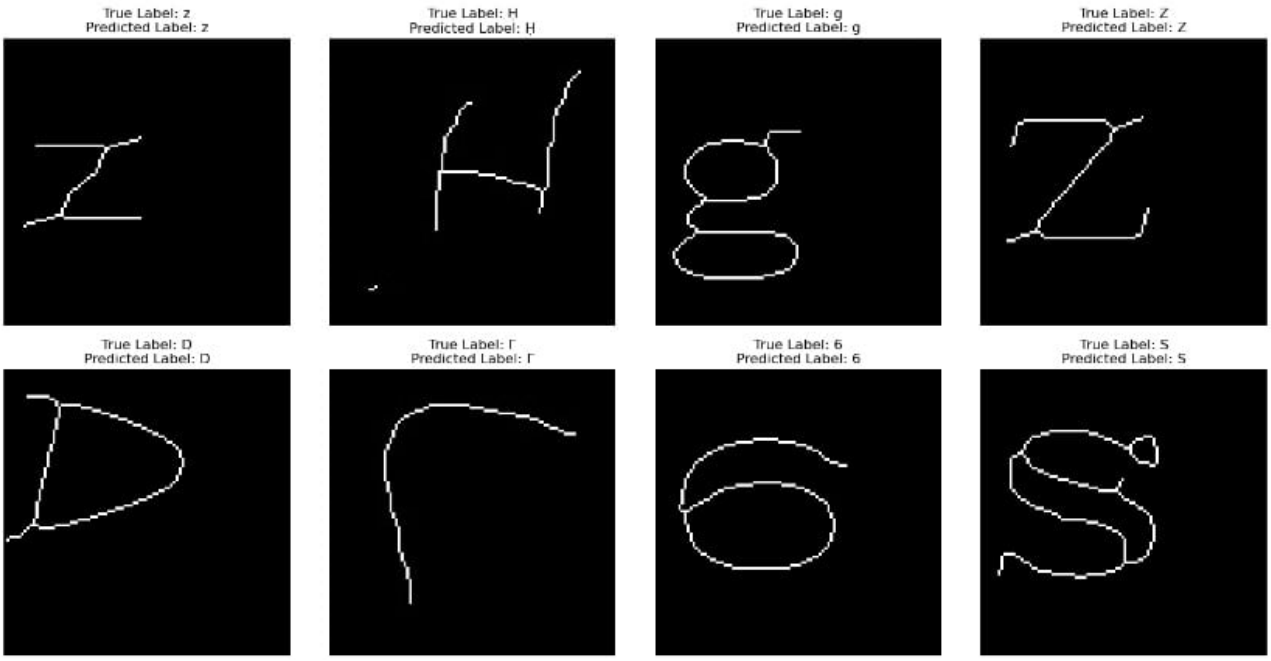


FIGURE 5.6 – Caractères bien prédits

En revanche, une image qui a été mal prédite, comme celle-ci :

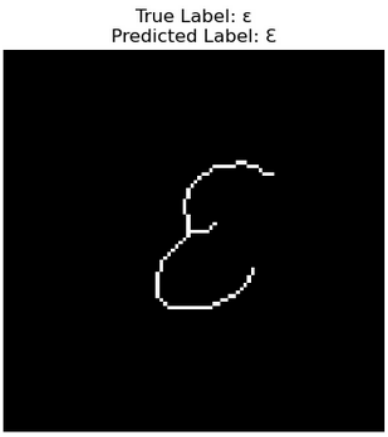


FIGURE 5.7 – Caractère mal prédit

5.3.5 Analyse et discussion

Notre modèle montre une bonne performance avec une précision globale de 94% et une perte de 0.2, mais présente des signes de surajustement. La courbe de précision de validation fluctue, suggérant des défis de généralisation malgré l'utilisation de techniques d'optimisation. Nous avons constaté un déséquilibre dans notre corpus, d'où la nécessité de faire une augmentation des données avec un nettoyage de certaines classes. Quelques classes, comme «ɣ» et «a», présentent des F1-scores faibles, probablement en raison de ce déséquilibre. En revanche, des classes comme «A» et «Z» atteignent des performances élevées, prouvant que notre modèle est efficace pour les classes avec un grand nombre d'instances.

5.4 Modèle prédéfini

5.4.1 Introduction

Concevoir et entraîner manuellement un réseau de neurones sur un corpus de données contenant plus de 300 000 images est une tâche complexe qui nécessite beaucoup de temps et une machine puissante.

Notre objectif est de trouver un modèle performant capable de prédire au mieux les données de notre corpus. C'est pourquoi nous avons opté pour l'utilisation d'un réseau CNN prédéfini. Ces réseaux sont optimisés et ajustés sur des ensembles de données massifs qui contiennent des millions d'images réparties en des milliers de catégories, ce qui permet au modèle d'apprendre à extraire les caractéristiques pertinentes.

Parmi les avantages de ces réseaux, il y a le fait que nous pouvons transférer les connaissances déjà acquises par ces réseaux vers nos propres données, au lieu de démarrer l'entraînement à partir de zéro. Cela permet de réduire le temps nécessaire pour atteindre une performance acceptable.

Les modèles comme DenseNet121 ont souvent des architectures sophistiquées qui captent des caractéristiques complexes. En utilisant ce dernier sur nos propres données, nous avons pu bénéficier de cette capacité à reconnaître nos images.

5.4.2 Architecture de DenseNet121

DenseNet121 est un type de CNN connu sous le nom de dense convolutional network, ou DenseNet. Son concept principal est l'utilisation de connexions denses entre les couches, où chaque couche reçoit les entrées de toutes les couches précédentes et transmet ses caractéristiques à toutes les couches suivantes.

Voici la structure du DenseNet121 :

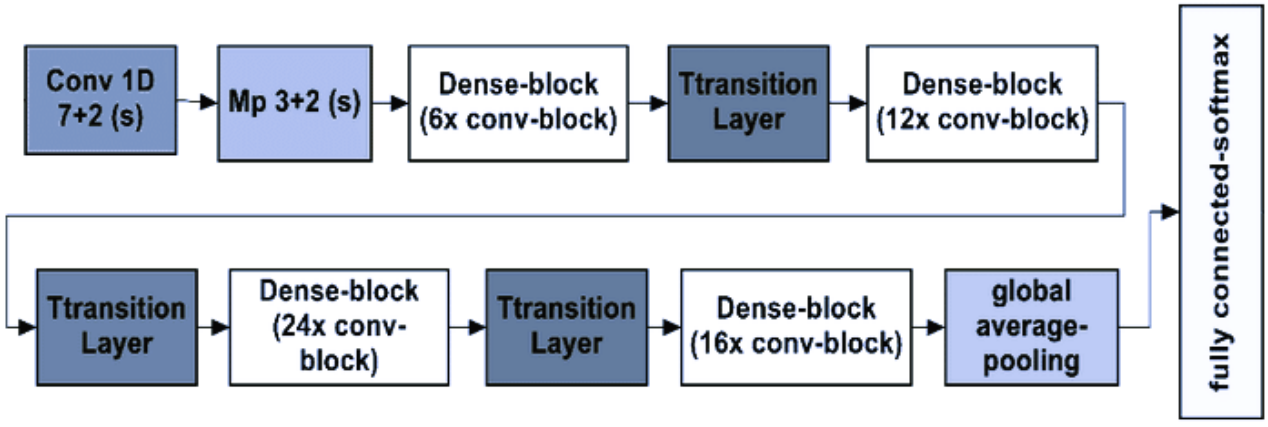


FIGURE 5.8 – Structure du DenseNet121

5.4.3 Résultat et interprétation

Ce modèle a été entraîné avec succès, avec un taux d'exactitude de 97.5% et une perte de moins de 0.1. Voici les graphes qui présentent l'historique de l'entraînement, et illustrent les courbes de perte et de précision pour les phases d'entraînement et de validation :

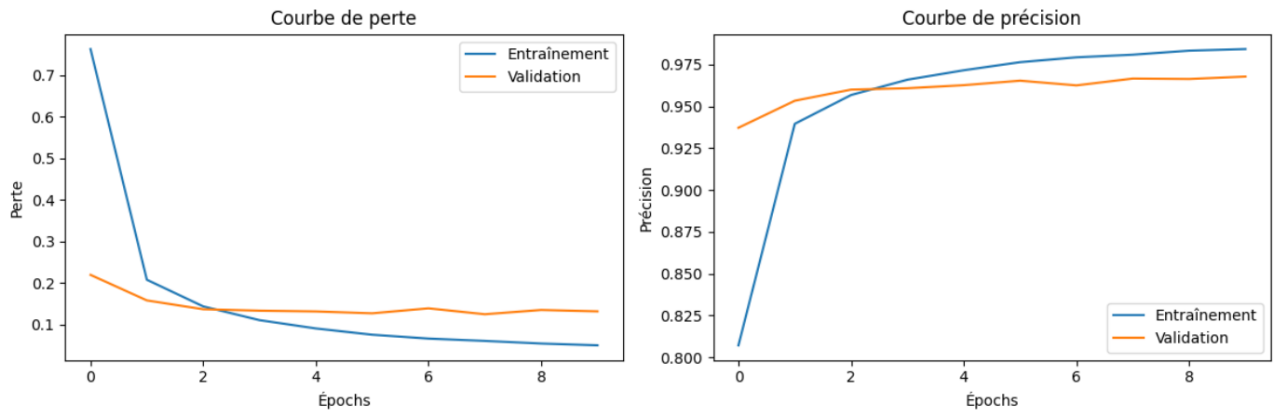


FIGURE 5.9 – L'historique de l'entraînement

5.4.4 Courbe de perte

La courbe de perte pour l'ensemble d'entraînement diminue rapidement, elle passe de plus de 0.7 à moins 0.1, cela indique un apprentissage efficace du modèle sur les données d'entraînement. La courbe de perte pour l'ensemble de validation suit une tendance similaire, mais plus stable, après environ 3 époques, elle se stabilise autour de 0.1.

5.4.5 Courbe de Précision (accuracy)

La courbe de précision montre une augmentation rapide atteignant environ 97.5% en moins de 10 époques, ce qui montre un fort apprentissage du modèle. La courbe de validation atteint également environ 97% et se stabilise après environ 3 époques, ce qui indique une bonne généralisation.

La proximité entre les courbes d'entraînement et de validation suggère que le modèle généralise bien. Donc ce modèle est performant et généralise de manière cohérente sur les données de validation.

5.4.6 Rapport de classification

Classe	Précision	Rappel	F1-score	Support	Classe	Précision	Rappel	F1-score	Support
0	0.98	0.99	0.98	935	1	0.96	0.92	0.94	1005
2	0.99	0.97	0.98	988	3	0.99	0.97	0.98	1016
4	0.99	0.97	0.98	1007	5	0.98	0.97	0.98	970
6	0.98	0.98	0.98	992	7	0.98	0.99	0.98	971
8	0.97	0.98	0.98	967	9	0.96	0.98	0.97	981
A	0.98	0.99	0.98	1528	B	0.98	0.97	0.98	1498
C	0.92	0.95	0.93	1497	D	0.98	0.97	0.97	1545
E	0.97	0.97	0.97	1500	F	0.98	0.98	0.98	1453
G	0.97	0.97	0.97	1497	H	0.96	0.98	0.97	1495
I	0.85	0.88	0.86	1469	J	0.96	0.98	0.97	1566
K	0.96	0.97	0.97	1488	L	0.98	0.98	0.98	1533
M	0.98	0.97	0.98	1547	N	0.97	0.97	0.97	1503
Q	0.94	0.99	0.96	1475	R	0.97	0.98	0.98	1500
S	0.95	0.93	0.94	1523	T	0.96	0.98	0.97	1451
U	0.96	0.98	0.97	1517	W	0.98	0.95	0.97	1549
X	0.93	0.95	0.94	1472	Y	0.97	0.97	0.97	1511
Z	0.91	0.97	0.94	1499	a	0.98	0.99	0.98	1177
b	0.99	0.97	0.98	1210	c	0.95	0.92	0.93	1166
d	0.98	0.99	0.99	1190	e	0.99	0.98	0.99	1208
f	0.98	0.98	0.98	1510	g	0.97	0.97	0.97	1161
h	0.98	0.98	0.98	1131	i	0.98	0.97	0.98	1547
j	0.98	0.97	0.98	1532	k	0.97	0.97	0.97	1224
l	0.82	0.80	0.81	1137	m	0.96	0.99	0.98	1131
n	0.99	0.98	0.98	1518	q	0.99	0.97	0.98	1198
r	0.97	0.98	0.98	1498	s	0.92	0.95	0.93	1248
t	0.99	0.98	0.98	1164	u	0.98	0.96	0.97	1144
w	0.95	0.96	0.96	1150	x	0.94	0.92	0.93	1213
y	0.99	0.96	0.97	1236	z	0.97	0.89	0.93	1466
Č	0.98	1.00	0.99	1516	č	0.98	0.97	0.97	305
Ɛ	0.94	0.85	0.89	927	Ɣ	0.92	0.68	0.78	53
Ğ	0.93	0.89	0.91	678	ğ	0.98	0.99	0.99	1140
ɛ	0.95	0.77	0.85	109	ɣ	0.94	0.99	0.96	1034
Γ	0.98	0.91	0.95	990	Ḍ	0.99	0.89	0.93	850
ḍ	0.97	1.00	0.99	1073	Ḥ	0.98	0.87	0.92	821
ḥ	0.95	1.00	0.97	1066	Ṛ	0.97	0.77	0.86	1260
ṛ	0.98	1.00	0.99	40	Ṣ	0.97	0.92	0.94	1008
ṣ	0.97	0.99	0.98	1077	Ṭ	0.99	0.94	0.96	1004
ṭ	0.94	0.94	0.94	35	Ẓ	0.99	0.87	0.93	752
ẓ	0.99	0.99	0.99	1125					
accuracy								0.97	93542
macro avg						0.97	0.97	0.97	93542
weighted avg						0.97	0.97	0.97	93542

TABLE 5.2 – Tableau de précision, rappel, f1-score et support de DenseNet121

D'après le tableau de précision ci-dessous, nous remarquons que :

- La plupart des classes ont des scores de précision, de rappel et de F1-score très élevés, souvent au-dessus de 0.97%, comme le «A», «ṛ», «ḍ», «ḥ», «n» et «č», ce qui montre que le modèle performe bien sur une large gamme de classes.

- Certaines classes telles que «Ɣ », «ɛ» et «Ṛ» montrent un rappel faible, le modèle rencontre des difficultés pour bien classifier ces classes.

Selon la matrice de confusion qui se trouve dans l'annexe 2, la majorité des prédictions se trouvent sur la diagonale principale, alors le modèle fait des prédictions correctes pour la plupart des instances.

Quelques classes ont été prédites comme d'autres classes, par exemple, le « W » majuscule a été prédit plusieurs fois comme un « w » minuscule, cela est dû au style des polices qui modifie la taille d'une majuscule et d'une minuscule. Même problème avec le « c » et le « C », le « Z » et le « z », le « S » et le « s », etc.

La classe «I» a été prédite comme classe «l» à 135 reprises, de même que la classe «l» a été prédite comme classe «I» à 187 reprises. Cela est dû à la ressemblance entre les deux caractères après le prétraitement, comme le montre cette figure :

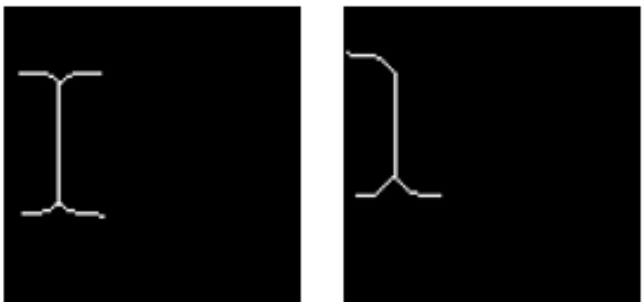


FIGURE 5.10 – La ressemblance entre «I» et «l»

5.4.7 Phase de test

Une fois le modèle exécuté, nous avons testé ce dernier sur un ensemble de test. Voici les résultats pour quelques images :

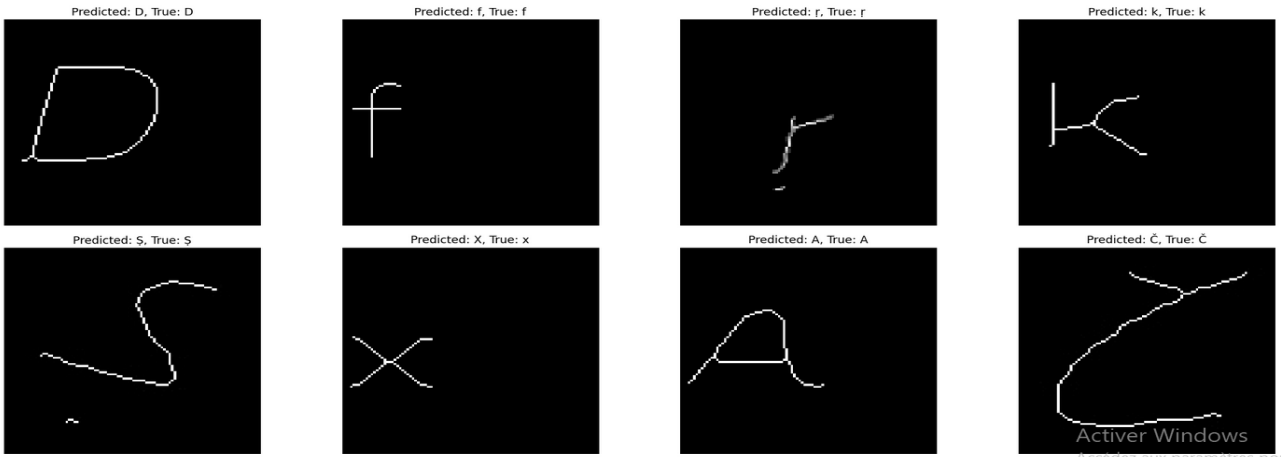


FIGURE 5.11 – Caractères bien prédits

En revanche, voici une image qui a été mal prédite :

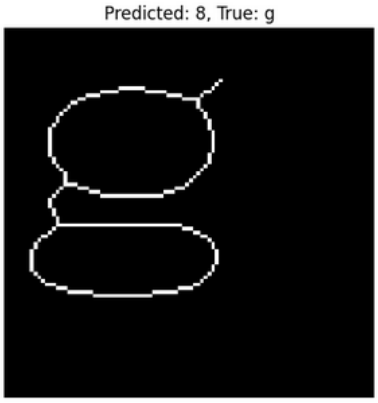


FIGURE 5.12 – Caractère mal prédit

5.4.8 Analyse et discussion

Le modèle montre une performance globale remarquable, avec une haute précision de 97% et une faible perte de 0.1.

L'analyse de la matrice de confusion et d'autres métriques fournit une vue d'ensemble claire des forces et des faiblesses de ce modèle. En général, ils donnent de bons résultats pour la plupart des classes.

Plusieurs facteurs ont contribué à la mauvaise reconnaissance de certains caractères, tels que le déséquilibre du nombre d'images entre les caractères. Les scores plus bas pour les classes avec peu d'images suggèrent un besoin de collecte de données pour améliorer leur précision.

Un deuxième facteur est la mauvaise écriture de certains caractères dans le corpus de l'ensemble de test, ce qui rend leur classification difficile même pour un humain. Cela est dû à l'utilisation de différentes polices, certaines ressemblent étroitement à l'écriture humaine.

Néanmoins, les classes avec un support important montrent des performances robustes, renforçant la fiabilité globale du modèle.

5.5 Comparaison entre les deux modèles

Le premier modèle comporte 4 couches CNN et 2 couches LSTM, tandis que le deuxième modèle est un modèle prédéfini, DenseNet121, qui contient de nombreuses couches convolutives denses et des couches de transition.

La courbe de perte des deux modèles en entraînement diminue régulièrement et significativement jusqu'à environ 0.1. En validation, la courbe de perte du premier modèle diminue rapidement au début, mais reste stable après les premières époques autour de 0.2. En revanche, la courbe de perte du deuxième modèle diminue jusqu'à atteindre 0.1.

La courbe de précision du premier modèle en entraînement augmente régulièrement pour atteindre presque 96%, tandis que celle du deuxième modèle augmente rapidement pour atteindre environ 97.5%. En validation, la courbe de précision du premier modèle fluctue légèrement mais reste autour de 94%. En comparaison, la courbe de précision du deuxième modèle augmente initialement, puis se stabilise

après l'époque 3 autour de 97%.

D'après les résultats, nous pouvons dire que le modèle 2 affiche des courbes plus stables en validation, sans signes évidents de surapprentissage. À l'inverse, le modèle 1 montre un potentiel de surapprentissage. En termes de précision, de rappel, de F1-score moyen pondéré et de précision macro moyenne, le modèle 2 présente une supériorité. De même, l'accuracy est également supérieure par rapport au modèle 1.

Le surapprentissage (overfitting) du premier modèle peut être dû à plusieurs raisons importantes. Parmi ces raisons, nous trouvons le déséquilibre des données entre chaque classe, ce qui a conduit à des courbes irrégulières, même si les performances semblent bonnes. En général, le modèle 2 semble meilleur en termes de métriques de performance.

Dans notre cas, il est nécessaire de bien entraîner le modèle et de garantir une bonne généralisation, afin d'assurer son bon fonctionnement avec des caractères nouveaux et variés, ainsi qu'avec différents styles d'écriture et tailles. Des erreurs dans la reconnaissance de ces caractères peuvent conduire à des incompréhensions pour de nouvelles données. Donc, le deuxième modèle est le meilleur choix. Un modèle stable implique une bonne généralisation des données.

Conclusion

Concevoir un système OCR pour la reconnaissance d'une nouvelle langue est un défi. Afin d'obtenir de bons résultats, plusieurs éléments doivent être présents. Le premier est le corpus de données, disposer d'un corpus annoté et varié joue un rôle important pour réaliser une reconnaissance efficace. Un autre point est la réalisation d'un prétraitement adéquat du corpus, afin d'extraire les caractéristiques pertinentes. Enfin, le choix du modèle pour la phase de classification. Il existe plusieurs types de modèles, allant de l'apprentissage automatique à l'apprentissage profond.

Dans ce chapitre, nous avons choisi d'utiliser les techniques de deep learning pour cette phase, grâce à leurs résultats impressionnants dans ce domaine ces dernières années.

Nous avons commencé par discuter de la phase d'apprentissage. Ensuite, nous avons présenté l'architecture de notre modèle proposé, en détaillant toutes les étapes que nous avons suivies. Nous avons également proposé un autre modèle prédéfini afin d'améliorer la précision et effectué une comparaison entre les deux modèles.

Enfin, nous avons réussi à créer un système OCR performant avec une précision de 97%.

Conclusion Générale et Perspectives

Dans ce travail, nous avons exploré la reconnaissance automatique des caractères latins en langue amazighe hors ligne. Pour ce faire, nous avons construit une base de données de caractères imprimés et proposé l'utilisation de l'apprentissage profond, en raison de ses résultats prometteurs dans le domaine du traitement de l'image.

Nous avons élaboré et préparé un corpus de données, structuré en trois niveaux : caractères, mots et phrases. On se focalisant principalement sur le premier niveau. Après la constitution du corpus, nous l'avons prétraité pour obtenir des images de haute qualité, en utilisant des techniques éprouvées telles que la méthode d'Otsu pour la binarisation, le filtre de moyennes non locales pour la réduction du bruit, et l'algorithme de Zhang-Suen pour la squelettisation.

Pour l'extraction de caractéristiques et la classification, nous nous sommes appuyés sur l'apprentissage profond, élaborant deux modèles distincts. Le premier modèle est une approche hybride qui combine des couches convolutionnelles et bidirectionnelles, tandis que le deuxième modèle repose sur DenseNet, une architecture préexistante. Une étude comparative de ces deux modèles sur notre corpus a montré des résultats impressionnants, avec des performances remarquables et satisfaisantes.

Cependant, plusieurs défis subsistent. La complexité des modèles nécessite des ressources informatiques puissantes, la phase d'apprentissage demeure extrêmement longue. La création du corpus a été un processus qui nécessite la collecte de nombreuses images dans diverses polices et tailles. Le prétraitement des images a présenté des difficultés, chaque méthode ne fonctionnant pas uniformément sur toutes les images. L'équilibrage des données des différents caractères dans le corpus a été une autre contrainte. Le manque de corpus de qualité et de références spécifiques dans le domaine de la reconnaissance des caractères amazighes a également entravé notre progression.

Malgré ces défis, nous avons réussi à atteindre un taux de reconnaissance de 97%, permettant à notre système de reconnaître la plupart des images de notre ensemble, y compris celles qui n'avaient jamais été vues auparavant. Cette avancée nous encourage à poursuivre nos efforts et à explorer de nouvelles perspectives dans le domaine de la reconnaissance d'images. Les perspectives envisageables sont les suivantes :

- Amélioration du corpus de données : enrichir le corpus et équilibrer les classes sont essentiels pour obtenir de meilleures performances.
- Amélioration du modèle : ajouter des couches, tester de nouveaux hyperparamètres, utiliser d'autres ajustements méthodologiques et appliquer des techniques d'optimisation supplémentaires.
- Utilisation d'autres méthodes de classification et continuation de ce travail pour la reconnaissance des mots et des phrases : en commencant par enrichir le corpus en utilisant des livres, des documents, etc.

En poursuivant ces axes d'amélioration, nous espérons faire progresser ce travail, et créer à l'avenir un système OCR reconnu pour la langue amazighe.

Bibliographie

[1] AHARRANE, N. *Contributions à la reconnaissance automatique hors ligne de l'écriture Amazighe imprimée et manuscrite en Tifinagh*. Thèse de doctorat, Université Sidi Mohamed Ben Abdellah, Faculté des Sciences-Dhar El Mahraz-, Fès, Décembre 2018.

[2] AICHOUCHE, A., AND MAROUANE, C. Recognition of arabic handwritten letters using deep learning approach. Master in artificial intelligence, Mohamed Boudiaf University - M'Sila, 2023.

[3] AIT ALI YAHIA, Y. Intégration des données dans un contexte big data. Mémoire de master, Université A. Mira - Béjaïa, 2020.

[4] AL-BADR, B., AND HARALICK, R. M. Symbol recognition without prior segmentation. In *Document Recognition* (1994), vol. 2181, Spie, pp. 303–314.

[5] AMARA, M. Reconnaissance des caractères arabes imprimés par l'approche neuro-génétique. Mémoire de master, Ecole nationale des sciences de l'informatique, 2012.

[6] AMROUCH, M. *Reconnaissance de caractères imprimés et manuscrits, textes et documents basée sur les modèles de Markov cachés*. PhD thesis, Université Ibnou Zohr, Faculté des Sciences, Agadir, 2012.

[7] AYECHE, F. *Traitement automatique d'images de visages algorithmes et architecture*. Thèse de doctorat en sciences en informatique, Université Farhat Abbas Sétif, avril 2021.

[8] BAKA, A. E., AND FILLALI, H. Traitement et reconnaissance des caracteres. Mémoire de master, Université M'hamed Bougara - Boumerdès, juin 2016.

[9] BENKESSIRAT, W., AND OUSSAMA, K. Elaboration d'un ocr basé sur les modèle de markov cachés : application aux textes arabe imprimé non voyellé. Mémoire de master, Université Saad Dahlab -Blida, 2020.

[10] BOUCETTA, A. Etude de l'effet des transformées de décorrélation en compression des images couleurs rgb. Mémoire de magister, Université de Batna 2, 2010.

[11] CENTRE DE RECHERCHE EN LANGUE ET CULTURE AMAZIGHES. Site officiel du centre de recherche en langue et culture amazighes, 2024. http://www.crlca.dz/.

[12] DAOUD, F., AND LOUALI, F. La reconnaissance des caractères arabes manuscrits par les réseaux neuronaux convolutionnels. Mémoire de master, Université Saad Dahlab -Blida, 2019.

[13] DE CAMPOS, T. E., AND BABU, BODLA RAKESH ET VARMA, M. Character recognition in natural images. In *International conference on computer vision theory and applications* (2009), vol. 1, Scitepress, pp. 273–280.

[14] DJABALLAH, M. A. E. Système de prédiction de la consommation d'énergie basé deep learning. Mémoire de master, Université de 8 Mai 1945 – Guelma, septembre 2021.

[15] EL GAJOUI, K., AND ALLAH, F. A. Vers un système de reconnaissance optique des caractères dans des documents multilingues : Français-amazighe. In *International Workshop IHMIM 14* (Hammamet, Tunisia, May 2014), Laboratoire de recherche en Informatique et Télécommunications.

[16] EL GAJOUI, K., ALLAH, F. A., AND OUMSIS, M. Diacritical language ocr based on neural network : Case of amazigh language. *Procedia computer science 73* (2015), 298–305.

[17] EL GAJOUI, K., ALLAH, F. A., AND OUMSIS, M. Recognition of amazigh language transcribed into latin based on polygonal approximation. *International Journal of Circuits, Systems and Signal Processing 10* (2016), 297.

[18] EL GAJOUI, K., ALLAH, F. A., AND OUMSIS, M. A corpus for amazigh transcribed to latin : Ocr systems' evaluation. *ARPN Journal of Engineering and Applied Sciences 13*, 22 (November 2018), 8797. ISSN 1819-6608.

[19] ES SAADY, Y. *Contribution au développement d'approches de reconnaissance automatique de caractères imprimés et manuscrits, de textes et de documents Amazighes*. Thèse de doctorat en sciences, Université Ibn Zohr, janvier 2012.

[20] GABARRA, E. *De la binarisation de documents vers la reconnaissance de symboles dans l'analyse de schémas électriques*. Thèse de doctorat, L'université de pau et des pays de l'adour côte basque, décembre 2008.

[21] GACI, Z. Quel système d'écriture pour la langue berbère (le kabyle) ? Mémoire de magister, Université Mouloud Mammeri De Tizi Ouzou, 2011.

[22] GAGAOUA, M. *Reconnaissance de l'écriture arabe manuscrite*. Thèse de doctorat, Université A.Mira - Bejaia, novembre 2021.

[23] HAITAAMAR, S. Segmentation de textes en caracteres pour la reconnaissance optique de l'ecriture arabe. Mémoire de magister, Université El-hadj Lakhdhar - Batna, juillet 2007.

[24] HU, S., WANG, Q. F., AND ET MIN WEN ET FRANS COENEN, K. H. Retrieval-based language model adaptation for handwritten chinese text recognition. *Journal of Machine Learning Research 25* (2022), 1001–1020.

[25] IOUKNANE, I., AND KHETACHE, B. Reconnaissance automatique de l'écriture amazighe. Mémoire de master, Université Abderrahmane Mira, 2022.

[26] KALAI, L. Corpus et contextualisation. *Journée d'Etude Doctorants et Jeunes chercheurs Institut Supérieur des Sciences Humaines, Université de Jendouba/Le corpus* (May 2017), 2. Jendouba, Tunisie. hal-03712999.

[27] KARDACHE, S., AND KHELIFA, D. Etude comparative de la dérivation en berbère (kabyle, tamazight du moyen atlas, touareg, mozabite et rifain). Mémoire de master, Université Mouloud Mammeri de Tizi-Ouzou, Laboratoire d'aménagement et d'enseignement de la langue amazighe, 2023.

[28] KARTHICK, K., RAVINDRAKUMAR, K., FRANCIS, R., AND ILANKANNAN, S. Steps involved in text recognition and recent research in ocr ; a study. *International Journal of Recent Technology and Engineering (IJRTE) 8*, 1 (May 2019), 2277–3878.

[29] KHALLA, I., AND SEDAIRIA, B. Apport de la lecture dans l'apprentissage de la phonétique cas des élèves de la 5 éme année primaire. Mémoire de master, Université 8 Mai 1945 -Guelma, juillet 2019.

[30] KIESSLING, B. *Avancées en Reconnaissance Optique des Caractères pour les Documents Arabes Historiques*. Thèse de doctorat, PSL University, Paris, France, avril 2021.

[31] LASRI, Y. Contribution à la reconnaissance optique (ocr) du texte arabe imprimé. Mémoire de master en sciences et techniques, Université Sidi Mohamed Ben Abdellah, juin 2014.

[32] MACHOUCHE, R., AND DAHMANI, F. La mise en place d'un système de contrôle interne au sein d'une entreprise, etude de cas : Centre de recherche en langue et culture amazighes bejaia. Mémoire de master, Université A.Mira - Bejaia, 2021.

[33] MEHENNAOUI, Z. Reconnaissance de l'ecriture arabe manuscrite a base des machines a vecteurs de supports. Mémoire de magister, Université Badji Mokhtar – Annaba, 2006.

[34] OUHNINI, A., AKSASSE, B., AND OUANAN, M. Towards an automatic speech-to-text transcription system : Amazigh language. *International Journal of Advanced Computer Science and Applications (IJACSA) 14*, 2 (2023), 413–418.

[35] OUTAHAJALA, M., AND ZENKOUAR, LAHBIB ET ROSSO, P. Construction d'un grand corpus annoté pour la langue amazighe. *Études et documents berbères 33*, 1 (2014), 77–94.

[36] PASCARELLA, A. *Neural Sequence Analysis Toolbox*. PhD thesis, 01 2021.

[37] RAKSHIT, R., KISKU, D. R., GUPTA, P., AND SING, J. Cross-resolution face identification using deep-convolutional neural network. *Multimedia Tools and Applications 80* (06 2021).

[38] REHAILIA, R. Une approche basée sur la prédiction de liens pour les systèmes de recommandation. Mémoire de master, Université de 8 Mai 1945 – Guelma, septembre 2021.

[39] SAIDJ, S. D. Techniques de nlp pour la détection des fausses nouvelles. Mémoire de master, Université Ibn Khaldoun -Tiaret, 2022.

[40] SEKLAOUI, W., AND AREZKI, R. Reconnaissance de mots manuscrits en utilisant le modèle de markov caché : Application aux noms d'auteurs arabes. Mémoire de master, Université Mouloud Mammeri de Tizi-Ouzou, juillet 2016.

[41] SEZGIN, M., AND SANKUR, B. Survey over image thresholding techniques and quantitative performance evaluation. *Journal of Electronic Imaging 13*, 1 (January 2004), 146–165.

[42] SHAN, L., LIU, Y., TANG, MIN ET YANG, M., AND BAI, X. Cnn-bilstm hybrid neural networks with attention mechanism for well log prediction. *Journal of Petroleum Science and Engineering 205* (2021), 108838.

[43] SHI, C., WANG, C., XIAO, B., ZHANG, Y., GAO, S., AND ZHANG, Z. Scene text recognition using part-based tree-structured character detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2013), pp. 2961–2968.

[44] SLIMANI, M., AND KHALED, A. Reconnaissance automatique des chiffres avec le deep learning. Mémoire de master, Université A.MIRA - Bejaïa, 2020.

[45] STUNER, B. *Cohorte de réseaux de neurones récurrents pour la reconnaissance de l'écriture*. Thèse de doctorat, Normandie Université, France, 2018. NNT : 2018NORMR024, tel-04214479.

[46] TRIER, O. D., AND JAIN, A. K. Goal-directed evaluation of binarization methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence 17* (1995), 1191–1201.

[47] TRIER, O. D., AND TAXT, T. Evaluation of binarization methods for document images. *IEEE Transactions on Pattern Analysis and Machine Intelligence 17* (1995), 312–314.

[48] ZAHAFI YASMINE NARIMENE, G. B. E.-G. A. Reconnaissance faciale par réseaux neuronaux convolutifs (cnns). Mémoire de master, Université Abdelhamid Ibn Badis - Mostganem, juin 2022.

[49] ZAIZ, F. *Technique basée puzzle/SVM pour l'amélioration de la reconnaissance du texte arabe manuscrit*. Thèse de doctorat en sciences, juillet 2017.

[50] ZARA, I. L'intelligence artificielle principe, outils et objectifs. Mémoire de master, Universite Badji Mokhtar - Annaba, 2019.

[51] ZHU, Y., YAO, C., AND BAI, X. Scene text detection and recognition : Recent advances and future trends. *Frontiers of Computer Science 10* (2016), 19–36.

Annexes

Annexe 1

Voici le code principal pour effectuer un prétraitement pour une image :

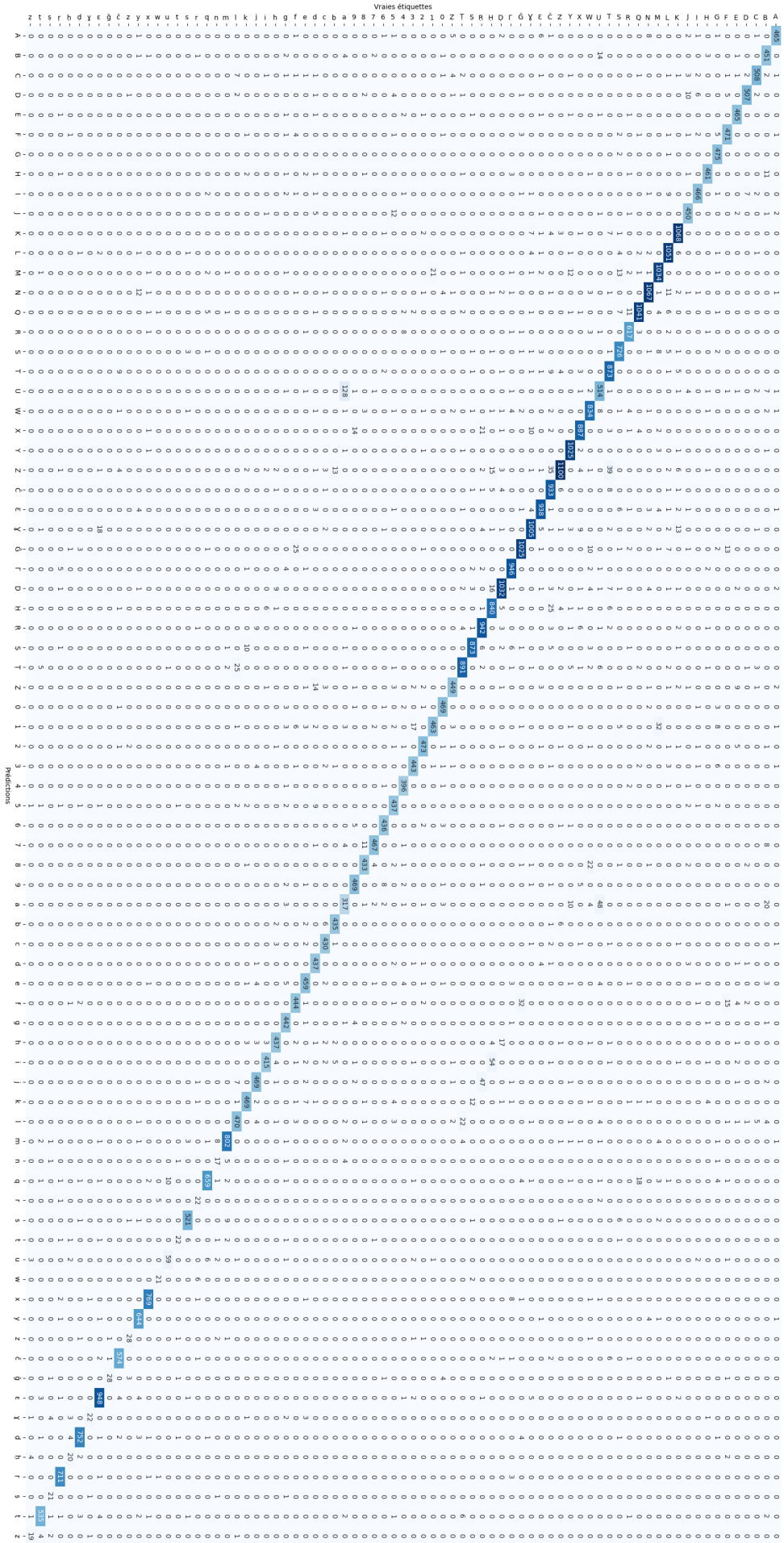
```
# Charger l'image :
image = cv2.imread('image.png')

# Convertir l'image en niveaux de gris :
gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

# Normaliser l'image :
normalized_gray_image = cv2.normalize(gray_image, None, 0, 255,
cv2.NORM_MINMAX)

# Reduire le bruit en appliquant la methode de non local means filter :
blurred_image = cv2.fastNlMeansDenoising(gray_image, None, 10,
7, 21)

# Inverser l'image seuillee pour obtenir l'ecriture en noir :
invert_thresholded = cv2.bitwise_not(blurred_image)

# Appliquer la binarisation d'Otsu sur l'image inversee :
_, binary_otsu = cv2.threshold(invert_thresholded, 0, 255,
cv2.THRESH_BINARY  + cv2.THRESH_OTSU)

# Appliquer la squelettisation :
skeleton = thinning(binary_otsu)

# Calculer la luminosite moyenne de l'image :
mean_brightness = np.mean(gray_image)

# Verifier si la luminosite est faible :
if mean_brightness < 200:
    # Appliquer la dilatation de l'ecriture :
    skeleton = apply_dilation(skeleton)
```

Annexe 2

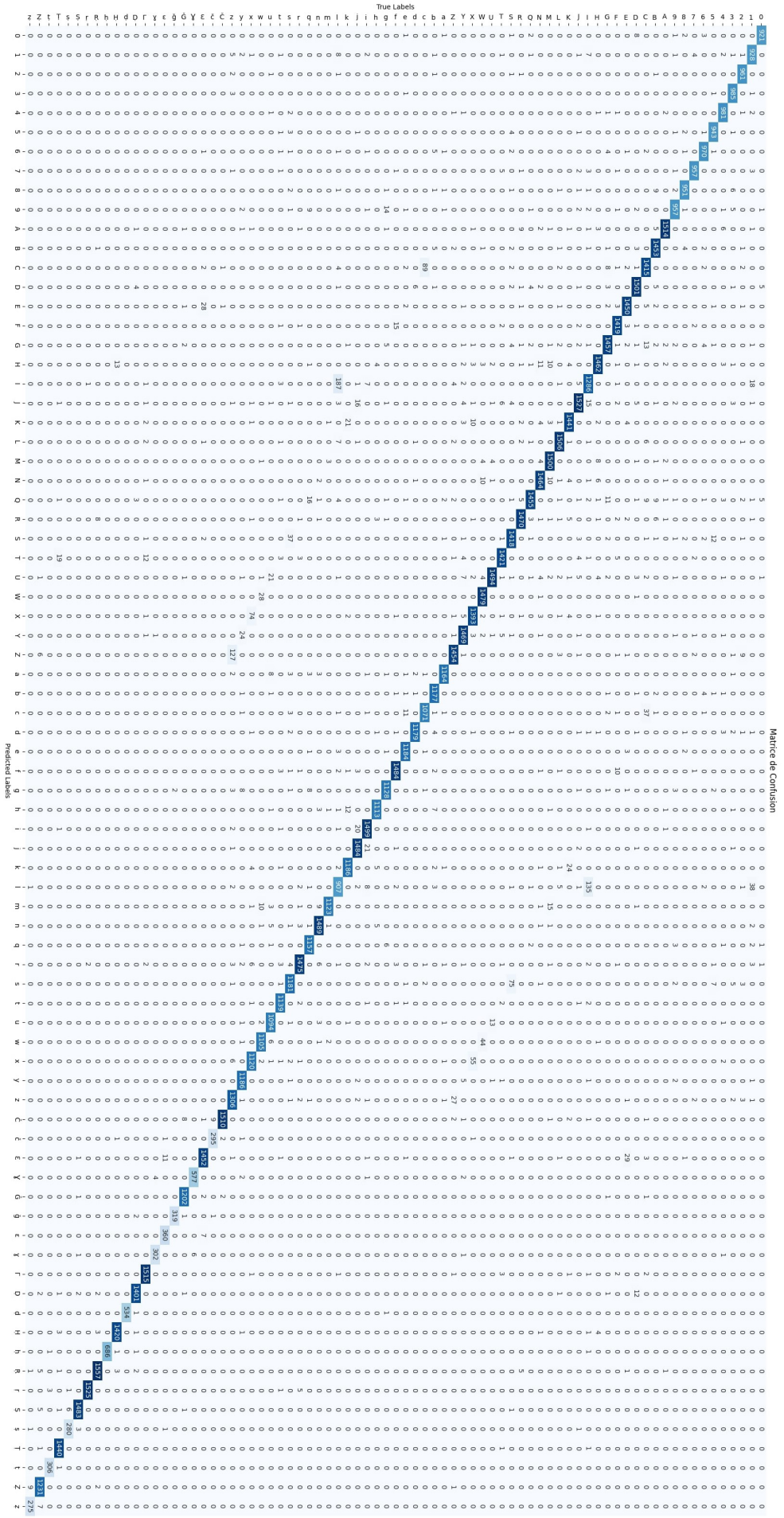
Nombre d'images d'entraînement, de validation et de test pour chaque label :

Label	Entraînement	Validation	Test	Label	Entraînement	Validation	Test
A	5056.0	1061.0	1118.0	0	2298.0	489.0	482.0
B	5118.0	1061.0	1112.0	1	2329.0	493.0	516.0
C	5168.0	1067.0	1099.0	2	2301.0	482.0	526.0
D	5115.0	1103.0	1101.0	3	2305.0	497.0	523.0
E	4942.0	1094.0	1077.0	4	2290.0	514.0	496.0
F	3153.0	677.0	648.0	5	2297.0	505.0	513.0
G	3593.0	800.0	779.0	6	2306.0	483.0	515.0
H	4397.0	906.0	959.0	7	2402.0	460.0	479.0
I	2876.0	602.0	609.0	8	2242.0	504.0	486.0
J	4424.0	954.0	896.0	9	2302.0	508.0	483.0
K	4396.0	963.0	924.0	a	2200.0	483.0	471.0
L	5108.0	1123.0	1068.0	b	2165.0	449.0	492.0
M	5082.0	1097.0	1131.0	c	2296.0	498.0	486.0
N	5097.0	1058.0	1034.0	d	2154.0	497.0	491.0
Q	4423.0	925.0	972.0	e	2301.0	488.0	472.0
R	5089.0	1091.0	1047.0	f	2239.0	490.0	424.0
S	5121.0	1077.0	1079.0	g	2116.0	517.0	477.0
T	4373.0	964.0	983.0	h	2215.0	462.0	455.0
U	5120.0	1127.0	1079.0	i	2240.0	450.0	480.0
W	4431.0	939.0	940.0	j	2204.0	479.0	453.0
X	4839.0	995.0	1031.0	k	2251.0	498.0	499.0
Y	4327.0	954.0	899.0	l	2230.0	470.0	472.0
Z	4380.0	955.0	939.0	m	2230.0	483.0	457.0
Č	3701.0	741.0	827.0	n	2263.0	407.0	454.0
Ɛ	3169.0	716.0	678.0	q	2217.0	481.0	481.0
Ɣ	199.0	47.0	34.0	r	2189.0	472.0	492.0
Ǧ	2292.0	544.0	537.0	s	2269.0	455.0	491.0
Γ	3663.0	792.0	782.0	t	2205.0	460.0	483.0
Ḍ	2942.0	641.0	676.0	u	2145.0	510.0	462.0
Ḥ	2907.0	593.0	597.0	w	2176.0	469.0	429.0
Ṛ	4431.0	956.0	982.0	x	2275.0	465.0	496.0
Ṣ	3539.0	806.0	767.0	y	2255.0	491.0	493.0
Ṭ	3689.0	788.0	730.0	z	2270.0	491.0	517.0
Ẓ	2531.0	537.0	554.0	č	149.0	28.0	32.0
				ǧ	155.0	31.0	27.0
				ɛ	349.0	64.0	70.0
				ɣ	137.0	33.0	28.0
				ḍ	140.0	33.0	35.0
				ḥ	142.0	35.0	30.0
				ṛ	143.0	34.0	25.0
				ṣ	156.0	24.0	33.0
				ṭ	156.0	21.0	31.0
				ẓ	150.0	17.0	29.0

Matrice de confusion du premier modèle :



Matrice de confusion du deuxième modèle :



Résumé

Cette étude se concentre sur la reconnaissance automatique hors ligne des caractères latins en langue amazighe. Un système OCR a été développé en utilisant l'apprentissage profond pour identifier les caractères. Un corpus de données a été construit, structuré en trois niveaux : caractères, mots et phrases. La méthodologie adoptée comprend la préparation et le prétraitement des images à l'aide de techniques de binarisation, de réduction du bruit et de squelettisation. Pour l'extraction des caractéristiques et la classification, deux modèles d'apprentissage profond ont été élaborés : un modèle hybride combinant des couches CNN et BiLSTM qui a atteint un taux de reconnaissance de 94%, et un modèle basé sur l'architecture DenseNet qui a obtenu un taux de reconnaissance de 97%.

Le résultat final montre une performance avec un taux de reconnaissance de 97% et une bonne précision pour la plupart des caractères étudiés dans ce travail.

Mots-clés : Reconnaissance automatique hors ligne, Caractères latins, Langue amazighe, Apprentissage profond, CNN, BiLSTM, DenseNet, Corpus, Modèle hybride.

Abstract

This study focuses on the offline automatic recognition of Latin characters in the Amazigh language. An OCR system was developed using deep learning to identify characters. A corpus of data was constructed, structured in three levels : characters, words and sentences. The methodology adopted includes the preparation and pre-processing of images using binarisation, noise reduction and skeletonisation techniques. For feature extraction and classification, two deep learning models were developed : a hybrid model combining CNN and BiLSTM layers that achieved a recognition rate of 94%, and a model based on the DenseNet architecture that achieved a recognition rate of 97%.

The final result shows a performance with a recognition rate of 97% and good accuracy for most of the characters studied in this work.

Keywords : Offline automatic recognition, Latin characters, Amazigh language, Deep learning, CNN, BiLSTM, DenseNet, Corpus, Hybrid model.