République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieure et de la Recherche Scientifique

Université Abderrahmane Mira

Faculté de la Technologie

Département d'Automatique, Télécommunication et d'Electronique

Projet de Fin d'Etudes

Pour l'obtention du diplôme de Master

Filière : Automatique

Spécialité : Automatique et informatique industrielle

Thème

Reconnaissance des caractères manuscrits de la langue berbère par Deep Learning appliqué à la base de données Berbère Dataset

Préparé par :

MESSAOUDI Sofiane

ARROUL Benyamine

Dirigé par :

Mr. L. TIGHZERT

Examiné par :

Mr. M. SADJI

Mem. M.GAGAOUA

Année universitaire : **2020/2021**

Remerciement

À l'issue de ce travail, on tient tout d'abord à exprimer notre entière reconnaissance à dieu tout puissant qui, sans lui, rien ne serait possible.

Patience, volonté et courage ont été nos armes afin d'aboutir à bon terme nos divers travaux au fil des années aboutissant à ce dernier, qui représente tant pour nous, récompense de nos efforts fournis mais aussi achèvement d'un long parcours marquant le début d'un second chapitre de notre vie.

La concrétisation de ce travail est tout d'abord le fruit de nos efforts et de notre dévouement à ce que la science peut fournir de meilleur mais c'est sans oublier l'encouragement et les conseils de personnes qui ont été au fil de ces années, nos guides et nos mentors afin de pouvoir servir l'intérêt commun.

Nos plus sincères et profonds remerciements à ***Mr TIGHZERT Lyes,*** *docteur, chercheur et encadreur, qui, en plus de sa bienveillance, nous a été d'une orientation remarquable, faisant de notre travail son intérêt majeur.*

On remercie vivement le président du jury ***Mr Sadji Moustapha*** *de nous avoir fait l'honneur de présider le jury.*

Nous tenons également à saluer ***Mme GAGAOUA Meriem*** *d'avoir accepté de juger ce modeste travail.*

Nos sincères et profonds remercîments également ver nos chers parents pour leurs soutiens et sacrifices qui ont contribué beaucoup d'efforts durant toute notre vie d'étude et notre cycle universitaire.

Nos profonds remerciements vont également à nos amis et toutes personnes ayant contribuer à ce modeste travail.

Dédicaces

C'est avec profonde gratitude et sincères mots, que je Dédie ce modeste travail de fin d'étude à :

Mes chers parents :

Les mots ne sauraient exprimer mon respect, mon amour, l'estime et ma considération pour tous les sacrifices que vous avez consenti pour mon instruction et mon bien être.

Je vous dédie ce travail, car c'est grâce à votre soutien durant tout mon parcours que cela a été réalisable. Je ne vous remercierai jamais assez pour vos efforts de jour comme de nuit pour mon éducation et ma formation.

J'espère qu'un jour, je pourrais vous rendre un peu de ce que vous avez fait pour moi, que dieu vous prête bonheur et longue vie.

A mes amis :

Khaled, Sara, Safia, Yousri, Nesrine, Hazam.

Pour mon binôme Benyamine :

Pour son sérieux et sa Compréhension pendant la réalisation de ce modeste travail.

Sofiane

Dédicaces

Je remercie Dieu ALLAH le tout puissant d'avoir pu achever ce modeste travail, que je dédie :

A ma chère MAMAN, unique dans son genre, et je tiens à la remercier, à l'occasion, pour ses longs sacrifices, sa belle compréhension et ses incessants soutiens et conseils répétitifs. Sachant que, ce présent temoignage de ma reconnaissance ne se résumeront JAMAIS : ni à ces quelques lignes, ni à ces quelques mots, vue qu'ils ne seront JAMAIS assez suffisants pour définir la gratitude que je lui dois et l'immense reconnaissance que je lui porte.

Je dédie aussi à ma seule et unique sœur Sara et mon seul frère younes, à lesquels je souhaite la réussite éternelle et le bonheur et l'épanouissement perpétuels dans l'avenir.

A mon binôme M^R MESSAOUDI Sofiane, sans oublier mes amis surtout MESSAOUDI Khaled pour son aide mais aussi d'autres amis qui m'ont aidé (Karim, Hazam, Yousri, Sara, Nesrine Kheloufi, Nesrine Teniche, Safia, Houda, Kenza).

Encore une autre fois, je vous remercie tous infiniment, sachant que je n'oublierai jamais votre immense service que vous m'avait rendu au long du cursus universitaire.

Sans oublier bien sûr, je dédie ce présent modeste travail de recherche, à l'ensemble de la communauté scientifique : chercheurs, enseignants, étudiants. Comme je le dédie à tous les honorables MARTYRS de notre cher pays, d'avoir tant sacrifié à l'indépendance de notre pays, en veillant à notre sérénité et sécurité et vie paisible.

Mes gratitudes vont bien droit à vous

ET à notre promotion 2020/2021 d'Automatique

Benyamine

Sommaire

Chapitre III

Liste des Figures

Liste des Tableaux

Introduction générale

Contexte

Nous vivons dans un monde ou le numérique prend de plus en plus de place dans nos vies, où les informations sont stockées, traitées, indexées et recherchées par des systèmes informatiques, ce qui fait de leur récupération une tâche peu coûteuse et rapide. Les documents manuscrits ne font pas exception à la règle. Avec la parole, l'écriture est encore aujourd'hui le moyen le plus naturel de communication et d'échange d'informations entre humains. Cependant, l'ère numérique demande des technologies capables de traduire ou de transcrire automatiquement des textes dans différentes langues, parlés ou écrites sur différents supports. La reconnaissance automatique de l'écriture manuscrite est une réponse à un tel besoin. La reconnaissance de l'écriture manuscrite consiste à transformer des images de textes manuscrits en leurs transcriptions ASCII ou Unicode [7]. Les enjeux de la reconnaissance automatique des documents manuscrits, ou textes manuscrits, sont multiples, allant du traitement des documents à leur indexation ou leur compréhension. En vision par ordinateur et en intelligence artificielle, la reconnaissance et l'analyse de texte basées sur des images jouent un rôle clé dans le processus de récupération de l'information qui y figure. Permettre à une technique d'apprentissage automatique de reconnaître les caractères manuscrits d'une langue spécifique nécessite un ensemble de données standard.

Problématique

La reconnaissance des textes manuscrits issus d'images présente deux difficultés majeures dans sa réalisation. La première est l'élaboration de bases de données suffisamment larges pour permettre d'entrainer et tester les systèmes de reconnaissance automatique. Il est primordial d'obtenir un ensemble de données représentatif de l'ensemble des cas que nous voulons généraliser. La reconnaissance optique de caractères manuscrits (OCR) est un domaine de recherche populaire en raison de ses diverses applications. De nombreux types de recherches ont déjà été menés pour la reconnaissance du bengali (BanglaLekha-Isolated), de la base de données des langues indiennes (Numeral Databases of Indian), de l'anglais (GoodNotes Handwriting Kollection), du chinois (HITMW database) et de l'arabe (AHTD), etc. Actuellement, la reconnaissance de caractères berbère est un domaine qui suscite de plus en plus d'intérêt, plusieurs personnes, dont des étudiants et des chercheurs se sont penché sur le

problème, parmi lesquels nous pouvons citer les travaux de Youcef Kherous et Djoudi Nadjet à l'université de Bejaia dans le cadre de leurs mémoires de fin de cycle « Réalisation d'une Nouvelle Base de Données pour l'Apprentissage Automatique de la Langue Amazigh par l'Intelligence Artificielle »[11], Youssef Es Saady, Ali Rachidi, Mostafa El Yassa, Driss Mammass « Amazigh Handwritten Character Recognition based on Horizontal and Vertical Centerline of Character »[12]**,** B. El Kessab, C. Daoui, B. Bouikhalene, R. Salouan « Handwriting Moroccan regions recognition using Tifinagh character»[4]. Cependant, à ce jour, il n'existe aucune base de données regroupant des images réelles de la langue berbère spécifique à la région kabyle algérienne. Nous nous sommes alors intéressés à cette problématique.

La deuxième difficulté est celle de l'approche à adopter pour accomplir cette tâche. La reconnaissance de caractères manuscrits n'est pas facile. Etonnamment, les humains sont prodigieusement doués pour donner un sens à ce que nos yeux nous montrent. Mais presque tout ce travail est fait inconsciemment. Et donc, nous ne comprenons généralement pas comment notre système visuel résout un problème de reconnaissance difficile. La difficulté de la reconnaissance visuelle des formes devient évidente si vous essayez d'écrire un programme informatique pour reconnaître des lettres. Ce qui semble facile quand nous le faisons nous-mêmes devient soudain extrêmement difficile. Des intuitions simples sur la façon dont nous reconnaissons les formes de tels caractères ne s'avèrent pas si simples à exprimer algorithmiquement. Lorsque vous essayez de rendre de telles règles précises, vous vous perdez rapidement dans une myriade d'exceptions, de mises en garde et de cas particuliers. On s'intéresse dans ce mémoire la reconnaissance automatique de la langue berbère [6].

Solution proposée

Pour y remédier, nous avons créé une base de données pour la reconnaissance des caractères manuscrits de la langue berbère, nommée Berber Dataset. Elle est constituée de 45 102 images manuscrites de l'alphabet berbère en néo tifinagh, étiqueté et séparé en 33 classes représentant chacune un caractère. Nous avons récolté un échantillon représentatif des variations de la forme des caractères, et cela écrit sur divers support papier (Feuille blanche, Feule à carreau, etc.)

Les réseaux de neurones convolutifs sont spécialement conçus pour gérer la variabilité des formes 2D, surpassent toutes les autres techniques. L'idée est de prendre un grand nombre de caractères manuscrits, appelés données d'apprentissage. Dans notre cas, il s'agit de la base de données *berbère dataset,* puis de développer un système qui peut apprendre de ces données.

En d'autres termes, le réseau de neurones utilise les exemples pour déduire automatiquement des règles de reconnaissance des lettres manuscrites. De plus, en augmentant le nombre de données d'apprentissage, le réseau peut en apprendre davantage sur l'écriture manuscrite et ainsi améliorer sa précision.

Organisation du mémoire

Ce mémoire est divisé en trois chapitres. Le premier chapitre est un état de l'art visant à définir diverses généralités concernant l'intelligence artificielle ainsi que ses applications. Le deuxième chapitre est consacré à l'étude approfondie des réseaux de neurones convolutifs (CNN) utilisés dans la reconnaissance de l'alphabet berbère. Et enfin, le chapitre trois explique les étapes de la création de la base de données ainsi que l'entrainement du réseau de neurones et les résultats obtenus.

CHAPITRE I

GENERALITES SUR L'INTELLIGENCE ARTIFFICIEL

I.1 Introduction

Dans ce chapitre, nous allons aborder le concept de vision artificiel et ses diffèrent domaines d'application notamment dans la classification automatique d'images qui permet la reconnaissance de formes, consistant à attribuer automatiquement une classe à une image à l'aide d'un système de classification. Pour ce faire, nous aurons recours à l'intelligence artificielle et plus précisément à l'apprentissage par Deep learning grâce aux algorithmes de réseaux de neurones artificiels. Ainsi, nous allons présenter clairement des généralités sur l'intelligence artificielle, l'apprentissage automatique, l'apprentissage profond et la vision Artificielle.

I.2 L'Intelligence Biologique

Le terme « intelligence » renvoie d'une part à une qualité personnelle, comme peuvent le laisser supposer des réflexions telles que « cet homme est intelligent » ou « qu'est-ce qu'elle est intelligente ! ». On peut d'emblée se demander ce qui fonde de telles appréciations, qui s'apparentent à des jugements de valeur. Certains auteurs ont défini l'intelligence comme une capacité générale à connaître ou à comprendre qui interviendrait quel que soit le domaine. Dans cette perspective, une personne intelligente sera une personne qui comprend bien et qui raisonne bien. Cette définition caractérise une conception unitaire de l'intelligence.[5]

I.3 L'Intelligence Artificielle :

On pourrait dire que l'Intelligence Artificielle (IA) est un ensemble de techniques permettant à des machines d'accomplir des tâches et de résoudre des problèmes normalement réservés aux humains et à certains animaux. Les tâches relevant de l'IA sont parfois très simples pour les humains, comme par exemple reconnaître et localiser les objets dans une image, planifier les mouvements d'un robot pour attraper un objet, ou conduire une voiture. Elles requièrent parfois de la planification complexe, comme par exemple pour jouer aux échecs ou au Go. Les tâches les plus compliquées requièrent beaucoup de connaissances et de sens commun, par exemple pour traduire un texte ou conduire un dialogue. Depuis quelques années, on associe presque toujours l'intelligence aux capacités d'apprentissage. C'est grâce à l'apprentissage qu'un système intelligent capable d'exécuter une tâche peut améliorer ses performances avec l'expérience. C'est grâce à l'apprentissage qu'il pourra apprendre à exécuter de nouvelles tâches et acquérir de nouvelles compétences. Les algorithmes et l'intelligence artificielle occupent une place déjà importante dans notre vie, et cela va s'accentuer avec le temps. Le « Deep Learning » est la technologie d'apprentissage automatique la plus

prometteuse, Le réseau peut traiter de grandes quantités d'informations et apprendre progressivement des images, du texte ou des données.[1]

I.4 Machine Learning

Les méthodes manuelles se sont avérées très difficiles à appliquer pour des tâches en apparence très simples comme la reconnaissance d'objets dans les images ou la reconnaissance vocale. Les données venant du monde réel, les échantillons d'un son ou les pixels d'une image, sont complexes, variables et entachées de bruit. Pour une machine, une image est un tableau de nombres indiquant la luminosité (ou la couleur) de chaque pixel, et un signal sonore une suite de nombres indiquant la pression de l'air à chaque instant.[1]

Le Machine Learning est une technique de modélisation qui implique des données. L'apprentissage automatique est une technique qui détermine le « modèle » à partir de « données ». Ici, les données signifient littéralement des informations telles que des documents, de l'audio, des images, etc. Le « modèle » est le produit final de l'apprentissage automatique.[2]

Le nom apprentissage lui-même indique que la technique analyse les données et trouve le modèle par elle-même plutôt que par un humain. Nous appelons cela « apprendre » parce que le processus ressemble à une formation avec des données pour résoudre le problème de la recherche d'un modèle. Par conséquent, les données utilisées par Machine Learning dans le processus de modélisation sont appelées données «de formation ».[2]

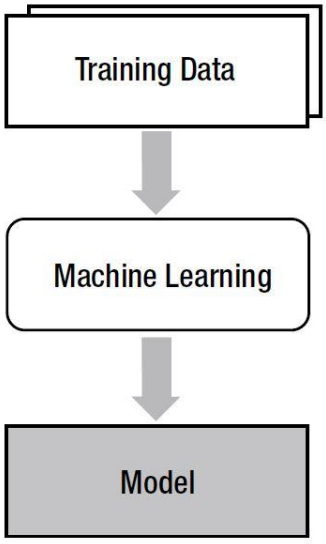

Figure I. 1 Que se passe-t-il pendant le processus d'apprentissage automatique [2]

On peut citer trois types d'apprentissage automatique :

1- Apprentissage supervisé.

2- Apprentissage non supervisé.

3- Apprentissage par renforcement.

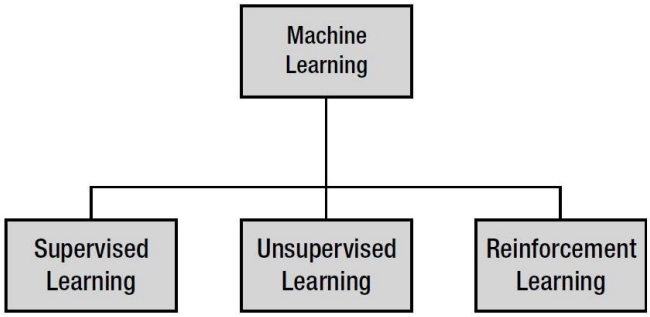

Figure I. 2 Les trois types d'apprentissage automatique [2]

I.5 Apprentissage supervisé

Dans sa forme la plus utilisée, l'apprentissage machine est supervisé : on montre en entrée de la machine une photo d'un objet, par exemple une voiture, et on lui donne la sortie désirée pour une voiture. Puis on lui montre la photo d'un chien avec la sortie désirée pour un chien. Après chaque exemple, la machine ajuste ses paramètres internes de manière à rapprocher sa sortie de la sortie désirée. Après avoir montré à la machine des milliers ou des millions d'exemples étiquetés avec leur catégorie, la machine devient capable de classifier correctement la plupart d'entre eux. Mais ce qui est plus intéressant, c'est qu'elle peut aussi classifier correctement des images de voiture ou de chien qu'elle n'a jamais vues durant la phase d'apprentissage. C'est ce qu'on appelle la capacité de généralisation [1]. Exemple d'apprentissage supervisé : créer un système qui reconnaisse et classe des images contenant une maison, une voiture ou un animal.

• 1ère étape : créer un grand jeu de données de ce type d'images. Chaque image est associée à une indication de la catégorie à laquelle elle appartient.

• 2ème étape - la formation / l'entraînement ("training") : une image est "montrée à la machine", ce qui produit un résultat sous forme d'un vecteur de scores pour chaque catégorie. La catégorie de l'image considérée doit obtenir le score le plus élevé (peu probable avant l'entraînement).

• 3ème étape : calcul d'une fonction objectif (moyennée sur tous les éléments du jeu

d'apprentissage) qui mesure la distance (l'erreur) entre ces scores de sortie et le modèle de scores souhaité. La machine (l'algorithme / le programme) modifie ensuite ses paramètres ajustables internes pour minimiser cette distance.

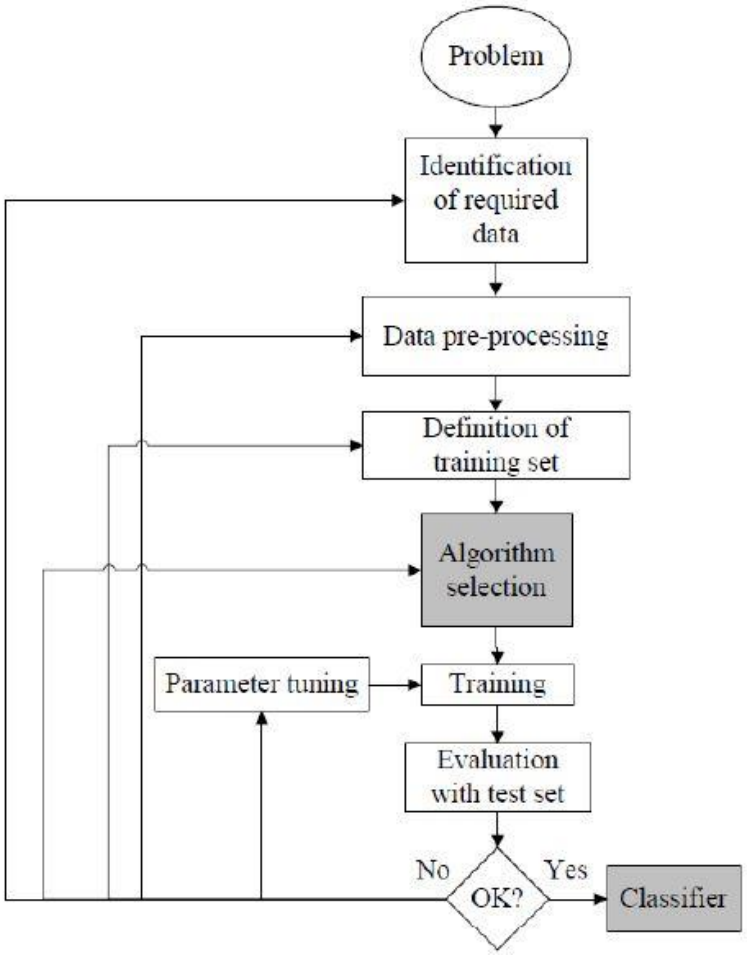

Figure I. 3 Le concept d'apprentissage supervisé en réseaux de neurones [2]

I.6 Deep learnning

Le problème de l'approche classique de la reconnaissance des formes est qu'un bon extracteur de caractéristiques est très difficile à construire, et qu'il doit être repensé pour chaque nouvelle application. C'est là qu'intervient l'apprentissage profond ou Deep learning en anglais. C'est une classe de méthodes dont les principes sont connus depuis la fin des années 80, mais dont l'utilisation ne s'est vraiment généralisée que depuis environ 2012. L'idée est très simple : le système entraînable est constitué d'une série de modules, chacun représentant une étape de traitement. Chaque module est entraînable, comportant des paramètres ajustables similaires aux poids des classifieurs linéaires. Le système est entraîné de bout en bout : à chaque exemple, tous les paramètres de tous les modules sont ajustés de manière à rapprocher la sortie

produite par le système de la sortie désirée. Le qualificatif profond vient de l'arrangement de ces modules en couches successives.

Dans sa réalisation la plus commune, une architecture profonde peut être vue comme un réseau multicouche d'éléments simples, similaires aux classifieurs linéaires, inter-connectés par des poids entraînables. C'est ce qu'on appelle un réseau neuronal multi-couches. Ce qui fait l'avantage des architectures profondes, c'est leur capacité d'apprendre à représenter le monde de manière hiérarchique. Comme toutes les couches sont entraînables, nul besoin de construire un extracteur de caractéristiques à la main. L'entraînement s'en chargera. De plus, les premières couches extrairont des caractéristiques simples (présence de contours) que les couches suivantes combineront pour former des concepts de plus en plus complexes et abstraits : assemblages de contours en motifs, de motifs en parties d'objets, de parties d'objets en objets, etc.[1]

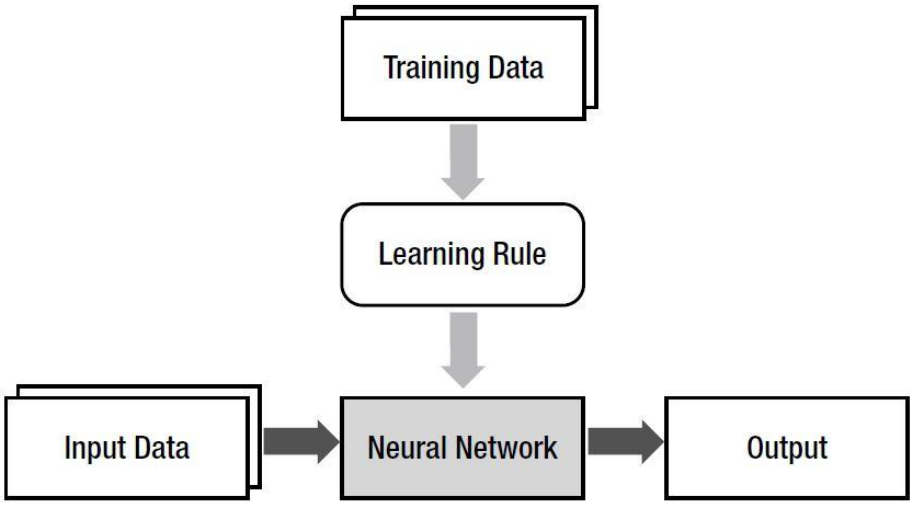

Figure I. 4 La relation entre le Machine Learning et les réseaux de neurones [2]

I.7 Neurone biologique

Le neurone est une cellule composée d'un corps cellulaire et d'un noyau. Le corps cellulaire se ramifie pour former ce que l'on nomme les dendrites. Celles-ci sont parfois si nombreuses que l'on parle alors de chevelure dendritique ou d'arborisation dendritique. C'est par les dendrites que l'information est acheminée de l'extérieur vers le soma, corps du neurone. L'information traitée par le neurone chemine ensuite le long de l'axone (unique) pour être transmise aux autres neurones. La transmission entre deux neurones n'est pas directe. En fait,

il existe un espace intercellulaire de quelques dizaines d'Angstroms (10-9 m) entre l'axone du neurone afférent et les dendrites (on dit une dendrite) du neurone efférent. La jonction entre deux neurones est appelée la synapse.[3]

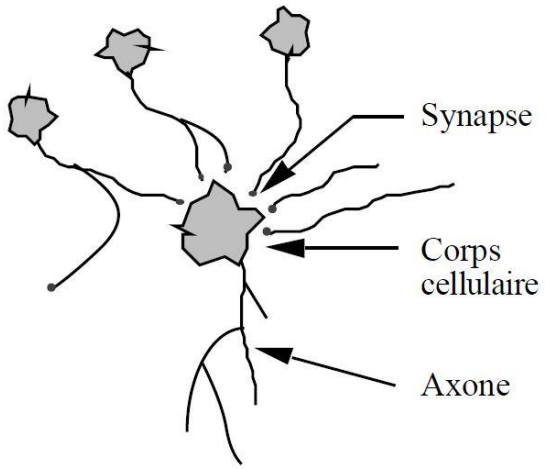

Figure I. 5 neurone biologique [3]

I.8 Neurone Artificiel

La figure 1-6 montre la structure d'un neurone artificiel. Chaque neurone artificiel est un processeur élémentaire. Il reçoit un nombre variable d'entrées en provenance de neurones amonts. A chacune de ces entrées est associée un poids w abréviation de weight (poids en anglais) représentatif de la force de la connexion. Chaque processeur élémentaire est doté d'une sortie unique, qui se ramifie ensuite pour alimenter un nombre variable de neurones avals. A chaque connexion est associé un poids.[3]

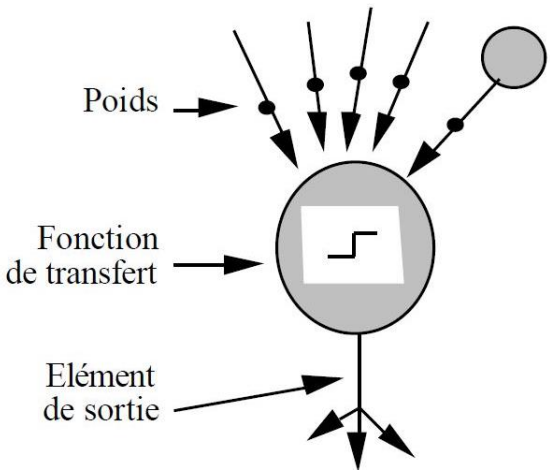

Figure I. 6 Un nœud (neurone) qui reçoit de multiples entrées [3]

I.9 Réseaux de neurones

Chaque fois que nous apprenons quelque chose, notre cerveau stocke les connaissances. L'ordinateur utilise la mémoire pour stocker des informations. Bien qu'ils stockent tous les deux des informations, leurs mécanismes sont très différents. L'ordinateur stocke des informations à des emplacements spécifiés de la mémoire, tandis que le cerveau modifie l'association des neurones. Le neurone lui-même n'a pas de capacité de stockage ; il ne fait que transmettre des signaux d'un neurone à l'autre. Le cerveau est un réseau gigantesque de ces neurones et l'association des neurones forme une information spécifique. Un réseau de neurones imite le mécanisme du cerveau. Comme le cerveau est composé de connexions de nombreux neurones, le réseau de neurones est construit avec des connexions de nœuds, qui sont des éléments correspondant aux neurones du cerveau. Le réseau de neurones imite l'association des neurones, qui est le mécanisme le plus important du cerveau, en utilisant la valeur de poids.[2]

Le perceptron est un exemple d'un simple réseau de neurones feedforward à une couche. Ce dernier était très prometteur, mais on a vite découvert qu'il présentait de sérieuses limitations car il ne fonctionne que pour les classes linéairement séparables. En 1969, Marvin Minsky et Seymour Papert ont démontré qu'il ne pouvait pas apprendre même une simple fonction logique telle que XOR. Cela a conduit à une baisse significative de l'intérêt pour le perceptron. Cependant, d'autres réseaux de neurones peuvent résoudre ce problème.

I.10 Réseaux de neurones multicouches

Un perceptron multicouche classique a plusieurs perceptrons interconnectés, tels que des unités organisées en différentes couches séquentielles (couche d'entrée, une ou plusieurs couches cachées et une couche de sortie). Chaque unité d'une couche est connectée à toutes les unités de la couche suivante. Tout d'abord, l'information est présentée à la couche d'entrée, puis nous l'utilisons pour calculer la sortie (ou l'activation), yi, pour chaque unité de la première couche cachée. Nous nous propageons vers l'avant, avec la sortie comme entrée pour les couches suivantes du réseau (donc feedforward), et ainsi de suite jusqu'à ce que nous atteignions la sortie. La façon la plus courante de former des réseaux de neurones consiste à utiliser une descente de gradient en combinaison avec une rétropropagation.

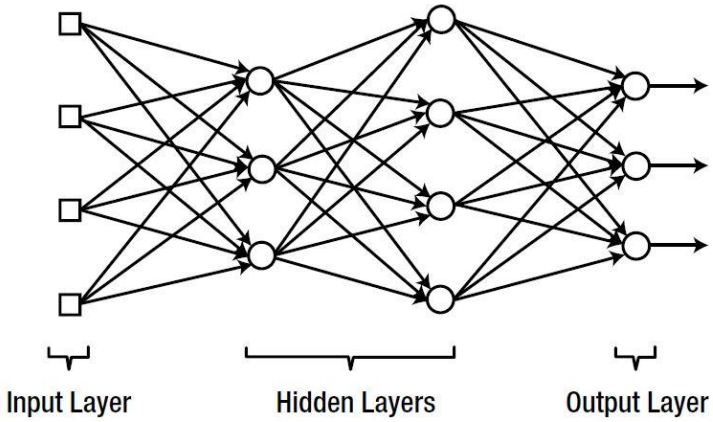

Figure I. 7 Une structure de réseaux de neurone profond [2]

I.11 Vision Artificielle

La vision artificielle est spécifiée par plusieurs disciplines, vision numérique, vision industrielle et vision par ordinateur, est une branche de l'intelligence artificielle qui permet de concevoir des machines qui peuvent voir, analyser, traiter et comprendre des images par un système d'acquisition, qui consiste à mettre en œuvre des outils et techniques (matériel et logiciel) pour permettre au système d'interpréter un élément visuel. La vision artificielle est une extension de la perception humaine. Elle utilise des moyens électronique et informatique pour doter les machines, autrement dit elle utilise des caméras avec des capteurs meilleurs que l'œil humain, les capteurs de la VA voient ce que l'homme ne voyons pas et fournie les connaissances nécessaires afin de permettre une interprétation non ambiguë.

I.12 Ce que le Deep learning permet à la vision par ordinateur

L'apprentissage en profondeur établit ses propres caractéristiques analytiques. Par conséquent, il peut identifier des fonctionnalités invisibles pour les humains, faciliter la recherche et automatiser les opérations, en économisant ainsi beaucoup de temps. Par exemple, les voitures autonomes utilisent ce système pour se localiser dans l'espace et réagir aux événements extérieurs. Le Deep Learning a également la capacité de transformer son réseau et d'inverser son processus. Autrement dit, une entrée similaire à la sortie précédente peut être fournie pour obtenir une image de sortie. Par conséquent, il permet à la vision par ordinateur d'approfondir ses principales fonctions et, surtout, de comprendre l'image analysée.

I.13 Les applications de l'intelligence artificielle

L'intelligence artificielle, définie comme intelligence présentée par les machines, a de nombreuses applications dans la société d'aujourd'hui. Plus précisément, c'est l'IA faible, la forme d'IA avec laquelle les programmes sont développés pour effectuer des tâches spécifiques, qui est utilisée pour un large éventail d'activités, y compris le diagnostic médical, le commerce électronique, le contrôle des robots et la télédétection. L'IA a été utilisée pour développer et faire progresser de nombreux domaines et industries, y compris la finance, la santé, l'éducation, le transport, et plus encore.

I.13.1 Surveillance et sécurité

La technologie de reconnaissance faciale basée sur l'apprentissage automatique est utilisée pour reconnaître les extrémistes dans les endroits bondés des visiteurs des centres de congrès, des aéroports et de divers autres événements importants. Maintenant, dans la situation pandémique de COVID-19, cette technologie s'avère très utile dans la communication et la sécurité sans contact. Ainsi, actuellement utilisé dans de nombreuses entreprises. En outre, la vision par ordinateur est utilisée dans la reconnaissance faciale à des fins de sécurité. Un algorithme reconnaît les visages de la personne puis autorise une accessibilité supplémentaire. De plus, utilisé pour la vérification automatique du système de présence dans les instituts professionnels. Cela offre une facilité par rapport aux méthodes conventionnelles telles que les clés, les cartes d'identité qui peuvent être facilement volées [5].

I.13.2 Reconnaissance de caractères manuscrits

Facilite le travail des organisations où les documents manuscrits sont volumineux. Par exemple, les universités, les centres d'examens, la police, etc. C'est un processus de numérisation et de numérisation de documents en quelques minutes [5].

I.13.3 La reconnaissance vocale

Un processus de traduction de mots prononcés en texte. Il offre des avantages aux systèmes de santé, militaires, dans les voitures ou pour créer des interfaces vocales et des assistants vocaux dans la vie quotidienne, car il contribue à améliorer l'accessibilité. La reconnaissance vocale est également appelée reconnaissance vocale et reconnaissance vocale automatique. Les divers algorithmes utilisés sont les réseaux de neurones artificiels, la quantification vectorielle, l'enveloppement dynamique du temps [5].

I.13.4 La santé

Grâce à la recherche, un logiciel spécial peut détecter avec précision toute variation chez les humains dans le service de santé. Il peut détecter divers paramètres en même temps et les traiter pour les dossiers médicaux dans des applications en temps réel. En outre, l'analyse statistique de la documentation médicale s'avère être une excellente référence [5].

I.13.5 La finance

Les prédictions basées sur des données historiques peuvent être effectuées à l'aide de l'apprentissage automatique. Diverses applications telles que les prévisions de prix des actions, la recherche scientifique, les campagnes de marketing et bien d'autres cas. Généralement, des réseaux de neurones artificiels et des algorithmes de forêt aléatoire sont utilisés pour les prédictions [5].

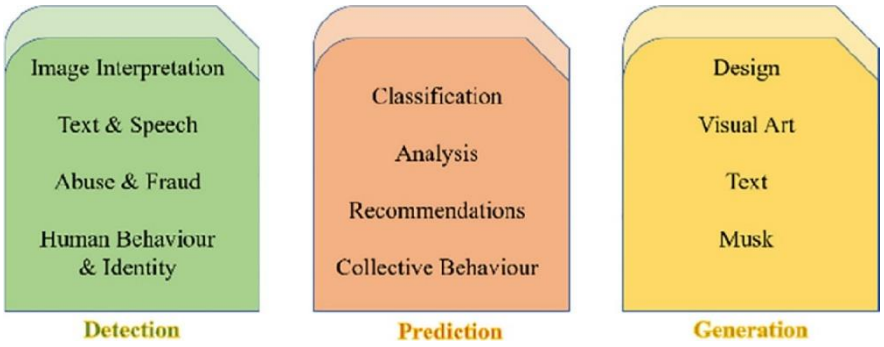

Figure I. 8 Applications of machine learning and deep learning **[5]**.

I.14 Conclusion

Dans ce chapitre, nous avons survolé les concepts de l'intelligence artificielle et on a donné une vision générale sur les méthodes ou approches de la classification. Comme toute technologie puissante, l'IA peut être utilisée pour le bénéfice de l'humanité entière ou pour le bénéfice d'un petit nombre aux dépens du plus grand nombre. Mais malgré tous ces progrès, nous sommes encore bien loin de produire des machines aussi intelligentes qu'un humain. Dans le chapitre suivant, nous allons nous focaliser sur une architecture neuronale artificielle bien particulière et très utilisée dans le domaine de l'intelligence artificielle. Il s'agit des réseaux CNN.

CHAPITRE II

ARCHITECTURE D’UN RESEAU DE NERUEUNE CONVOLUTIF

II.1 Introduction

Dans ce chapitre, nous allons voir en profondeur le fonctionnement des réseaux de neurones artificiels à travers les équations mathématiques qui définissent cette méthode. La façon la plus courante de les former consiste à utiliser l'algorithme de la descente du gradient à travers la rétropropagation. Ainsi, nous nous concentrerons sur un typer particulier de réseaux qui sont les réseaux de neurones convolutifs (ConvNet ou CNN) pour la classification automatique d'images. Ce même réseau sera exploité dans le chapitre suivant pour reconnaitre l'alphabet berbère.

II.2 Réseau de neurones convolutifs (ConvNet ou CNN)

Une architecture profonde particulièrement répandue est le réseau convolutif. Les réseaux convolutifs sont une forme particulière de réseau neuronal multi-couches dont l'architecture des connexions est inspirée de celle du cortex visuel des mammifères. Par exemple, chaque élément n'est connecté qu'à un petit nombre d'éléments voisins dans la couche précédente. Les réseaux convolutifs ont d'abord été utilisés pour la reconnaissance de caractères. Mais ces méthodes étaient plutôt difficiles à mettre en œuvre avec les ordinateurs de l'époque, et malgré ce succès, les réseaux convolutifs, et les réseaux neuronaux plus généralement, ont été délaissés par la communauté de la recherche entre 1997 et 2012.

En 2011-2012, trois évènements ont soudainement changé la donne. Tout d'abord, les GPUs (Graphical Processing Units) capables de plus de mille milliards d'opérations par seconde sont devenus disponibles à des prix plus abordables. Ces puissants processeurs spécialisés, initialement conçus pour le rendu graphique des jeux vidéo, se sont avérés être très performants pour les calculs des réseaux neuronaux. Deuxièmement, des expériences menées simultanément à Microsoft, Google et IBM avec l'aide du laboratoire de Geoff Hinton, ont montré que les réseaux profonds pouvaient diminuer de moitié les taux d'erreur des systèmes de reconnaissance vocale. Troisièmement, plusieurs records en reconnaissance d'image ont été battus par des réseaux convolutifs. L'évènement le plus marquant a été la victoire éclatante de l'équipe de Toronto dans la compétition de reconnaissance d'objets "ImageNet". La diminution des taux d'erreurs était telle qu'une véritable révolution d'une rapidité inouïe s'est déroulée. Du jour au lendemain, la majorité des équipes de recherche en parole et en vision ont abandonné leurs méthodes préférées et sont passées au réseaux convolutifs et autres réseaux neuronaux [1].

II.3 Architecture du réseau de neurone convolutif conçu

Le réseau d'extraction d'entités contient une seule couche de convolution avec 20 filtres de convolution de 7×7. La sortie de la couche de convolution passe par la fonction ReLU, suivie de la couche de regroupement (*pooling*). La couche de *pooling* utilise le processus *mean pooling* de deux par deux. Le réseau neuronal de classification se compose d'une seule couche cachée et d'une couche de sortie. Cette couche cachée a 330 nœuds (neurones) qui utilisent la fonction d'activation ReLU. Puisque nous avons 33 classes à classer, la couche de sortie est construite avec 33 nœuds. Nous utilisons la fonction d'activation softmax pour les nœuds de sortie. La figure suivante montre l'architecture de ce réseau de neurones.

.

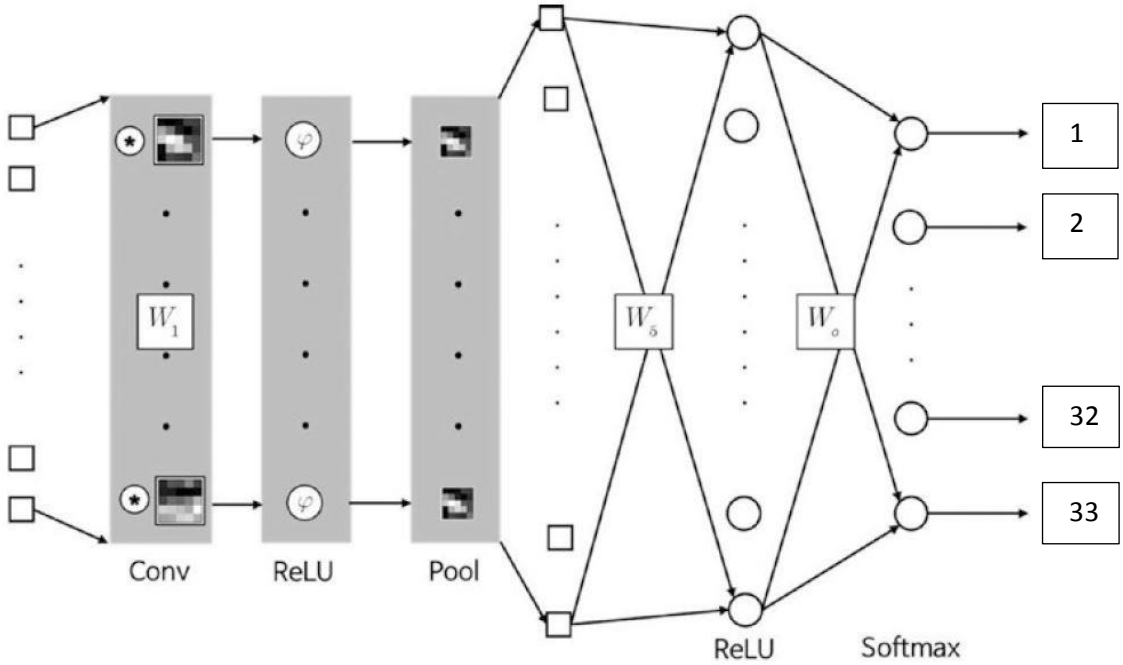

Figure II. 1 Architecture du réseau de neurone convolutif [2]

Bien qu'il comporte de nombreuses couches, seules trois d'entre elles contiennent les matrices de poids qui nécessitent une adaptation ; ils sont W1, W5 et Wo dans les blocs carrés. W5 et Wo contiennent les poids de connexion du réseau neuronal de classification, tandis que W1 est le poids de la couche de convolution, qui est utilisé par les filtres de convolution pour le traitement d'image

II.3.1 Couche de convolution

Cette section explique comment fonctionne la couche de convolution, qui sert à l'extraction d'entités à partir d'images. La couche de convolution génère de nouvelles images appelées cartes d'entités. La carte des entités accentue les caractéristiques uniques de l'image d'origine. La couche de convolution fonctionne de manière très différente par rapport aux autres couches du réseau neuronal. Cette couche n'utilise pas une somme pondérée des connexions. Au lieu de cela, elle contient des filtres qui *convertissent les images*. Nous appellerons ces filtres des filtres convolutifs. Le processus d'entrée de l'image à travers les filtres de convolution donne la carte des caractéristiques (des entités) [2].

II.3.2 Convolution matricielle

Une image est représentée généralement sur ordinateur comme une matrice. En noir est blanc, le cas le plus simple, serait de considérer chaque pixel en binaire (noir =1, blanc=0). Le pixel peut contenir aussi le niveau en gris (méthode utilisée dans les écrans noir et blanc). Les images en couleurs sont codées en RGB, c'est-à-dire trois matrices (R pour le niveau de la couleur rouge, G pour le niveau de la couleur verte, B pour le niveau de la couleur bleue), ceci donne trois valeurs pour chaque pixel. Nous allons alors considérer l'image comme une matrice [2].

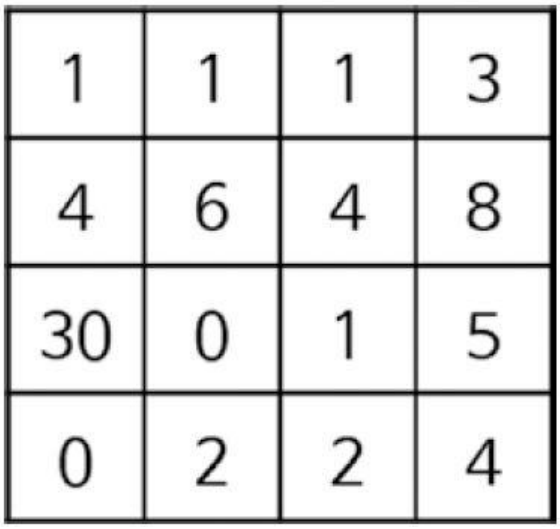

Figure II. 2 Image d'entrée de dimension 4x4 [2]

Il convient de noter que les filtres du ConvNet réel sont déterminés par le processus d'apprentissage et non par décision manuelle. Voici l'exemple de filtre que nous utiliserons pour illustrer le processus de convolution.

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Regardons maintenant la convolution suivante entre une image de 4×4 pixels et un filtre de 2×2.

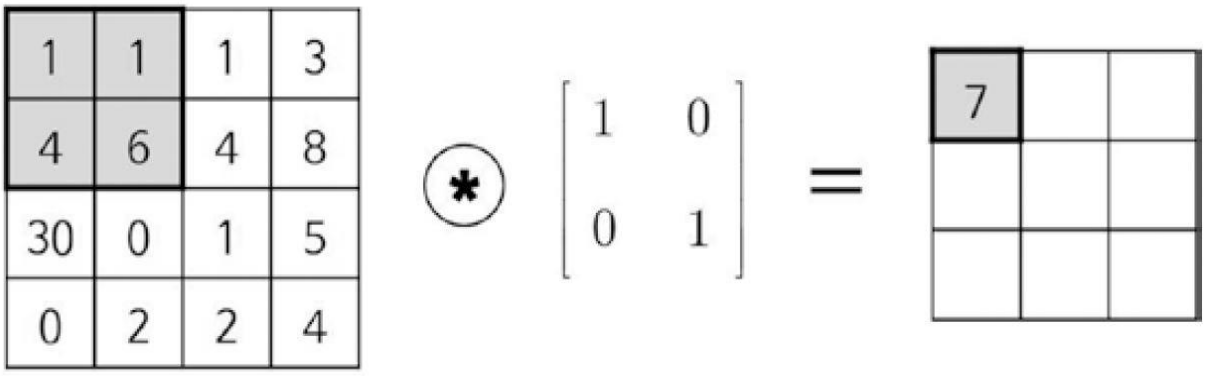

Figure II. 3 L'opération de convolution commence dans la partie supérieur droite **[2]**

L'opération de convolution est la somme des produits des éléments qui se trouvent aux mêmes positions des deux matrices. Le résultat dans le calcul précédent est calculé comme suit

$$(\mathbf{1} * \mathbf{1}) + (\mathbf{1} * \mathbf{0}) + (\mathbf{4} * \mathbf{0}) + (\mathbf{6} * \mathbf{1}) = \mathbf{7}$$

L'opération sera répétée jusqu'à l'obtention du résultat montré dans la **figure II.4**. La convolutions complète donne alors :

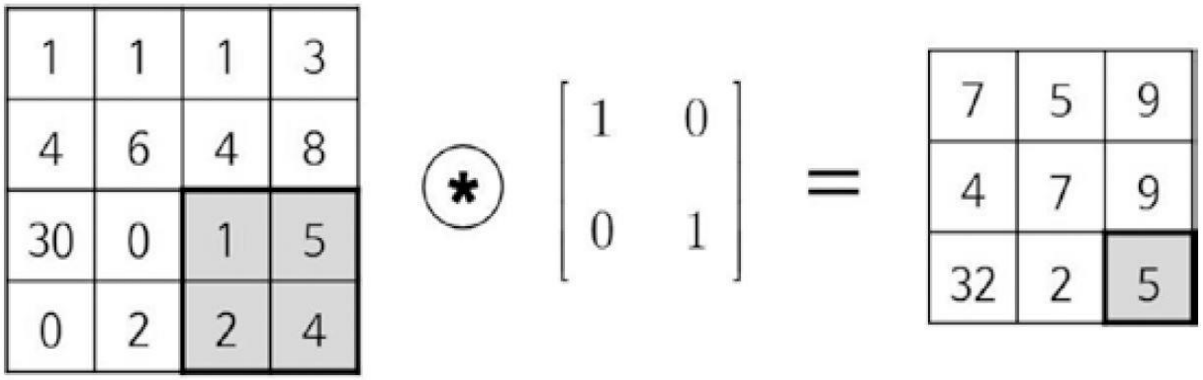

Figure II. 4 Le résultat final de la convolution **[2]**

II.3.3 Couche de pooling

La couche de regroupement (*pooling*) réduit la taille de l'image, car elle combine les pixels voisins d'une certaine zone de l'image en une seule valeur représentative. Afin de mener les opérations dans la couche de *pooling*, nous devons déterminer comment sélectionner les pixels de *pooling* de l'image et comment définir la valeur représentative. Les pixels voisins sont généralement sélectionnés dans la matrice carrée et le nombre de pixels combinés diffère d'un problème à l'autre. La valeur représentative est généralement définie comme la moyenne (mean pooli*n*g) ou le maximum (max *pooling*) des pixels sélectionnés. Le *pooling* permet de gros gains en puissance de calcul. Cependant, en raison de la réduction agressive de la taille de la représentation (et donc de la perte d'information associée), la tendance actuelle est d'utiliser de petits filtres (type 2x2) [12]. Considérons l'image d'entrée de 4 × 4 pixels, qui est exprimée par la matrice de **la figure II.2** a laquelle nous appliquerons l'opération de pooling [2] :

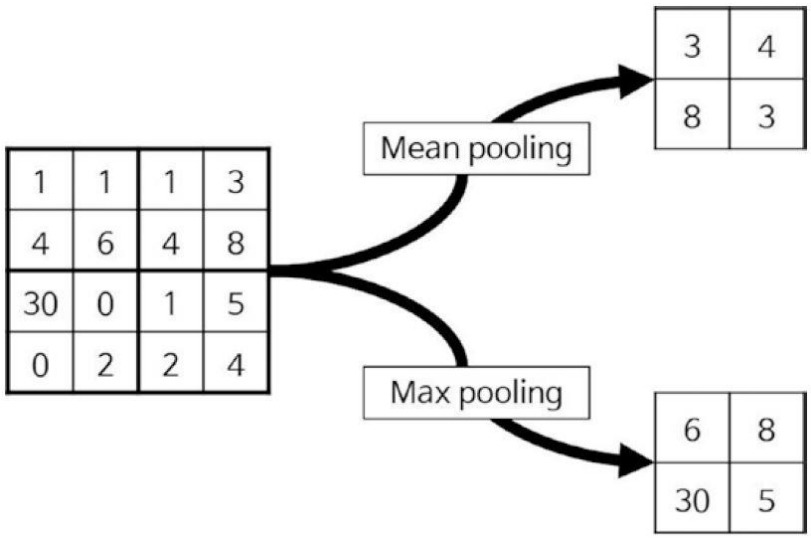

Figure II. 5 Illustration de l'opération de pooling **[2]**

En fait, dans un sens mathématique, le processus de regroupement (pooling) est un type d'opération de convolution. La différence avec la couche de convolution est que le filtre de convolution est stationnaire et que les zones de convolution ne se chevauchent pas. La couche pooling compense dans une certaine mesure les objets excentriques et inclinés. Par exemple, elle peut améliorer la reconnaissance d'un chat, qui peut être décentré dans l'image d'entrée. De plus, comme le processus pooling réduit la taille de l'image, il est très bénéfique pour soulager la charge de calcul et empêcher la surinterprétation.

II.3.4 Couche Rectified Linear Units ReLU

La fonction d'activation d'unité linéaire rectifiée (ReLU) a été proposée par Nair et Hinton 2010, et depuis, a été la fonction d'activation la plus utilisée pour les applications d'apprentissage en profondeur avec des résultats de pointe à ce jour. ReLU est une fonction d'activation permettant un apprentissage plus rapide, elle s'est avéré être la fonction la plus efficace et la plus largement utilisée. Elle offre les meilleures performances et généralisations en apprentissage profond par rapport aux fonctions d'activation Sigmoïde et tanh. ReLU représente une fonction presque linéaire et préserve donc les propriétés des modèles linéaires qui les rendaient faciles à optimiser, avec des méthodes de descente de gradient. La fonction ReLU ou Rectified Linear Units est appliquée après la fonction de convolution et laisse les dimensions inchangées. Elle désigne une fonction réelle non linéaire définie par :

$$f(x) = \max(0, x) = \begin{cases} x_i, & if\ x_i \geq 0 \\ 0, & if\ x_i < 0 \end{cases} \textbf{(II.1)}$$

La couche de correction ReLU remplace donc toutes les valeurs négatives reçues en entrées par des zéros. Elle joue le rôle de fonction d'activation et a pour but d'introduire des complexités non-linéaires au réseau. La figure 2-6 montre la courbe de la fonction d'activation Relu [10].

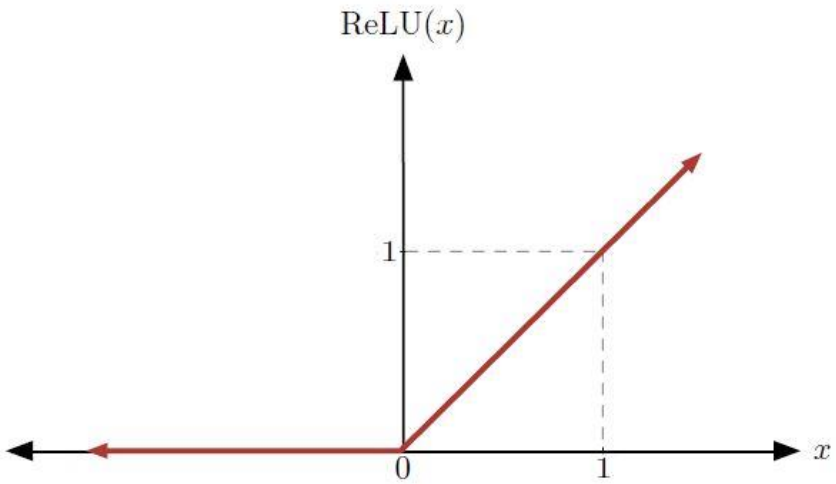

Figure II. 6 courbe de la fonction Relu **[10]**

II.3.5 Reshape

La fonction d'activation Reshape est une opération importante dans notre réseau de neurones puisqu'elle transforme les données de son entrée, qui sont un ensemble de matrices, en un seul vecteur.

II.3.6 Couche softmax

La fonction Softmax est un autre type de fonction d'activation utilisée dans les réseaux neuronaux. Elle est utilisée pour calculer la distribution de probabilité à partir d'un vecteur de nombres réels. La fonction Softmax produit une sortie qui est une plage de valeurs entre 0 et 1, avec la somme des probabilités égale à 1. La fonction Softmax est calculée en utilisant la relation suivante : [10]

$$f(x_i) = \frac{exp(x_i)}{\sum_j exp(x_j)} \quad \textbf{(II.2)}$$

La fonction Softmax est utilisée dans les modèles multi-classes où elle renvoie les probabilités de chaque classe, la classe ciblée ayant la probabilité la plus élevée. La fonction Softmax apparaît dans presque toutes les couches de sortie des architectures de deep learning, où elles sont utilisées.[10]

II.4 Apprentissage d'un réseau CNN

La propagation directe du signal et la rétropropagation des erreurs suivent des règles spéciales pour chaque couche. La première phase est appelée propagation directe, où le signal est propagé depuis les entrées du CNN jusqu'à sa sortie, Dans la dernière couche, la sortie est comparée à la valeur souhaitée par fonction de coût puis l'erreur est estimée. Dans la deuxième phase, un algorithme de rétropropagation est utilisé pour optimiser les paramètres variables du réseau grâce à un algorithme de descente de gradient.

II.4.1 Propagation directe

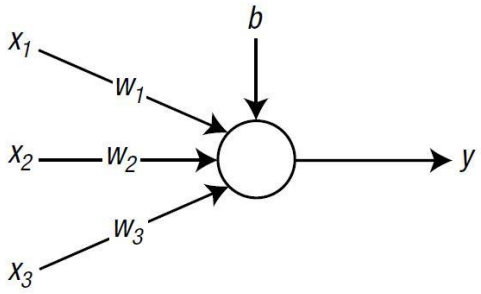

Figure II. 7 neurone recevant trois entrées **[2]**

Le cercle et la flèche de la figure désignent respectivement le nœud (un neurone) et le flux de signal. x1, x2 et x3 sont les signaux d'entrée. w1, w2 et w3 sont les poids pour les signaux correspondants. Enfin, b est le biais, qui est un autre facteur associé au stockage d'informations. Le signal entrant de l'extérieur est multiplié par le poids avant qu'il n'atteigne le noeud. Une fois que les signaux pondérés sont collectés au nœud, ces valeurs sont ajoutées pour constituer la somme pondérée. La somme pondérée de cet exemple est calculée comme suit :

$$v = (w1\ x1) + (w2\ x2) + (w3\ x3) + b \ (\mathbf{II.3})$$

L'équation de la somme pondérée peut être écrite avec des matrices comme :

$$v = wx + b \ \mathbf{(II.4)}$$

Où

$$w = [w1\ w2\ w3] \ \mathbf{(II.5)}$$

$$x = [x1\ x2\ x3] \ \mathbf{(II.6)}$$

Enfin, le nœud entre la somme pondérée dans la fonction d'activation, notée souvent φ, et renvoie sa sortie. La fonction d'activation détermine le comportement (aussi la sortie) du nœud.

$$y = \varphi(v) \ \mathbf{(II.7)}$$

Cette section explique la propagation directe en prenant comme exemple notre réseau neuronal convolutif. Nous allons lui injecter une entrée pour voir comment les données sont traitées lorsqu'elles traversent les couches [2]

La figure 2-8 est une image de 50x50 pixels qui représente notre entrée

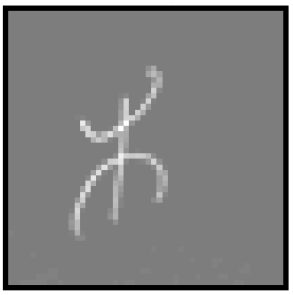

Figure II. 8 Image d'entrée

Couche de convolution $y^1 = \varphi(v^1)$

L'entrée passe d'abord par la couche de convolution, **la figure II.9** représente les 20 filtres de convolution. Chaque filtre est une image affichant les valeurs des éléments sous forme de nuances de gris. Plus la valeur est élevée, plus l'ombre devient claire. Ces filtres sont ce que ConvNet a déterminé comme étant les meilleures caractéristiques pouvant être extraites de l'image.

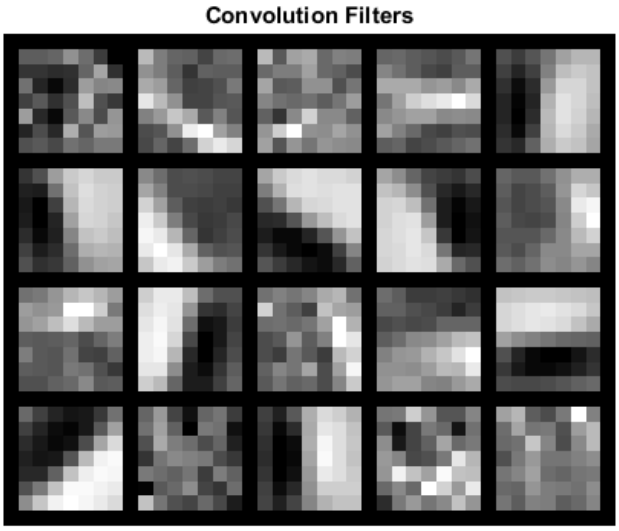

Figure II. 9 filtre de la couche de convolution

La figure II.10 est le résultats (y1) du traitement d'image de la couche de convolution. Elle se compose de 20 images de (44x44) pixels. Les différentes altérations de l'image d'entrée dues au filtre à convolution sont perceptibles sur cette figure.

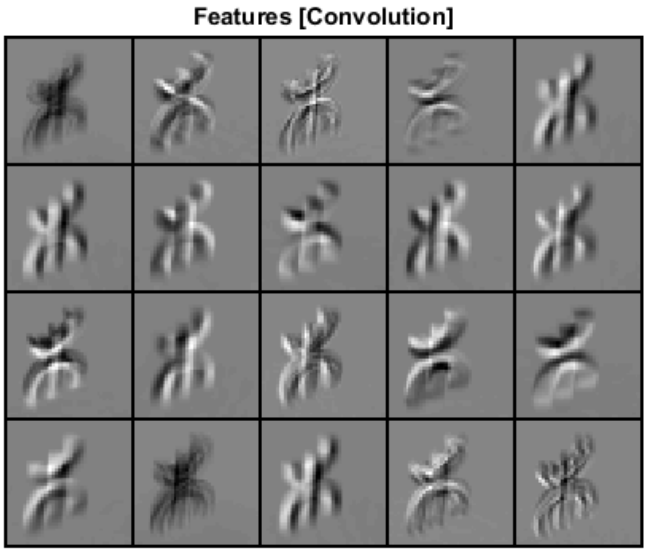

Figure II. 10 sortie de la couche de convolution

Couche de Relu $y^2 = \varphi(y^1)$

La figure II.11 est ce que la fonction ReLU a traité à partir de la sortie de la couche de convolution. Les pixels sombres de l'image précédente sont supprimés et les images actuelles ont principalement des pixels blancs sur la lettre. C'est un résultat raisonnable lorsque l'on considère la définition de la fonction ReLU.

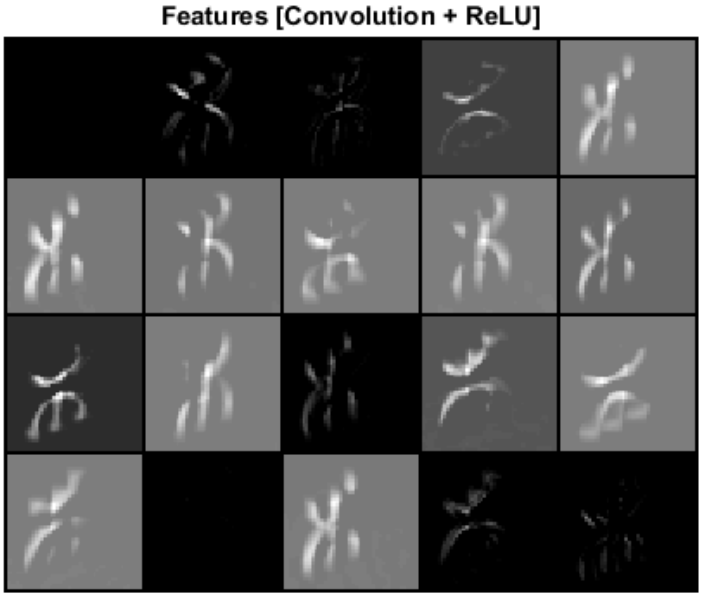

Figure II. 11 sortie de la couche de ReLu

II.4.2 Couche de pooling

La figure II.12 montre le résultat du mean pooling sur la sortie de la couche ReLu Chaque image hérite de la forme de l'image précédente dans une dimension de (22x22) pixels, soit la moitié de la taille précédente. Cela montre à quel point la couche de *pooling* peut réduire les *ressources*.

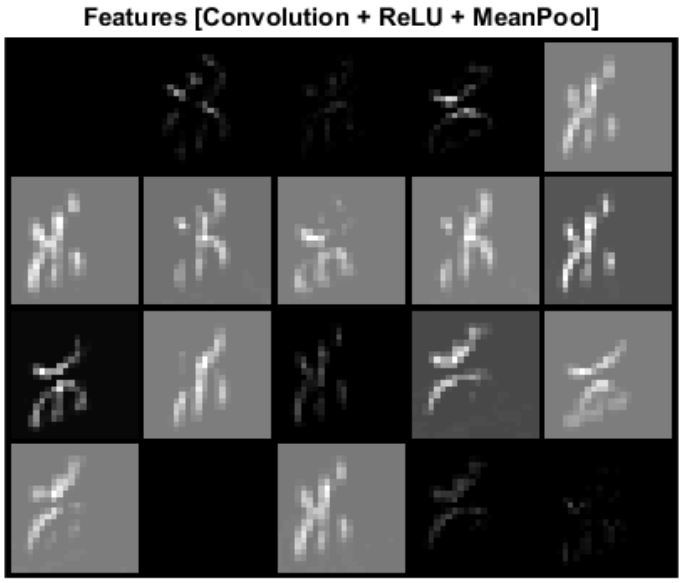

Figure II. 12 sortie de la couche de pooling

Couche reshape $y^4 = \varphi(y^3)$

La couche reshape transforme les images de la sortie de Relu en un vecteur unidimensionnel

Couche ReLu $y^5 = \varphi(w^5 y^4)$

Le vecteur y4 sera multiplié par la matrice de poids de la couche Relu et le résultat sera injecté dans cette dernière a nouveau

Couche softmax $y = w^o y^5$

Et pour finir cette sortie sera à son tour multiplier par la matrice de poids de la couche softmax, et elle sera appliqué au résultat du produit pour crée un vecteur des probabilités de chaque classe. Ce résultat présentera une nature fortement aléatoire car la propagation directe ne représente qu'une partie de l'apprentissage.

II.4.3 Backpropagation

Dans l'algorithme de rétro-propagation, l'erreur de sortie commence à partir de la couche de sortie et recule jusqu'à atteindre la couche cachée immédiatement à droite de la couche d'entrée. Ce processus est appelé rétro-propagation, car il ressemble à une erreur de sortie se propageant en arrière. Même en rétro-propagation, le signal passe toujours par les lignes de connexion. La seule différence est que les signaux d'entrée et de sortie circulent dans des directions opposées. C'est durant cette étape que les dérivées partielles d'une certaine fonction de coût J (cross entropie fonction) par rapport aux paramètres du réseau sont rétro-propagés. Enfin, les poids du réseau sont mis-à-jour en fonction de cette dérivée partielle.

II.4.4 Fonction de coût et règle d'apprentissage

La fonction de coût est un concept plutôt mathématique associé à la théorie de l'optimisation (autre grande bronche de l'IA). La fonction de coût est liée à l'apprentissage supervisé du réseau neuronal. Ce dernier est un processus d'ajustement des poids pour réduire l'erreur des données d'entraînement. Dans ce contexte, la mesure de l'erreur du réseau neuronal est la fonction de coût. Plus l'erreur du réseau de neurones est grande, plus la valeur de la fonction de coût est élevée. Il existe deux principaux types de fonctions de coût pour l'apprentissage supervisé du réseau de neurones qui sont données ici [2].

$$E = \sum_{i=1}^{M} \frac{1}{2}(d_i - y_i)^2 \quad \textbf{(II.8)}$$

$$E = \sum_{i=1}^{M} \{-d_i \ln(y_i) - (1 - d_i)\ln(1 - y_i)\} \quad \textbf{(II.9)}$$

Où d est la valeur désirée, y est la sortie du réseau et i et l'indice de la sortie (première sortie, deuxième sortie …). M est le nombre de sorties du réseau. Il est clair que la valeur de la fonction de coût est proportionnelle à l'erreur. Cette relation est si intuitive qu'aucune autre explication n'est nécessaire[2].

II.4.4.1 Erreur quadratique

La plupart des premières études sur le réseau neuronal ont utilisé la première fonction de coût (erreur quadratique) pour dériver des règles d'apprentissage. Non seulement la règle delta dont nous parlerons plus tard est dérivée de cette fonction, mais l'algorithme de rétro propagation l'était également. Les problèmes de régression utilisent toujours cette fonction de coût [2]. La fonction est illustrée dans la figure II.13

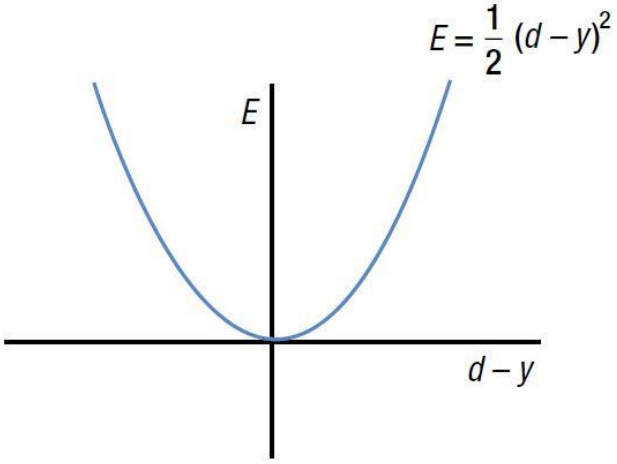

Figure II. 13 courbes de l'erreur quadratique **[2]**

II.4.4.2 Fonction d'entropie croisée (cross entropie)

La fonction de cross entropie affecte directement la règle delta utilisée plus bas, de sorte que, à la couche de sortie on aura :

$$\delta = E \text{ à la place de } \delta = \Psi'(v) * E$$

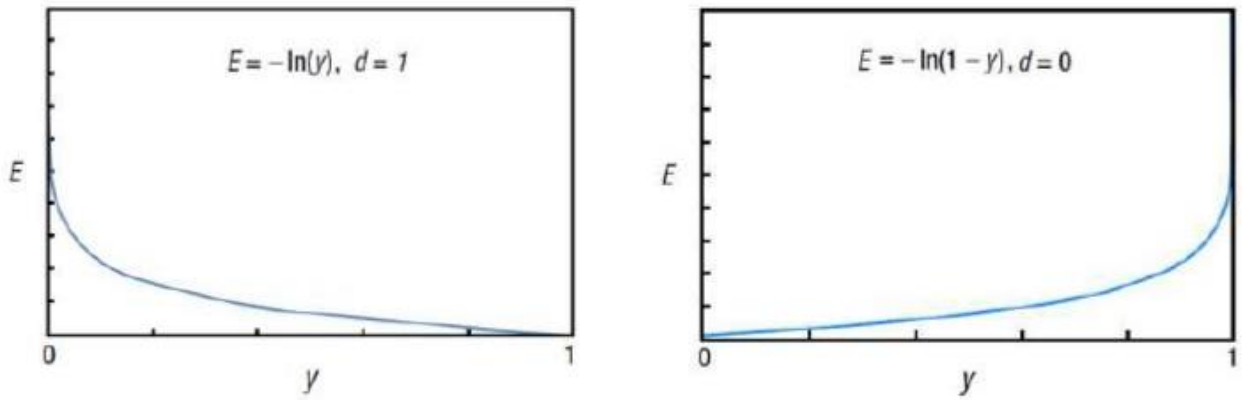

Figure II. 14 courbes de la fonction d'entropie croisée **[2]**

La principale différence entre la fonction d'entropie croisée et la fonction quadratique est son accroissement géométrique. En d'autres termes, la fonction d'entropie croisée est beaucoup plus sensible à l'erreur. Pour cette raison, les règles d'apprentissage dérivées de la fonction d'entropie croisée sont généralement connues pour donner de meilleures performances.

Il est recommandé d'utiliser les règles d'apprentissage axées sur l'entropie croisée, sauf dans les cas inévitables tels que la régression [2].

II.4.5 Méthode de la descente du gradient stochastique

Dans le but d'ajuster les poids et les biaises du réseau d'une manière à minimiser une fonction cout donnée, un apprentissage supervisé basé sur l'algorithme de descente du gradient stochastique (SGD) est utilisé. La descente du gradient stochastique (SGD) calcule l'erreur pour chaque donnée d'entrainement et ajuste les poids immédiatement (en utilisant les équations **(II.10) et (II.11)**). Si nous avons 100 points de données d'entrainement, le SGD ajuste les poids 100 fois [2].

$$w_{ij} = w_{ij} + \alpha\delta_i x_i \quad \textbf{(II.10)}$$

x_j = le signal d'entrée pour le poids correspondant.

w_{ij} = le poids entre le nœud de sortie et l'entrée nœud

α = taux d'apprentissage $(0 < \alpha < 1)$

δ_i est défini comme suit:

$$\delta_i = \varphi'(v_i)e_i \quad \textbf{(II.11)}$$

e_i = l'erreur du noeud de sortie

v_i = la somme pondérée du noeud de sortie

φ' = La dérivée de la fonction d'activation du nœud de sortie

La façon dont l'algorithme de descente de gradient fonctionne est de calculer à plusieurs reprises le gradient, puis de se déplacer dans la direction opposée, "en descendant" la pente vers la vallée. On peut le visualiser sur la figure II.15

E

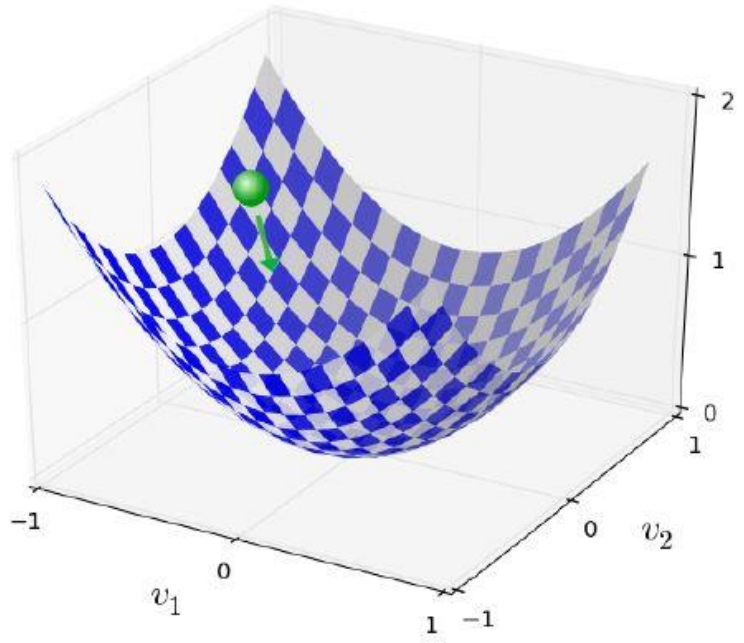

Figure II. 15 illustration de la descente du gradient **[6]**

La rétro propagation a été créé en généralisant la loi d'apprentissage de Widrow-Hoff (règle du delta) à des réseaux de neurones multicouches La règle delta impose toujours

$$\Delta w = -\alpha \delta_i x_i \quad \textbf{(II.12)}$$

II.4.6 La méthode BATCH, par lot

Dans la méthode par lots, chaque mise à jour de poids est calculée pour toutes les erreurs des données d'apprentissage et la moyenne des mises à jour de poids est utilisée pour ajuster les poids. Cette méthode utilise toutes les données d'apprentissage et ne se met à jour qu'une seule fois [2].

$$\Delta w_{ij} = \frac{1}{N} \sum_{K=1}^{N} \Delta w_{ij}(k) \quad \textbf{(II.13)}$$

II.4.7 La méthode Mini Batch, Mini Lot

La méthode des mini lots est un mélange des méthodes SGD et batch. Elle sélectionne une partie des données d'apprentissage et l'utilise pour l'apprentissage de la méthode par lots. Par conséquent, il calcule les mises à jour de poids des données sélectionnées et entraine le réseau de neurones avec la mise à jour de poids moyen. Par exemple, si 20 points de données arbitraires sont sélectionnés sur 100 points de données d'apprentissage, la méthode par lots est appliquée aux 20 points de données. Dans ce cas, cinq ajustements de poids au total sont

effectués pour terminer le processus d'entrainement pour tous les points de données (5 = 100/20) [2].

II.4.7 La méthode du moment

Le moment, m, est un terme qui est ajouté à la règle du delta pour ajuster les poids. L'utilisation du terme de *Momentum* conduit l'ajustement du poids dans une certaine direction Dans une certaine mesure, plutôt que de produire un changement immédiat. Il agit de manière similaire au moment physique, ce qui entrave la réaction du corps aux forces externes [2].

$$m^t = \Delta w^{t-1} + \beta m^{t-1} \textbf{ (II.14)}$$

$$w^t = w^{t-1} + m^t \textbf{ (II.15)}$$

II.4.8 Pseudocode complet de l'algorithme utilisé

La figure suivante donne une idée très précise de l'algorithme que nous avons utilisé pour implémenter le réseau de neurone CNN et faire le deepleanring.

Pseudo code de l'algorithme utilisé

1- Importation de la base de données
2- Initialisation des paramètres
N,m ←ordre du filtre
Nf ←nombre de filtre
W0,w5,w1 ← les valeurs initiales des poids des couches de convolution, de la cinquième couche, de la sixième couche.
H // nombre de pixels après le max pooling
Alpha, Beta // le taux de la modification du poids à chaque itération
3- L'apprentissage
For i← nombre d'époque do
// propagation direct
Initialisation des paramètres de l'époque
momentum1 ← 0

momentum5 ←0

momentumo ←0

Bsize // la dimension de chaque mini batch

blist //nombre de mini batch

// one epoche loop

For batch← la dimension de blist do

dw1←0

dw5 ←0

dwo ←0

//mini batch loop

- For k← le contenu de la batch do

//Détermination de sortie y

x← l'entrée // image de dimension [100 100]

Y1←conv(x.w1)

Y2←ReLu(y1)

Y3←pool(y2)

Y4←reshape(y3)

Y5←ReLu(y4.w5)

Y←softmax(y5,w0) // la sortie

D ← train label

e←(D-y) //l'erreur

rétro propagation

//Calcul de l'erreur dans les couches cachées

delta = e;

e5 ←$w6^T$ *delta

delta5 //si y5>0 delta5 ← y6*e6 sinon 0

e4←$w5^T$*delta5

e3 ←reshape(e4 , size(y3))

e2 //matrice nulle de dimension égale à la dimension de y2

W3 // Nf matrices empilées d'identité de dimension égale à la

dimension de y2 divisé par (4)

- For c ← 1 to Nf do

  e2(c) = (e3 © $\begin{bmatrix}1 & 1\\ 1 & 1\end{bmatrix}$)*.w3(c) // © est le produit de Kronecker

- end

delta2 : si y2>0 delta2 ← y2*e2 sinon 0

delta1_x // matrice nulle de dimension égale à la dimension de w1

- for c = 1: Nf do

  delta1_x(c) = conv(x,c)

- end

// ajustement des poids

dW1 = dW1 + delta1_x

dW5 ← dW5 + delta5*$y4^T$

dWo ← dWo + delta *$y6^T$

- end

//update weights

dW1 ← dW1 / bsize

dW5 ← dW5 / bsize

dW6 ← dW6 / bsize

dWo ← dWo / bsize

momentum1 ← alpha * dW1 + beta * momentum1

W1 ← W1 + momentum1

momentum5 ← alpha * dW5 + beta * momentum5

W5 ← W5 + momentum5

momentumo ← alpha * dWo + beta * momentumo

Wo ← Wo + momentumo

end

- end

II.5 Conclusion

Nous avons présenté dans ce chapitre les réseaux de neurones profonds de type convolutif. Nous avons détaillé l’architecture que nous avons choisie pour faire de l’apprentissage profond. Dans le chapitre suivant, nous allons appliquer ce réseau de neurones à la reconnaissance des caractères de la langue berbère.

CHAPITRE III

CREATION DE LA BASE DE DONNEES ET APPRENTISSAGE DU CNN

III.1 Introduction

Dans ce chapitre, nous présenterons la conception et la mise en œuvre de notre réseau de neurones profonds destiné à la reconnaissance automatique des caractères manuscrits de la langue berbère. Nous détaillerons les étapes de la création de notre base de données Berber Dataset qui a servie à l'apprentissage de notre réseau de neurones convolutif. Puis nous exposerons nos multiples tests d'apprentissage et nous discuterons les résultats obtenus en fonction des différents paramètres du réseau de neurones.

III.2 Bases de données

Un jeu de données (Dataset) est un ensemble de valeurs (ou données) où chaque valeur est associée à une variable (ou attribut) et à une observation. Une variable décrite l'ensemble des valeurs décrivant le même attribut et une observation contient l'ensemble des valeurs décrivant les attributs d'une unité.

III.2.1 Bases de données en machine learning

Les bases de données font partie intégrante du domaine du machine-learning, c'est-à-dire : elles sont nécessaires pour réussir un apprentissage. Des avancées majeures dans ce domaine peuvent donner des progrès en algorithmes d'apprentissage (tels que Deep learning), du matériel informatique et, de manière moins intuitive, par la disponibilité de bases de données de haute qualité. Ces dernières reçoivent des étiquettes (labels) afin d'être utilisées pour divers apprentissages automatiques supervisés et semi-supervisés, mais elles sont généralement difficiles et coûteuses à produire en raison du temps nécessaire pour étiqueter les données.

L'étiquetage des données est le processus d'identification des données brutes (images, fichiers texte, vidéos, etc.) et d'ajout d'une ou plusieurs étiquettes significatives et informatives pour fournir un contexte afin qu'un modèle du machine learning puisse apprendre correctement. Par exemple, les étiquettes peuvent indiquer si une photo contient un oiseau ou une voiture, quels mots ont été prononcés dans un enregistrement audio ou si une radiographie contient une tumeur. L'étiquetage des données est requis pour une variété de cas, notamment la vision par ordinateur, le traitement du langage naturel et la reconnaissance vocale.

Les réseaux de neurones apprennent et mémorisent, dans les poids synaptiques, toutes les solutions de la tâche assignée. Un phénomène appelé surinterprétation peut alors se produire, ceci engendre généralement des performances médiocres ; pour cette raison, les données d'entrée sont généralement divisées en deux ensembles, un pour l'apprentissage, la phase où le

réseau s'améliore et apprend, et un pour tester et vérifier les performances du système, pour vérifier s'il n'y a pas eu surinterprétation et valider les performances. Les échantillons utilisés pour les tests ne doivent jamais être les mêmes que ceux utilisés pour la formation. La quantité de données nécessaires à la phase d'apprentissage est généralement assez importante, et réunir de tel quantité de données est très ardu. Historiquement, il a été jugé pratique d'avoir préalablement organisé des données prêtes à être fournies aux algorithmes, et ainsi plusieurs bases de données ont été créés, chacun avec sa propre caractérisation.

III.2.2 MNIST Dataset

La base de données MNIST (Modified National Institute of Standards and Technology Database) est une grande base de données de chiffres manuscrits comme son nom l'indique, cet ensemble de données a été créé en modifiant une base de données de chiffres manuscrits fournie par l'NIST. L'ensemble de données contiennent 70 000 images de (28x28) pixels de dimension, dont 60 000 images d'entraînement et 10 000 images de test. Chaque image est un scan d'un chiffre manuscrit de 0 à 9. Les étiquettes associées aux images correspondent aux valeurs des dix chiffres. Des exemples de chiffres dans la base de données MNIST sont illustrés dans la figure (III-1).

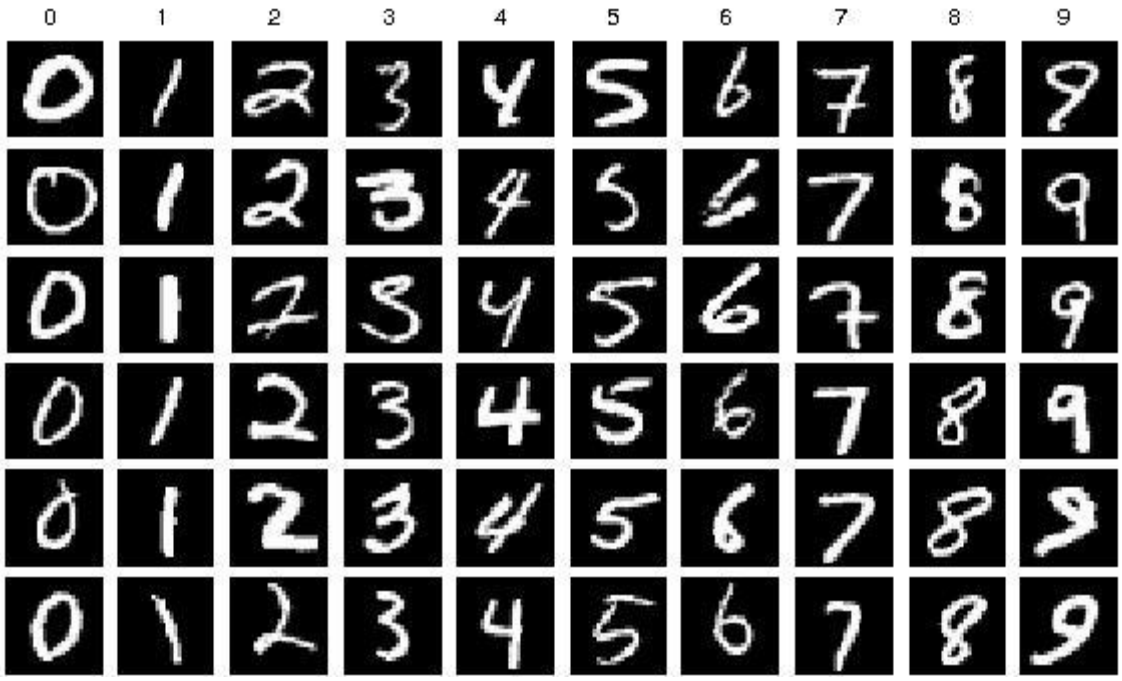

Figure III. 1 Echantillon de la base de données MNIST

III.2.3 HIT-MW Dataset

Une base de données de texte manuscrit chinois, HIT-MW, est présenté pour faciliter la reconnaissance manuscrite de texte chinois hors ligne. La version actuelle de HIT-MW comprend 186 444 caractères écrits par plus de 780 participants [9].

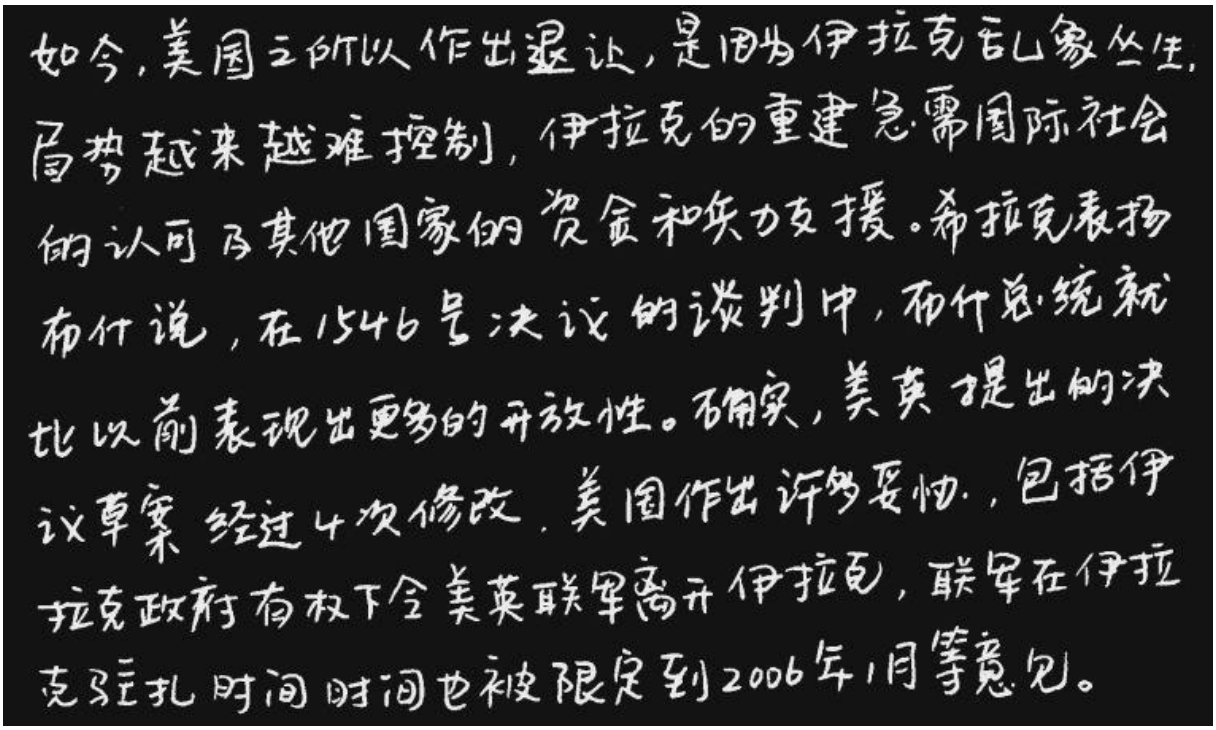

Figure III. 2 Echantillon de la base de données HIT-MW **[9]**

III.2.4 Base de données berbère dataset (BBDS)

Le tifinagh ou alphabet touareg est l'écriture utilisée par les Amazighs (Berbères) en Afrique du Nord pour écrire leur langue, le tamazight. La base de données *berbère dataset* que nous avons réalisée est constituée d'un ensemble de 45 102 d'images de caractères manuscrits représentant les 33 lettres de l'alphabet berbères, en noir et blanc, normalisées, centrées, de dimension (50x50) pixels. 40 000 d'entre elles sont destinées à l'apprentissage du réseau de neurone, et 5000 pour le test.

III.2.5 Etapes de la création de la base de données BBDS

III.2.5.1 Réalisation des images

La première étape consiste à générer en moyenne 1000 images de chaque lettre de l'alphabet amazigh. Dans l'optique d'assurer un apprentissage satisfaisant, nous avons récolté une variété d'échantillons qui englobent les multiples nuances que peut prendre l'écriture de tel caractères, ainsi que le choix de différents supports.

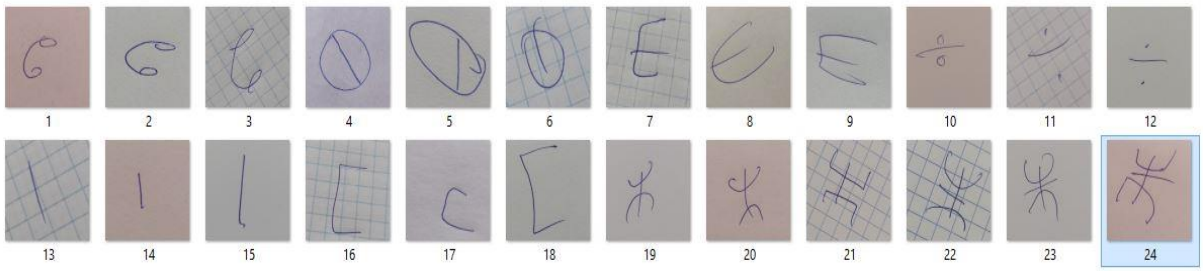

Figure III. 3 Echantillon de la base de données berbère dataset brute

Ensuite, il a fallu séparer les images en 33 classes, une classe par lettre, à qui on a attribué une étiquette (Label) pour chacune d'entre elle.

Figure III. 4 Création des classes de la base de données

III.2.5.2 Réduction de la dimension et conversion en niveau de gris

Comme notre base de données est relativement volumineuse, son utilisation tel qu'elle pour un apprentissage pourrait demander une puissance de calcul très importantes. Pour y remédier nous avons diminué sa taille en réduisant la dimension des images a (50x50) pixels, et comme plusieurs algorithmes de DeepLearning destiné à la vision par ordinateur se basent sur des images en niveau de gris, nous avons converti toutes les images de l'RGB au niveau de gris. Ainsi nous nous retrouvons avec une base de données composé uniquement d'objet simple, et cette simplicité est un avantage car la base de données peut être utilisé pour réaliser des apprentissages plus rapides.

1. ImageGray=double(rgb2gray(imresize(double(image)/255,[50 50])));

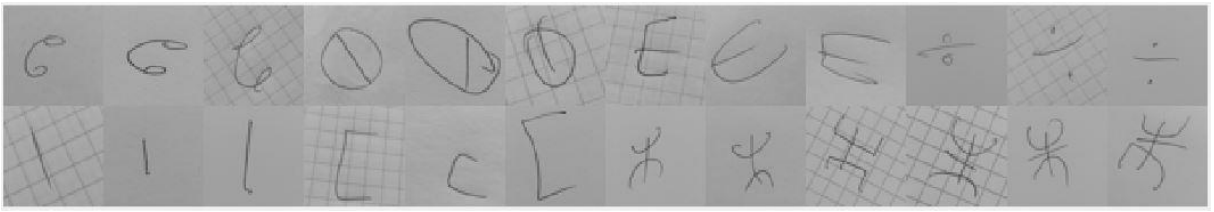

Figure III. 5 Conversion en niveau de gris

III.2.5.3 Inversion des niveaux de noire et blanc et correction des images

Inverser une image binaire signifie inverser les valeurs des pixels. D'un point de vue visuel, lorsque nous inversons une image binaire, les pixels blancs seront convertis en noir et les pixels noirs seront convertis en blanc. Ensuite pour améliorer le rendu des images, nous avons créé et appliqué un filtre de telle sorte à assombrir les pixels noirs et éclaircir les pixels blancs d'une image en niveau de gris.

2. ImagG_Inv=1-min(max((ImageGray)/mean(mean(ImageGray)),0),1);
3. ImagG= ImagG_Inv /max(max(ImagG_Inv));

Figure III. 6 Inversion des niveaux de noir et de blanc

III.2.5.4 Ajout de bruit

Les données venant du monde réel, les échantillons d'un son ou les pixels d'une image, sont complexes, variables et entachées de bruit. L'ajout de bruit artificiel à nos images permet de simuler cela et ainsi d'augmenter la robustesse de notre réseau de neurones pour éviter certain problème tel que la surinterprétation. Le bruit ajouté est un bruit gaussien sous Matlab.

4. ImagGbruit=ImagG+b*rand(size(ImagG))

III.2.5.5 Randomisation de la base de données

Et pour finir, nous procèdent à une randomisation (réarrangement aléatoire) de notre base de données pour assurer un apprentissage satisfaisant. **La figure III.6** représente un échantillon de notre base de données.

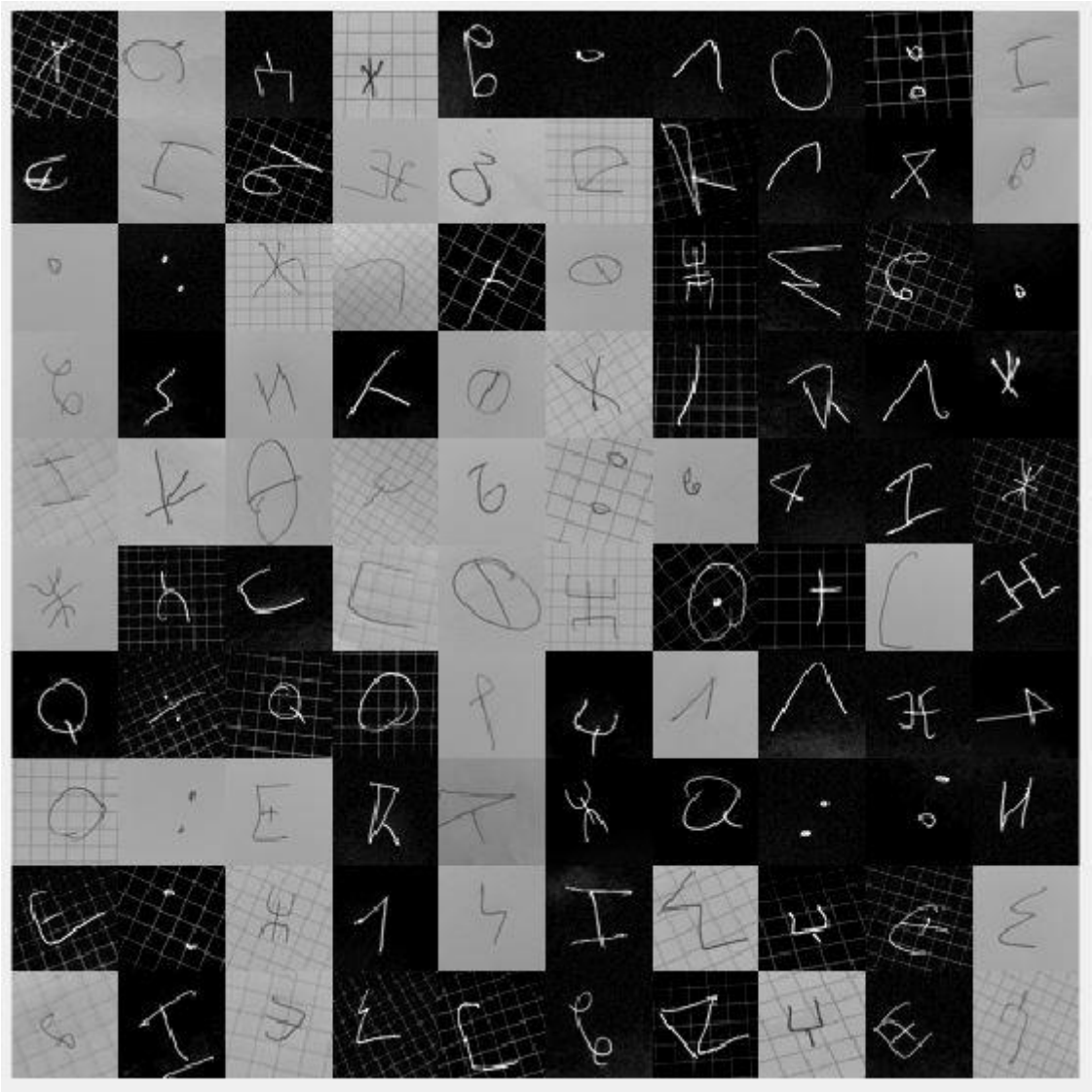

Figure III. 7 Echantillon final de la base de données berbère Dataset

Tableau III. 1 Classes associée à chaque lettre de la base de données

Ordre des lettres	Lettres-en Tifinaghe	Correspondance en Latin	Correspondance en Arabe
01	ⵄ	A	أ
02	ⴱ	B	ب
03	ⵛ	Š	ش
04	ⴷ	D	د
05	ⴹ	Ḍ	ض
06	ⵃ	ḏ	ظ
07	ⵉ	E	-
08	ⴼ	F	ف
09	ⴳ	G	گ
10	ⵖ	ɛ	ع
11	ⵀ	H	ه
12	ⵃ	Ḥ	ح
13	ⵉ	I	ئ
14	ⵊ	J	ج
15	ⴽ	K	ك
16	ⵍ	L	ل
17	ⵎ	M	م
18	ⵏ	N	ن
19	ⵒ	p	پ
20	ⵇ	Q	ق
21	ⵔ	R	ر
22	ⵕ	ṛ	ر.
23	ⵙ	S	س
24	ⵚ	Ṣ	ص
25	ⵜ	T	ت
26	ⵟ	Ṭ	ط
27	ⵓ	U	أ
28	ⵅ	V	غ
29	ⵡ	W	و
30	ⵅ	X	خ
31	ⵢ	Y	ي
32	ⵣ	Z	ز
33	ⵥ	Ẓ	ژ

III.3 Implémentation sur Matlab

MATLAB_ (Matrix LABoratory) est un logiciel interactif basé sur le calcul matriciel. Il est utilisé dans les calculs scientifiques et les problèmes d'ingénierie parce qu'il permet de résoudre des problèmes numériques complexes, et ce grâce à une multitude de fonctions intégrées et à plusieurs programmes outils testés et regroupés selon usage dans des dossiers appelés boites à outils ou "toolbox". Son objectif, par rapport aux autres langages, est de simplifier au maximum la transcription en langage informatique d'un problème mathématique, en utilisant une écriture la plus proche possible du langage naturel scientifique.

III.4 Apprentissage et validation

La base de données *berbère Dataset* présentée plus haut a été utilisée pour l'apprentissage de notre réseau de neurones convolutif. Nous avons réalisé une multitude d'essais en variant les différents paramètres d'entrée du réseau, afin d'étudier leurs influences sur la performance de ce dernier, et ainsi minimiser au mieux l'erreur de classification.

Dans cette section, nous allons exposer un échantillon des tests réalisés pour l'apprentissage. **Le tableau III.2** présente les paramètres de l'essai qui nous a fourni les résultats les plus abouties. De ce fait, il servira de valeur étalon pour évaluer les performances des autres essais.

Tableau III. 2 paramètres du meilleur essai

Paramètres	n	m	Nf	Nn5	alpha	Beta	N	Ns	Bsize	Epoch	Précision
The best	7	7	20	330	0.01	0.85	40000	33	55	15	91.0018

(n*m) : la dimension des filtres de convolution
Nf : nombre de filtres de convolution
Nn5 : nombre de neurones de la couche cachée
Ns : nombre de sorties
Alpha : le pas d'apprentissage
Beta : le coefficient du moment
N : le nombre de données utilisé pour l'apprentissage
Bsize : taille des lots
Epoch : un epoch correspond à un apprentissage sur toutes les données

Les paramètres obtenus dans **le tableau III.2** sont les résultats de plusieurs tests effectués au cours de la période de la préparation de ce projet avec un nombre de 45 tests.

Tableau III. 3variation du Paramètre alpha

Paramètres	n	m	Nf	Nn5	alpha	Beta	N	Ns	Bsize	Epoch
The best	7	7	20	330	0.01	0.85	40000	33	55	15
Test 01	7	7	20	330	0.1	0.85	40000	33	55	15
Test 02	7	7	20	330	0.001	0.85	40000	33	55	15

Tableau III. 4 précision de validation pour différentes valeurs de alpha

Paramètres	Précision sur les données de validation
The best	91.0018
Test 01	63.8502
Test 02	43.3837

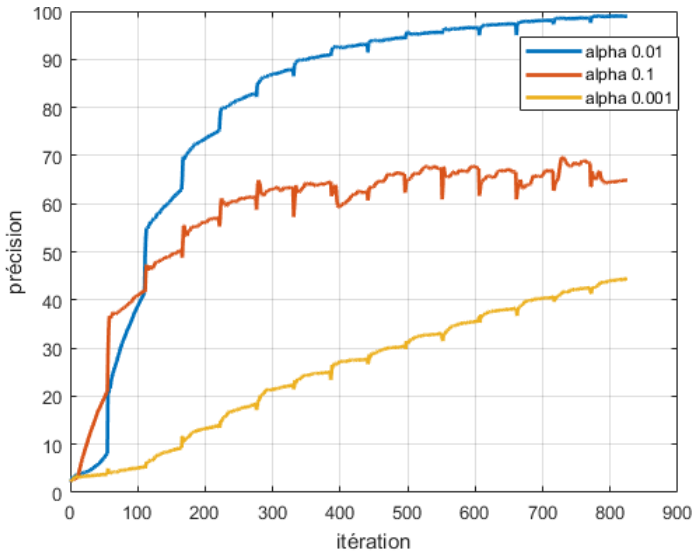

Figure III. 8 précision d'apprentissage pour différentes valeurs de alpha

On remarque dans la figure **Figure III.8** pour que alpha=0.001 la précision est trés faible donc il faut augmenter alpha(le pas)pour avoir une convergence plus rapide, nous avons augmenter le pas afin d'améliorer la précision avec une valeur de alpha= 0.1, une augmenatation de la precision est constaté mais toujours avec une faible valeur, mais meme si on augmente le pas plus que ça ,ça restera toujours faible ,alors on a constaté qu'il faut choisir une valeur alpha=0.01 c'est-à-dire entre ces deux dernières valeurs [0.001 0.1] et ça nous a permet d'avoir une meilleure précision .

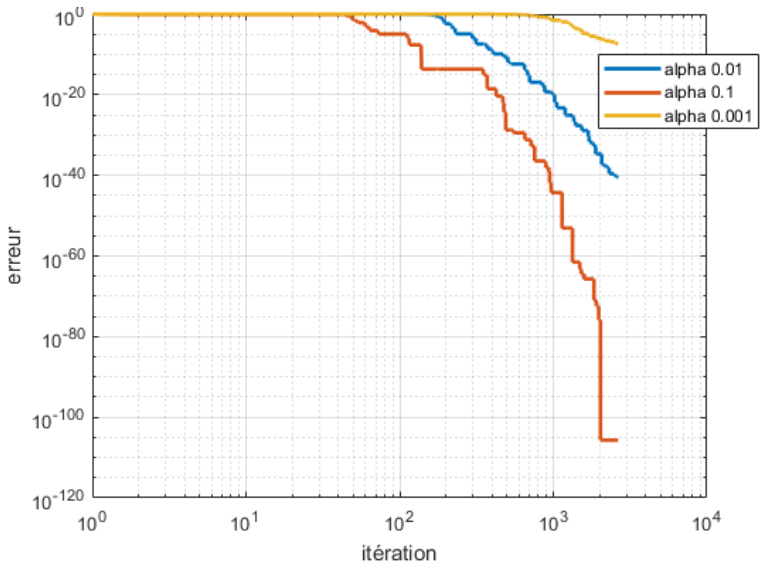

Figure III. 9 erreur d'apprentissage pour différentes valeurs de alpha

Tableau III. 5variation du nombre du neurones

Paramètres	N	M	Nf	Nn5	alpha	Beta	N	Ns	Bsize	Epoch
The best	7	7	20	330	0.01	0.85	40000	33	55	15
Test 01	7	7	20	500	0.01	0.85	40000	33	55	15
Test 02	7	7	20	100	0.01	0.85	40000	33	55	15

Tableau III. 6 précisions de validation pour différents nombres de neurones

Paramètres	Précision sur les données de validation
The best	91.0018
Test 01	89.0218
Test 02	63.1053

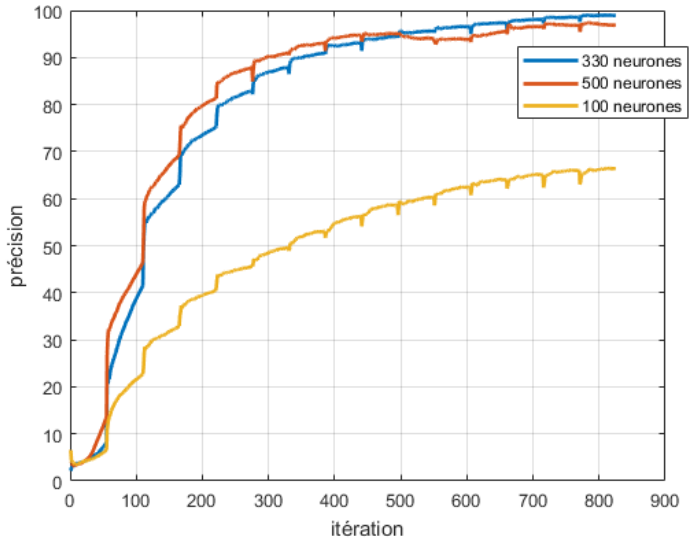

Figure III. 10 précisions d'apprentissage pour différents nombres de neurones

Les neurones sont l'unité principale des algorithmes d'apprentissage, donc il faut bien choisir le nombre de ces derniers. Et à chaque fois leurs nombre est élevé, l'algorithme sera capable d'absorber et d'apprendre un grand nombre de données, dans notre cas il suffit de choisir 330 neurones sachant que l'addition de plus que ça affecte négativement sur la précision finale.

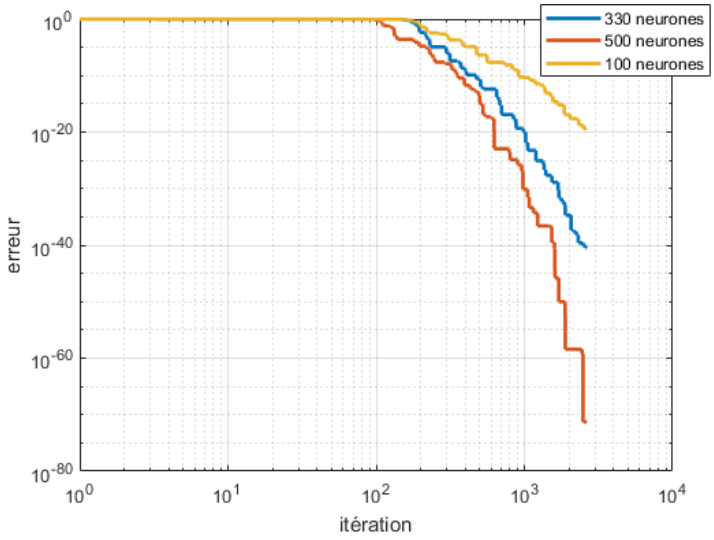

Figure III. 11 erreur d'apprentissage pour différents nombres de neurones

Tableau III. 7 variations du Nombre de filtres

Paramètres	n	M	Nf	Nn5	alpha	Beta	N	Ns	Bsize	Epoch
The best	7	7	20	330	0.01	0.85	40000	33	55	15
Test 01	7	7	12	330	0.01	0.85	40000	33	55	15
Test 02	7	7	40	330	0.001	0.85	40000	33	55	15

Tableau III. 8 précisions de validation pour différents nombres de filtres

Paramètres	Précision sur les données de validation
The best	91.0018
Test 01	87.2378
Test 02	60.7724

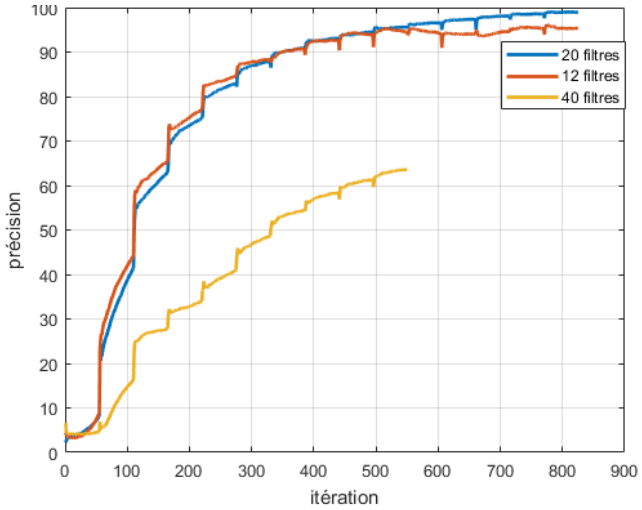

Figure III. 12 précision d'apprentissage pour différents nombres de filtres

Les filtres ont une grande influence sur l'apprentissage donc tout changement de ce paramètre aura un grand impact sur les résultats finals présentés par une précision, Et dans notre cas la meilleure valeur du filtre qui a donné une meilleure précision était celle de 20 filtres.

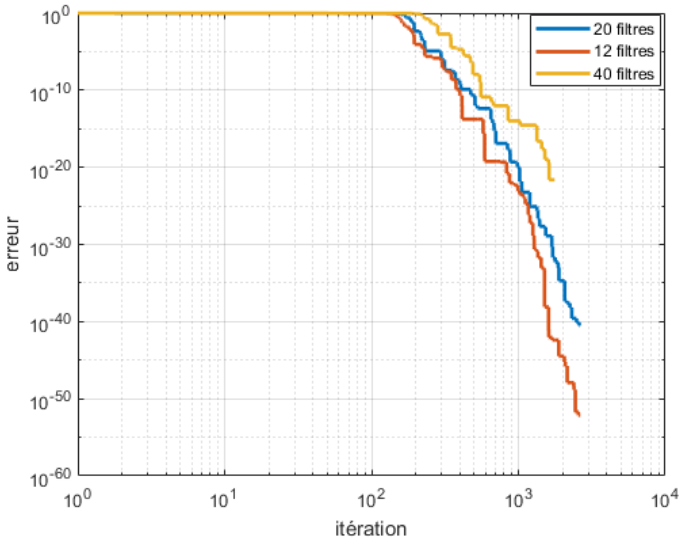

Figure III. 13 erreur d'apprentissage pour différents nombres de filtres

Tableau III. 9 variation du paramètre beta

Paramètres	n	M	Nf	Nn5	Alpha	Beta	N	Ns	Bsize	Epoch
The best	7	7	20	330	0.01	0.85	40000	33	55	15
Test 01	7	7	20	330	0.01	0.75	40000	33	55	15
Test 02	7	7	20	330	0.001	0.95	40000	33	55	15

Tableau III. 10 précision de validation pour différents beta

Paramètres	Précision sur les données de validation
The best	91.0018
Test 01	88.7473
Test 02	87.2378

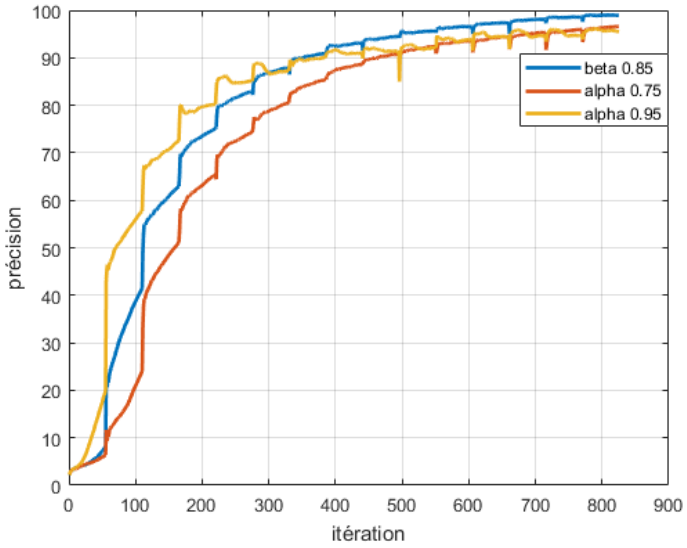

Figure III. 14 précision d'apprentissage pour différentes valeurs de beta

On remarque dans **Figure III.14** que pour beta=0.75 la précision est trés faible donc il faut augmenter beta pour avoir une convergence plus rapide, nous avons augmenter ce paramètre afin d'améliorer la précision avec une valeur de beta= 0.95, une augmenatation de la precision est constaté mais toujours avec une faible valeur, mais meme si on augmente beta plus que ça ,ça restera toujours faible ,alors on a constaté qu'il faut choisir une valeur beta=0.85 c'est-à-dire entre ces deux dernières valeurs [0.75 0.95] et ça nous a permet d'avoir une meilleure précision.

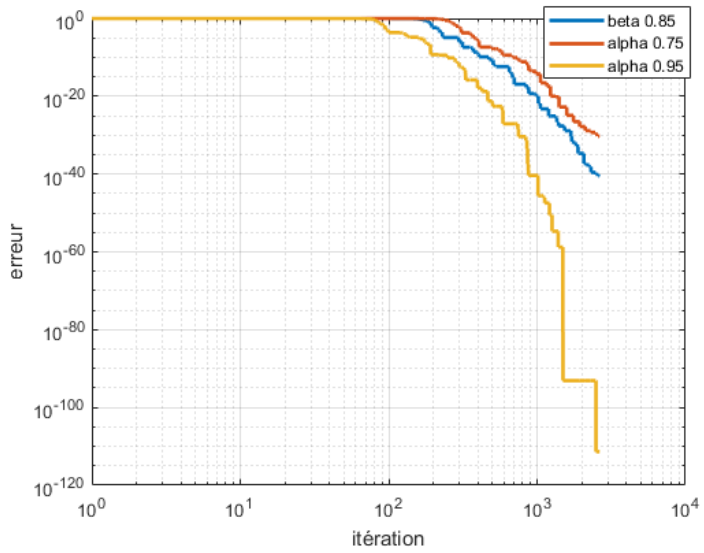

Figure III. 15 précision d'apprentissage pour différentes valeurs de beta

Pour tous les cas notre objectif est d'avoir une précision très élevée et une très faible erreur. Mais dans le but d'avoir une meilleure précision une faible marge d'erreur est acceptable et ce qu'on peut observer à travers les figures suivantes : **figure III.9, figure III.11, figure III.13**, **figure III.15.**

III.5 Segmentation de texte

Dans le but de pourvoir faire reconnaitre a notre réseau de neurone, des phrases ou des paragraphes, il important de réaliser une segmentation afin d'isoler chacune des lettres, et les injecter séparément dans le CNN. Pour ce faire, nous utiliseront la méthode des projection vertical et horizontal présenté dans l'exemple suivant. La figure III.16 représente la suite de lettre que nous voulons séparer

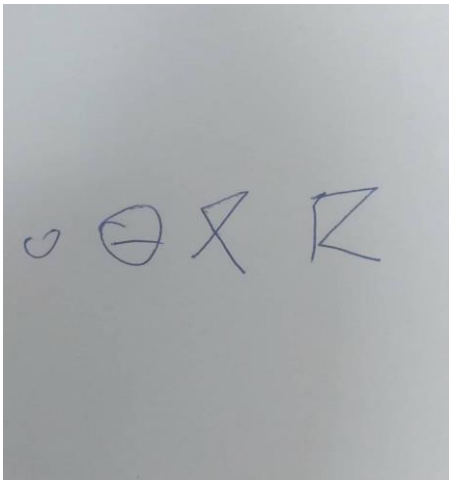

Figure III. 16 Suite de caractère berbère

III.5.1 Binarisation de l'image

En traitement d'image, la binarisation est une opération qui produit une image ayant deux classes de pixels, on parle alors d'une image binaire. En général, les pixels sont représentés par des pixels noirs et des pixels blancs dans une image binaire.

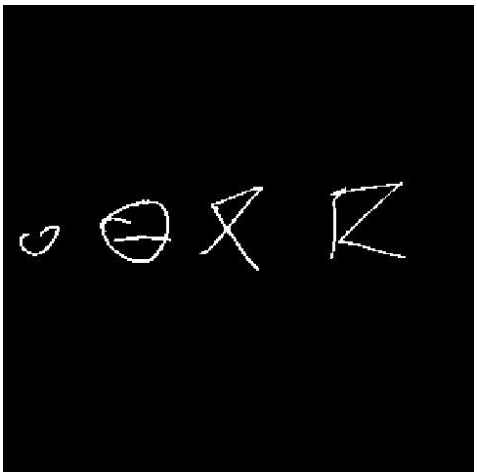

Figure III. 17 Image binarisé

III.5.2 Projection vertical

Cette méthode consiste a calculer la somme des pixels de chaque colonnes de l'image. Le résultat et un vecteur d'une dimension égale au nombre de colonne de l'image d'origine.

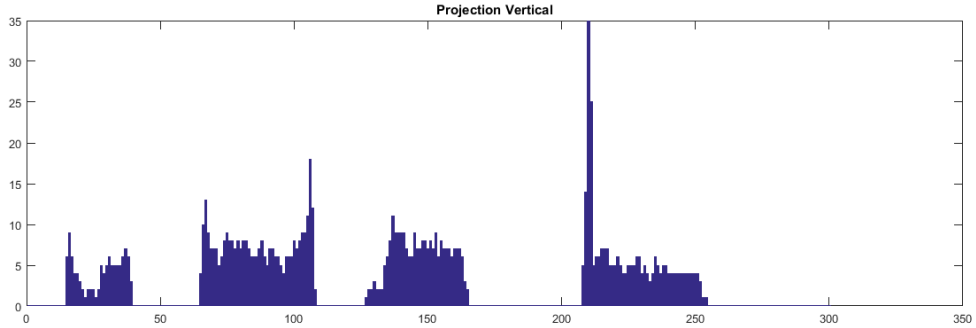

Figure III. 18 Nombre de pixel dans chaque colonne

Les piques élevé traduisent la présence d'un grand nombre de pixel dans les colonnes correspondante.

III.5.3 Projection Horizontal

Ici nous calculons la somme des pixels de chaque ligne de l'image. Le résultat et un vecteur d'une dimension égale au nombre de lignes de l'image d'origine.

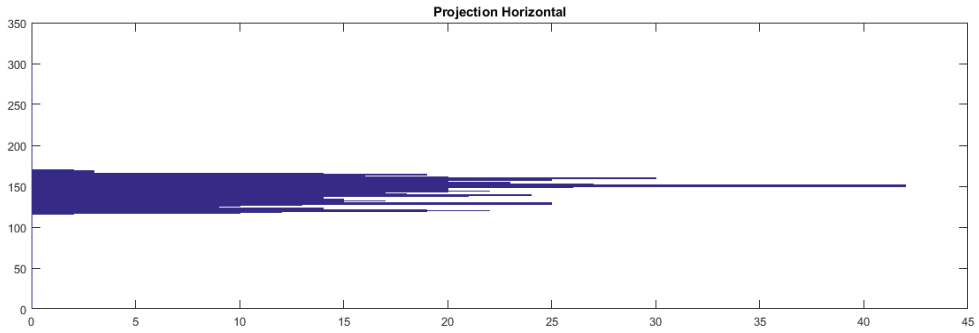

Figure III. 19 Nombre de pixel dans chaque ligne

Les piques élevé traduisent la présence d'un grand nombre de pixel dans les lignes correspondante. Maintenant que nous possédons les coordonnées de chaque caractère, il est possible de les isoler séparément.

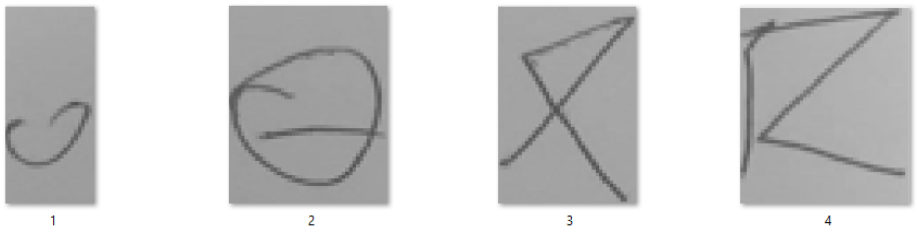

Figure III. 19 Résultat final de la segmentation

III.5 Pseudo code de la segmentation par projection

Pseudo code de l'algorithme utilisé

1. Binarisation de l'image d'entrée
2. for i=1: (nombre de ligne de l'image)
3. **for** j=1: (nombre de colonne de l'image)
4. prov(j) = prov(j)+ I(i,j) ← somme des pixels de chaque colonne
5. **end**
6. **end**

7. for j=1: (nombre de colonne de l'image)
8. for i=1: (nombre de ligne de l'image)
9. proh(i) = proh(i)+ I(i,j) ← somme des pixels de chaque ligne
10. **end**
11. **end**

III. 6 Conclusion

Nous avons présenté dans ce chapitre notre base de données berbère DataSet. Nous avons expliqué les méthodes qui nous ont permet de la crée. Enfin, on a discuté les résultats obtenus après l'apprentissage en utilisant un réseau de neurones profonds de type convolutif**,** on a détaillé également les paramètres qu'on a choisi pour obtenir ces meilleurs résultats présentés par une précision de 91%.

Conclusion Générale et Perspectives

Ce mémoire est consacré au développement d'une nouvelle base de données pour la langue berbère écrite dans le style d'écriture néo tifinagh. Le berbère n'a pas de base de données hors-ligne standard disponible publiquement. Le motif principal de la préparation de la base de données *berbère dataset* est de recueillir du texte en néo tifinagh et de le mettre gratuitement à la disposition de la communauté de recherche pour entrainer des réseaux de neurones et servir en intelligence artificielle. La base de données que nous avons développée nous a servi à l'apprentissage d'un CNN pour la reconnaissance et la classification des caractères berbère.

Le chapitre 1 de ce mémoire est une vue d'ensemble du concept d'intelligence artificielle, ainsi que les liens étroits existant entre l'apprentissage automatique et l'apprentissage profond. Puis, nous avons présenté les applications de ces différentes technologies à divers domaines de la vision artificielle (vision par ordinateur) par réseaux de neurones profonds.

Ensuite, dans le chapitre 2 nous avons présenté les réseaux de neurones profonds de type convolutif, en détaillent, une par une, chaque couche qui constituent l'architecture que nous avons choisie pour faire l'apprentissage des caractères de la langue berbère. Cela nous a permis d'approfondir nos connaissances sur le processus d'apprentissage d'un CNN à travers les étapes de propagation directe et de rétropropagation.

Le chapitre 3 retrace les étapes de la création de la base de données, en partant de l'acquisition des lettres écrites, le traitement d'images, a la finalisation de la base de données. Ce qui a permis la conduite de multiples tests d'apprentissage sur le réseau de neurones convolutif conçu spécialement pour reconnaitre la langue berbère. Au bout de plusieurs essais, nous avons pu obtenir une précision de validation de 91%.

Les résultats obtenus lors de la phase de test sont très encourageants bien qu'ils sont largement améliorables. En effet, ce travail est le premier consacré à ce sujet et reste ouvert pour des travaux à venir. Plusieurs perspectives sont envisageables pour améliorer la précision du CNN. En élargissent la base de données, en implémentant des algorithmes de segmentation plus précis pour permettre la reconnaissance de phrases et de paragraphes entiers et en améliorant la précision du réseau

Bibliographie

[1] Les Enjeux de la Recherche en Intelligence Artificielle Yann LeCun

[2] P. Kim, MATLAB Deep Learning, DOI 10.1007/978-1-4842-2845-6_1

[3] Claude Touzet. LES RESEAUX DE NEURONES ARTIFICIELS, INTRODUCTION AU CONNEXIONNISME : COURS, EXERCICES ET TRAVAUX PRATIQUES. EC2, 1992, Collection de l'EERIE, N. Giambiasi. ffhal-01338010

[4] B. El Kessab, C. Daoui, B. Bouikhalene, R. Salouan « Handwriting Moroccan regions recognition using Tifinagh character», Data in Brief Volume 4, September 2015

[5] N. Sharma, R. Sharma and N. Jindal Global Transitions Proceedings 2 (2021)

[6] Neural Networks and Deep Learning, Michael Nielsen Determination press,2015

[7] Théodore Bluche. Deep Neural Networks for Large Vocabulary Handwritten Text Recognition. Computers and Society [cs.CY]. Université Paris Sud - Paris XI, 2015. English. ffNNT : 2015PA112062ff. fftel-01249405f

[8] A Universal Way to Collect and Process Handwritten Data for Any Language, AKM Shahariar Azad Rabby et al. / Procedia Computer Science 143 (2018) 502–509 509 AKM Shahariar Azad Rabby et al./ Procedia Computer Science 00 (2018) 000–000 7

[9] Tonghua Su, Tianwen Zhang, Dejun Guan. HIT-MW Dataset for Offline Chinese Handwritten Text Recognition. Tenth International Workshop on Frontiers in Handwriting Recognition, Université de Rennes 1, Oct 2006, La Baule (France). ffinria-00103725f

[10] Activation Functions: Comparison of Trends in Practice and Research for Deep Learning, Chigozie Enyinna Nwankpa, Winifred Ijomah, Anthony Gachagan, and Stephen Marshall

[11] Youcef Kherous et Djoudi Nadjet « Réalisation d'une Nouvelle Base de Données pour l'Apprentissage Automatique de la Langue Amazigh par l'Intelligence Artificielle », Decembre 2020

[12] Youssef Es Saady, Ali Rachidi, Mostafa El Yassa, Driss Mammass « Amazigh Handwritten Character Recognition based on Horizontal and Vertical Centerline of Character ». International Journal of Advanced Science and Technology Vol. 33, August, 2011

Résumé

Le travail présenté dans ce mémoire a pour objectif principal de proposer la création d'une base de données un système de reconnaissance hors-ligne de l'écriture manuscrite berbère. Une étude approfondie de ce domaine nous indique que la difficulté de la reconnaissance vient principalement de l'ambiguïté existante souvent à cause de la présence de bruit, de la grande variation des styles d'écriture et de la similarité qui existe entre les entités à reconnaître. Et pour palier a cela, nous avons opter pour un apprentissage via les réseaux de neurones convolutif.

Mots clés : Intelligence artificiel, vision artificiel, apprentissage profond, réseaux de neurones convolutif, base de données.

Abstract

The main objective of the work presented in this thesis is to propose the creation of a database for an offline recognition system for Berber handwriting. An in-depth study of this field tells us that the difficulty of recognition comes mainly from the existing ambiguity often due to the presence of noise, the great variation in writing styles and the similarity that exists between the entities to be recognized. And to overcome this, we have opted for learning via convolutional neural networks.

Key words : Machin learning, artificial vision, deep learning, convolutional neural networks, datasets

الملخص

الهدف الرئيسي للعمل المقدم في هذه الأطروحة هو اقتراح إنشاء قاعدة بيانات لنظام التعرف على خط اليد البربري دون اتصال بالإنترنت. تخبرنا دراسة متعمقة لهذا المجال أن صعوبة التعرف تأتي أساسًا من الغموض الموجود غالبًا بسبب وجود ضوضاء، والاختلاف الكبير في أنماط الكتابة والتشابه الموجود بين الكيانات المراد التعرف عليها. وللتغلب على هذا، اخترنا التعلم عبر الشبكات العصبية التلافيفية.

الكلمات المفتاحية: التعلم الآلي، الرؤية الاصطناعية، التعلم العميق، الشبكات العصبية التلافيفية ، مجموعات البيانات.