République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieure et de la Recherche Scientifique

Université Abderrahmane Mira

Faculté de Technologie

Département d'Automatique, Télécommunication et d'Electronique

Projet de Fin d'étude

Pour l'obtention du diplôme de Master

Filière : Télécommunications

Spécialité : Réseaux Télécommunications

Thème

Deep Learning pour La Reconnaissance des caractères manuscrits

Réalisé par :

- **REDJRADJ Amine**
- **OULEBSIR Melisa**
-

Dirigé par :

Mme GAGAOUA .M

Examiné par :

Mr .MOKRANI Karim
Mr .SADJI Mustapha

Année universitaire : **2020/2021**

Remerciements

Nous remercions tout d'abord Dieu tout puissant de nous avoir donné la volonté et la persévérance pour réaliser ce travail.

A nos parents qui par leurs prières et leurs encouragements, on a pu surmonter tous les obstacles.

Nous tenons à exprimer nos sincères remerciements à Mm : Gagaoua qui nous a permis de bénéficier de son encadrement, les conseils qu'elle nous a prodigué, la confiance qu'elle nous a témoigné ont été déterminants dans la réalisation de notre travail.

Nos vifs remerciements vont également au membre de jury Mr : Mokrani et Mr : Sadji pour l'intérêt qu'ils ont porté à notre travail en acceptant d'examiner notre projet de fin de d'études.

Nous souhaitons adresser nos remerciements les plus sincères aux personnes qui nous ont apporté leur aide et qui ont contribué à l'élaboration de ce mémoire ainsi qu'à la réussite de cette formidable année universitaire.

Dédicaces

Du profond de mon cœur, je dédie ce travail à tous ceux qui me sont chers.

A mes chers parents

Aucune dédicace ne saurait être assez éloquente pour exprimer mon respect, mon amour éternel et ma considération pour les sacrifices que vous avez consenti pour mon instruction et mon bien être

Que ce modeste travail soit l'exaucement de vos vœux tant formulés, puisse dieu le très haut, vous accorde santé, bonheur et longue vie

A mes chères sœurs et mon frère

Leticia et Ramy pour leurs encouragements, leur patience, soutien et leurs sentiments d'amour aux moments les plus difficiles.

En particulier à mon adorable petite sœur Hana qui sait toujours comment procurer la joie et le bonheur pour toute la famille.

A mes grands parents

Ceci est ma profonde gratitude pour votre éternel amour, et à la mémoire de mes grands-parents paternels qu'ils apprécient cet humble geste comme preuve de reconnaissance de la part d'une petite fille qui a toujours priés pour le salut de leurs âme.

A mon binôme et cher ami

Amine pour sa patience et les efforts dont il a fait preuve et sa compréhension tout au long de ce projet

A mes merveilleux amis

Soulef, Malika et Cylia pour leurs encouragements, leurs soutiens moral et leurs présences quotidiennes.

A tous mes amis qui m'ont toujours encouragé, et à qui je souhaite plus de succès

Melisa.

Dédicaces

Du profond de mon cœur, je dédie ce travail à tous ceux qui me sont chers.

A mes chers parents

Aucune dédicace ne saurait être assez éloquente pour exprimer mon respect, mon amour éternel et ma considération pour les sacrifices que vous avez consenti pour mon instruction et mon bien être.

Que ce modeste travail soit l'exaucement de vos vœux tant formulés, puisse dieu le très haut, vous accorde santé, bonheur et longue vie.

A ma chère sœur

Mélina, qui n'a cessé de m'encourager, de sa patience et de son soutien dans les moments difficiles.

A mes chères grandes mères

Ceci est ma profonde gratitude pour votre éternel amour : Houria et Zoulikha

A mes chers tantes et oncles

Karima, Lila, Sonia, Nacera, Réda, Hakim, Omar et Farid, pour leurs amours et générosité.

A la mémoire de mon grand père

Je dédie ce travail à mon grand-père, qui m'a toujours motivé dans mes études, qu'il apprécie cet humble geste comme preuve de reconnaissance de la part d'un petit fils qui a toujours prié pour le salut de son âme.

A mes chers cousins et cousines

A mon binôme et chère amie

Mélisa, pour son amitié, sa patience et efforts dont elle a fait preuve

A mes merveilleux amis

Narimane, Nassim, Manel, Amine, Kenza, Ihssan, Sara, Melissa,Amel et Danny. Qui m'ont toujours soutenus et pousser à donner le meilleur de moi-même.

Amine

Table Des Matières

Chapitre III Implémentation et Résultats

Liste Des Figures

Chapitre I

Chapitre II

Chapitre III

Liste Des Tableaux

Chapitre I

Chapitre II

Chapitre III

Liste Des Acronymes

AI : **A**rtifcial **I**ntelligence

ASCII : **A**merican **S**tandard **C**ode for **I**nformation **I**nterchange

ASMO : **A**rabic **S**tandard **M**etrology **O**rganization

AVG : **A**vera**g**e

CNN : **C**onvolutional **N**eural **N**etwork

CPU : **C**entral **P**rocessing **U**nit

DBN : **D**eep **B**elief **N**etwork

DL : **D**eep **L**earning

DNN : **D**eep **N**eural **N**etwork

IFHCDB : **I**solated **F**arsi **H**andwritten **C**haracter **D**ata**B**ase

FC : **F**ully **C**onnected

GPU : **G**raphics **P**rocessing **U**nit

LSTM : **L**ong **S**hort-**T**erm **M**emory

ML : **M**achine **L**earning

MLP : **M**ulti**L**ayer **P**erceptron

MNIST : Modified ou **M**ixed **N**ational **I**nstitute of **S**tandards and **T**echnology

OCR : **O**ptical **C**haracter **R**ecognition

ReLU : **R**ectified **L**inear **U**nit

RGB : **R**ed **G**reen **B**l

TPU : **T**ensor **P**rocessing **U**nit

Introduction

Depuis son invention à la fin du 4ème millénaire avant J-C, l'écriture reste un moyen de communication privilégié entre les êtres humains. Bien qu'avec l'invention de l'imprimerie et l'évolution de l'informatique aient permis son automatisation, l'importance de l'écriture manuscrite demeure toujours et de manière massive. Et pour ce, des méthodes à automatisé l'écriture manuscrite ont été créé tel que l'OCR (Optical Character Recognition).

Les analystes et experts en charge de traitement de la reconnaissance des caractères mettent en exergue la difficulté du traitement des caractères arabes compte tenu de leurs spécificités qui se distinguent par le rang ou l'emplacement du caractère dans un mot, ainsi que les formes innombrables que prennent ces derniers selon les utilisateurs.
La reconnaissance de chiffres manuscrits, fait l'objet d'un nombre important de travaux de recherches. De ce fait, la nécessité de recourir à des procédés plus performants de reconnaissance est primordiale

Dans notre étude nous nous sommes penchés sur les réseaux de neurones profonds (deep learning), qui au cours de la dernière décennie a dominé l'apprentissage automatique. Un sous-type d'un réseau neuronal appelé réseau de neurone convolutif (CNN), convient bien aux taches permettant d'extraire et de classifier des caractéristiques des caractères, permettant ainsi la reconnaissance automatique de l'écriture manuscrite.

Pour aboutir aux objectifs fixés, ce présent mémoire sera organisé en trois chapitres précédé de cette introduction et suivie d'une conclusion générale :

- Le premier chapitre est consacré à la représentation des concepts généraux liés à la reconnaissance de l'écriture manuscrite, citant les principales étapes sur les quelles sont basés les systèmes de reconnaissance de l'écriture.
- Le deuxième chapitre sera dédié à la présentation des réseaux de neurones profonds, en particulier le CNN.
- Enfin, le troisième chapitre se portera sur la simulation et l'interprétation des différents résultats obtenue suite à l'implémentation des différents algorithmes de deep learning sur deux bases de données, IFHCDB pour les lettres arabes manuscrites et MNIST pour les chiffres manuscrits.

Chapitre I

La Reconnaissance De L’écriture Manuscrite

I.1 Introduction

L'écriture manuscrite est un moyen naturel de communication qui présente l'avantage d'être familier à la majorité des gens. De ce fait elle présente un moyen d'interaction facile avec l'ordinateur. L'un des objectifs les plus recherchés est de doter les ordinateurs de capacités de l'être Humain. La reconnaissance optique de caractères (OCR) est une opération informatique rapide permettant de réaliser la transformation d'un texte écrit sur papier en un texte sous forme d'un fichier informatique en représentation symbolique (par exemple pour les écritures latines, le codage opéré est le code ASCII (American Standard code for Information Interchange), tandis que pour l'arabe on utilise généralement le code ASMO (Arabic Standard Metrology Organization).

I-2 Différents aspects de l'OCR

Il n'existe pas de système universel d'OCR qui permet de reconnaître n'importe quel caractère dans n'importe quelle fonte. Tout dépend du type de données traitées et bien évidemment de l'application visée [1]. Il existe plusieurs modes de classification des systèmes OCR parmi lesquels on peut citer :

- Les systèmes qualifiés de « enligne » ou « hors-ligne » suivant le mode d'acquisition.
- Les approches globales ou analytiques selon que l'analyse s'opère sur la totalité du mot, ou par segmentation en caractères.
- Les approches statistiques, structurelles ou stochastiques relatives aux traits caractéristiques extraits des formes considérées.

I. 3 Mode d'acquisition des données :

Il existe deux différents modes d'OCR, ayant chacun ses outils propres d'acquisition et ses algorithmes correspondants de reconnaissance.

a) La reconnaissance en ligne :

Ce mode de reconnaissance s'opère en temps réel (pendant l'écriture) et est réservé généralement à l'écriture manuscrite. Les symboles sont reconnus au fur et à mesure qu'ils ont écrits à la main. C'est une approche « signal » où la reconnaissance est effectuée sur des données à une dimension. L'écriture est représentée comme un ensemble de points dont les coordonnées sont fonction du temps, [2]. L'avantage majeur de la reconnaissance en ligne est la possibilité de correction et de modification de l'écriture de manière interactive vu la

réponse en continu du système [11]. L'acquisition de l'écrit est généralement assurée par une tablette graphique munie d'un stylo électronique. [2]

b) La reconnaissance hors-ligne

L'écriture en mode hors-ligne est obtenue par la saisie d'un texte déjà existant, obtenue par un scanner ou une caméra. [3] Ce mode peut être considéré comme le cas le plus général de la reconnaissance de l'écriture. Il se rapproche du mode de la reconnaissance visuelle. L'interprétation de l'information est indépendante de la source de génération.

Figure I .1 comparaison entre l'écriture en ligne et hors ligne

c) comparaison entre la reconnaissance en ligne et hors-ligne

Reconnaissance hors-ligne	Reconnaissance en ligne
Utilisation de papiers	Utilisation de stylos numérique
Exigence de point	Exigence d'échantillon
Taux de reconnaissance faible	Taux de reconnaissance élevé
Faible précision	Grande précision

Tableau I.1 Comparaison de la reconnaissance en ligne et hors ligne

I .4 Approches de reconnaissance

Il existe deux approches pour la reconnaissance des mots : la reconnaissance globale et la reconnaissance analytique

I.4.1 Approche globale

Dite aussi holistique, considère le mot comme une seule entité et le décrit indépendamment des caractères qui le constituent. Cette approche présente l'avantage de garder le caractère dans son contexte avoisinant, ce qui permet une modélisation plus efficace des variations de l'écriture et des dégradations qu'elle peut subir. Cependant cette méthode est pénalisante par la taille mémoire, le temps de calcul et la complexité du traitement qui croient linéairement avec la taille du lexique considéré, d'où une limitation du vocabulaire [2].

I.4.2 Approche Analytique

Cette approche permet de surmonter les limites de l'approche globale, mais nécessite une interprétation locale basée sur la segmentation des mots. La reconnaissance consiste à identifier les entités segmentées et à viser ensuite l'identification des mots, tâche assez délicate qui peut produire différents types d'erreurs [4]. La difficulté de cette approche a été explicitement mentionnée par Sayre en 1973 et peut être résumée par le dilemme suivant : " Pour identifier les lettres, il faut segmenter les lignes, et pour segmenter les lignes, il faut identifier les lettres " [3].

I.5 Les caractéristique morphologique de l'écriture arabe

L'arabe s'écrit horizontalement de droite à gauche et l'alphabet est composé de 28 caractères ayant chacun deux à quatre formes dont 16 incluent dans leurs formes des points diacritiques qui peuvent être au nombre de 1, 2 ou 3. Ces points font la différence entre les caractères ayant un corps identique tel que **(ب, ت, ث).** Ils peuvent être situés au-dessus ou au-dessous du corps du caractère de base ou au milieu.

La forme d'une lettre écrite dépend de son contexte, elle diffère selon que le caractère apparaît en position initiale, médiane ou isolée dans une chaîne de caractère comme illustré dans le tableau I.2

Lettre	*Isolé*	*For.F/M/I*	*Lettre*	*Isolé*	*For.F/M/I*
Alif	ا	ـا ـا ا	Dhad	ض	ضـ ـضـ ـض
Ba	ب	بـ ـبـ ـب	Tad	ط	طـ ـطـ ـط
Ta	ت	تـ ـتـ ـت	Thad	ظ	ظـ ـظـ ـظ
Tha	ث	ثـ ـثـ ـث	Ayn	ع	عـ ـعـ ـع
Jim	ج	جـ ـجـ ـج	Gyn	غ	غـ ـغـ ـغ
Ha	ح	حـ ـحـ ـح	Fa	ف	فـ ـفـ ـف
Kha	خ	خـ ـخـ ـخ	Qaf	ق	قـ ـقـ ـق
Dal	د	ـد ـد د	Kaf	ك	كـ ـكـ ـك
Dhal	ذ	ـذ ـذ ذ	Lam	ل	لـ ـلـ ـل
Ra	ر	ـر ـر ر	Mim	م	مـ ـمـ ـم
Zay	ز	ـز ـز ز	Nun	ن	نـ ـنـ ـن
Sin	س	سـ ـسـ ـس	Ha	ه	هـ ـهـ ـه
Shin	ش	شـ ـشـ ـش	Waw	و	ـو ـو و
Sad	ص	صـ ـصـ ـص	Ya	ي	يـ ـيـ ـي

Tableau I.2 différentes formes des caractères arabes.

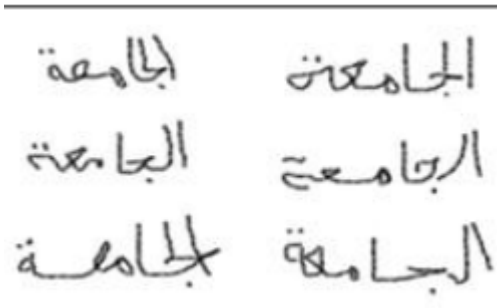

Figure I.2 différents styles et tailles pour le même mot

I.6 Reconnaissance des chiffres manuscrits

Il s'agit de la tâche la plus élémentaire d'un système numérique de reconnaissance des chaines de caractères. L'effort d'analyse se concentre sur un élément à la fois (vu comme une forme globale), il existe de nombreuses méthodes de reconnaissance, qui dépendent soit du choix du type de repère visuel de la méthode de reconnaissance, des paramètres, ou des primitives .Ces paramètres peuvent être topologiques, géométriques ou métriques (distance, taille, courbure et angle), ou enfin les statistique liées à l'observation de points.

La difficulté d'identifier le problème vient de la variabilité des formes de chiffres, quand on se trouve dans un environnement omni-scripteur. Ces différences sont visibles dans la figure I.3

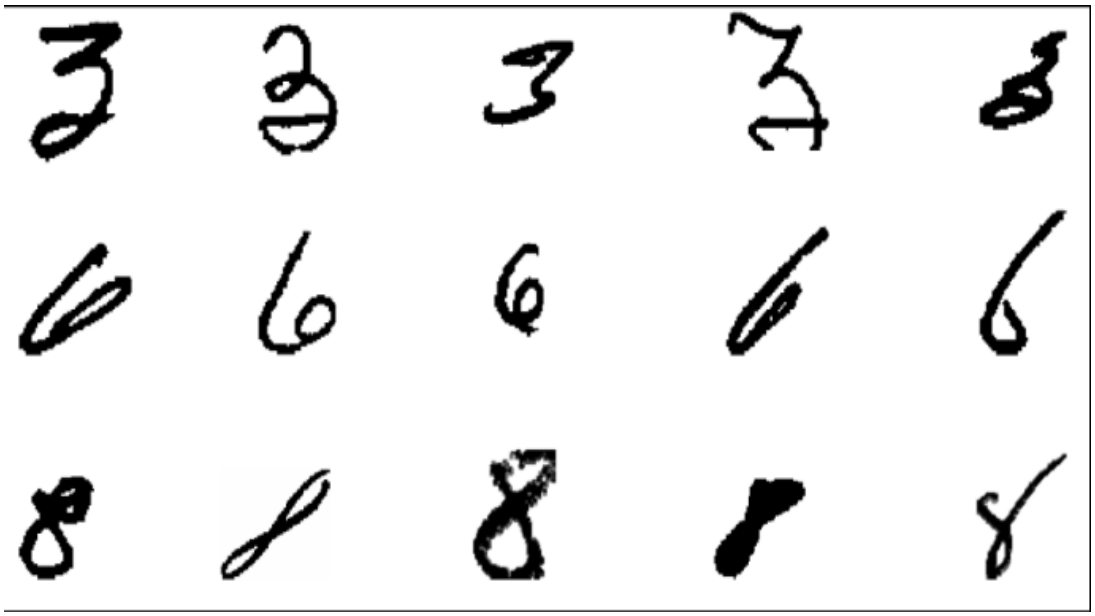

Figure I.3 Exemples de prototypes 3,6 et 8 extraits à partir de la base MNIST

I.7 L'organisation générale d'un système de reconnaissance de l'écriture

Les systèmes de reconnaissance de l'écriture manuscrite sont généralement basés sur les étapes principales suivantes : Acquisition, prétraitements, segmentation, extraction des caractéristiques, classification, suivis éventuellement d'une phase de post-traitement

Figure I-4 résume l'organisation générale de ces étapes en ordre de fonctionnement dans de reconnaissance

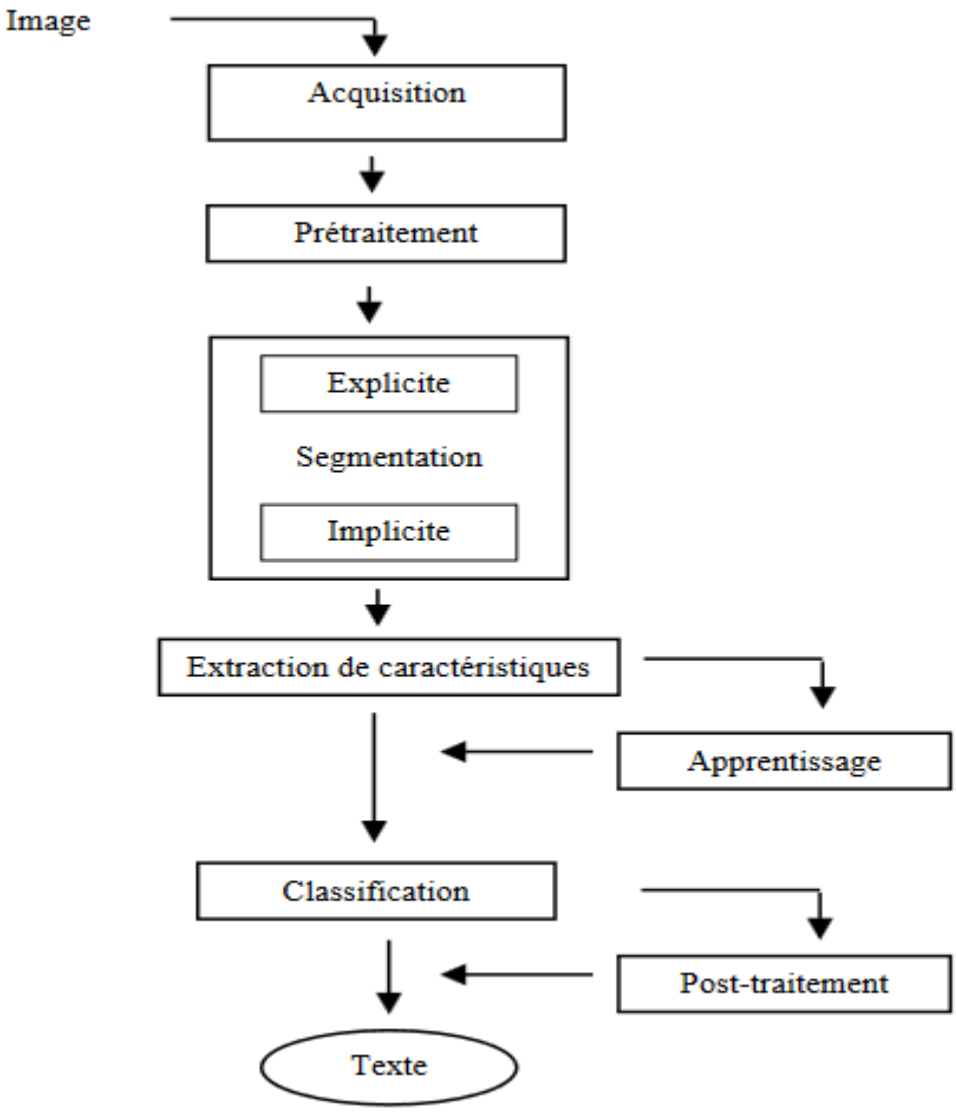

Figure I.4 Organisation générale d'un système de reconnaissance d'écriture

Pour assurer une bonne reconnaissance, après l'acquisition des données, ces dernières passent par un ensemble d'opérations de prétraitement, puis une étape de segmentation aura

lieu. Dans le cas de reconnaissance de mot en utilisant l'approche holistique le système passe directement à l'´étape d'extraction de caractéristiques. Ces dernières seront utilisées dans

l'étape de classification, puis à la fin une étape d'amélioration des résultats pourra avoir lieu.

I.8 Processus de la reconnaissance

I .8.1 Phase acquisition

La première étape dans le système OCR consiste à convertir l'écriture en grandeur numérique adaptées au système de traitement avec un minimum de dégradations possibles en mode hors ligne, selon l'outil d'acquisition utilisé, une image en couleur ou en niveau de gris est obtenue

I .8.2 phase de prétraitement

Le prétraitement est l'étape crucial dans tout système de reconnaissance, ces prétraitements ne sont pas spécifiques à la reconnaissance de l'écriture, mais sont des prétraitements classiques en traitement d'image. Le prétraitement a pour but de préparer l'image du tracé à la phase suivante d'analyse [5]. Il s'agit essentiellement de réduire le bruit superposé aux données et ne garder, autant que possible, que l'information significative de la forme présentée. Le bruit peut être dû au dispositif d'acquisition, aux conditions d'acquisition (éclairage, mise incorrecte du document…), ou encore à la qualité du document d'origine [6]. Parmi les opérations de prétraitements généralement utilisées, citons : la binarisation, la normalisation. L'extraction des composantes connexes, le redressement de l'écriture, le lissage, et la squelettisation

La figure I.4 illustre un exemple ou on a appliqué toutes les opérations précédentes

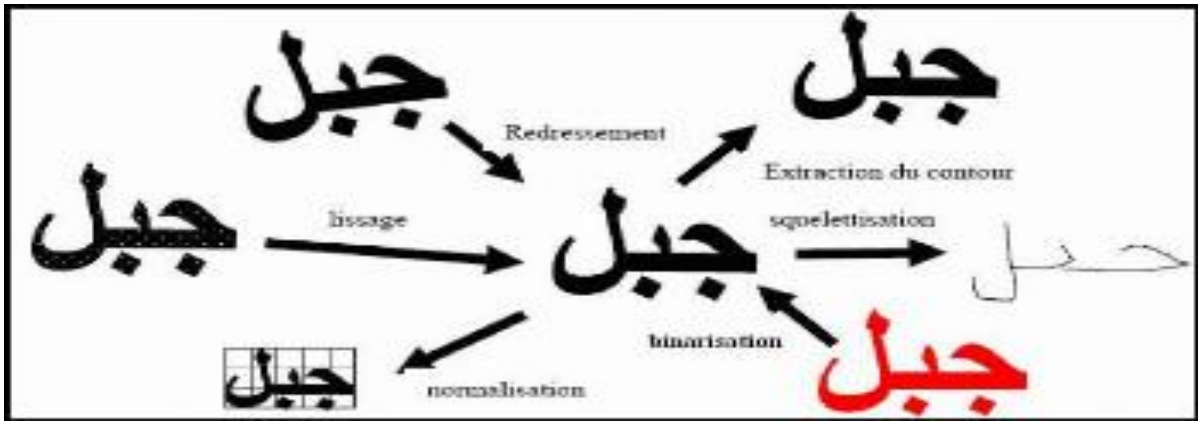

Figure I.5 effets de certaines opérations de prétraitement [14].

I.8.3 Phase de segmentation

Dans un système de reconnaissance de l'écriture manuscrite, les données à traiter sont des images. La mise en œuvre d'une étape de segmentation permet de diviser l'image en différentes régions connexes présentant une homogénéité selon certain critère, et de taille moins importantes qui peuvent être des graphèmes, des lettres ou bien des sous mots. Cependant une image segmentée reste une matrice de pixels [7]. On distingue quatre types de segmentation :

- Segmentation de la page
- Segmentation d'un bloc de texte en ligne
- Segmentation des lignes en mots
- Segmentation d'un mot en caractères

I.8.4 Phase d'extraction de caractéristique

Dans les textes manuscrits et imprimés, les caractéristiques saisissent les informations extraites de l'image textuelle. Les attributs de ces informations doivent êtres filtré tout en préservant les propriétés qui différencient un caractère ou un mot d'un autre, C'est une étape critique lors de la construction d'un système de reconnaissance, l'une des raisons pour laquelle cette phase pose un problème est qu'une plusieurs techniques d'extraction s'accompagne d'une perte d'information. De ce fait, il faut effectuer un compromis entre la quantité et la qualité de l'information [8].

La réduction du nombre de caractéristiques a de nombreux avantages : elle permet d'améliorer la visualisation et la compréhension des données, de réduire les temps d'apprentissage et de classification des systèmes, d'améliorer les performances en classification, et permet de réduire la taille des bases d'apprentissage.

La phase d'extraction d'attributs et de classification est regroupée en une seule étape dans le Deep Learning comme le schématise la figure I.6.

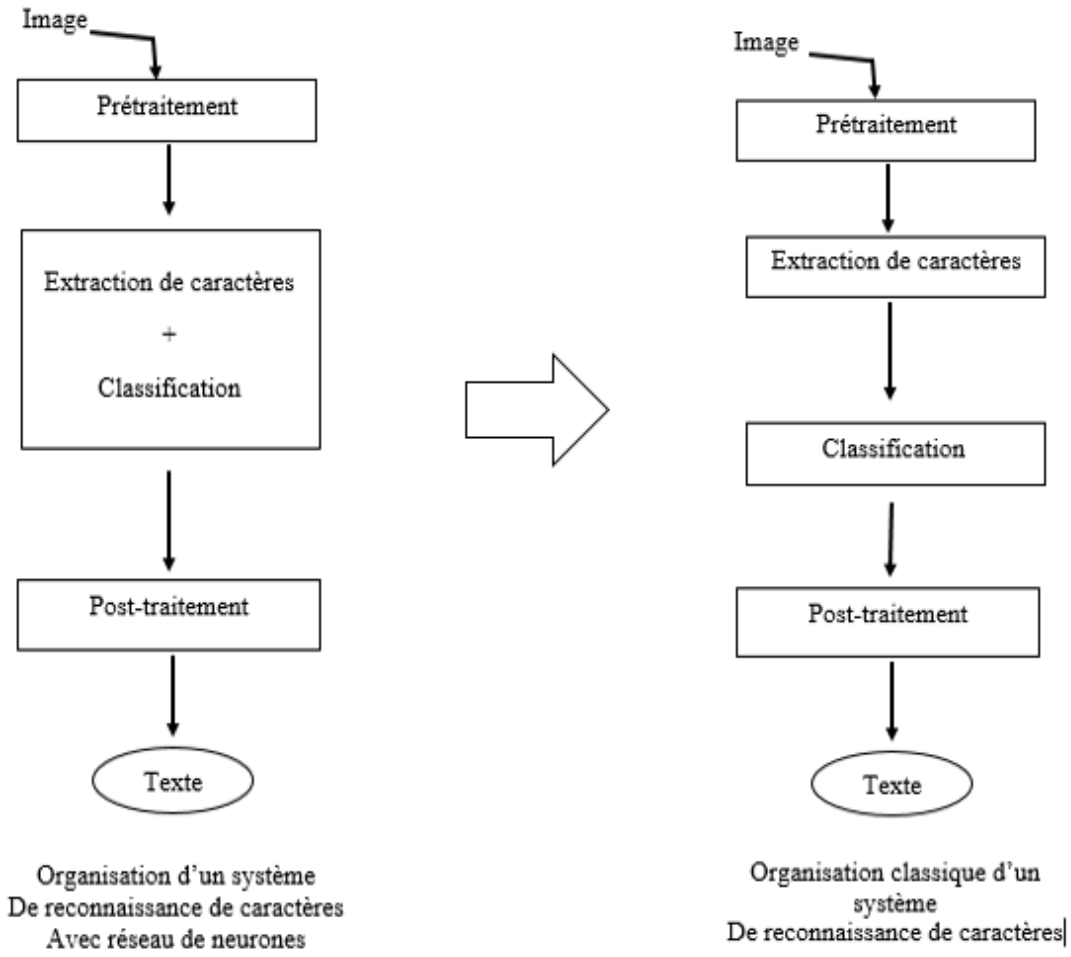

Figure I.6 extraction manuelle et automatique des attributs

I.8.5 Phase de classification

Après l'extraction d'attributs, la technique de classification est basée sur une stratégie de décision qui permet de catégoriser un objet au plus près selon certains critères d'optimisation. Et les attributs caractérisant la forme sont transformés en appartenances de classe (passées de l'espace de codage à l'espace de décision) [13]. A ce stade, un problème de classification se pose. Afin de trouver une solution, des chercheurs de différents domaines ont utilisé des réseaux neuronaux. De nombreuses études, telles que [15,16], ont comparé plusieurs fois les performances des classificateurs neuronaux à celles des classificateurs classiques.

D'après ces références, les classificateurs basés sur les réseaux de neurones ont donné des résultats prometteurs dans la reconnaissance. C'est pour cela que les chercheurs les ont appliqués avec succès à diverses tâches de classification du monde réel

Le type d'une méthode de classification se décline généralement en deux familles : Le mode supervisé et le mode non supervisé [9].

I.8.5.1 Classification supervisée

Cette technique est basée sur l'étiquetage des observations en affectant chaque observation à une classe (supervisés où la sortie correcte doit être fournie à l'avance).

I.8.5.2Classification non supervisée

Aucune des observations n'est étiquetée (non supervisés où la sortie correcte n'est pas exigée à l'avance, elle résulte après une étape d'apprentissage). Généralement, pour évaluer la performance d'un classificateur, on poursuit deux étapes Principales qui sont l'apprentissage et le test (Reconnaissance et décision) :

- **Étape d'apprentissage** : Les données d'apprentissage sont les données utilisées pour construire un classificateur capable de reconnaître des formes inconnues pour caractériser les classes [10].
- **Étape de Reconnaissance et décision** : Elle cherche parmi les modèles de référence en présence, ceux qui lui sont les plus proches. Le problème revient à affecter une forme inconnue à l'une des classes obtenues pendant l'apprentissage [11].

Si toutes les données sont employées pour l'apprentissage et le test en même temps, il est possible que le classificateur soit incapable de reconnaître d'autres formes inconnues. C'est pourquoi il est important d'avoir deux ensembles de données pour améliorer la généralisation d'un classificateur : le premier ensemble est spécifié pour l'apprentissage, le deuxième pour le test.

I.8.6 Phase de post-traitement

Le but du post-traitement est d'améliorer le taux de reconnaissance des mots (le taux de reconnaissance des caractères recommandé). Cette étape est généralement mise en œuvre sous la forme d'un ensemble d'outils liés à la fréquence d'occurrence des caractères dans une chaîne, au lexique et à d'autres informations contextuelles. [7].Comme la classification peut aboutir à plusieurs candidats possibles, l'objectif du post-traitement est d'utiliser des informations de plus haut niveau (syntaxique, lexical, sémantique ...). Pour sélectionner une solution. Le post-traitement peut également vérifier si la réponse est correcte (même si elle est unique) en se basant sur d'autres informations non disponibles pour le classificateur.

I.9 Conclusion

Dans ce chapitre nous avons traité la problématique de la reconnaissance des caractères et les méthodes à cet effet .le principe de reconnaissance est basé sur l'obtention d'une base de données contenant des images de chaque caractère puis en passant par trois étapes nécessaire qui sont : le prétraitement, l'extraction des attributs et la classification puis le post-traitement. Nous allons utiliser le Deep Learning comme classificateurs qui sera détaillés ainsi que ces algorithmes y afférent dans les chapitres qui suivent

Chapitre II

Deep Learning

II.1 Introduction

Deep Learning est l'un des principaux paradigmes pour performer au de-là du machine learning. Au cours des dernières décennies les informaticiens ont développées avec sucées une large gamme de différents algorithmes qui permettent aux ordinateurs d'apprendre à résoudre des problèmes à travers d'exemples. En raison de la vague de technologies et de la recherche modernes avancées, Le DL est à la hausse depuis qu'il s'est avéré extrêmement efficace lorsqu'il s'agit d'apprendre à nos ordinateurs à faire ce que le cerveau humain peut faire naturellement et sans effort.

II.2 Deep learning

Le Deep Learning est un nouveau domaine de recherche du Machine Learning, qui a été introduit dans le but de rapprocher le ML de son objectif principal : l'intelligence artificielle (AI)
Le terme Deep learning a été introduit pour la première fois au ML par Dechter, et aux réseaux neurones artificiels par Ainzenberg et Al [17].

C'est une branche de l'apprentissage automatique basée sur un ensemble d'algorithmes qui tentent de modéliser des abstractions de haut niveau dans les données en utilisant plusieurs couches de traitement, composées de réseaux neuronaux artificiels. Cette technique a connu un succès récent en raison de l'augmentation de la puissance de traitement, des grandes quantités de données et de l'amélioration des algorithmes. Dans le DL de nombreuses couches d'opérations linéaire et non linéaire sont connectées en séquence pour former un réseau profond [18]. Dans ce biais, nous allons examiner les bases de l'apprentissage profond.

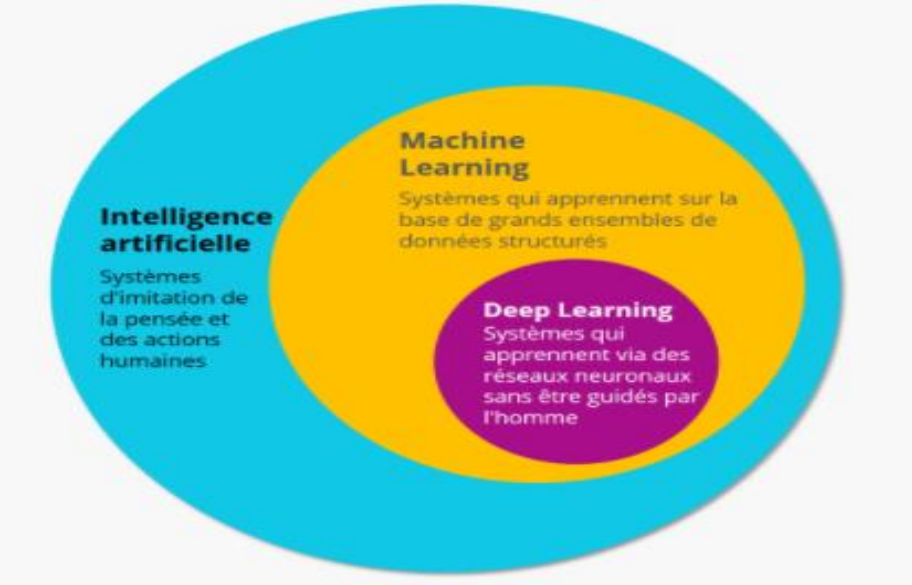

Figure II.1 Relation entre l'IA, ML et le DL [lm1].

II.2.1 Historique du deep learning

Les algorithmes à base de deep learning qui ont connu une énorme évolution durant la dernière décennie semblent prêts à résoudre bien des problèmes : reconnaitre des écritures, des visages comme le propose DeepFace, bientôt permettre l'autonomie de conduite de voitures ou encore la recherche de cellules cancéreuses.

Le tableau 2.1 présente l'historique des recherches effectué :

Année	Contributeur	Contribution
300 AC	Aristole	Introduction de l'associationnisme, début de l'histoire des humains qui tentent de comprendre le cerveau
1873	Alexander Bain	Introduction du Neural Groupings comme les premiers modèles de réseaux de neurones
1943	McCulloch and Pitts	Introduction du McCulloch–Pitts (MCP) modèle considérer comme L'ancêtre des réseaux de neurones artificiels
1949	Donald Hebb	Considérer comme le père des réseaux de neurones, il introduit la règle d'apprentissage de Hebb qui servira de fondation pour les réseaux de neurones modernes
1958	Frank Rosenblatt	Apparition du premier perceptron
1974	Paul Werbos	Arrivée de la retro propagation
1980	Teuvo Kohonen	Introduction des cartes auto organisatrices
1980	Kunihiko Fukushima	Introduction du Neocognitron, qui a inspiré les réseaux de neurone convolutif
1985	Hilton and Sejnowski	Apparition des machines de Boltzmann
1986	Paul Smolensky	Introduction d'Harmonium, qui sera connu plus tard comme machines de Boltzmann restreintes
1986	Michael I. Jordan	Définition et introduction des réseaux de neurones récurrents
1990	Yann LeCun	Introduction de LeNet et montra la capacité desréseaux de neurones profonds

1997	Hochreiter and Schmidhuber	Introduction de LSTM, qui a résolu le problème du vanishing gradient dans les réseaux de neurones récurrent
2006	Geoffrey Hinton	Introduction des Deep belief Network
2009	Salakhutdinov and Hinton	Apparition des Deep Boltzmann Machine
2012	Alex Krizhevsky	Arrivée d'AlexNet qui remporta le challenge ImageNet

Tableau II.1 Deep Learning à travers les années

II.3 Algorithme de Deep Learning

Il existe différents algorithmes de Deep Learning : Nous pouvons ainsi citer :

a) **Deep Neural Networks (DNN)** : Les réseaux de neurones profonds sont similaires aux réseaux perceptron multicouche MLP, mais avec plus de couches cachées. L'augmentation du nombre de couches permet à un réseau de neurones de détecter de légères variations du modèle d'apprentissage, favorisant le sur-apprentissage ou sur-ajustement « overfitting ».

b) **Convolutional Neural Networks(CNN) :** Le problème des réseaux de neurones convolutionnel est divisé en sous parties, et pour chaque partie, un «cluster» de neurones sera créé afin d'étudier cette portion spécifique. Par exemple, pour une image en couleur, il est possible de diviser l'image sur la largeur, la hauteur et la profondeur (les couleurs).

c) **Deep Belief Network (DBN)** : Les algorithmes de la machine de Boltzmann profonde fonctionnent suivant une première phase non supervisée, suivi de l'entraînement classique supervisé. L'étape d'apprentissage non-supervisée, permet, en outre, de faciliter l'apprentissage supervisé.

II.4 Neurone biologique

Le neurone est une cellule nerveuse. Elle se compose d'un corps cellulaire appelé « Somma »qui contient le noyau et de prolongement appelé «Neurite ».

Ces dernières sont de deux types :

- Les dendrites : qui servent de canal d'entrées.
- L'axone, unique : qui est le canal de sortie.

Au point de vue fonctionnel, le neurone est considéré comme une entité polarisée, l'information est transmise des dendrites vers l'axone.

Le neurone reçoit grâces aux dendrites des informations venant d'autres cellules nerveuses, on aura donc une sommation, au niveau du Somma de toutes ces informations via un signal électrique. Le résultat de l'analyse va transiter le long de l'axone jusqu'aux terminaisons synaptiques, a cet endroit, lors de l'arrivée du signal, des vésicules synaptiques vont venir fusionner avec la membrane cellulaire, ce qui va permettre la libération de neurotransmetteurs dans la fente synaptique. Le signal électrique ne pouvant pas passer la synapse (dans le cas d'une synapse chimique), les neurotransmetteurs permettent donc le passage des informations, d'un neurone à un autre. Les synapses possèdent une sortie de « mémoire » qui leur permet d'ajuster leur fonctionnement [18]. En fonction de leur activation répétée ou non entre deux neurones, les connexions synaptiques vont donc se modifier. Ainsi, la synapse va faciliter ou non le passage des influx nerveux. Cette plasticité est à l'origine des mécanismes d'apprentissage (figure II.2).

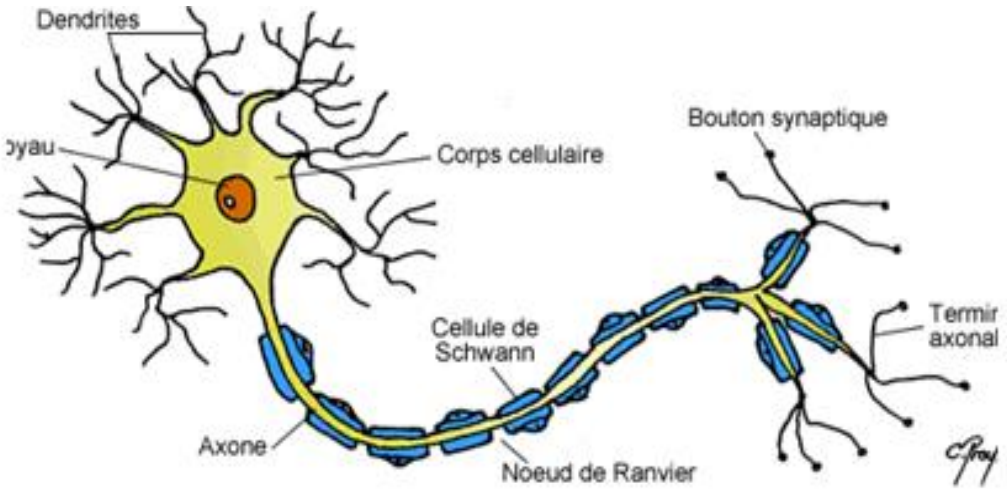

Figure II.2 Modèle d'un neurone biologique [Im2].

II.5 Réseaux de neurones

L'idée de base derrière les réseaux de neurones est de s'inspirer des propriétés du cerveau pour construire des systèmes de calcul capable de mieux résoudre le type de problèmes que les êtres vivant savent résoudre [19].

Les réseaux de neurones ressemblent au cerveau en deux points :

- La connaissance est acquise au travers d'un processus d'apprentissage
- Les poids des connections entre les neurones sont utilisés pour mémoriser la connaissance

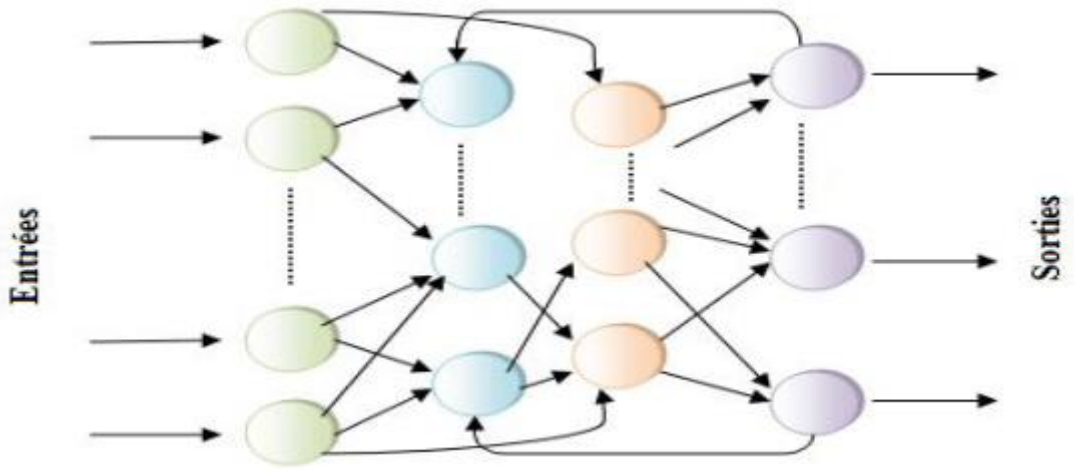

Figure II .3 Réseaux de neurones [Im3]

II.5.1 Neurone formel

C'est une modélisation mathématique qui reprend les principes du fonctionnement du neurone biologique. C'est une fonction non linéaire, paramétrée, à valeurs bornées, elle se caractérise par un état interne, des signaux d'entrée $X = \{x_1, x_2, \dots, x_n\}$ et une fonction de transition ou d'activation f (II.1) [20] comme suit :

$$f(s) = f\left(\sum_{i=1}^{n} w_i x_i + b\right) \qquad (\mathbf{II.1})$$

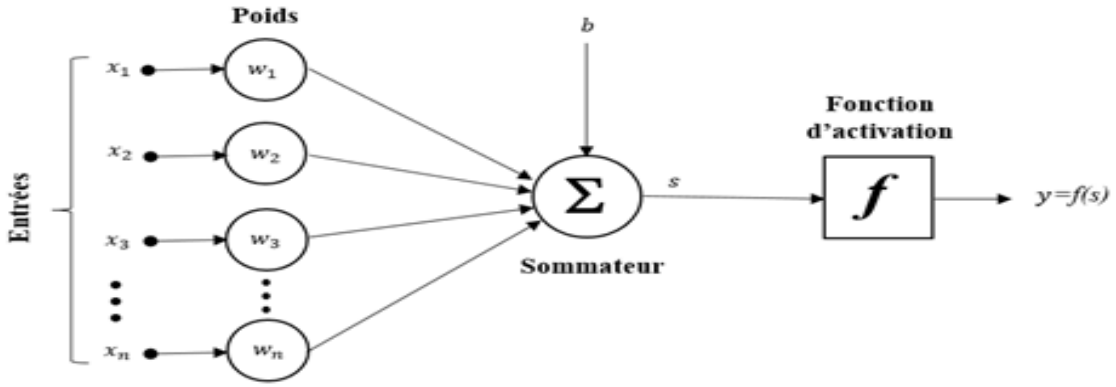

Figure II.4 Modèle d'un neurone artificiel

La figure II.4 ci-dessus illustre un modèle de neurone artificiel tel que :

$\{x_1, x_2, x_3, \dots, x_n\}$ **:** représentent les entrées.

$\{w_1, w_2, w_3, \dots w_n\}$ **:** représentent les poids de chaque entrée.

b **:** désigne le biais

s **:** est la somme pondérée des entrées.

La fonction d'activation dites fonction de seuillage, ou encore fonction de transfert est une transformation d'une combinaison des signaux d'entrée. Son rôle est d'introduire une non-linéarité dans le fonctionnement du neurone.

II.5.2 Perceptron

C'est le neurone artificiel le plus simple (monocouche), qui effectue des calculs afin de détecter des caractéristiques dans les données d'entrées. Il s'agit d'un algorithme pour l'apprentissage supervisé de classificateur binaire, qui permet aux neurones artificiels d'apprendre et de traiter les éléments d'un ensemble de données.

II.5.3 Perceptron multicouche

Un MLP est un réseau une ou plusieurs couches de neurones cachés (couches cachées) entre les couches d'entrée et de sortie, comme il est montré dans la figure II.5 d'où :

La couche d'entrée : reçoit les données d'entrée et les transmet à la première couche cachée.

Les couches cachées : effectuent des calculs mathématiques sur nos entrées. L'un des défis de la création de réseaux de neurones consiste à décider du nombre de couches cachées, ainsi que du nombre de neurones pour chaque couche.

Le terme "**Deep**" (profond) dans Deep Learning fait référence au fait d'avoir plus d'une couche cachée.

La couche de sortie : renvoie les données de sortie.

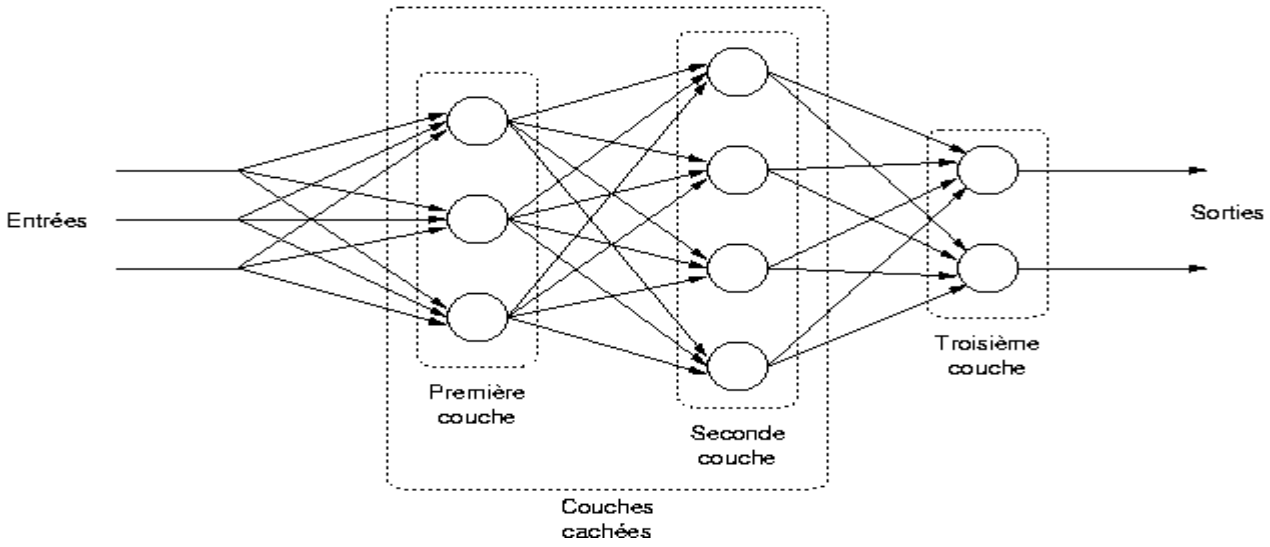

Figure II.5 Schéma d'un perceptron multicouche [Im4]

II.6 Algorithme du gradient

Cet algorithme permet de réaliser un apprentissage du réseau de neurones. On cherche à obtenir du réseau une réponse préétablie comme étant correcte. On dispose d'une base de connaissance de type entrée (p) – une sortie attendue (d). On compare ensuite la sortie (a) à la sortie attendue. On introduit la fonction d'erreur e = d − a que l'on va chercher à minimiser en modifiant les poids du réseau. Une fois les poids définis par cet algorithme à partir d'exemples connus de l'utilisateur, on va chercher à extrapoler le réseau, en lui fournissant des entrées inconnues.

Le but de l'algorithme de rétro-propagation est d'ajuster les poids synaptiques des neurones, de sorte que le réseau génère la sortie souhaitée. Il décrit le processus d'apprentissage. Le résultat de cet algorithme est un réseau neuronal configuré pour minimiser l'erreur lors de la résolution d'un problème.

L'apprentissage doit être effectué sur les données marquées et donc supervisée. Avant le commencement de l'algorithme, les poids doivent être initialisés à certaines valeurs. Il peut y avoir des rapproche différentes c'est pour cela que l'initialisation ne fait pas partie de la spécification de l'algorithme, la plus courante étant l'initialisation la plus banale - aléatoire. Ensuite, l'algorithme d'apprentissage commence.

II.7 Le processus d'apprentissage

Pour entraîner un perceptron, c'est-à-dire apprendre les poids de connexion, le but est de chercher à minimiser l'erreur de prédiction sur le jeu d'entraînement.
L'entraînement d'un perceptron est donc un processus **itératif.** Après chaque observation, il est nécessaire d'ajuster les poids de connexion de sorte à réduire l'erreur de prédiction faite par le perceptron dans son état actuel. Pour cela, l'algorithme du gradient est utilisé.

Un problème courant rencontré dans le processus d'apprentissage est le sur-apprentissage, qui est produit généralement lorsque l'apprentissage se fait trop longtemps, et surtout lorsque l'ensemble de ce dernier est trop faible pour représenter uniformément tous les types des modèles à partir du domaine des entrées possibles du réseau. Dans un tel cas, l'apprentissage peut ajuster le réseau aux fonctions aléatoires présentes dans ses données. L'accouplement est observé pendant le processus, lorsque la performance prédictive du réseau s'améliore sur l'ensemble d'apprentissage, tout en aggravant les données de test précédemment non vues.

Pour lutter contre ce problème, les données marquées sont divisées en un ensemble d'apprentissage et un ensemble de validation. La principale raison pour laquelle on utilise l'ensemble de validation est qu'il montre les taux d'erreur sur les données indépendantes des données sur lesquelles nous nous entraînons. Une étude de Guyon suggère que le rapport optimal entre la taille de l'ensemble des données d'apprentissage et de validation dépend du nombre des classes reconnues et de la complexité des caractéristiques de classe.
Une estimation de la complexité des caractéristiques est cependant assez lourde.
Un bon point de départ pour déterminer ce ration est de mettre 80% des données dans l'ensemble dans l'apprentissage et 20% dans l'ensemble de validations. Une expérimentation supplémentaire peut aider à se rapprocher du rapport optimal. Tout en apprenant, la performance de RNA est régulièrement examinée sur l'ensemble des données de validation. Lorsque les erreurs récupérées atteignent un point d'arrêt, le processus d'apprentissage est arrêté (FigureII.6) et le réseau est considéré comme formé.

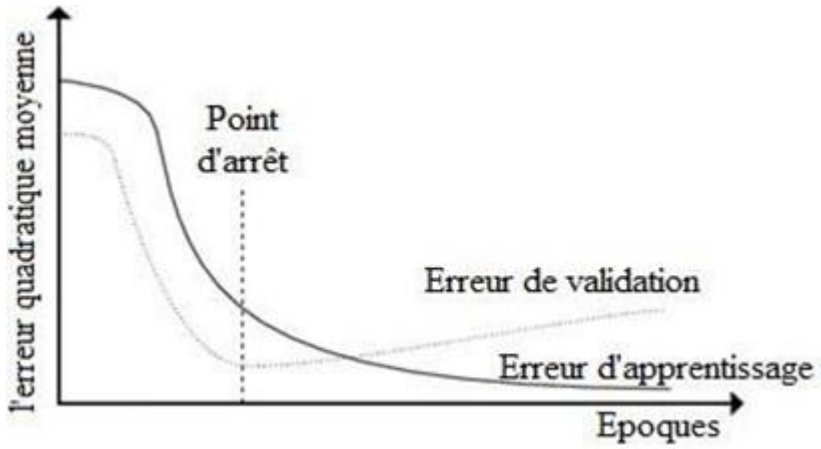

Figure II.6 Evolution de l'erreur d'apprentissage par rapport à l'erreur de validation

II.8 Techniques utilisées par les réseaux de neurones

Les techniques courantes de Machine Learning entrant dans la conception d'applicationsde réseaux de neurones comprennent :

II.8.1 Apprentissage supervisé

Les réseaux de neurones supervisés sont conçu afin de générer les sorties voulues en réponse aux échantillons d'entrée, dans le but de les adapter tout particulièrement à la modélisation et au contrôle de systèmes dynamiques, à la classification de données bruyantes et à la prédiction des événements futurs. Deep Learning Toolbox™ inclut quatre types de réseaux supervisés : **contrôleur boucle ouvert, base radiale, dynamique et quantification de vecteur d'apprentissage** [21] [22].

II .8.1.1Classification

La classification est un type de Machine Learning supervisé dans lequel un algorithme «apprend » à classifier de nouvelles observations à partir d'exemples de données étiquetées.

II.8.1.2Régression

Les modèles de régression décrivent la relation entre une variable réponse (sortie) et une ou plusieurs variables prédictives (entrée). Ici on n'attribue pas une classe mais une valeur mathématique : un pourcentage ou une valeur absolue.

II .8.2 Apprentissage non supervisé

La formation d'un réseau de neurones non supervisé s'effectue en le laissant s'ajuster continuellement aux nouvelles entrées. Il est utilisé pour tirer des conclusions à partir d'ensembles de données composés de données d'entrée sans réponses labellisées. On peut l'utiliser pour découvrir des distributions naturelles, des catégories et des relations de catégories ausein des données.

Deep Learning Toolbox inclut deux types de réseaux non supervisés : les couches cachées et les cartes auto-organisatrices de Kohonen [22].

II.9 Fonctions d'activation

On dit qu'une fonction d'activation est activée si sa sortie est non nulle, et pour une forte activation si la sortie est relativement élevée et vice versa. Il est en outre souhaité que la fonction $f(x) \approx x$ lorsque x s'approche de 0. Les fonctions d'activation doivent être non linéaires car il s'agit d'une caractéristique nécessaire pour que le réseau neural soit une approximation universelle. Pour les méthodes d'optimisation basées sur les gradients, des fonctions d'activation continuellement différentiables sont nécessaires.

Les fonctions d'activation monotone garantissent une surface d'erreur convexe d'un modèle à une seule couche. Les réseaux peuvent s'entraîner plus efficacement si $f(x) \approx x$ lorsque x s'approche de 0. Il existe plusieurs fonctions d'activation à choisir.

La fonction sigmoïde, la tangente hyperbolique et la fonction d'activation unité de rectification linéaire (ReLU) qui sont introduites et définies dans cette section. Certaines fonctions de classification et de régression utilisées dans la couche finale des réseaux de neurones sont également introduites.

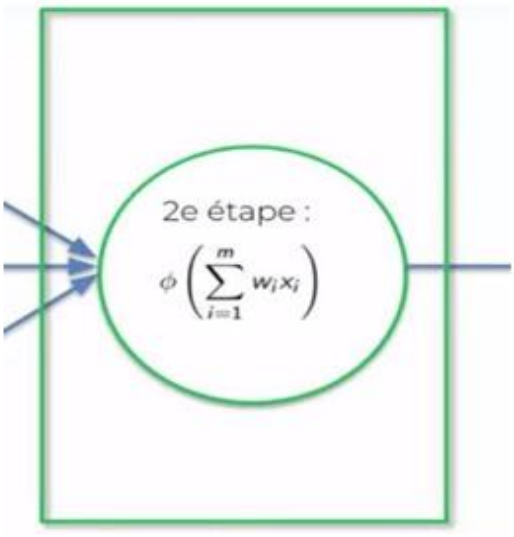

Figure II.7 Placement de fonction d'activation dans le modèle de réseau neuronal

II.9.1 Sigmoïde

La fonction sigmoïde logistique donnée par l'équation II.2, est une fonction d'activation largement utilisée, biologiquement plus plausible que la tangente hyperbolique. L'une des raisons pourlesquelles la fonction sigmoïde est largement utilisée est le fait qu'elle soit différente à chaque point [15].

$$f(x) = \frac{1}{1 + e^{-x}} \qquad (\mathbf{II.2})$$

II.9.2 Tangente hyperbolique

La fonction tangente hyperbolique donnée par est l'une des fonctions d'activation les plus populaires. L'entrée est une combinaison linéaire pondérée des entrées du nœud. Elle fonctionne le plus efficacement sur les entrées dans la plage (0 ; 1), produisant des sorties en intervalle (-1 ; 1)

$$\tanh(x) = \frac{\sinh(x)}{\cosh(x)} - \frac{\mathrm{e}^{\mathrm{x}} - \mathrm{e}^{-\mathrm{x}}}{\mathrm{e}^{\mathrm{x}} + \mathrm{e}^{-\mathrm{x}}} - \frac{\mathrm{e}^{2\mathrm{x}} - 1}{\mathrm{e}^{2\mathrm{x}} + 1} \qquad (\mathbf{II.3})$$

II.9.3 Unité de rectification linéaire (ReLU)

La fonction de l'unité de Rectification Linéaire (equation II.4) est utilisée dans le but d'augmenter la non-linéarité du réseau. Les neurones de rectification sont considérés comme biologiquement plus plausibles que les neurones tangentiels sigmoïdes ou hyperboliques tangents. Ils bénéficient de leur simplicité, entraînant une formation plus rapide et des améliorations de performance dans des cas particuliers, et donc souvent utilisés dans les RNC. ReLU est donné par l'équation :

$$f(x) = \max(0, x) \qquad (\mathbf{II.4})$$

II.9.4 Softmax

La fonction d'activation de softmax (equation II.5) est habituellement utilisée dans la dernière couche du réseau, lorsque le problème de classification n'est pas binaire mais contient plusieurs classes, cette dernière est généralement utilisée, mais est également viable pour le cas binaire. Il s'agit d'une généralisation de la fonction logistique et n'est qu'un autre nom pour un modèle de classification multinomiale quand on suppose qu'il n'existe aucune hiérarchie parmi les classes. La fonction softmax est agréable car elle donne une approximation de la probabilité qu'une classe soit correcte. L'approche la plus simple consiste à simplement choisir la classe avec la probabilité la plus élevée et ignorer

le reste. Mais étant donné qu'il s'agit d'une fonction probabiliste, il peut également être utilisé pour un modèle générateur. Les scores de softmax (probabilités) sont calculés par la fonction de normalisation.

$$\sigma(x)i = \frac{e^{x_i}}{\sum_{j=1}^{k} e^{x_j}} \qquad (\mathbf{II.5})$$

II.10 Réseaux de neurones convolutionnels (CNN)

Les réseaux de neurones convolutionnels, désigné par l'acronyme CNN, de l'anglais (Convolutional Neural Network), comportent deux parties distinctes. En entrées, une image est fournie sous la forme d'une matrice pixels à deux dimensions en niveau de gris, la couleur est représentée par une troisième dimension, de profondeur 3 pour les couleurs fondamentales RVB [23].

Parmi les réseaux convolutive célèbre on trouve :

- **LeNet :** C'est une structure de réseau de neurones convolutifs, il est utilisé pour lire les codes postaux, les chiffres…
- **AlexNet :** c'était le premier travail qui a popularisé les réseaux convolutifs dans la vision par ordinateur. AlexNet avait une architecture très similaire à LeNet, mais était plus profond, plus grand et comportait des couches convolutives empilées les unes sur les autres.
- **ZFnet :** c'était une amélioration de d'AlexNet en ajustant les hyper-paramètre de l'architecture, en particulier en élargissant la taille de couches convolutifs et en réduisant la taille du noyau sur la première couche.
- **GoogLeNet :** Sa principale contribution a été le développement d'un module inception qui a considérablement réduit le nombre de paramètre dans le réseau. En outre, ce module utilise le global AVG pooling au lieu de PMC à la fin du réseau, ce qui élimine une grande quantité de paramètre.
- **ResNet :** Residual network présente des sauts de connexion et une forte utilisation de la batch normalisation. Il utilise le global AVG pooling au lieu du PMC à la fin.

II.11 Construction d'un réseau CNN

La partie convolutive étant la première partie d'un CNN, fonctionne comme un extracteur de caractéristiques d'images.

La carte de convolution est créée à partir d'une image passée à travers une succession de

filtres ou de noyaux de convolutions. Certains des filtres intermédiaires réduisent la résolution de l'image par une opération de maximum local.

Finalement, les cartes de convolutions sont mises à plat et concaténé en un vecteur de caractéristiques appelé code CNN [23].

En sortie de la partie convolutive le code CNN est branché en entrée d'une deuxième partie, constituée de perceptron multicouche, son rôle est de combiner les caractéristiques de ce code pour classer l'image, comme illustré par la figure II.8

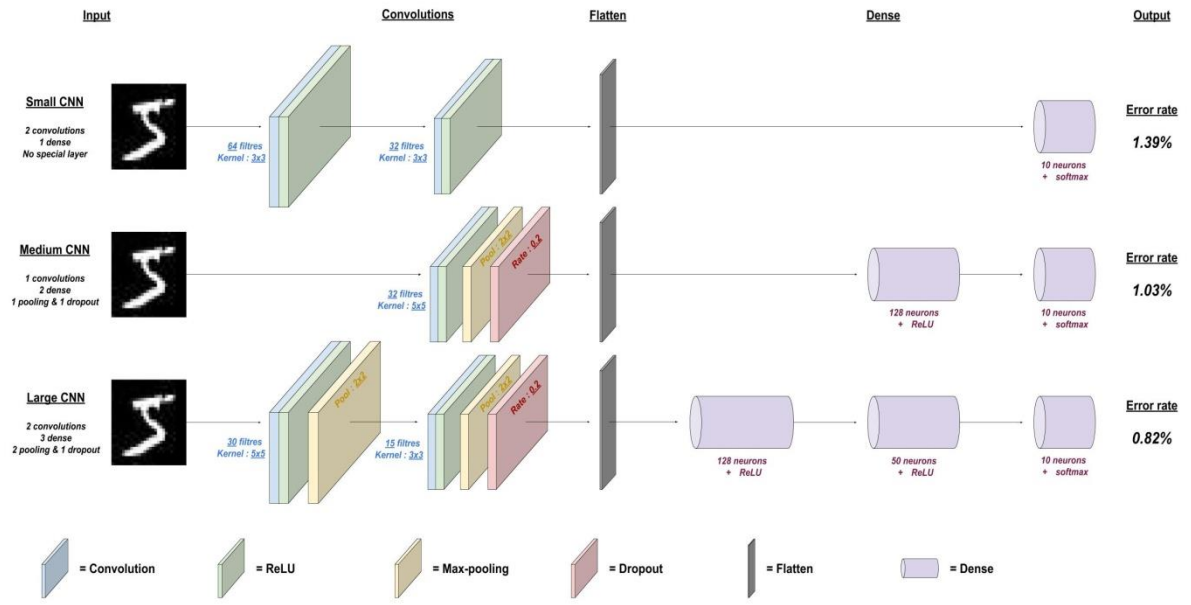

Figure II.8 Architecture d'un réseau CNN [Im5].

II.12 Architecture de réseaux de neurones convolutionnels

II.12.1 La couche de convolution

La couche de convolution représente la composante clé des réseaux de neurones convolutifs et constitue au moins leur première couche. C'est un outil mathématique simple qui est très largement utilisé pour le traitement d'image [24].

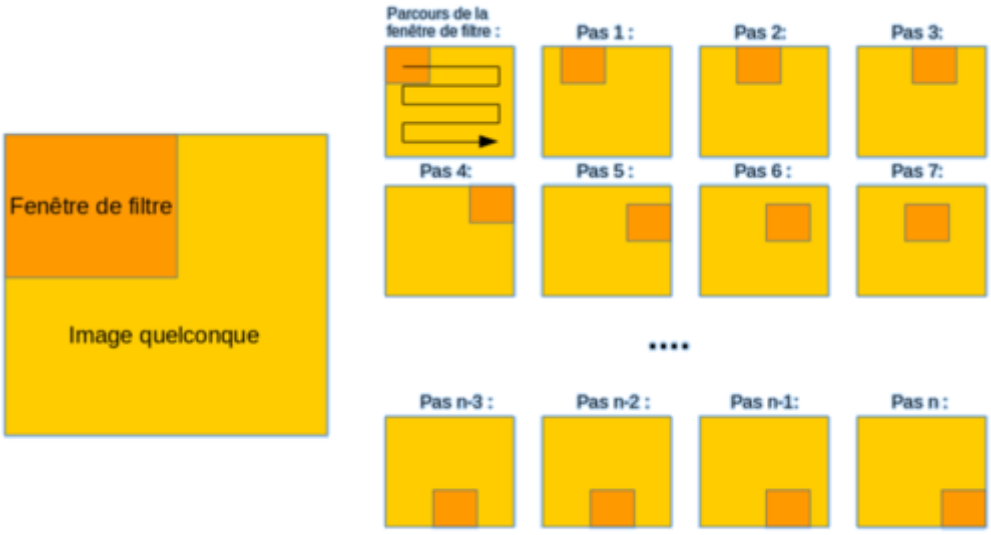

Figure II.9 schéma du parcours de la fenêtre de filtre sur l'image [Im6].

Elle a pour but de repérer la présence d'un ensemble de features dans les images reçu en entrée, grâce à un filtrage par convolution, qui consiste à faire 'glisser' une fenêtre représentant la feature sur l'image, ainsi que de calculer le produit de convolution entre ce dernier et chaque portion de l'image balayée. Par conséquent une feature est vue comme un filtre.

Pour chaque pair (image, filtre), on obtient une carte d'activation appelée feature map, qui indique ou se situent les features dans l'image.

Donnant l'exemple d'une convolution dont la configuration est représenté par l'illustration II.10 ci-dessous :

Opération= Argument maximal, Pas horizontal= 1px, Pas vertical=1Px [25].

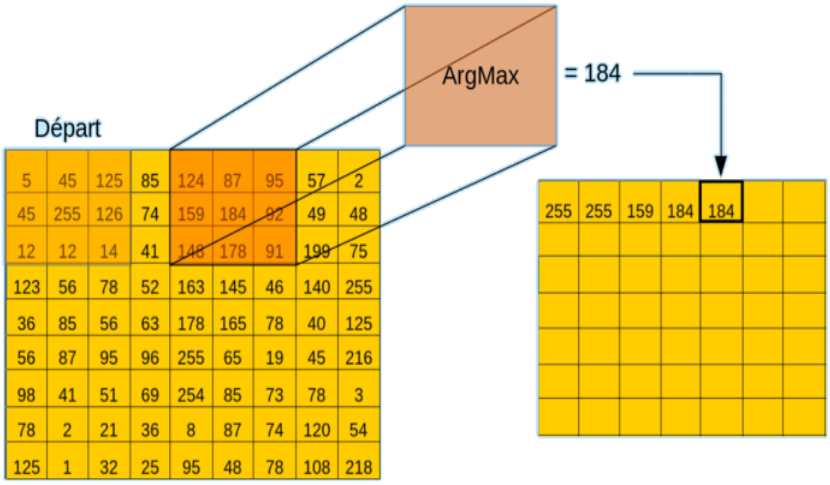

Figure II.10 Exemple d'une convolution [Im7].

II.12.2 Couche de mise en commun

Appelée couche pooling, elle est souvent placée entre deux couches de convolution : elle reçoit en entrée plusieurs feature maps, et applique à chacune d'entre elles l'opération de pooling. Cette dernière dite sub-sampling consiste à réduire progressivement la taille spatiale de la représentation, afin de diminuer la quantité de paramètres et de calculs dans le réseau, et donc de contrôler le sur-ajustement tout en préservant leurs caractéristiques importantes. Pour cela, on découpe l'image en cellules régulière. On améliore ainsi l'efficacité du réseau. En pratique, on utilise souvent des cellules carrées de petite taille pour ne pas perdre trop d'informations. Les choix les plus communs sont des cellules adjacentes de taille 2 × 2 pixels qui ne se chevauchent pas, ou des cellules de taille 3 × 3 pixels, distantes les unes des autres d'un pas de 2 pixels (qui se chevauchent). On obtient en sortie le même nombre de feature maps qu'en entrée [24] [25].

La couche de pooling fonctionne indépendamment sur chaque tranche de profondeur de l'entrée et la redimensionne spatialement, en utilisant l'opération MAX. Ainsi, la couche de pooling rend le réseau moins sensible à la position des features, le fait qu'une feature se situe un peu plus en haut ou en bas, ou même qu'elle ait une orientation légèrement différente ne devrait pas provoquer un changement radical dans la classification de l'image

Il existe plusieurs types de pooling :

- **Max Pooling**

Max Pooling est un processus de convolution où le Kernel (noyau) extrait la valeur maximale de la zone du feature map. Il est de loin le plus utilisé car son calcul est immédiat et les images sont efficacement simplifiées, comme le montre la figure II.10

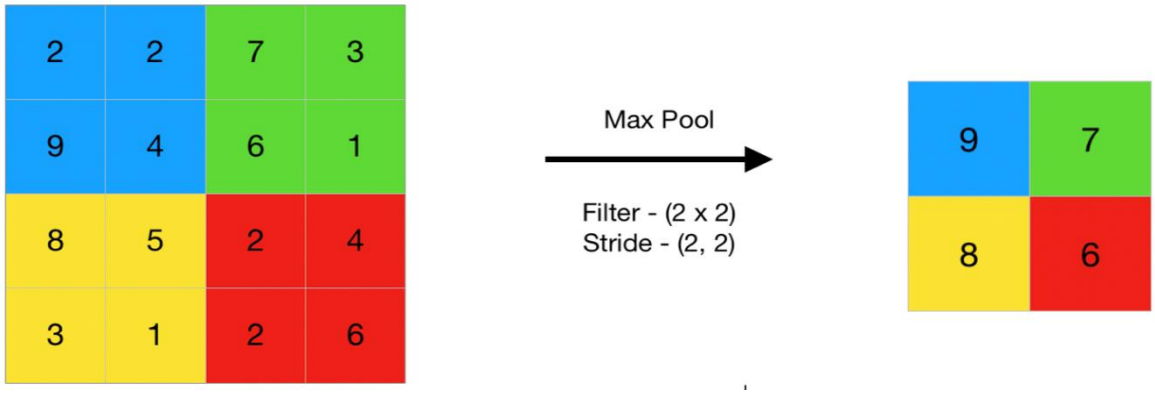

Figure II.11 Représentation de max pooling [Im8]

- **Mean Poolin**

Appelé aussi average pooling, calcule la moyenne des éléments présent dans le feature map couverte par le filtre. Alors que le max pooling donne la feature la plus importante dans un patch particulier de la feature map, le mean pooling donne la moyenne des fonctionnalités présentes dans un patch, comme le montre la figure II.11

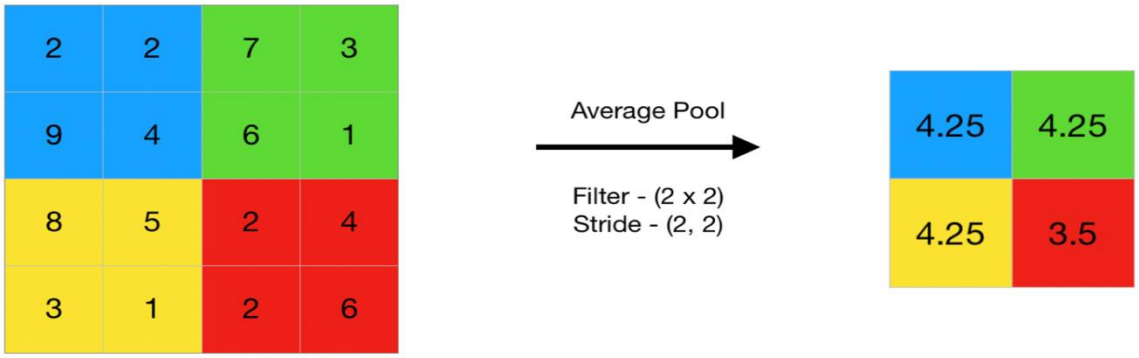

Figure II.12 Représentation de l'average pooling [Im8].

- **Sum Pooling**

C'est la moyenne sans avoir à divisé par le nombre de valeurs (on ne calcule que leur somme

II.12.3 La couche ReLU

Pour améliorer l'efficacité du traitement en intercalant entre les couches de traitement, une couche va opérer une fonction mathématique (fonction d'activation) sur les signaux de sortie. Dans ce cadre on trouve ReLU (Rectified Linear Units) désigne la fonction réelle non-linéaire définie par ReLU(x)= max (0, x). La couche de correction ReLU remplace toutes les valeurs négatives reçues en entrées par des zéros. Elle joue souvent le rôle de fonction d'activation [25], la correction ReLU est choisie car il en résulte la formation de réseau neuronal plus rapide.

II.12.4 Couche fully-connected

Elle comporte toujours la dernière couche d'un réseau de neurones, convolutif ou non, elle n'est donc pas une caractéristique d'un CNN. Ce type de couche reçoit un vecteur en entrée et produit un nouveau vecteur en sortie. A cet égard, elle applique une combinaison linéaire puis éventuellement une fonction d'activation aux valeurs reçues en entrée. La couche fully-Connected (FC) permet de classifier l'image en entrée du réseau en renvoyant un vecteur de taille N, où N est le nombre de classes dans notre problème de classification d'images. Chaque élément du vecteur indique la probabilité pour l'image en entrée d'appartenir à une classe.

Chaque valeur du tableau en entrée opte pour classe. Les votes n'ont pas tous la même importance : la couche leur accorde des poids qui dépendent de l'élément du tableau et de la classe. Pour calculer les probabilités, la couche FC multiplie donc chaque élément en entrée par un poids, fait la somme, puis applique une fonction d'activation (logistique si N=2, softmax si N>2) : Il est possible de remplacer les couches entièrement connectées d'un CNN par des couches convolutionnelles, ce qui le rend entièrement convolutionnel.

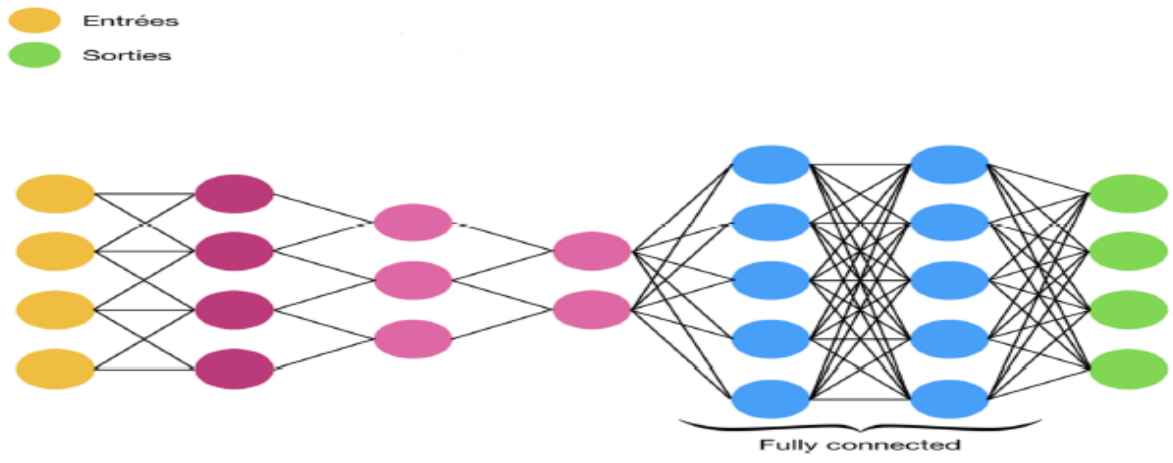

Figure II.13 Représentation de la couche fully-connected [Im9].

II.13 Conclusion

Dans ce chapitre, nous avons traité les réseaux de neurones et particulièrement les réseaux de neurones convolutionnels, en détaillant leur conception et leur utilisation dans le traitement de la reconnaissance.

Dans le chapitre qui suit nous allons procéder à l'application de ces réseaux de neurone convolusionnels dans la reconnaissance de chiffres et des caractères arabes. Ces réseaux sont capables d'extraire des caractéristiques d'images présentées en entrée et de classifier ces caractéristiques. Ils implémentent l'idée de partage des poids permettant ainsi de réduire d'une part le nombre de paramètres libres de l'architecture, d'autre part de réduire les temps de calcul, l'espace mémoire nécessaire, et améliorer ainsi les capacités de généralisation du réseau.

Chapitre III

Implémentation et Résultat

III .1 Introduction

Le Deep Learning a montré de bonnes performances dans de nombreuses taches d'intelligence artificielle et d'apprentissage automatique

Les systèmes de reconnaissance de chiffre et des caractères arabes manuscrits font face à plusieurs défis, y compris la variation illimitée de l'écriture humaine et des grandes bases de données publique

Dans ce chapitre, on va faire lumière sur la conception de nos différentes architectures de CNN pour deux bases, MNIST pour la reconnaissance des chiffres manuscrit et IFHCDB pour la reconnaissance des caractères arabe manuscrites, et montrer comment réaliser notre classificateur et créer notre modèle, via le langage de programmation python.

III.2 conception

Avant d'entrer dans l'implémentation, on va expliquer comment concevoir notre programme.

Il y a deux grands processus qui englobent notre conception

III.2.1 Training

C'est le processus le plus important parce qu'on va créer notre modèle grâce à des configurations précises.

- **La base de données** : c'est une base de données d'images répertoriées en classes. Par exemple, Si on prend une base de données de caractères arabe on peut trouver des classe comme 'ba' 'alif' 'jiim' ... etc.
- **labels_classes** : c'est un fichier texte qui portera les noms des classes de notre dataset.
- **CNN et paramètres** : c'est notre algorithme de création d'un réseau neurones convolutionels qui sera configuré avec des paramètres, par exemple : nombre d'époques, nombre de filtre, nombre de couche ...etc.

On va exécuter la dataset sur notre algorithme CNN qui sera paramétrée pour générer un modèle puis on va utiliser ce modèle pour le Test.

III.2.2.Test

Dans le processus Test on retrouve :

- **La base de données** : c'est une base de données d'images répertoriées en classes contenant un nombre inférieure d'images par rapport au training.

- **Le modèle** : c'est un fichier généré dans notre training.
- **L'affichage de la classification** : son nom résume son travail, on va afficher le résultat sortie du modèle qui est le nom d'une classe.

III.3 Logiciel et librairies utilisé dans l'implémentation

III.3.1 Python

Python est un langage de programmation de haut niveau interprété et orienté objet avec une sémantique dynamique. Il est très sollicité par une large communauté de développeurs et de programmeurs.

Python est un langage simple, facile à apprendre et permet une bonne réduction du cout de la maintenance des codes. Les bibliothèques (packages) encouragent la modularité et la réutilisabilité des codes.

Python et ses bibliothèques sont disponibles (en source ou en binaires) sans charges pour la majorité des plateformes et peuvent être redistribués gratuitement.

III.3.2 Tensorflow

TensorFlow est une bibliothèque logicielle open source pour le calcul numérique de haute performance. Son architecture flexible permet un déploiement facile du calcul sur diverses plates-formes (CPUs ,GPUs ,TPUs), et des ordinateurs de bureau aux clusters de serveurs, aux périphériques mobiles. Initialement développé par des chercheurs et des ingénieurs de l'équipe de Google Brain au sein de l'organisation de l'IA de Google, il s'appuie sur l'apprentissage automatique et l'apprentissage en profondeur.

III.3.3 keras

Keras est une API de réseaux neuronaux de haut niveau, écrite en Python et capable de s'exécuter sur TensorFlow, CNTK ou Theano. Il a été développé dans le but de permettre une expérimentation rapide. Être en mesure de passer de l'idée au résultat le plus rapidement possible, est la clé pour faire de la recherche :

- Permet un prototypage facile et rapide (grâce à la convivialité, à la modularité et à l'extensibilité).
- Prend en charge les réseaux convolutionnels et les réseaux récurrents ainsi que les combinaisons des deux.
- Fonctionne de manière transparente sur le processeur et le processeur graphique [27].

III.3.3.1 Module keras utilisés

Les modules de Keras qu'on a utilisé sont :

-Keras.models : Sequential

-Keras.layers : Conv2D MaxPooling2D Flatten Dense Dropout

-Keras.preprocessing.image : ImageDataGenerator

III .4 configuration utilisés dans l'implémentation

La configuration utilisée du matériel dans notre configuration est :

- PC portable HP 250 core i3 CPU 2.5GHZ
- RAM de taille 8 GO
- Carte graphique Nvidia GeForce 920M
- Disque dur de taille 500 GO
- Système d'exploitation windows 8.1 professional

III .5 Base de données

- **MNIST**

L'acronyme MNIST (Modified ou Mixed National Institute of Standards and Technology*)*, est une base de données de chiffres écrits à la main. C'est un jeu de données très utilisé en apprentissage automatique.

La reconnaissance de l'écriture manuscrite est un problème difficile, et un bon test pour les algorithmes d'apprentissage. La base MNIST est devenue un test standard. Elle regroupe 60,000 images d'apprentissage et 10,000 images de test, issues d'une base de données antérieure, appelée simplement NIST. Ce sont des images en noir et blanc, normalisées centrées de 28 pixels de côté.

Figure III.1 structure de la base MNIST

- **IFHCDB**

Isolated Farsi Handwritten Character DataBase a été créé au département d'ingénierie électrique du département d'ingénierie électrique de l'université technologique d'Amirkabir (AUT), à Téhéran, en Iran, en 2006. L'IFHCDB est un sous-ensemble d'une grande base de données recueillie dans le cadre d'un projet de recherche. Son objectif principal est d'aider les chercheurs à développer de nouvelles techniques, technologies et algorithmes pour la reconnaissance automatique des caractères manuscrits Farsi/Arabe [26].

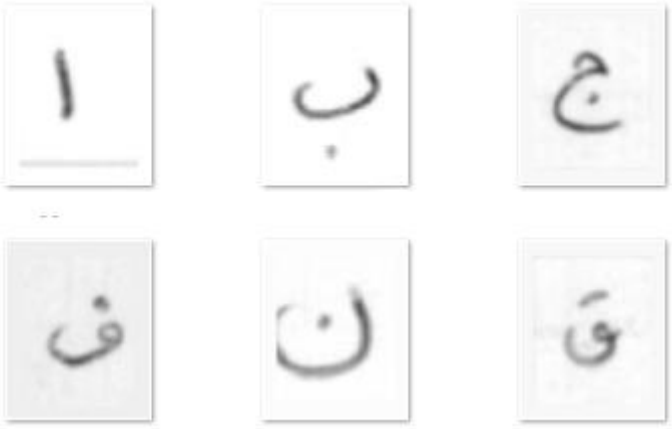

Figure III.2 structure de la base de données IFHCDB

III.6 Architecture du réseau CNN

Au cours de nos expérimentations, nous avons créé trois modèles avec différentes architectures qu'on a appliqué à la base de données MNIST et la base IFHCDB et pour chaque modèle, on a fait une évaluation sur le nombre d'époques. Dans ce qui suit, on présentera l'architecture des 3 modèles avec les deux bases et le résultat obtenu à chaque fois

III.6.1 Réseau neuronal convolutif à 2 couches

L'architecture du réseau à 2 couches pour la reconnaissance des caractères arabes manuscrit est basée sur le réseau de neurones convolutionnel dont la structure est présentée par la **figure III .3**

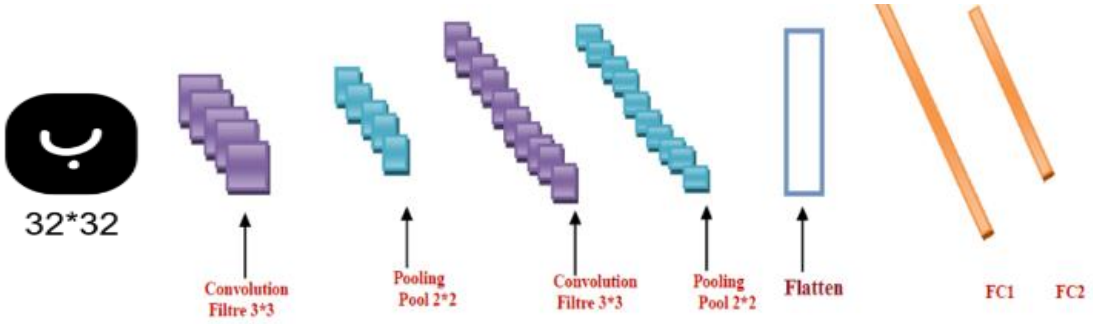

Figure III.3 Architecture du modèle à 2couches

Le modèle est composé de deux couches de convolution, deux couches de maxpooling et deux couches de fully connected.

L'image en entrée est de taille 32*32, elle passe d'abord à la première couche de convolution. Cette couche est composée de 32 filtres de taille 3*3, Chacune de nos couches de convolution est suivie d'une fonction d'activation ReLU cette fonction force les neurones à retourner des valeurs positives, Le Maxpooling est appliqué après pour réduire la taille de l'image 32 features maps (images caractéristiques) de taille 28*28 seront créés. Ensuite on applique Maxpooling pour réduire la taille de l'image ainsi la quantité de paramètres et de calcul. À la sortie de cette couche, nous aurons 32 feature maps de taille 15*15. On répète la même chose avec la deuxième couche de convolution cette couche est composée de 64 filtres, la fonction d'activation ReLU est appliquée toujours sur chaque convolution. Une couche de Maxpooling est appliquée après la deuxième couche de convolution. À la sortie de cette couche, nous aurons 64 feature maps de taille 6*6. Pour ne pas tomber dans le problème de sur apprentissage il faut utiliser l'instruction Dropout elle est très efficace pour les réseaux de

neurones, elle permet de désactiver un nombre de neurones selon notre configuration, cette dernière sera utilisée aussi à la sortie de la première couche de fully connected. Le vecteur de caractéristiques issu des convolutions a une dimension de 2304. Après ces couches de convolution, nous utilisons un réseau de neurones composé de deux couches fully connected, la première couche possède 128 neurones où la fonction d'activation utilisée est le ReLU, pour la deuxième couche sa fonction d'activation est softmax qui permet de calculer la distribution de probabilité des 28 classes (nombre de classe dans la base donnée IFHCDB).

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 30, 30, 32)	896
max_pooling2d (MaxPooling2D)	(None, 15, 15, 32)	0
conv2d_1 (Conv2D)	(None, 13, 13, 64)	18496
max_pooling2d_1 (MaxPooling2	(None, 6, 6, 64)	0
dropout (Dropout)	(None, 6, 6, 64)	0
flatten (Flatten)	(None, 2304)	0
dense (Dense)	(None, 128)	295040
dropout_1 (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 28)	3612

Total params: 318,044
Trainable params: 318,044
Non-trainable params: 0

Figure III.4 Configuration du modèle à 2 couches

III .6 .2 réseau neuronal convolutif à 4 couches

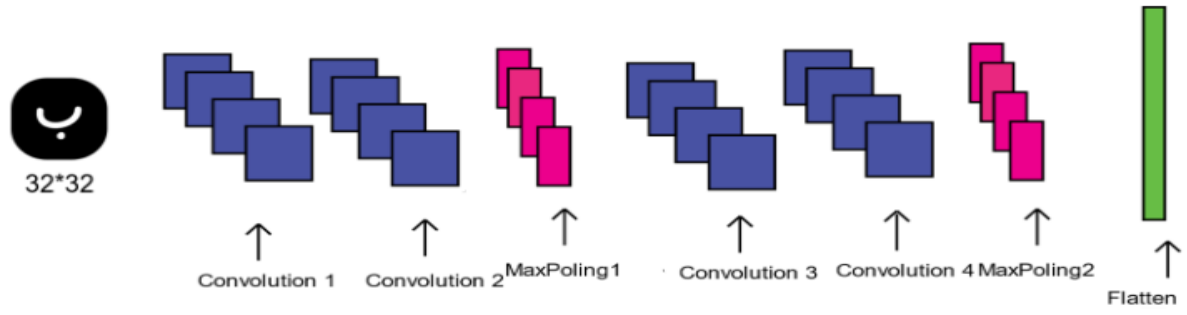

Figure III.5 Architecture du modèle à 4couches

Le modèle que nous présentons dans **la figure III.5** est composé de quatre couches de convolution, deux couches de maxpooling et de trois couches de fully connected.

L'image en entrée est de taille 32*32, chaque couche de convolution composée de plusieurs filtres (32 pour les 2 première couche), où la taille de chaque filtre est de 5*5. La fonction d'activation ReLU est utilisée à chaque fois qu'on passe par une couche de convolution, après cette convolution, 32 features maps de taille 28*28 seront créés.
Le Maxpooling est appliqué la fin de la deuxième couche de convolution
Dans la troisième et quatrième couche, on change quelques paramètres comme le nombre de filtres qui devient 64 au-lieu de 32, la ReLU reste la même tout comme maxpooling, nous aurons 64 feature maps de taille 5*5 puis on utilise Dropout.
Après ces quatre couches nous utilisons un réseau de neurones composé de trois couches fully- connected. Pour convertir nos données 3D en 1D, nous utiliserons la fonction Flatten, Les deux premières couches ont chacune 256 neurones où la fonction d'activation utilisée est le ReLU, tant dis que pour la troisième couche, c'est un softmax, illustré par la **figure III.6**

Layer (type)	Output Shape	Param #
conv2d_20 (Conv2D)	(None, 32, 32, 32)	2432
conv2d_21 (Conv2D)	(None, 28, 28, 32)	25632
max_pooling2d_10 (MaxPooling	(None, 14, 14, 32)	0
conv2d_22 (Conv2D)	(None, 12, 12, 64)	18496
conv2d_23 (Conv2D)	(None, 10, 10, 64)	36928
max_pooling2d_11 (MaxPooling	(None, 5, 5, 64)	0
flatten_5 (Flatten)	(None, 1600)	0
dense_15 (Dense)	(None, 256)	409856
dropout_10 (Dropout)	(None, 256)	0
dense_16 (Dense)	(None, 256)	65792
dropout_11 (Dropout)	(None, 256)	0
dense_17 (Dense)	(None, 28)	7196

Total params: 566,332
Trainable params: 566,332
Non-trainable params: 0

Figure III.6 configuration de modèle (4 couche)

III.6.3 réseau neuronal à 6 couches

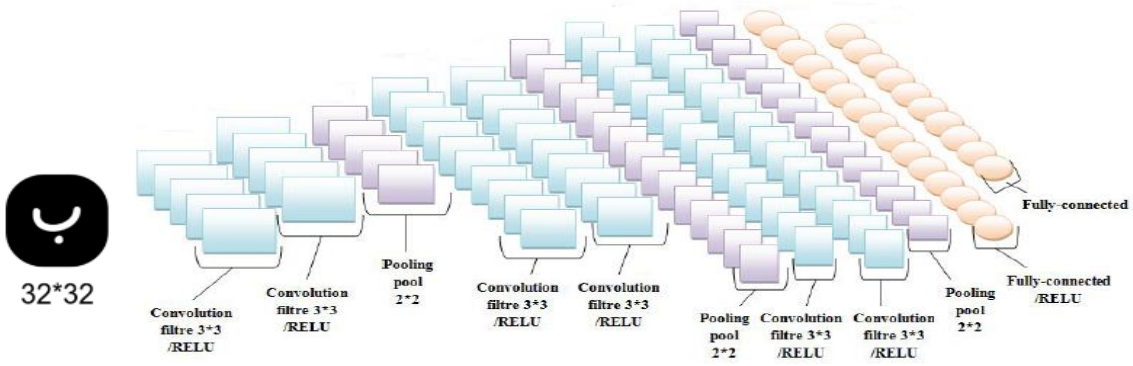

Figure III.7 Architecture du modèle à 6 couches

Sur ce troisième modèle, on a modifié le nombre de couches pour la convolution et le maxpooling, maintenant notre architecture est composée de six couches convolution, trois couches de maxpooling et de trois dernières couches de fully connected.

Comme pour les deux autres modèles, l'image en entrée a une taille de 32*32, chaque couche de convolution composée de plusieurs filtres (32 pour les deux premières couches, 64 pour les deux autres couches et 128 filtres pour les deux dernières couches), la taille de chaque filtre est inchangeable pour les 6 couches de convolution (elle est de 3*3), la fonction d'activation utilisée pour toutes ces couches est ReLU, à la fin de chaque deux couches on utilise le Maxpooling. .

On utilise Dropout avec un pourcentage de 20% après chaque couche de convolution et de regroupement,

Après ces six couches nous utilisons un réseau de neurones composé de trois couches fully-connected en passant par la fonction flatten qui convertira les données à une dimension.

Layer (type)	Output Shape	Param #
conv2d_44 (Conv2D)	(None, 32, 32, 32)	896
conv2d_45 (Conv2D)	(None, 30, 30, 32)	9248
max_pooling2d_22 (MaxPooling	(None, 15, 15, 32)	0
dropout_16 (Dropout)	(None, 15, 15, 32)	0
conv2d_46 (Conv2D)	(None, 13, 13, 64)	18496
conv2d_47 (Conv2D)	(None, 11, 11, 64)	36928
max_pooling2d_23 (MaxPooling	(None, 6, 6, 64)	0
dropout_17 (Dropout)	(None, 6, 6, 64)	0
conv2d_48 (Conv2D)	(None, 4, 4, 128)	73856
conv2d_49 (Conv2D)	(None, 2, 2, 128)	147584
max_pooling2d_24 (MaxPooling	(None, 1, 1, 128)	0
dropout_18 (Dropout)	(None, 1, 1, 128)	0
flatten_2 (Flatten)	(None, 128)	0
dense_6 (Dense)	(None, 256)	33024
dense_7 (Dense)	(None, 128)	32896
dropout_19 (Dropout)	(None, 128)	0
dense_8 (Dense)	(None, 28)	3612

Total params: 356,540
Trainable params: 356,540
Non-trainable params: 0

Figure III.8 Configuration du modèle à 6 couches

III.7 code source de notre architecture

```
train_datagen=ImageDataGenerator(rescale=(1./255),shear_range=0.2,zoom_range=0.2,horizontal_flip=(True))
test_datagen=ImageDataGenerator(rescale=(1./255),shear_range=0.2,zoom_range=0.2,horizontal_flip=(True))
training_set= train_datagen.flow_from_directory(path_train,target_size=(32,32),class_mode='categorical')
test_set= test_datagen.flow_from_directory(path_test,target_size=(32,32),class_mode='categorical')
```

- Charger et ajuster la base de données d'image
- 35988 images d'apprentissage
- 15041 images de test
- Compiler le modèle

```
model = Sequential()
model.add(Conv2D(32, kernel_size=(5, 5),activation='relu',padding='same',input_shape=(32,32,3)))
model.add(Conv2D(32, kernel_size=(5, 5),activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(64, kernel_size=(3, 3),activation='relu'))
model.add(Conv2D(64, kernel_size=(3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2),strides=(2,2)))
model.add(Flatten())
model.add(Dense(units=256, activation='relu'))
model.add(Dropout(0.25))
model.add(Dense(256, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(28, activation='softmax'))
```

- Construction de notre modèle CNN à 4 couches

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

- Compiler le modèle

```
history=model.fit(training_set,epochs=20,validation_data=test_set,validation_steps=105)
```

- Lancement de l'apprentissage

```
score= model.evaluate(test_set)
```

- Phase d'évaluation

III.8 Résultats et discussions pour la base IFHCDB

III.8.1 Résultats obtenus pour le modèle à 2 couches

- **Nombre d'époques = 10**

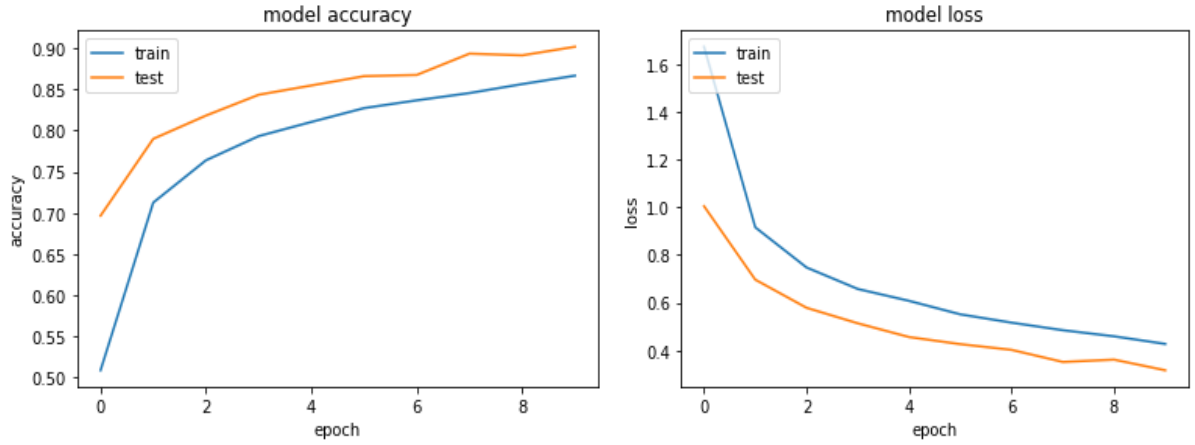

Figure III.9 Précision et erreur du modèle à 2 couches (10 époques)

D’après la figure III.9, la précision d’apprentissage (train) et celle du test augmente avec le nombre d’époque.

Contrairement à l'erreur qui diminue avec l’augmentation du nombre d’époques.

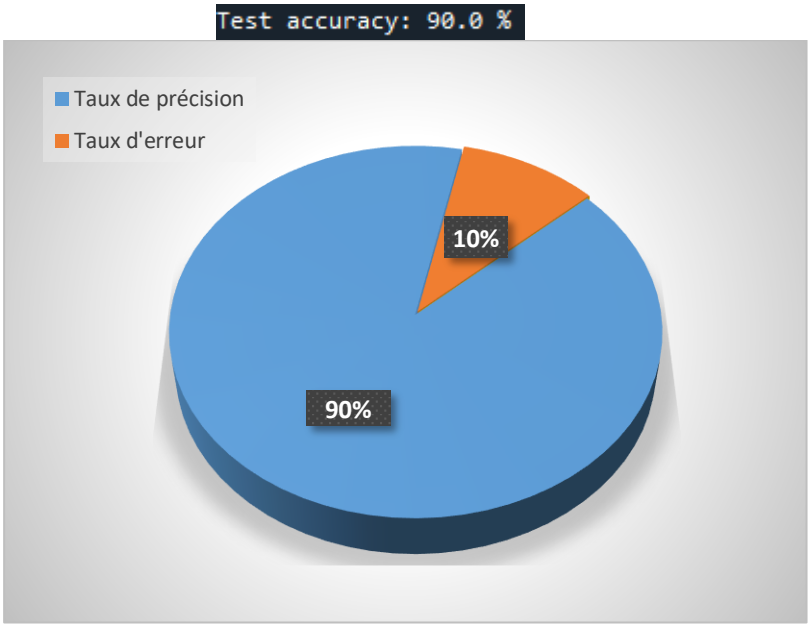

Figure III.10 Taux de précision et d’erreur du modèle à 2couche (10époques)

La figure III.10, illustre un taux de précision de 90% qui correspond à un total de 13537 caractères bien reconnus, et un taux d’erreur de 10% qui correspond à un total de 1504 caractères mal reconnus.

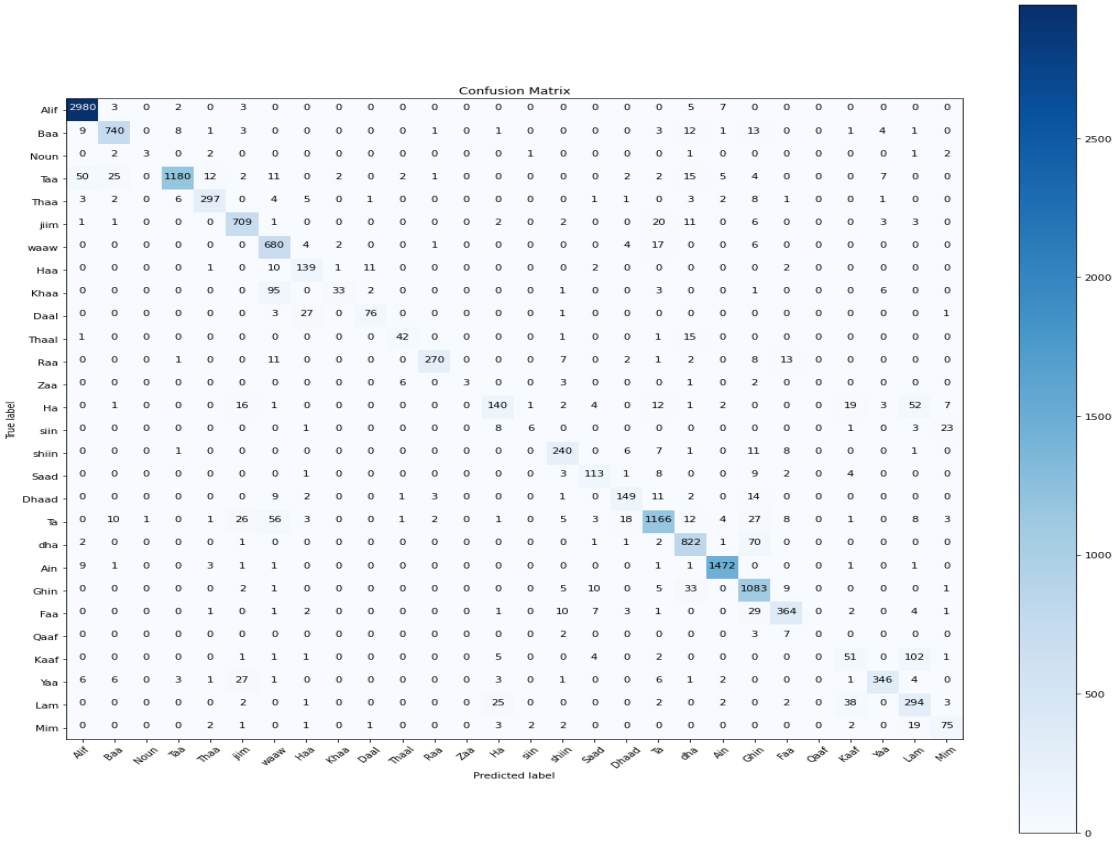

Figure III.11 Matrice de confusion pour le modèle à 2 couches (10époques)

La matrice de confusion permet d'évaluer la performance de notre modèle, puisqu'elle reflète les métriques du Vrai positif, Vrai négatif, Faux positif et Faux négatif. La figure III.11 illustre de près la position de ces métriques pour chaque classe. On prend pour exemple, le modèle a bien classé les caractères Alif, Ain et TA et a mal classé les caractères Qaaf, Siin et Noun.

- **Nombre d'époques = 20**

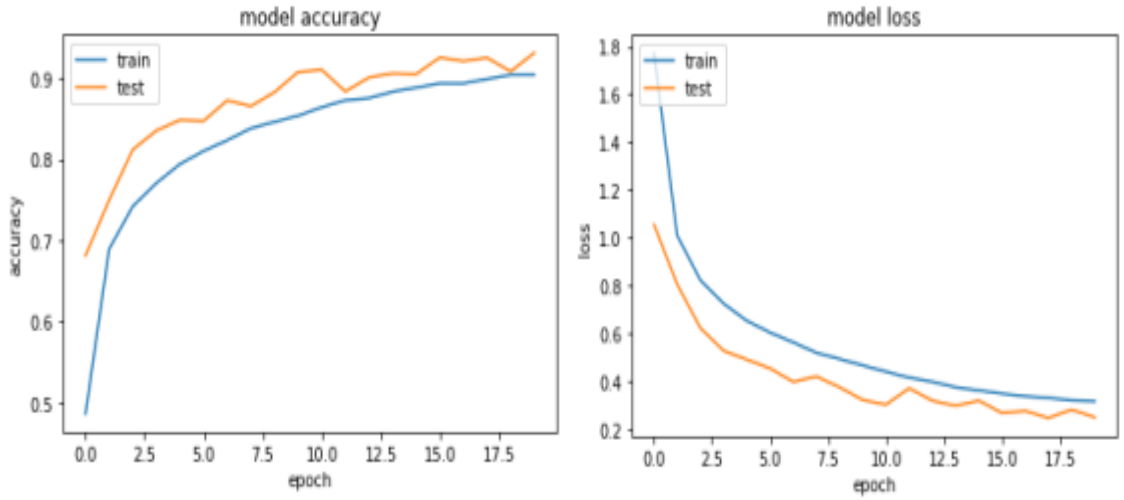

Figure III.12 Précision et erreur du modèle à 2 couches (20 époques)

D'après la figure III.12, on remarque que la précision de l'apprentissage et du test dans le model accuracy augmente avec le nombre d'époques. Ainsi, on voit que pour 20 époques le modèle à reconnu les caractères d'une manière plus précise par rapport au modèle à 10 époques avec une précision de pratiquement 80% dès les 2 première époques, sur l'intervalle [2.5-10] époques, la précision d'apprentissage (train) et de test ne cesse d'augmenter atteignant les 90% pour le test , puis sur l'intervalle [10-11] époques, la précision du test chute de 5% ,pour ensuite augmenter sur l'intervalle [11-17] et chuter sur l'intervalle [17-17.5] pour enfin augmenter jusqu'à stabilisation

De même, l'erreur d'apprentissage et du test est proportionnellement contraire à celle de précision.

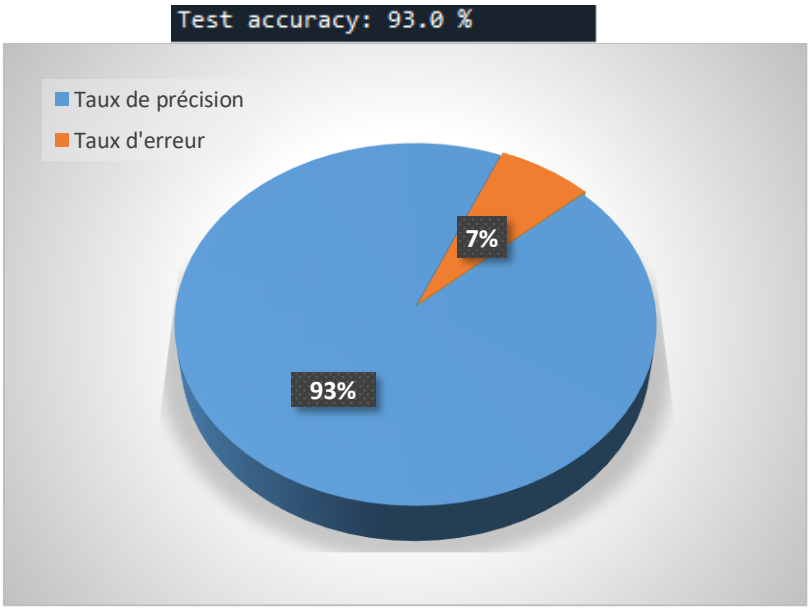

Figure III.13 Taux de précision et d'erreur du modèle à 2 couches (20 époques)

La figure III.13, illustre un taux de précision de 93% qui correspond à un total de 13988 caractères bien reconnus, et un taux d'erreur de 7% qui correspond à un total de 1052 caractères mal reconnus.

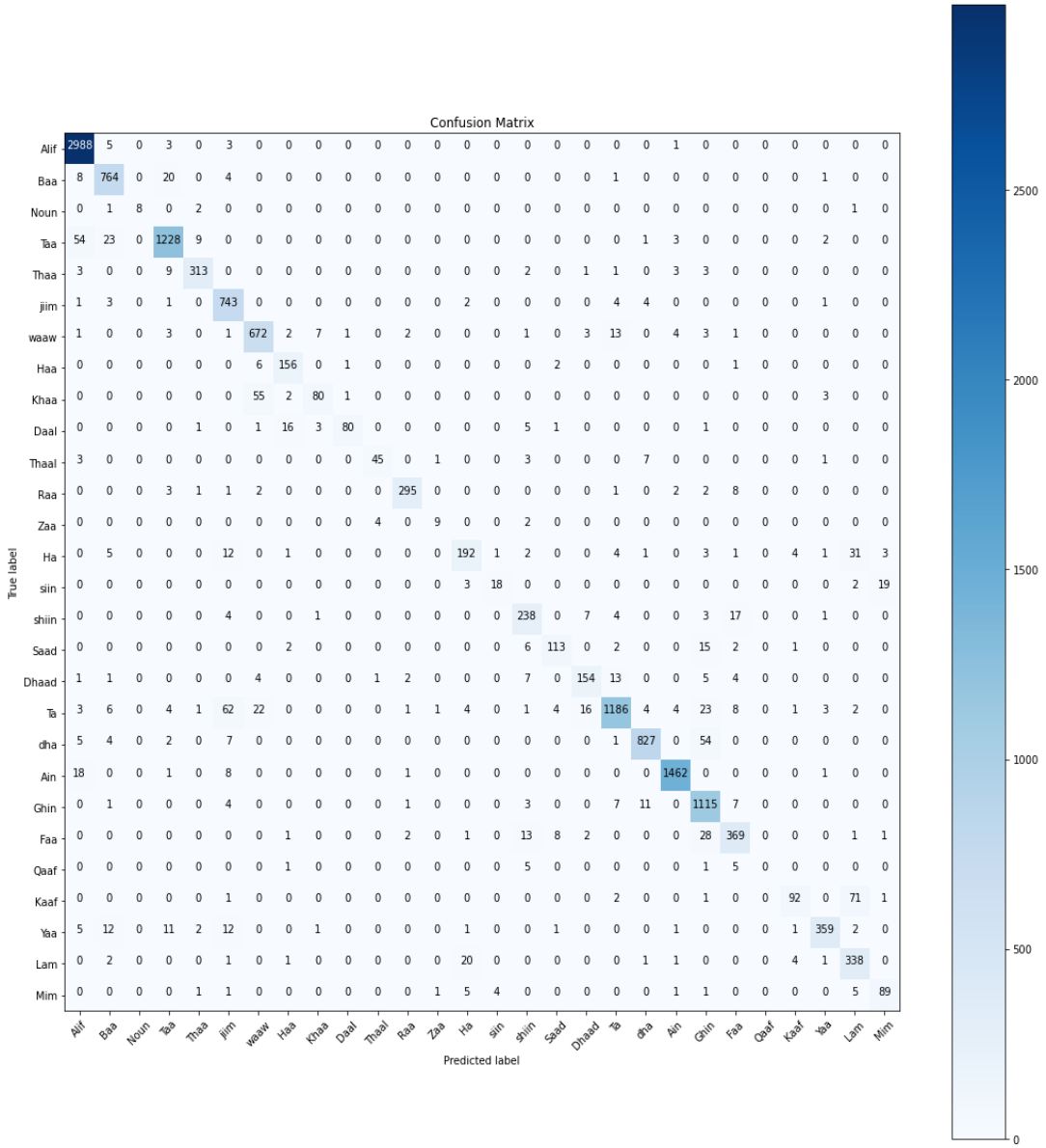

Figure III.14 Matrice de confusion pour le modèle à 2 couches (20époques).

La figure III.14, Illustre la position des métriques pour chaque classe. A titre d'exemple le modèle a bien classé les caractères alif, Taa, TA, Ain et a mal classé Qaff, Noun et Zaa, mais on remarque une légère différence pour ces deux dernier on les classant un peu mieux que pour le premier modèle à 10 époques. Pour le reste ce n'est pas une question de bien ou mal classé mais du nombre de caractère existant dans chaque classe

On remarque aussi qu'il y a une meilleure reconnaissance de ces caractères par rapport au modèle à 10 époques.

- **Nombre d'époques = 30**

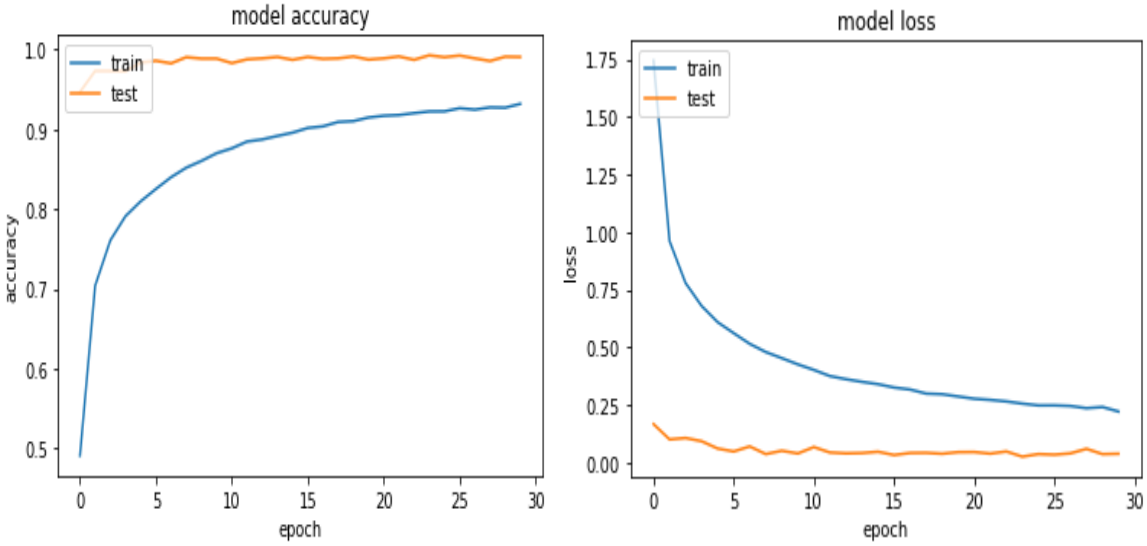

Figure III.15 Précision et erreur du modèle à 2 couches (30 époques)

D'après la figure III.15, on remarque que la précision de l'apprentissage et du test dans le model accuracy augmente avec le nombre d'époques. Ainsi, on voit que pour 30 époques le modèle à reconnu les caractères d'une manière plus précise à (90%) dès les 5 première époques, dans l'intervalle [10-30] époques, la précision d'apprentissage (train) ne cesse d'augmenter et la validation (test) se stabilise au niveau de 0.94

De même, l'erreur d'apprentissage et du test diminue avec le nombre d'époques et on remarque que les pertes ne sont pas conséquentes dès les premières époques pour le test jusqu'à stabilisation.

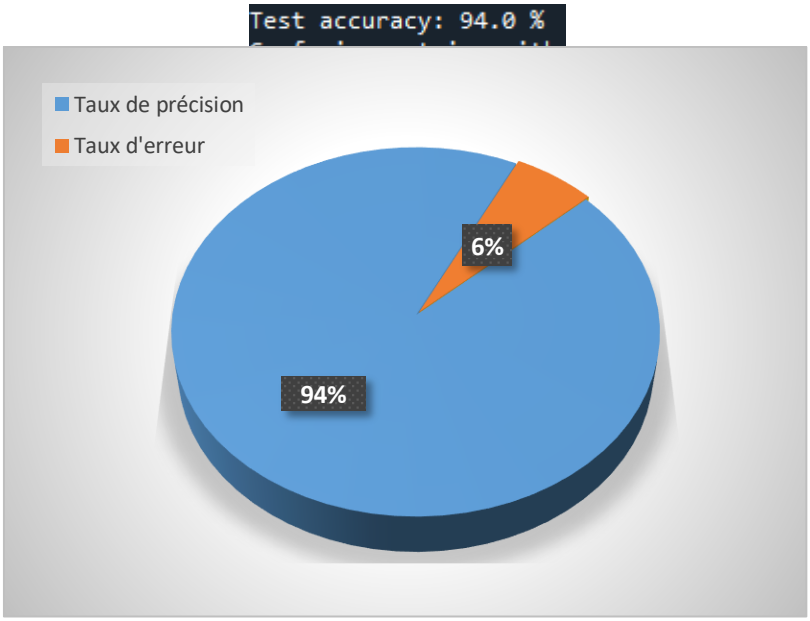

Figure III.16 Taux de précision et d'erreur du modèle à 2 couches (30 époques)

La figure III.16, illustre un taux de précision de 93% qui correspond à un total de 14138 caractères bien reconnus, et un taux d'erreur de 7% qui correspond à un total de 902 caractères mal reconnus.

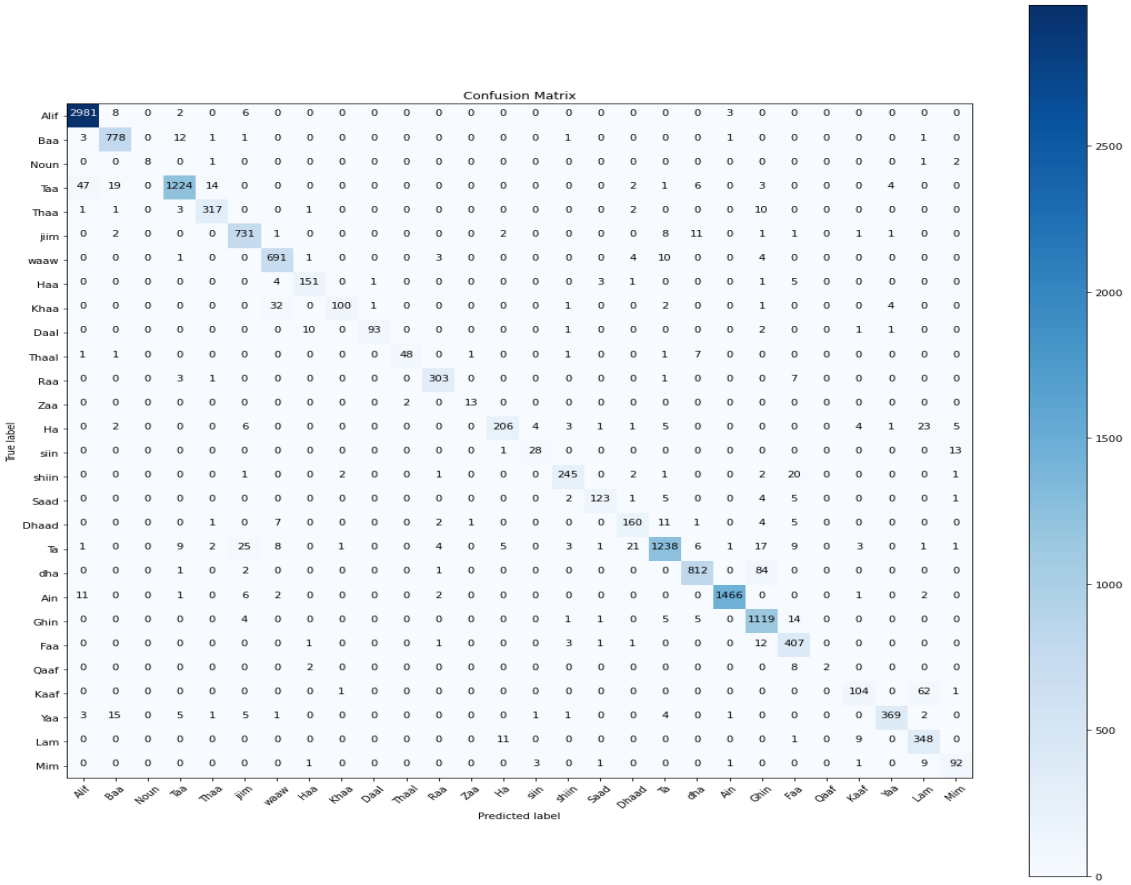

Figure III.17 Matrice de confusion pour le modèle à 2 couches (30 époques)

La figure III.17, illustre la position des métriques pour chaque classe. A titre d'exemple le modèle a bien classé les caractères alif, Taa, TA, Ain, Ghiin et a mal classé Qaff, daal et Zaa. Pour le reste ce n'est pas une question de bien ou mal classé mais du nombre de caractère existant dans chaque classe.

On remarque aussi qu'il y a une meilleure reconnaissance de ces caractères par rapport aux modèles à 10 et 20 époques.

III.8.2 Résultats obtenus pour le modèle à 4 couches

- **Nombre d'époques = 10**

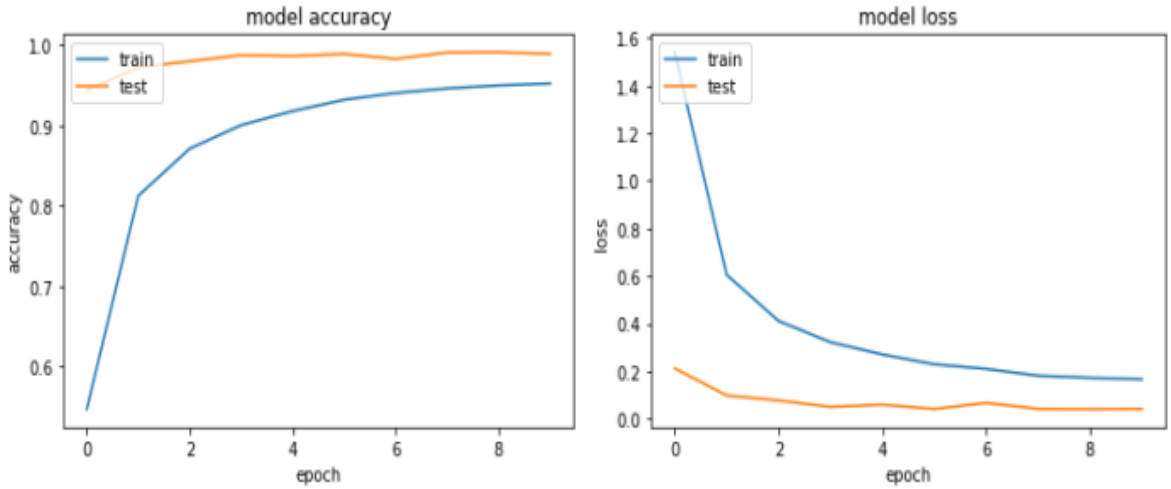

Figure III.18 Précision et erreur du modèle à 4 couches (10 époques)

D'après la figure III.18, la précision d'apprentissage (train) et celle du test augmente avec le nombre d'époque atteignant les 0,946

Contrairement à l'erreur qui diminue avec l'augmentation du nombre d'époques.

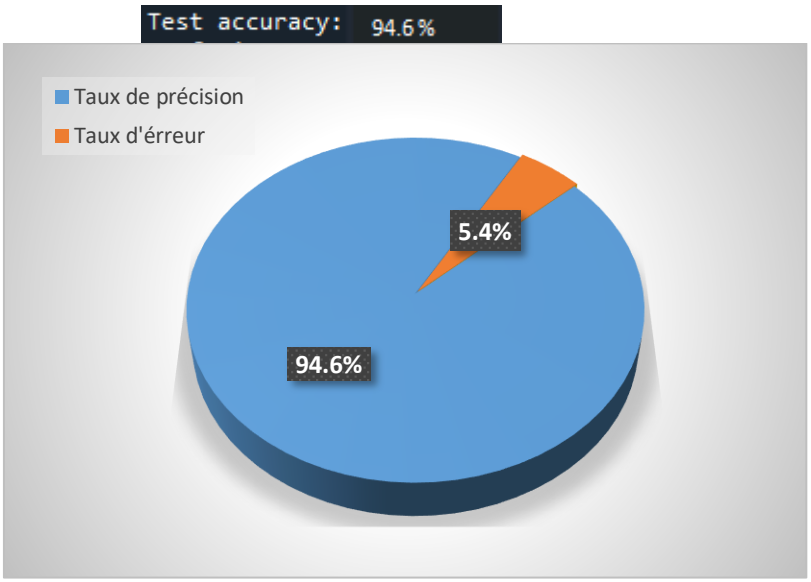

Figure III.19 Taux de précision et d'erreur du modèle à 4 couches (10 époques)

La figure III.19, illustre un taux de précision de 94.6% qui correspond à un total de 14229 caractères bien reconnus, et un taux d'erreur de 5.4% qui correspond à un total de 812 caractères mal reconnus.

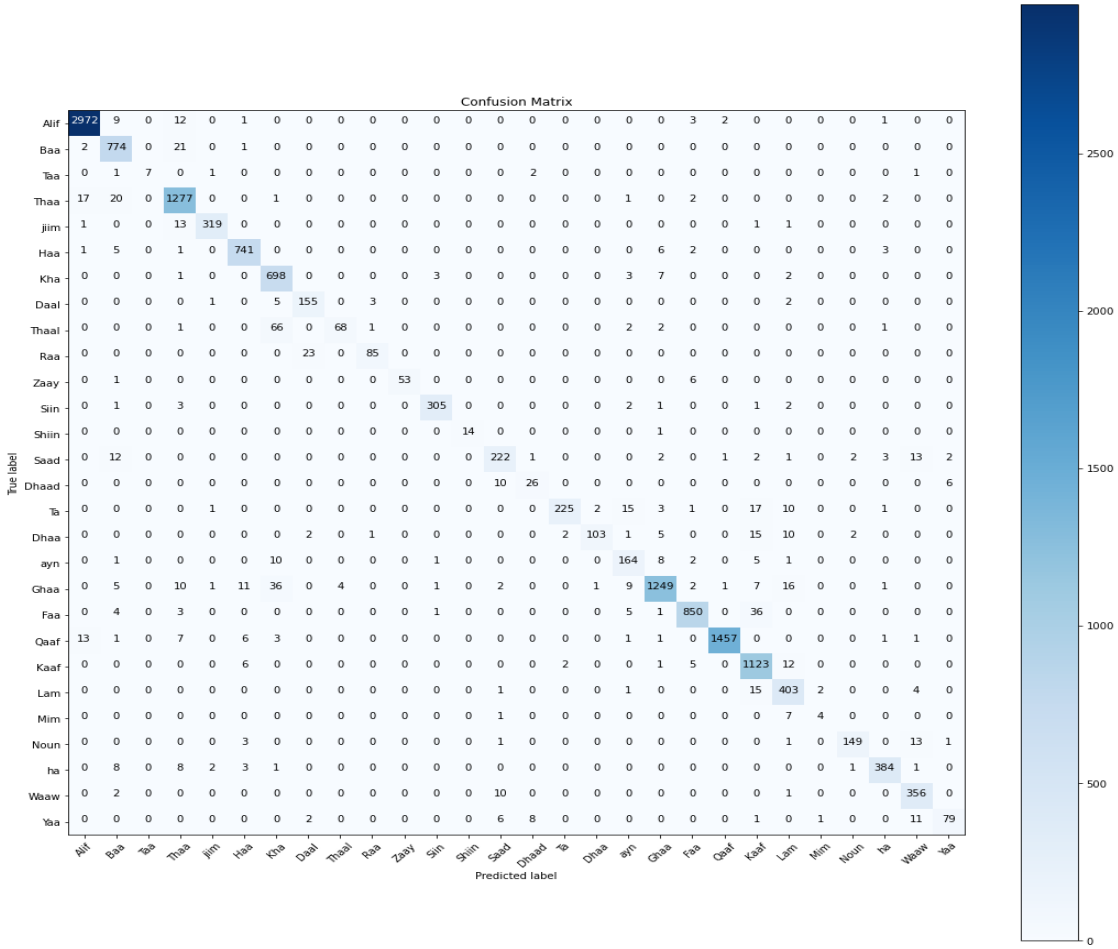

Figure III.20 Matrice de confusion pour le modèle à 4 couches (10 époques)

La figure III.20 illustre la position des métriques pour chaque classe. A titre d'exemple le modèle a bien classé les caractères alif, Taa, TA, Ain, Ghiin et a mal classé Qaff, daal et Zaa. Pour le reste ce n'est pas une question de bien ou mal classé mais du nombre de caractère existant dans chaque classe.

On remarque aussi, que tous les caractères sont mieux classés par rapport au modèle à 2 couches

- **Nombre d'époques = 20**

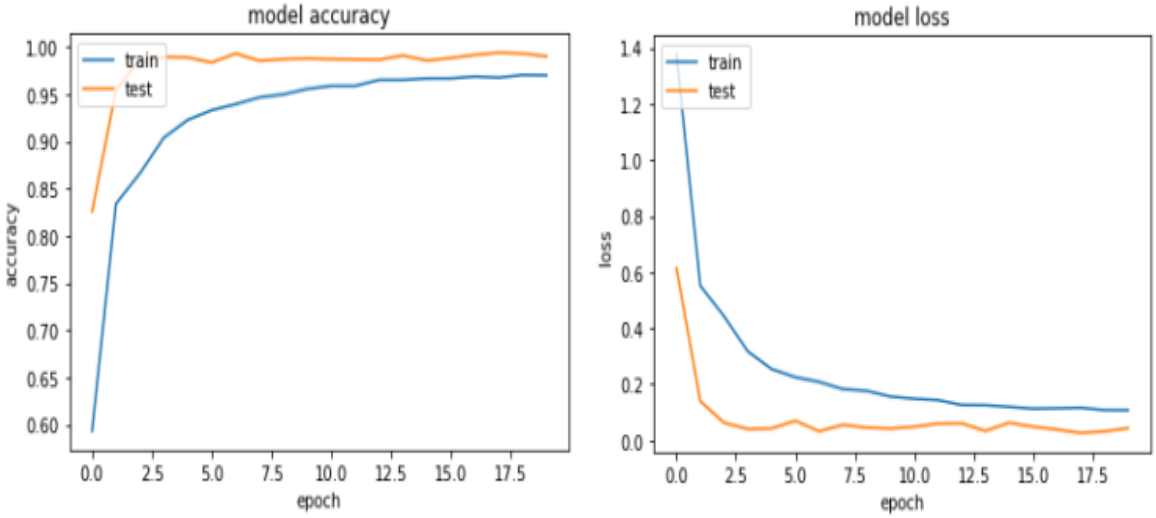

Figure III.21 Précision et erreur du modèle à 4 couches (20 époques)

D'après la figure III.21, on remarque que la précision du train ne cesse d'augmenter, tous comme celle du test qui à certaines époques à diminuer légèrement dans un laps de temps, pour ensuite se stabiliser en atteignant son summum.

En ce qui concerne l'erreur, elle est inversement proportionnelle à la précision, c'est-à-dire, elle diminue quand la précision augmente.

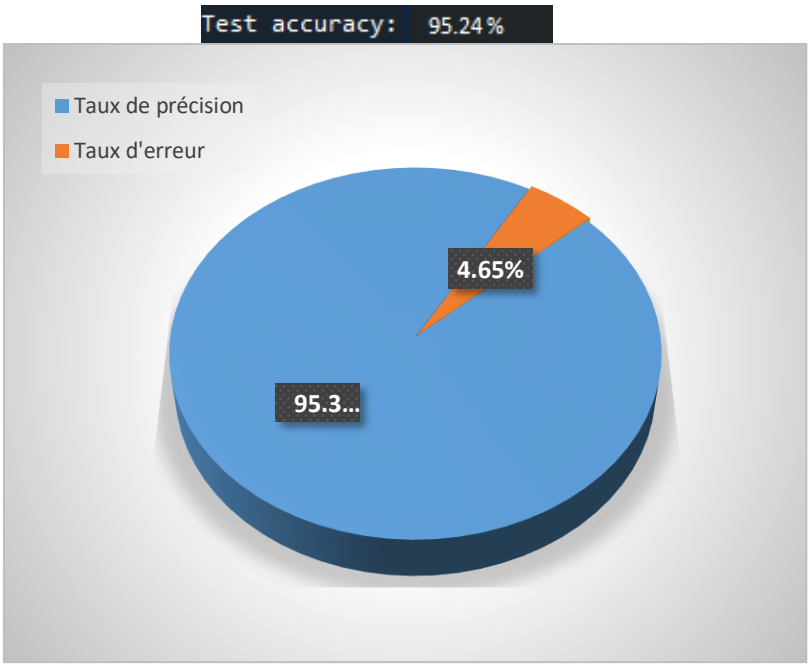

Figure III.22 Taux de précision et d'erreur du modèle à 4 couches (20 époques)

La figure III.22, illustre un taux de précision de 95.35% qui correspond à un total de 14342 caractères bien reconnus, et un taux d'erreur de 4.65% qui correspond à un total de 699 caractères mal reconnus.

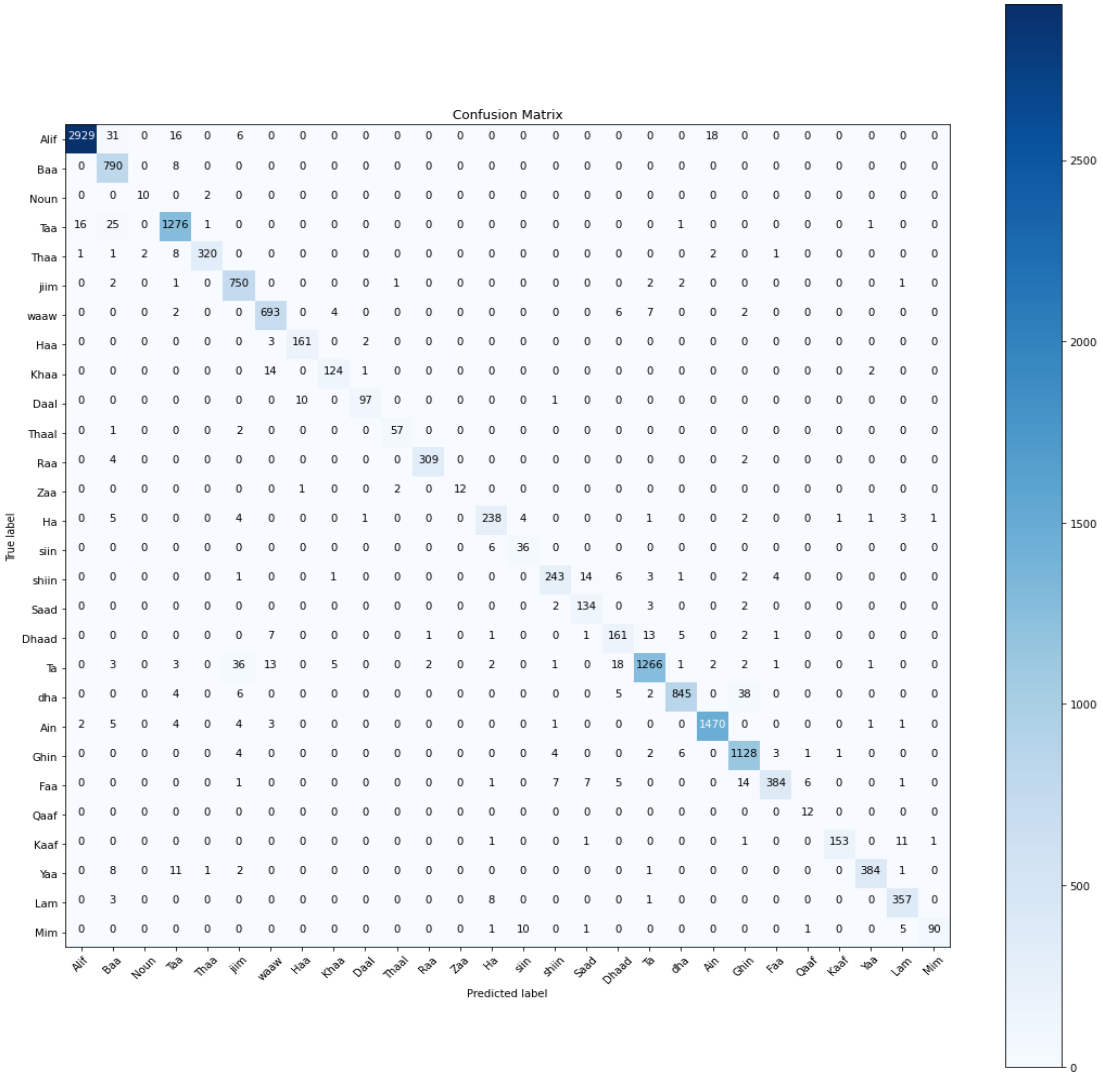

Figure III.23 Matrice de confusion pour le modèle à 4 couches (20 époques)

La figure III.23, illustre la position des métriques pour chaque classe. On remarque que le modèle a bien classé les caractères alif, Taa, TA, Ain, Ghaa, et a mal classé Qaff et Zaa. Pour le reste ce n'est pas une question de bien ou mal classé mais du nombre de caractère existant dans chaque classe.

On remarque aussi, que tous les caractères sont mieux classés par rapport au modèle à 2 couches, ainsi que pour un nombre d'époque inférieur.

- **Nombre d'époques = 30**

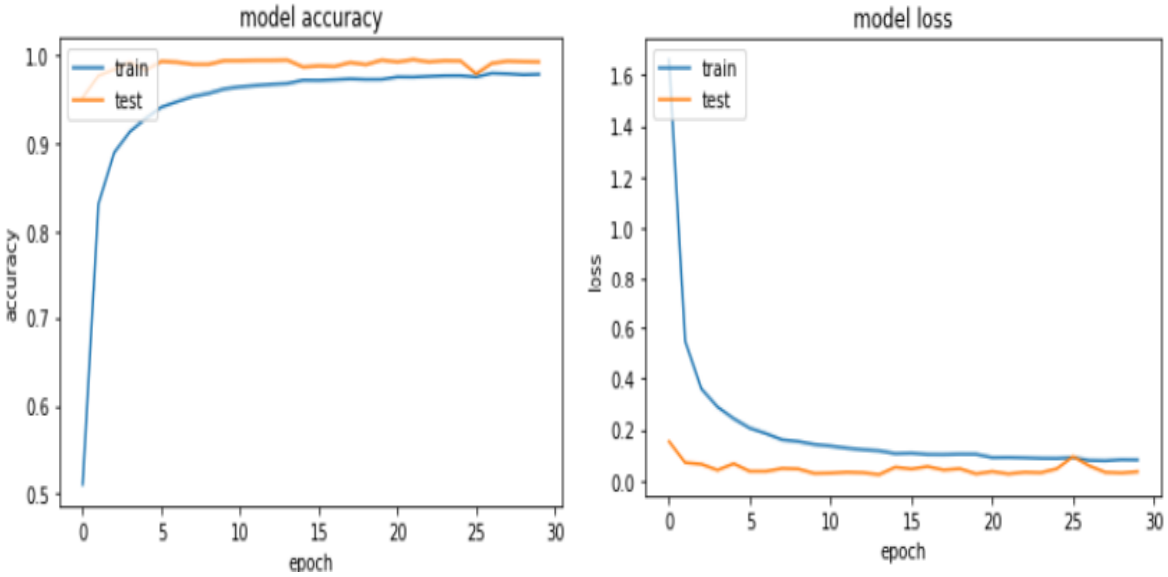

Figure III.24 Précision et erreur du modèle à 4 couches (30 époques)

D'après la figure III.24, la précision du train augmente de manière significative durant les 30 époques, tout comme celle du test qui commence direct par une précision qui dépasse les 0.9 (90%) pour ensuite atteindre 0.97 sur l'intervalle [3-13] époques, diminuer de 0.1 sur l'intervalle [14-24] époques, et rechuter d'encore 0.1 sur [24-25] époques, pour enfin augmenter et se stabiliser à 0.96.

Pour le modèle d'erreur, c'est tous le contraire de la précision, quand l'un augmente l'autre diminue et vice- versa.

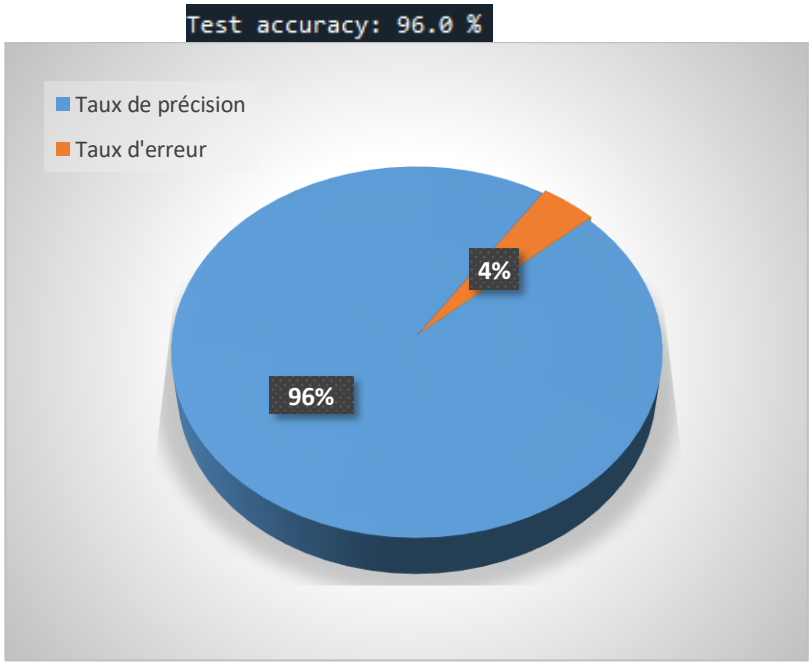

Figure III.25 Taux de précision et d'erreur du modèle à 4 couches (30 époques)

La figure III.25, illustre un taux de précision de 96% qui correspond à un total de 14439 caractères bien reconnus, et un taux d'erreur de 4% qui correspond à un total de 602 caractères mal reconnus

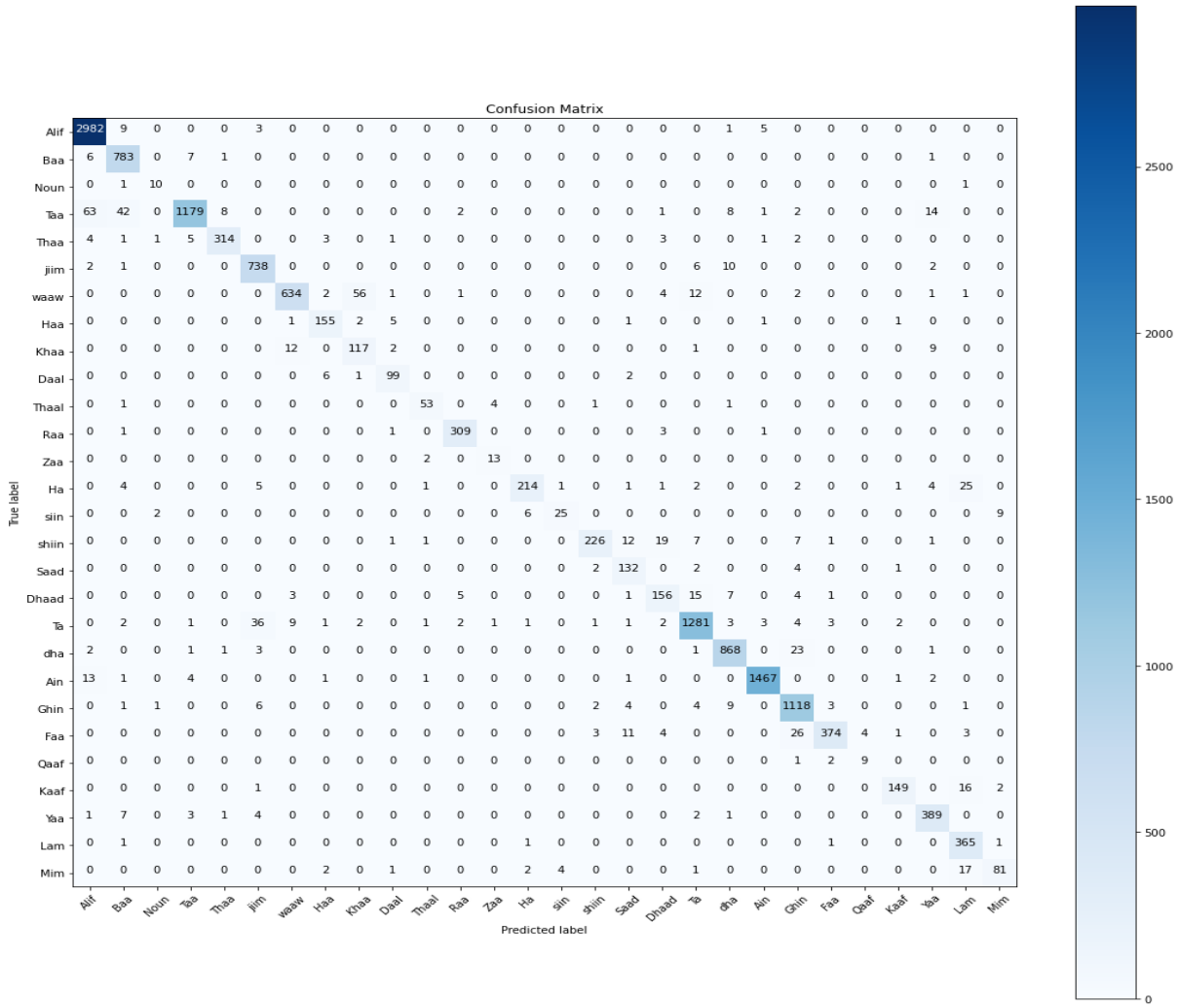

Figure III.26 Matrice de confusion pour le modèle à 4 couches (30 époques)

La figure III.26, illustre la position des métriques pour chaque classe. On remarque que le modèle a bien classé les caractères alif, TA, Ain, Ghaa, et a mal classé Qaff. Pour le reste ce n'est pas une question de bien ou mal classé mais du nombre de caractère existant dans chaque classe.

On remarque aussi, que tous les caractères sont mieux classés par rapport au modèle à 2 couches, ainsi que pour 20 et 10 époques.

III.8.3 Résultats obtenus pour le modèle à 6 couches

- **Nombre d'époques = 10**

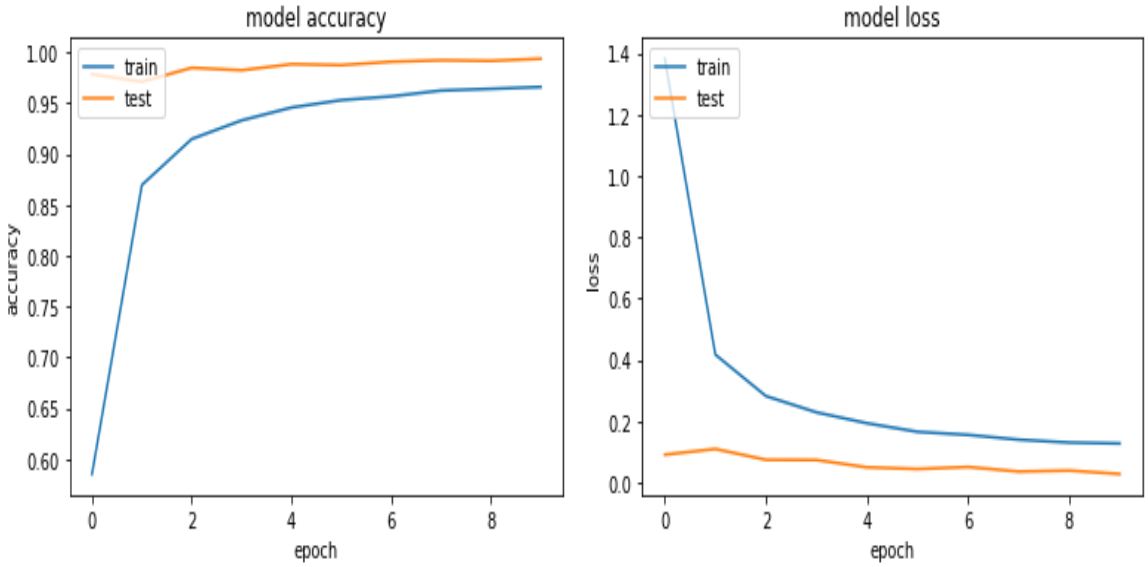

Figure III.27 Précision et erreur du modèle à 6 couches (10 époques)

D'après la figure III.27, la précision du train augmente de manière significative durant les 30 époques, tout comme celle du test qui commence direct par une précision qui dépasse les 0.95 pour ensuite diminuer à 0.93 sur l'intervalle [1-] époques, pour enfin augmenter et se stabiliser à 0.96.

Pour le modèle d'erreur, c'est tous le contraire de la précision, quand l'un augmente l'autre diminue et vice- versa

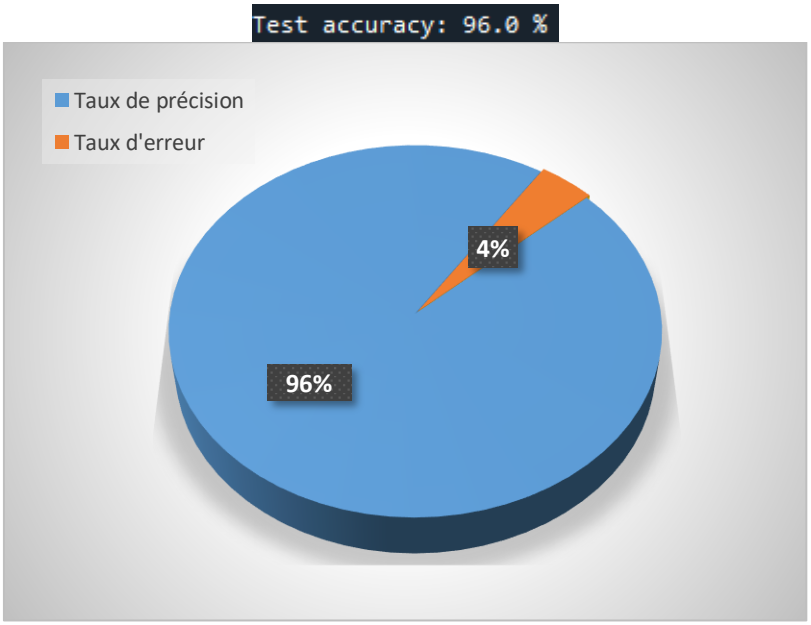

Figure III.28 Taux de précision et d'erreur du modèle à 6 couches (10 époques)

La figure III.28, illustre un taux de précision de 96% qui correspond à un total de 14439 caractères bien reconnus, et un taux d'erreur de 4% qui correspond à un total de 602 caractères mal reconnus

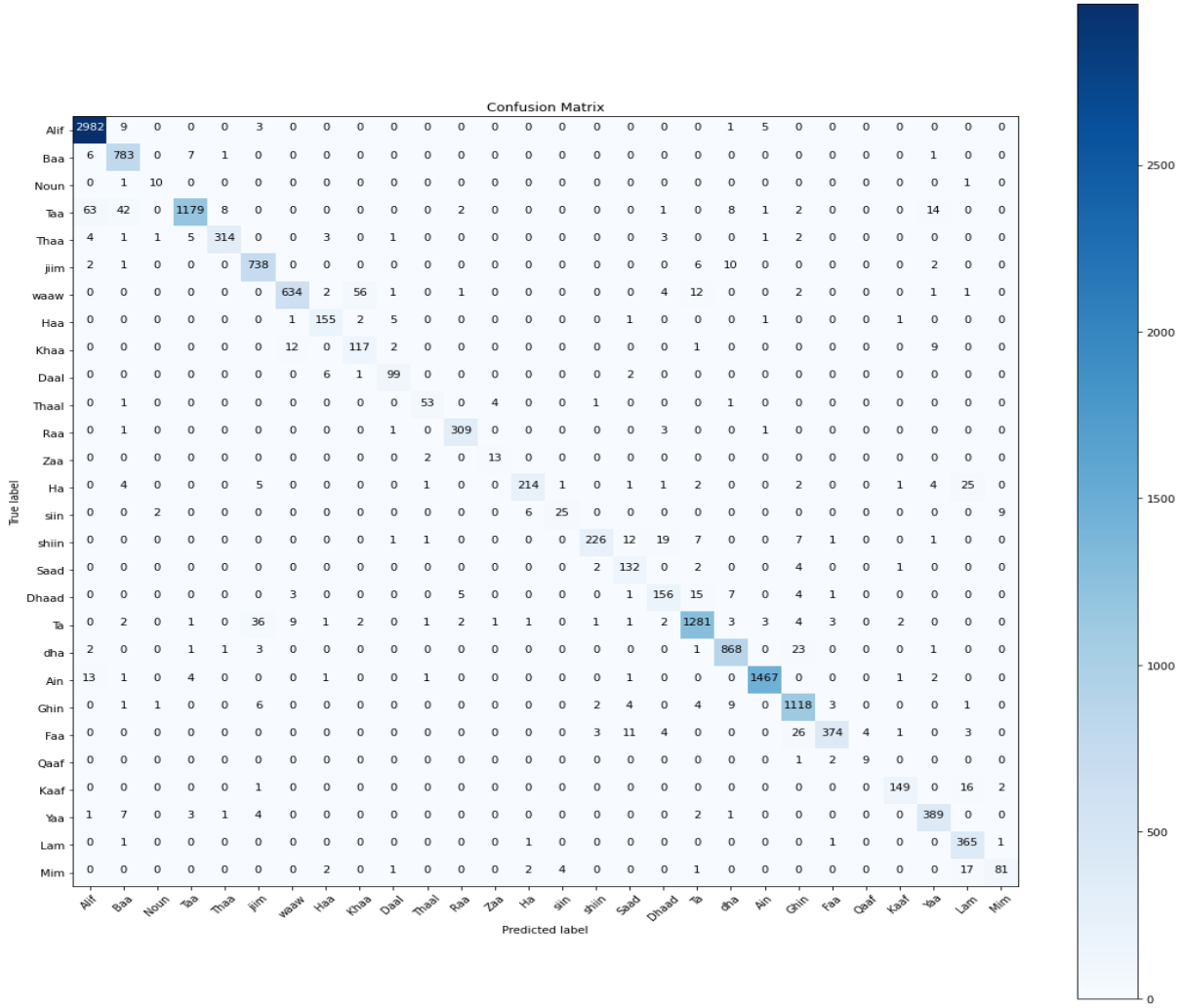

Figure III.29 Matrice de confusion pour le modèle à 6 couches (10 époques)

La figure III.29, illustre la position des métriques pour chaque classe. On remarque que le modèle a bien classé les caractères alif, Baa, Taa, TA, Ain, Ghin, et a mal classé Qaff et Noun. Pour le reste ce n'est pas une question de bien ou mal classé mais du nombre de caractères existant dans chaque classe.

On remarque aussi, que tous les caractères sont mieux classés par rapport au modèle à 2 et 4 couches.

- **Nombre d'époques = 20**

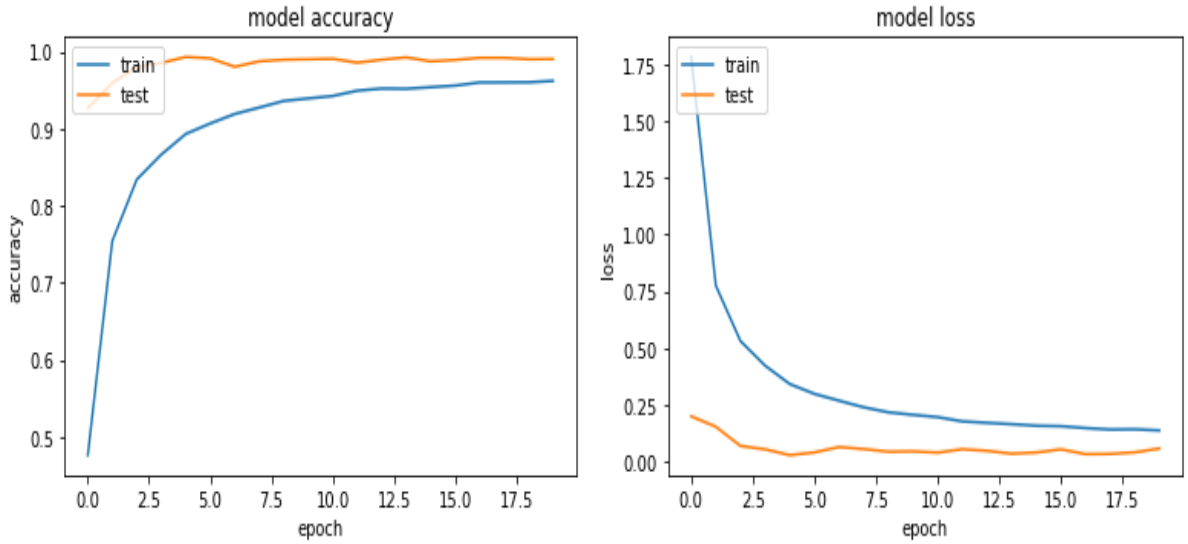

Figure III.30 Précision et erreur du modèle à 6 couches (20 époques)

D'après la figure III.30, on remarque que la précision du train ne cesse d'augmenter, tous comme celle du test qui atteint les 0.97 sur l'intervalle [2-4] époques, puis à certaines époques diminue légèrement dans un laps de temps, pour ensuite se stabiliser en atteignant les 0.9654.

En ce qui concerne l'erreur, elle est inversement proportionnelle à la précision, c'est-à-dire, elle diminue quand la précision augmente

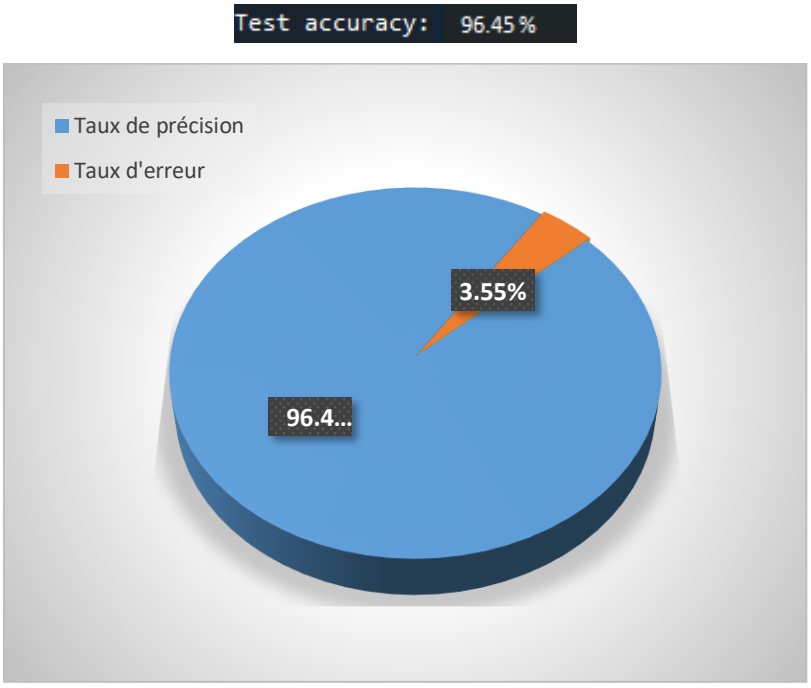

Figure III.31 Taux de précision et d'erreur du modèle à 6 couches (20 époques)

La figure III.31, illustre un taux de précision de 96.45% qui correspond à un total de 14507 caractères bien reconnus, et un taux d'erreur de 3.55% qui correspond à un total de 534 caractères mal reconnus

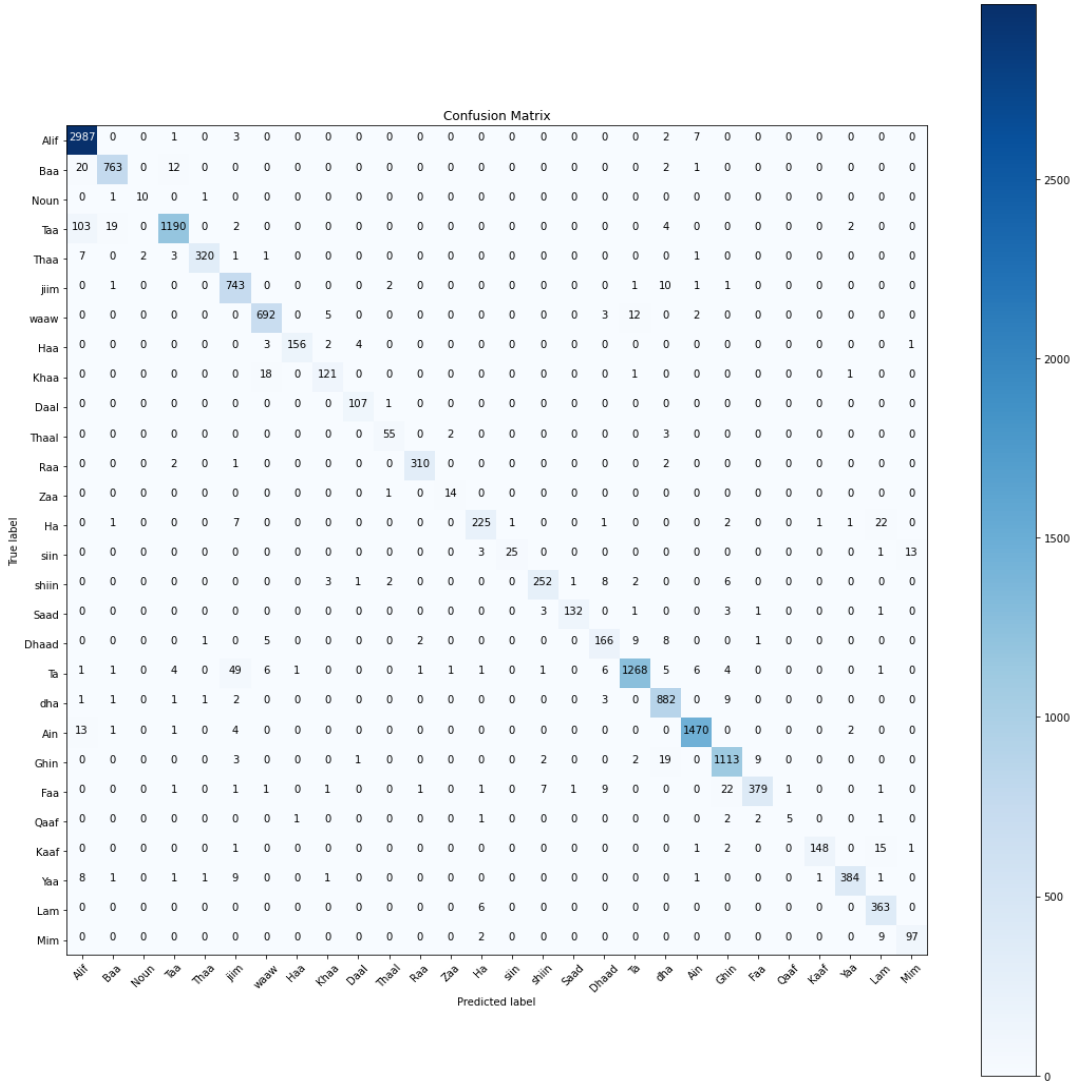

Figure III.32 Matrice de confusion pour le modèle à 6 couches (20 époques)

La figure III.32 illustre la position des métriques pour chaque classe. On remarque que le modèle a bien classé les caractères alif, Baa, Taa, TA, Ain, Ghin, et a mal classé Qaff et Noun. Pour le reste ce n'est pas une question de bien ou mal classé mais du nombre de caractères existant dans chaque classe.

On remarque aussi, que tous les caractères sont mieux classés par rapport au modèle à 2 et 4 couches, ainsi que pour 10 époques

- **Nombre d'époques= 30**

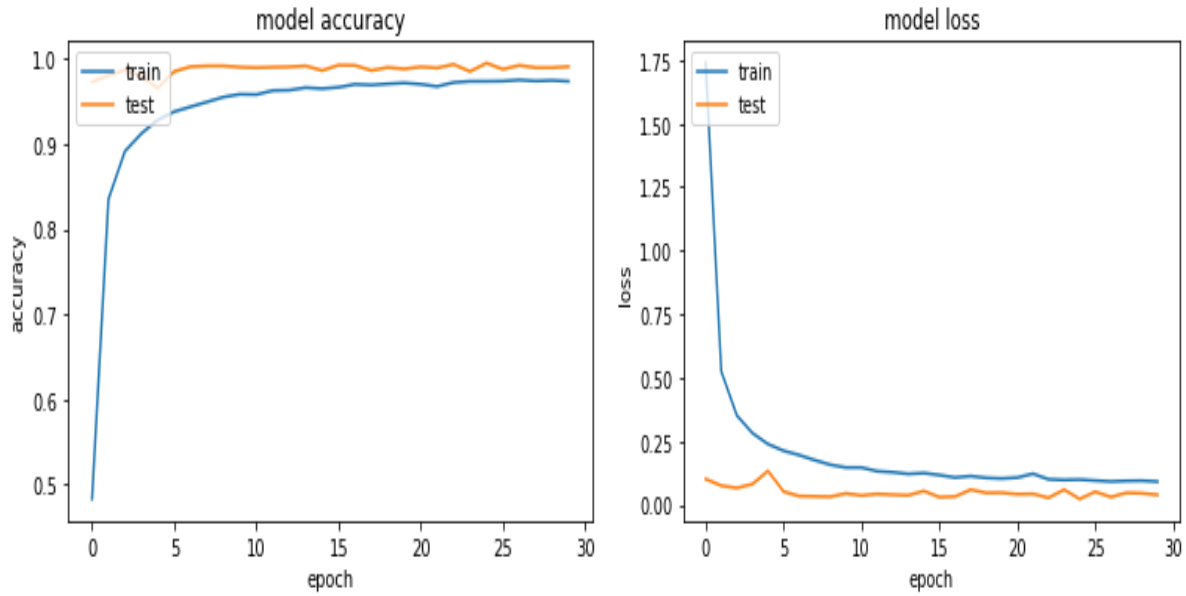

Figure III.33 Précision et erreur du modèle à 6 couches (30 époques)

D'après la figure III.33, la précision de train ne cesse d'augmenter jusqu'à la dernière époque, pareillement pour le test qui commence quant à lui avec une précision de 0.95, pour perdre 0.1 sur l'intervalle [4-5] époques et remonter petit à petit jusqu'à stabilisation sur 0.97. En ce qui concerne l'erreur, elle est inversement proportionnelle à la précision, c'est-à-dire, elle diminue quand la précision augmente

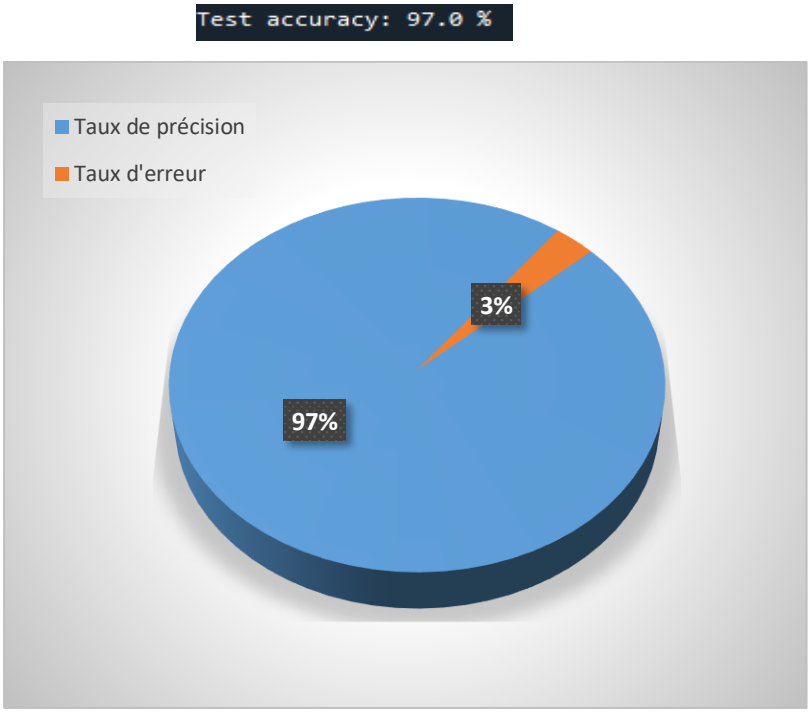

Figure III.34 Taux de précision et d'erreur du modèle à 6 couches (30 époques)

La figure III.34, illustre un taux de précision de 97% qui correspond à un total de 14590 caractères bien reconnus, et un taux d'erreur de 3% qui correspond à un total de 451 caractères mal reconnus

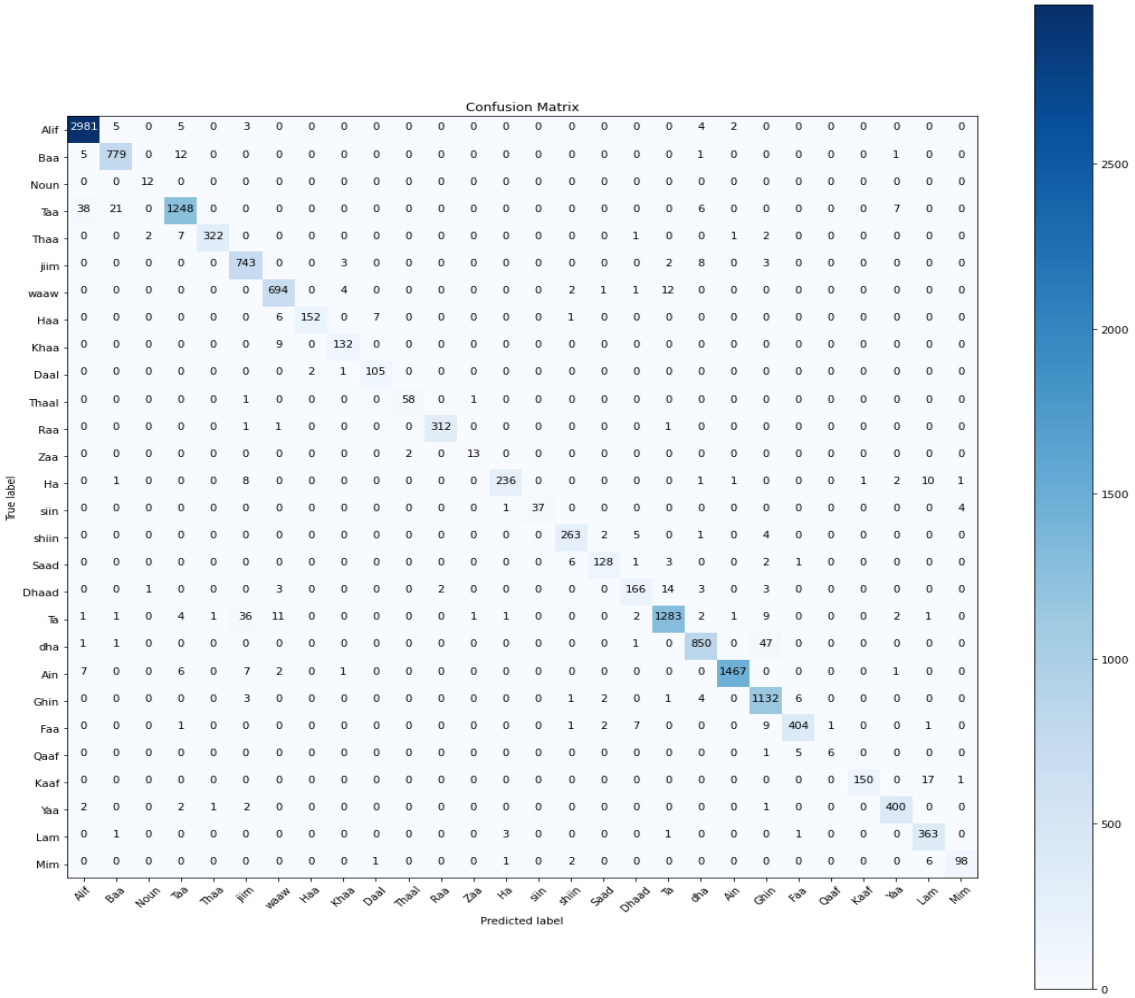

Figure III.35 Matrice de confusion du modèle à 6 couches (30 époques)

La figure III.35, illustre la position des métriques pour chaque classe. On remarque que le modèle a bien classé les caractères alif, Baa,Taa, TA, Ain, Ghin, et a mal classé Qaff. Pour le reste ce n'est pas une question de bien ou mal classé mais du nombre de caractères existant dans chaque classe.

On remarque aussi, que tous les caractères sont mieux classés par rapport au modèle à 2 et 4 couches, ainsi que pour 20 et 10 époques

	Architecture utilisée			Nombre d'époques	Précision	Erreur
	Couche de convolution	Couche de pooling	Couche de fully connected			
Modèle 1 (2couches)	2	2	2	10	90%	10%
				20	93%	7%
				30	94%	6%
Modèle 2 (4couches)	4	2	3	10	94.6%	5.4%
				20	95.24%	4.76%
				30	96%	4%
Modèle 3 (6couche)	6	3	3	10	96%	4%
				20	96.45%	3.55%
				30	97%	3%

Tableau III.1 Comparaison entre les différentes architectures et époques (IFHCDB)

Le tableau III.1, montre l'architecture utilisée dans chaque modèle ainsi que le nombre d'époques utilisées. Les résultats obtenus sont exprimés en termes de précision et d'erreur

Pour les trois modèles on remarque qu'à chaque fois qu'on augmente le nombre d'époques, le taux de précision augmente et le taux d'erreur diminue.

Si on compare entre nos trois modèles, on voit que le taux de précision augmente de façon très significative, prenons pour exemple 30 époques de ces trois derniers on à 93% pour le modèle à 2 couches, 96% pour le modèle à 4 couches et enfin 97% pour notre modèle à 6couches

On constate ainsi que plus l'architecture est profonde plus les résultats sont meilleurs par exemple dans notre tableau l'ajout de 4 couches au premier modèle à permis d'augmenter la précision de 7%, ce qui est énorme.

III.9 Résultats et discussion pour la base mnist

III.9.1 Résultats obtenus pour le modèle à 4 couches

- **Nombre d'époques = 30**

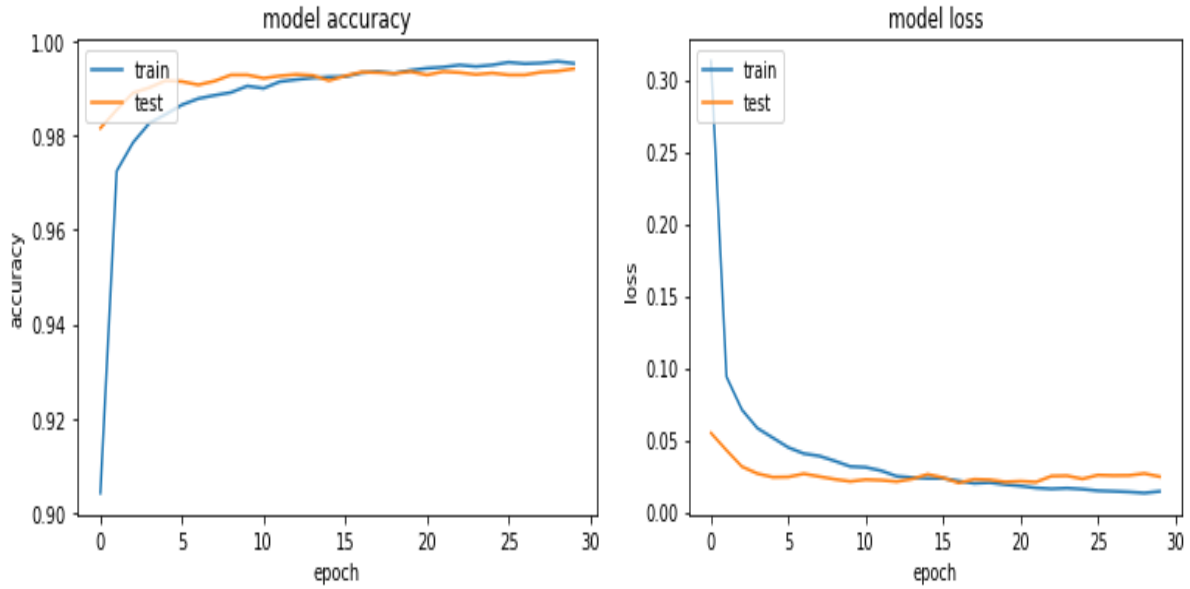

Figure III.36 Précision et erreur du modèle à 4 couches (30 époques)

D'après la figure III.36, on remarque que la précision du train ne cesse d'augmenter, tous comme celle du test qui atteint les 0.98 dès la première époque, mais sur l'intervalle [18-30] époques, on voit que la précision de l'apprentissage augmente et le test reste stable.

En ce qui concerne l'erreur, elle est inversement proportionnelle à la précision, c'est-à-dire, elle diminue quand la précision augmente

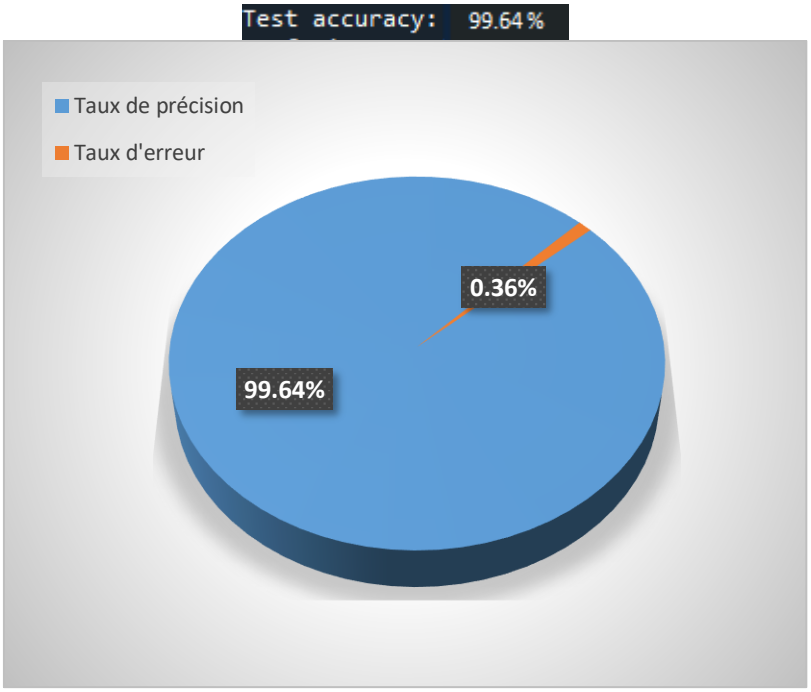

Figure III.37 Taux de précision et d'erreur du modèle à 4 couches (30 époques)

La figure III.37 Illustre un taux de précision de 99.64% qui correspond à un total de 9960 caractères bien reconnus, et un taux d'erreur de 0.36% qui correspond à un total de 36 caractères mal reconnus

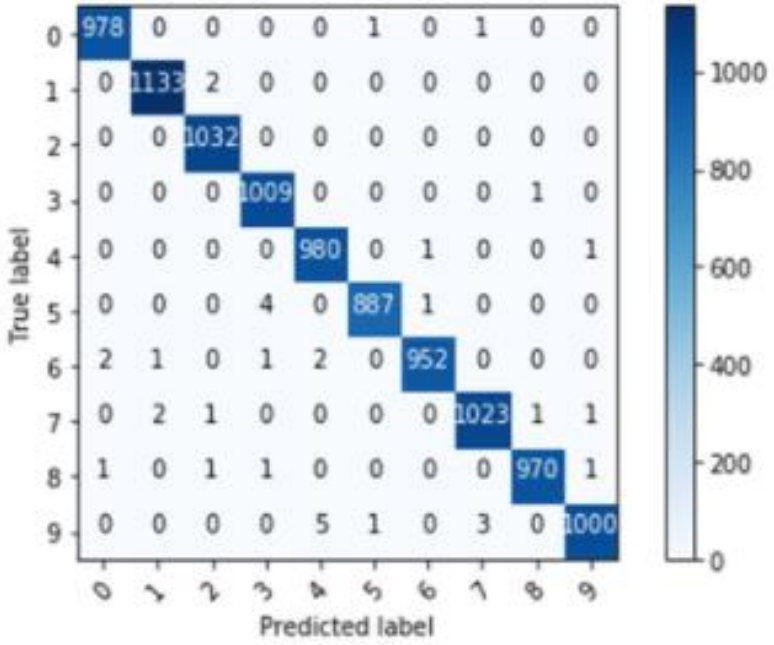

Figure III.38 Matrice de confusion du modèle à 6 couches (30 époques)

La figure III.38, illustre la position des métriques pour chaque classe. On remarque que le modèle a bien classé tous les caractères, On remarque aussi, que tous les caractères

sont mieux classés par rapport au modèle à couches inférieurs, et aussi pour des époques moindres, malgré la faible différence entres-elles.

	Architecture utilisée			Nombre d'époques	Précision	Erreur
	Couche de convolution	Couche de pooling	Couche de fully connected			
Modèle 1 (2couches)	2	2	2	10	98.7%	1.3%
				20	99.18%	0.82%
				30	99.34%	0.66%
Modèle 2 (4couches)	4	2	2	10	99.41%	0.59%
				20	99.5%	0.5%
				30	99.64%	0.63%

Tableau III.2 Comparaison entre les différentes architectures et époques (MNIST)

Le tableau III.2 illustre l'architecture utilisée dans chaque modèle ainsi que le nombre d'époques, utilisant MNIST comme base de données. Les résultats obtenus sont exprimés en termes de précision et d'erreur

Pour les deux modèles on remarque qu'à chaque fois qu'on augmente le nombre d'époques, le taux de précision augmente et le taux d'erreur diminue.

On constate ainsi que plus l'architecture est profonde plus les résultats sont meilleurs par exemple dans notre tableau l'ajout de 2 couches au premier modèle a permis d'avoir des résultats presque parfait.

III.10 Conclusion

Nous avons présenté dans ce chapitre une approche de classification basée sur les réseaux neurones convolutionnels, pour cela, on a utilisé deux base de donnée, avec différentes architectures et nombre d'époques puis on a interprété les différents résultats obtenus en termes de précision et d'erreur.

Les résultats obtenus sont très encourageant avec un taux d'exactitude de classification de 97% pour la reconnaissance des lettres arabe manuscrite malgré la complexité de chaque caractère, et un taux de 99.64% pour la reconnaissance des chiffres manuscrits.

Conclusion générale

Au cours de ce mémoire, nous avons rappelé les différentes techniques de reconnaissance de l'écriture manuscrite. On a pu constater que cette dernière a connu des progrès très importants durant ces dernières années, permettant désormais de faire face à la variabilité de l'écriture, ainsi de reconnaître de manière satisfaisante des entités manuscrites : lettres et chiffres.
Dans ce travail, nous nous sommes focalisé sur la reconnaissance des lettres arabes manuscrites et chiffre manuscrit par les réseaux de neurones convolutifs.

Nous avons discuté des notions fondamentales sur le Deep Learning, les réseaux de neurones en générale et les réseaux de neurones convolutionnels en particulier, qu'on a introduit en présentant les différents types de couches utilisées dans la classification : la couche convolutionnelle, la couche de rectification, la couche de pooling et la couche fully connected. Nous avons parlé aussi sur les méthodes de régularisation (dropout) utilisées pour éviter le problème de sur apprentissage.

On a implémenté trois modèles de réseaux neurones convolutifs avec différentes architectures pour la base IFHCDB ; et deux modèles pour la base MNIST pour ensuite appliqué pour chacun de ces modèles des tests qui consistent à changer le nombre d'époques, et afficher à la fin la matrice de confusion pour récupérer le résultat des images bien et mal classées.
L'implémentation a été faite avec le langage de programmation python, on a utilisé des bibliothèques pour faciliter la tâche de création de nos modèles et pour l'accélération du training

Enfin on a terminé avec deux tableaux récapitulatifs et comparatifs des résultats obtenus qui s'avèrent être très satisfaisants et encourageants.

Bibliographie

a.bibliographie

[1] N.Benamra : « utilisation des modéles de MarKov cachés planaires en reconnaissance de l'écriture arabe imprimé » thèse de doctorat, spécialité génie électrique, université des sciences, des techniques et de médecine de tunis II, 1999.

[2] B.Al-Badr, S.A.Mahmoud :«Survey and Bibliography pf arabic optical text recognition» Signal processing, vol41, pp.49-77 ,1995.

[3] G.Lorette, Y.Lecourtier, "Reconnaissance et Interprétation de Texts Manuscripts Hors-line: Un Problème d'Analyse de Scène", Bigre Num 80CNED Colloque National sur l'Ecrit et le Document, Nancy, CNED, Juillet 1992.

[4] N.Essoukri Ben Amara, Utilisation des Modèles de Markov Cachés Planaires en Reconnaissance de l'Ecriture Arabe Imprimé, Thèse de Doctorat, Ecole National d'Ingénieur de Tunisie, Février 1999.

[5] H. El-Abed and V. Margner, « Comparison of Different Preprocessing and Feature Extraction Methods for Offline Recognition of Handwritten Arabic Words », In Ninth International Conference on Document Analysis and Recognition, vol. 2, (2007), pp. 974-978.

[6] H. Al-Rashaideh, « Preprocessing Phase for Arabic Word Handwritten Recognition », Russian Academy of Sciences, Russian Federation, vol. 6, no. 1, (2006), pp. 11-19.

[7] S.Haithaamar : « segmentation de textes en caracteres pour la reconnaisance optique de l'écriture arabe », spécialité informatique industriel université de El.Hadj Lakhdhar Batna, (2006).pp.45-66

[8] M.Khoscheed : «off-line arabic character recognition areview»,patter analysis application,vol 5,(2002),pp.31-45 .

[9] N. Benahmed, 2002. : « Optimisation des réseaux de neurones pour la reconnaissance de chiffres manuscrits isolés : Sélection et pondération des primitives par algorithmes génétiques» Thèse de Master, École de technologie supérieure, MONTRÉAL.

[10] A. El-Yacoubi, R. Sabourin, M. Gilloux, and C.Y. Suen. Improved « model architecture and training phase in an off-line HMM-based word recognition system. In Proc. Of the 13th International Conference on Pattern Recognition, Brisbane, Australia, 1998

[11] P.M Lallican, C. Viarp-Gaudin and S. Knerr : « From off-line to on-line handwriting recognition ». Proc. 7th workshop on frontiers in handwriting recognition, Amsterdam, 2000.

[12] Salima Nabti : «Reconnaissance de caractères manuscrits par intelligence collective», thèse de doctorat université Ferhat Abass-Sétif.Algeria ,2013.

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

[14] R.BOUSLIMI : «Système de reconnaissance hors-ligne des mots manuscrits arabe pour multi-scripteurs ». Thèse de magister, FSJEGJ Jendouba ,2006

[15] S.P.CurramandJ.Mingers : «Neural Networks,Decision Tree Induction and Discriminant Analysis : an Empirical Comparison ». Journal of the Operational Research Society, 45(4) :440–450, Apr.1994

[16] D.Michie,D.J.Spiegel halter,C.C.Taylor,and J.Campbell,editors : « MachineLearning, Neural and Statistical Classification » Ellis Horwood, Upper Saddle River, NJ, USA, 1994

[17] ZACCONE Giancarlo, MD REZAUL Karim, MENSHAWY Ahmed : « Deep learning with tensorflow ». 2017

[18] K. Fukushima. Neocognitron : « A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position ». Biological Cybernetics, vol. 36, (1980) pp. 193–202.

[19] K. Fukushima : « A neural network model for selective attention in visual pattern recognition». Biological Cybernetics, vol., 55, (1986) pp.5–15.

[20] G.Dreyfur : « Réseaux de neurones : Méthodologie et application ». Édition Eyrolls, 2004

[21] DJOKHRAB Ala eddine : « Planification et optimisation de trajectoire d'un robot manipulateur à 6 ddl par des techniques neuro_floues ». 2015.

[22] TOUZET Claude : « Les réseaux de neurones artificiels, introduction à la connexion nisme » 1992.

[23] HUBEL David hunter, WIESEL Torsten nils : «Ferrier lecture-functional architecture of macaque monkey visual cortex ». 1977.

[24] WANG Peng, XU Jiaming, XU Bo, LIU Chenglin, ZHANG Heng, WANG Fangyuan HAO Hongwei : « Semantic clustering and convolutional neural networks for short text categorization ». 2015.

[25] O'SHEA Keiron, NASH Ryan : « An introduction to convolutional neural networks ». 2015.

[26] K.faez : « IFHCDB Release agreement ». Electrical Engineering Departement AmirKabir University of Technology, Hafez avenue tehran 15914, Iran.

b.Webgraphie

[27] https://keras.io/ Consultez le 24 /juillet /2021.

[Im1] https://www.ionos.fr/digitalguide/web-marketing/search-engine-marketing/deep-learning-vs-machine-learning/

[Im2] https://www.aquiladata.fr/insights/intelligence-artificielle-introduction-reseaux-de-neurones-13/

[Im3] https://www.becoz.org/these/memoirehtml/ch06s04.html

[Im4] https://penseeartificielle.fr/tp-reseau-de-neurones-convolutifs/

[Im5] https://datascientest.com/convolutional-neural-network

[Im6] https://www.natural-solutions.eu/blog/la-reconnaissance-dimage-avec-les-rseaux-de-neurones-convolutifs

[Im7] https://www.geeksforgeeks.org/cnn-introduction-to-pooling-layer/ (pour les pooling)

[Im8] http://newexplorers.fr/deep_learning

Résumé

Dans notre travail, nous utilisons le deep learning, plus précisément, les réseaux de neurones convoltionnel (CNN) pour la reconnaissance des caractères manuscrits arabes et chiffres, qui sont des réseaux neuronaux multicouches spécifiquement conçus pour les tâches de reconnaissance de formes. Un avantage majeur des réseaux convolutifs est l'utilisation de poids uniques associés aux signaux de tous les neurones entrant dans un seul noyau convolutif. Cette approche réduit l'empreinte mémoire, améliore les performances et permet l'invariance de la traduction dans le traitement par rapport à d'autres algorithmes de classification d'images, les réseaux de neurones convolutifs utilisent relativement peu de prétraitement. Ce projet consiste à utiliser et créer un classificateur de caractères avec trois modèles de réseaux neurones convolutionnels différents et à évaluer le meilleur d'entre eux en modifiant des hyper paramètres.

Les résultats obtenus ont montré que le choix du nombre d'époque, la taille de la base de données ainsi que la profondeur du réseau ont une grande influence pour avoir de meilleurs résultats.

Mot clé : Deep learning - réseaux de neurones convoltionnel (CNN – Caractères manuscrits

Abstract

In our work, we use deep learning, specifically the convolutional neural networks for the recognition of Arabic handwritten characters and digits, which are multi-layer neural networks specifically designed for pattern recognition tasks. A major advantage of convolutional networks is the use of unique weights associated with the signals of all neurons entering a single convolutional core. This approach reduces the memory footprint, improves performance and allows translation invariance in processing.

Compared to other image classification algorithms, convolutional neural networks use relatively little preprocessing. This project consists of using and creating a character classifier with three different convolutional neural network models and evaluating the best of them by modifying hyper parameters.

The results obtained showed that the choice of the number of epochs, the size of the database and the depth of the network have a great influence in obtaining better results.

Keyword : Deep learning - convolutional neural networks (CNN) – handwritten characters