République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieure et de la Recherche Scientifique

Université Abderrahmane Mira

Faculté de la Technologie

Département d'Automatique, Télécommunication et d'Electronique

Projet de Fin d'Etudes

Pour l'obtention du diplôme de Master

Filière : Electronique, Automatique

Spécialité : Instrumentation, Automatique et système

Thème

Réalisation d'un système de lecture automatique des caractères de Tamazight manuscrits par Deep Learning

Préparé par :

BALI Tafrara

SOUAGUI Siham

Dirigé par :

Dr. TIGHZERT Lyes

Examiné par :

M. M. KACIMI

M. A. DIBOUN

Année universitaire : **2021/2022**

Remerciement

Nous remercions le Dieux le plus puissant qui nous a fourni plusieurs personnes à qui nous voudrions témoigner toute nos reconnaissances, et de nous avoir permis de finaliser ce travail dans les meilleures conditions.

Nous voudrions tout d'abord adresser toute notre gratitude au promoteur de ce mémoire Mr Lyes Tighzert, pour sa patience, sa disponibilité, son soutien et surtout ses judicieux conseils, qui nous a dirigé et contribué à l'améliorer de notre réflexion.

Nous remercions vivement le président du jury de nous avoir fait l'honneur de présider le jury.

Nous tenons également à saluer d'avoir accepté d'évalué ce modeste travail.

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

Enfin nos profonds remercîments vont également à nos amis et à toutes personnes ayant contribué à ce modeste travail.

DEDICACE

Je dédie ce modeste travail

A mes chers parents

Djaafar BALI et Anissa BEN MESSAOUD

A mon cher frère

Jugurtha

A mes chères sœurs

Tanina, Tinhinan et Tania

A mes chers amis

Qui ont était toujours à mes côtés et ils n'ont jamais cessé de m'encouragé et de m'entourer de toute leur affection et leur amour.

Tafrara

DEDICACE

A la mémoire de mon défunt père, qui nous a quittés mais qui est toujours présent dans mon cœur, j'espère que je serai toujours à la hauteur de ses espérances.

Je dédie ce modeste travail à ma chère mère que je remercie infiniment pour leur encouragement et leur soutien, merci d'être là à mes côtés, que Dieu tout puissant vous garde et vous procure Santé, bonheur et longue vie.

A Mon chère frère : Mokran

A Ma chère sœur : Radia

A mes chères amies : Yamina, Amira, Tafrara, Nadjat

À tous ceux qui m'aiment… je les remercie tous.

Siham.

Sommaire

Chapitre III

Liste des figures

Liste des Algorithmes

Listings

Liste des Tableaux

L’introduction générale

Contexte :

La technologie est l'ensemble des connaissances et des techniques qui sont appliquées de manière ordonnée pour atteindre un certain objectif ou résoudre un problème, elle est une réponse au désir de l'homme de transformer l'environnement et d'améliorer sa qualité de vie.

L’intelligence artificielle est débutée en 1943 avec la publication de l’article (A logicial calculer of ideas immanent in nervous activity) par waren Mccuough et walter pitts, dans ce document les scientifiques présentent le premier modèle mathématique pour la création d’un réseau de neurones.

L’intelligence artificielle (IA) est une technologie si vaste et révolutionnaire qu’elle est difficile d’en donner une définition précise, on peut considérer qu’il s’agit d’une branche du domaine de l’automatique ayant pour but la création des machines capables d’effectuer des tâches nécessitant traditionnellement une intelligence humaine, le machine Learning (apprentissage automatique) est une catégorie d’IA et le Deep Learning est une technique du machine Learning. Elle est aujourd’hui utilisée dans les entreprises de toutes les industries.

Problématique :

De nos jours, l’automatisation des tâches complexes est de plus en plus souhaitée et encouragée par les scientifiques et les industriels. La problématique de la reconnaissance de l’écriture amazighe manuscrite est alors posée. Dans le domaine de la reconnaissance automatique des caractères, plusieurs recherches scientifiques ont été effectuées sur le caractère latin, arabe, et autre. Ceci a permis le développement de plusieurs approches de reconnaissance automatique de ces caractères et, par conséquent, reconnaître automatiquement des documents scannés. Par contre, les caractères tifinagh sont très peu traités. La recherche de solutions pour résoudre de telles problématique est fréquemment motivée par le fait que des méthodes d’opération sont inaccessibles pour un être humain, ennuyeuses, ou couteuses en temps. l’IA et le deep Learning sont parmi les outils les plus privilégiés pour résoudre ce type de problèmes. Dans notre document, nous allons traiter la résolution du problème de reconnaissances des caractères tifinagh manuscrits par Deep Learning.

L'objectif principal est de réaliser un système capable de recevoir, d'acquérir, une image en temps réel d'une caméra contenant un caractère manuscrit en tifinagh, de le transférer à un réseau de neurones profond et de le lire sur le haut-parleur (prononciation).

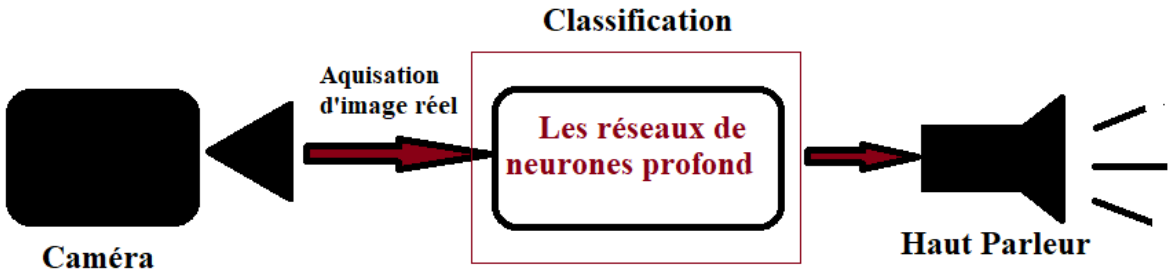


Schéma Synaptique

Présentation de nos approches :

La solution choisie pour résoudre la problématique est l'usage des réseaux de neurones de convolution qui sont l'élément clé en vision artificielle moderne. Afin d'y parvenir, nous faisons appel aux outils de développement softwars de matlab pour l'apprentissage du réseau de neurones convolutif. Pour une application pratique, qui fonctionne sur des réelles images, nous avons choisi de réaliser un système de lecture automatique des caractères de la langue berbère manuscrite par Deep Learning en combinant Matlab et Labview. Ce dernier, facile à programmer, permet de réaliser des interfaces graphiques. Nous l'avons alors utilisé pour créer notre application.

Organisation du mémoire

Ce mémoire est organisé et divisé en trois chapitres :

- Le premier chapitre est consacré pour la définition des différents domaines de l'intelligence artificielle (IA), Machine Learning (ML) et les différents types d'apprentissage et le Deep Learning (DL).
- Le deuxième chapitre est dédié aux réseaux de neurones convolutifs (ConvNet ou CNN), à apprentissage automatique d'un réseau de neurones, les différentes couches de réseau de CNN, et à la fin nous avons exploré les différentes architectures les plus populaires et les plus largement utilisées.

- Le troisième chapitre aborde l'implémentation du réseau de neurones entrainé sur la mise en œuvre pratique du système de reconnaissance en se basant sur Matlab et Labview. Ce travail abouti à la conception et l'implémentation d'un réseau de neurones convolutif qui reconnaitre les lettres manuscrites, et qui peut identifier et prononcer les lettres de tifinagh en temps réel.

I.1 Introduction

Un réseau neuronal est un ensemble de neurones formels interconnectés permettant la résolution de problèmes complexes tels que la reconnaissance des formes ou le traitement du langage naturel NPL, grâce à l'ajustement des coefficients de pondération dans une phase d'apprentissage. Essentiellement, ils résolvent les problèmes par essais et erreurs. Le nom de réseaux de neurones et leur structure sont inspirés par le cerveau humain. En effet, ces réseaux imitent la façon dont les neurones s'envoient mutuellement des signaux.

Dans ce chapitre nous allons présenter différents domaines de l'intelligence artificielle, ensuite nous allons passer au Machine Learning (ML), le Deep Learning (DL) et Démontrer une connaissance des réseaux de neurones et des réseaux de neurones profonds en particulier.

I.5 Vision artificielle (Computer Vision)

La vision artificielle est un domaine scientifique et une technique d'intelligence artificielle, permettant d'analyser des images captées par un équipement tel qu'une caméra. La vision artificielle se représente comme un outil basé sur l'IA capable de reconnaitre une image, de la comprendre, et de la traiter, la vision par ordinateur est l'équivalent en termes de l'IA des yeux humains et de la vision humaine de la capacité de notre cerveau à traiter et analyser les images perçues. La reproduction de la vision humaine par des ordinateurs constitue d'ailleurs l'un des grands objectifs de la vision par ordinateur.

Le Deep Learning est la technologie d'apprentissage automatique la plus prometteuse, le réseau peut traiter de grandes quantités d'information et appondre progressivement des images, du texte ou des données.

I.2 Intelligence artificielle

L'Intelligence Artificielle (IA) est la simulation de processus de l'intelligence humaine par des machines. Plus particulièrement, par des algorithmes. Ces processus comprennent trois phases. Tout d'abord, l'apprentissage, c'est-à-dire l'acquisition de l'information et ses règles d'utilisation. Puis, le raisonnement, soit l'utilisation des règles pour tirer des conclusions approximatives ou définitives. Et enfin, l'autocorrection. Les applications particulières de l'IA comprennent la narrow AI, la reconnaissance faciale et la vision par ordinateur.

Les taches relevant à l'IA sont parfois simples et faciles pour les humains, comme par exemple reconnaitre et localiser les objets dans une image. Comme elle peut faire des taches plus compliquées et requiert beaucoup de connaissance et de sens commun, par exemple pour traduire un texte ou mener un dialogue.

C'est grâce à l'apprentissage qu'un système intelligent est capable d'exécuter une tâche et peut aussi améliorer ses performances avec de l'expérience.

Il existe plusieurs domaines dans intelligence artificielle et parmi ces domaines :

I.2.1 Systèmes experts

Soit un logiciel capable de simuler le comportement d'un humain effectuant une tâche très précise. C'est un domaine où l'intelligence artificielle est incontestablement un succès, dû au caractère très précis de l'activité demandée à simuler.

I.2.2 Calcul formel (opposé au calcul numérique)

Traiter les expressions symboliques. Des logiciels sur le marché, comme Mathematica, Maple, etc., effectuent tous des **calculs formels**.

I.2.3 La Swarm Intelligence

La Swarm Intelligence ou intelligence distribuée est le comportement collectif de systèmes naturels ou artificiels décentralisés et **auto-organisés**. Plus précisément, il s'agit généralement d'un comportement collectif résultant des interactions locales entre plusieurs individus ou avec leur environnement.

On trouve aussi parmi les domaines que nous avons cités le Machine Learning (ML).

II.3 Machine Learning (ML)

Machine Learning (ML) est un domaine scientifique d'analyse de donné (Analytique and data science) et plus particulièrement une sous-catégorie de l'intelligence artificielle.

Il consiste à laisser des algorithmes découvrir des patterns à savoir des motifs récurrents, dans les ensembles de données. Ces données peuvent être des chiffres, des mots, des images des statistiques …

Les algorithmes du Machine Learning apprennent d'une manière autonome (ceci n'exclut pas la supervision) à partir de données et améliorent leurs performances au fil du temps.

Il existe 4 familles de type d'approches utilisées en ML :

- **Apprentissage basé sur l'information**
- **Apprentissage basé sur la similarité**
- **Apprentissage basé sur l'erreur**
- **Apprentissage basé sur la probabilité**

On trouve 3 types de l'apprentissage automatique :

- **Le supervisé**
- **Le non-supervisé**
- **Par renforcement**

I.3.1 Apprentissage supervisé

Pour cet apprentissage, nous avons des données en entrée (*Features*) et le résultat attendu (Label). Il nous permet de faire des prédictions basées sur un modèle*qui est obtenu à partir de données d'historique et de l'algorithme choisi.

I.3.2 Apprentissage non-supervisé

Avec cet apprentissage on a toujours des features, mais pas de *label*, car nous n'essayons pas de prédire quoi que ce soit, il sert généralement à découvrir des structures et des modèles dans les données.

I.3.3 Apprentissage par renforcement

Avec ce type d'apprentissage l'algorithme apprend un comportement étant donné une observation. L'action de l'algorithme sur l'environnement produit une valeur de retour qui guide l'algorithme d'apprentissage. On utilise souvent ce type dans le cadre de la théorie, des robots et des véhicules autonomes.

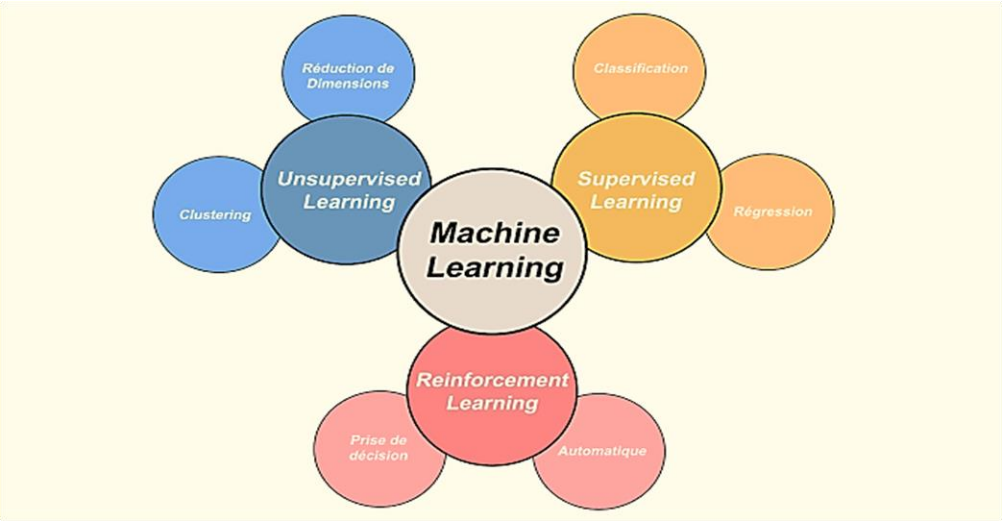


Figure I.1 les types de l'apprentissage automatique.

I.4 Les réseaux de neurones

Le réseau de neurones artificiel est un modèle de calcul dont la conception est très schématiquement inspirée du fonctionnement des neurones biologiques. Les réseaux de neurones sont généralement optimisés par des méthodes d'apprentissage de type probabiliste ils sont placés d'une part dans la famille des applications statistiques et d'autre part dans la famille des méthodes de l'intelligence artificielle**.** Un réseau de neurones est un ensemble de neurones inter connectés d'une manière précise, qui sont organisés sous forme de couches séparant les entrées des sorties du réseau.

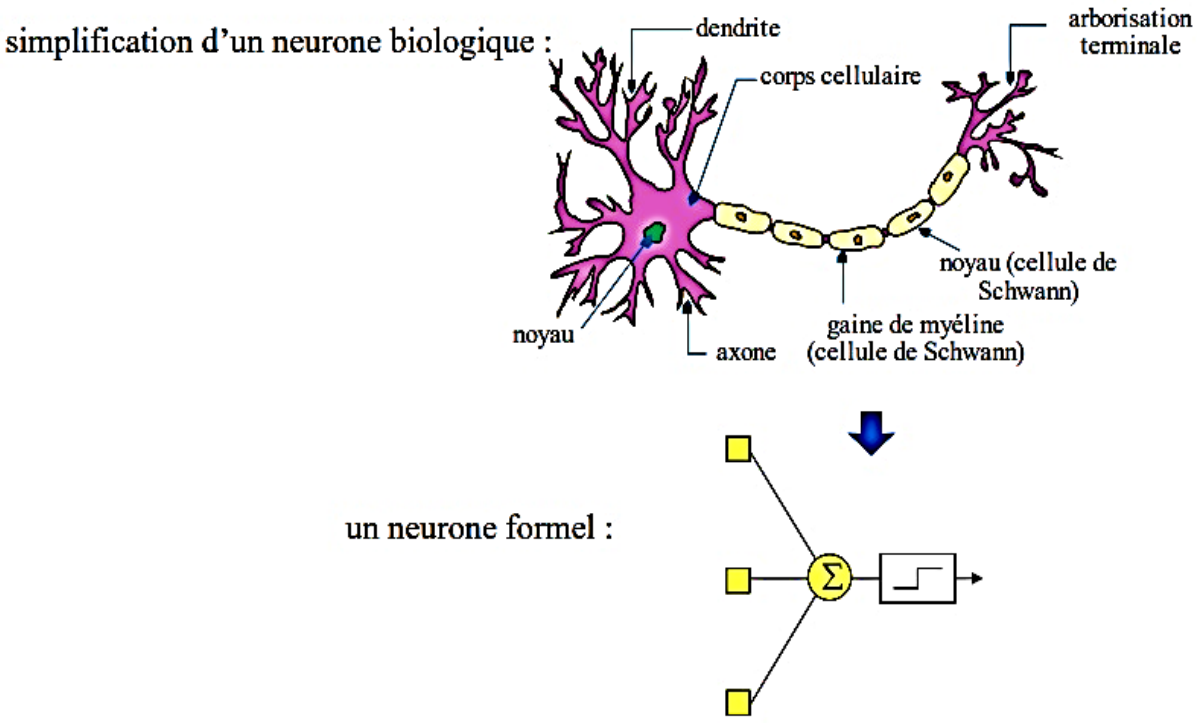


Figure I.2 Schéma du neurone biologique au neurone formel

I.6 Deep Learning (Apprentissage profond (DL))

Le Deep Learnig (DL) est une technologie qui fait partie de la Machine Learning (ML), qui à son tour appartient à l'Intelligence Artificielle (IA). Le DL intervient dans la construction d'un extracteur de caractéristique pour reconnaissance des formes.

Un réseau de neurone profond est composé d'une couche d'entrée ou les données sont insérées dans un réseau contenant plusieurs couches, et le DL consiste à modéliser et entrainer ce dernier.

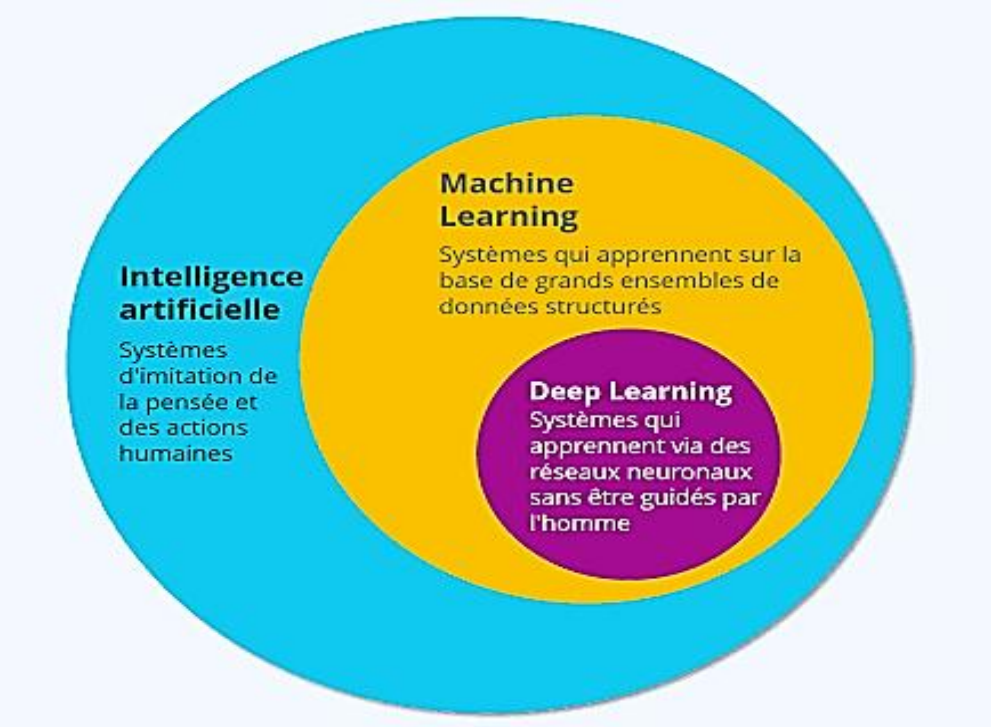


Figure I.5 Deep Learning discipline qui fait partie de Machine Learning.

I.7 La base de données

Une base de données est un ensemble d'informations qui sont organisées de manière à être facilement accessibles, gérées et mises à jour. Elle est utilisée par les organisations comme méthode de stockage, de gestion et de récupération de l'information.

Les données sont organisées en lignes, colonnes et tableaux et sont indexées pour faciliter la recherche d'informations. Les données sont mises à jour, complétées ou encore supprimées au fur et à mesure que de nouvelles informations sont ajoutées. Elles contiennent généralement des agrégations d'enregistrements ou de fichiers de données, tels que les transactions de vente, les catalogues et inventaires de produits et les profils de clients.

Généralement, l'administrateur de la base de données régule les accès des utilisateurs afin de contrôler leurs actions et d'analyser les usages. Pour garantir la cohérence des données et l'intégralité des transactions.

Les bases de données utilisées en DL sont conçues de sorte à contenir une diversité acceptable afin de réaliser et réussir l'apprentissage ou l'entrainement d'un réseau profond.

I.8 Conclusion

Dans ce chapitre, nous avons pu aborder la définition de l'intelligence artificielle ainsi que quelques domaines et parmi eux le Machine Learning (ML) et le Deep Learning (DL). Ensuite, nous avons présenté quelques notions élémentaires des réseaux de neurones artificiels, particulièrement les réseaux de neurones profonds. Nous avons clôturé ce chapitre avec la définition et le fonctionnement des bases des données. Les notions acquises dans ce chapitre nous permettront de mieux comprendre les réseaux de neurones convolutifs qui seront présentés dans le chapitre II.

II.1 Introduction

Dans ce chapitre, nous allons prendre quelques architectures et le fonctionnement des réseaux de neurones artificiels à travers les équations mathématiques qui définissent cette méthode, et découvrir les différentes couches. Le RN est une imitation algorithmique des fonctions du cerveau humain. Ces algorithmes sont utilisés pour la résolution de diverses problématiques d'apprentissage du Machine Learning. Ils sont beaucoup plus performants dans cet exercice que les techniques de régressions classiques. La façon la plus courante de les former consiste à utiliser l'algorithme de la descente du gradient à travers la rétro propagation. Nous nous concentrerons sur un type de RN particulier qui sont les réseaux de neurones convolutifs (ConvNet ou CNN) pour la classification automatique d'images. Ce dernier sera exploité dans le chapitre suivant pour reconnaitre les symboles de l'écriture Tifinagh.

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

Un réseau de neurones est un ensemble de neurones interconnectés d'une manière précise. Ils sont organisés sous forme de couches séparant les entrées des sorties du réseau [1]. Un CNN, convolutional neural network, est un type de réseau neuronal artificiel utilisé dans la reconnaissance et le traitement d'images et spécifiquement conçu pour traiter les données. Les CNN sont de puissants systèmes de traitement d'images, d'intelligence artificielle (IA) qui utilisent un apprentissage profond (Deep Learning).

Les RN dit CNN sont disposées de manière à ressembler davantage à ceux du lobe frontal, la zone responsable du traitement des stimuli visuels chez l'homme et d'autres animaux. Les couches de neurones sont disposées de manière à couvrir l'ensemble du champ visuel, évitant ainsi le problème du traitement d'images par morceaux des réseaux neuronaux traditionnels [1].

Un CNN utilise un système semblable à un perceptron multicouche qui a été conçu pour des besoins de traitement réduits. Les couches d'un CNN se composent d'une couche d'entrée, d'une couche de sortie et d'une couche cachée qui comprend plusieurs couches convolutionnelles, des couches de regroupement, des couches entièrement connectées et des couches de normalisation [1].

II.3 Les couches d'un réseau de neurones

Une couche dans un réseau de neurones est une superposition de plusieurs neurones. Les données (entréesXi) entrantes dans chaque neurone sont multipliées par un poids synaptiquesWi, la sortie passe par une fonction d'agrégation qui fait à son tour la somme entre Vm et les entrées Xi ensuite passe par une fonction d'activation $\varphi_{(v_m)}$ qui permet de produire une sortie Y.

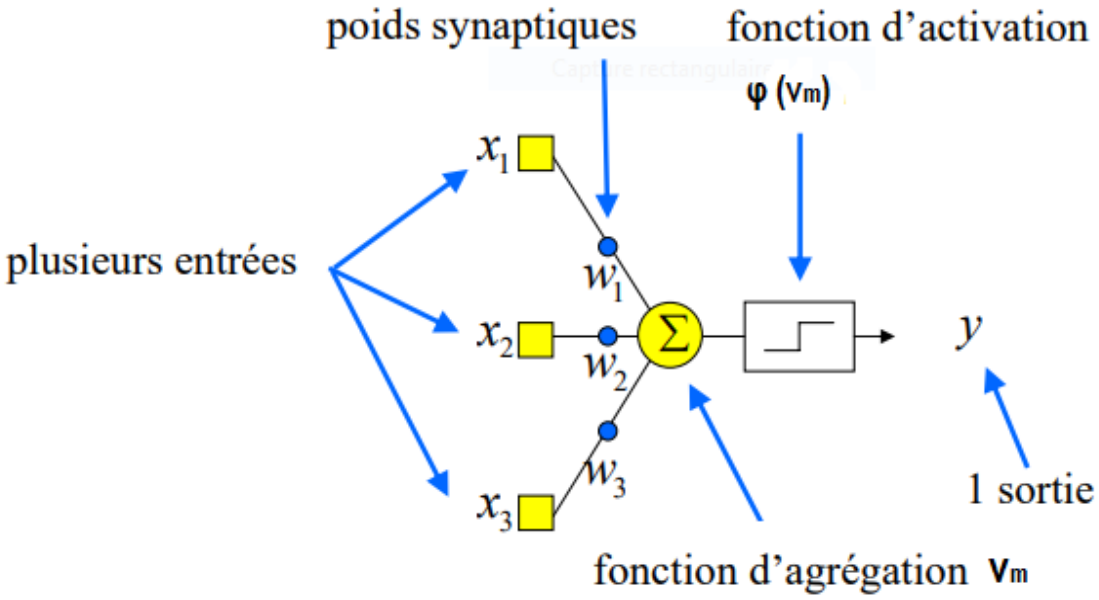


Figure II.1 Neurone Artificiel

II.3.1 Quelques définitions des composants d'un neurone artificiel

II.3.1.1 Le poids synaptique

Le poids synaptique (PS) représente le poids de la relation entre deux neurones connectés, autrement dit la probabilité d'obtenir une réponse de l'élément Post-Synaptique (fonction d'activation) en fonction de l'entrée, venant de l'élément Pré-Synaptique (poids synaptique). On le retrouve dans tous les réseaux de neurones modélisée et biologiques.

II.3.1.2 La fonction d'agrégation

En traitement des bases de données, la fonction d'agrégation ou la fonction de combinatoire est un opérateur permettant de réduire des groupes de lignes à une valeur calculée à partir de l'une des colonnes en jeu. Les fonctions les plus simples sont : la somme, la moyenne, le maximum et le minimum des valeurs.

II.3.1.3 La fonction d'activation

La fonction d'activation est une fonction mathématique appliquée à un signal de sortie d'un neurone artificiel, la fonction d'activation sert avant tout à modifier de façon non-linéaire les données. Dit simplement, la fonction d'activation permet de changer notre manière de voir une donnée. Parmi les fonctions d'activation que nous utilisons :

II.3.1.4 La fonction ReLu

Elle donne x si x est supérieur à 0, et 0 sinon. Autrement dit, c'est le maximum entre x et 0 : Fonction $Relu(x) = max\ (x, 0)$

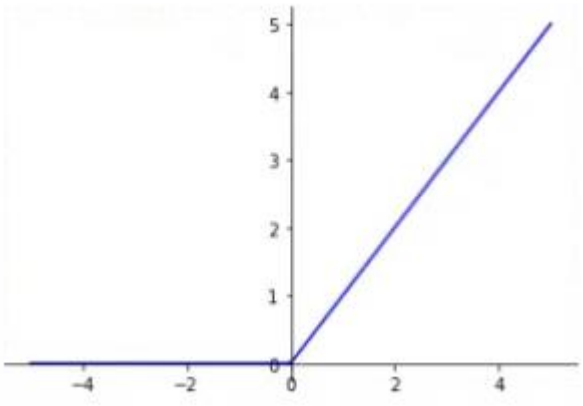


Figure II.2 la fonction ReLu – Rectified Linear Unit

Cette fonction permet d'effectuer un filtre sur nos données. Elle laisse passer les valeurs positives (x > 0) dans les couches suivantes du réseau de neurones. Elle est utilisée presque partout dans le réseau de neurones, mais surtout pas dans la couche finale, elle est utilisée dans les couches intermédiaires [5].

II.3.1.5 La fonction sigmoïde

La fonction Sigmoïde donne une valeur entre 0 et 1, une probabilité. Elle est donc très utilisée pour les classifications binaires, lorsqu'un modèle doit déterminer seulement deux labels [5]. Fonction Sigmoïde $(x) = \frac{1}{1 + exp(-x)}$

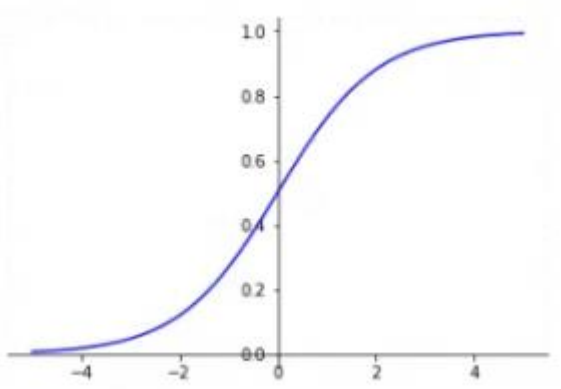


Figure II.3 Fonction Sigmoïde.

II.3.1.6 La fonction Softmax

La fonction Softmax permet de transformer un vecteur réel en vecteur de probabilité. On l'utilise souvent dans la couche finale d'un modèle de classification, notamment pour les problèmes multiclasse. Dans cette fonction, chaque vecteur est traité indépendamment [5].

$$f_{Softmax(x_j)} = \frac{e^{x_j}}{\sum_{i=1}^{n} e^{x_j}}$$

On trouve aussi plusieurs fonctions comme : Softplus, Softsign … etc. Cette propriété permet de créer des optimisations basées sur les gradients, c'est-à-dire lors de l'apprentissage, l'erreur se propage de sortie vers l'entrée et les sorties des fonctions d'activations doivent être différentiables afin de pouvoir utiliser des méthodes d'optimisation SGD (Descente de Gradient).

II.4 La méthode SGD

La descente de gradient stochastique (stochastic gradient descent, SGD) est une méthode qui permet de sélectionner au hasard des valeurs, au lieu de prendre toutes les données, nous travaillons sur des échantillons à chaque epochs et nous choisirons ces échantillons de manière aléatoire, voilà c'est exactement ce que propose une descente de gradient stochastique (ou SGD).

II.4.1 Méthode de la Descente de Gradient avec Momentum SGDM

SGD avec Momentum est une méthode d'optimisation stochastique qui ajoute un terme de momentum à la descente de gradient stochastique. Notre but est de trouver le point le plus bas de la fonction d'optimisation et cette méthode nous aide à réaliser ça, pour appliquer cette dernière notre fonction doit être différentiable. Cependant, la direction réelle de ce point n'est pas connue. Nous ne pouvons regarder que localement et de ce fait la direction du gradient négatif est la meilleure information dont nous disposons. Faire un petit pas dans cette direction ne peut que nous rapprocher du minimum. Une fois que nous avons fait ce petit pas, nous calculons à nouveau le gradient et nous nous déplaçons un peu dans cette direction. Nous répétons le processus jusqu'à ce que nous atteignions la vallée. Par conséquent, la descente de gradient ne fait essentiellement que suivre la direction de la descente la plus raide (pente négative) [6].

II.4.2 La propagation directe

La propagation directe c'est de faire propager les données depuis l'entrée d'un réseau de neurones à sa sortie. Chaque sortie de chaque couche est calculée en fonction de ses poids synaptiques et des sorties de la couche précédente.

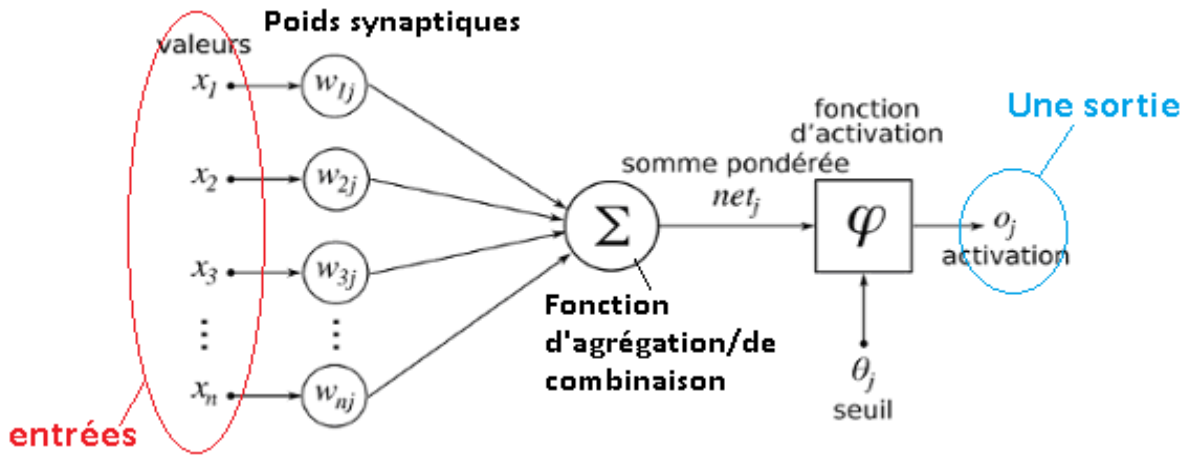


Figure II.6 – Neurone formel

L'expression de la somme V est exprimée comme suit :

$$v = (\sum_{i=1}^{n} w_i \times x_i) + b \text{ ..... (II.1)}$$

La sortie y est alors produite grâce à la fonction d'activation φ :

$$y = \varphi(v) \text{ ..... (2)}$$

Les poids w et les entrées x peuvent être exprimés sous forme matricielle, soit :

$$W = [w_1 \quad w_2 \quad \dots \quad w_n] \text{ et } X = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \text{ ..... (3)}$$

Le résultat v sera alors exprimé :

$$v = W \times X + b \text{ ..... (4)}$$

$$v = [w_1 \quad w_2 \quad \dots \quad w_n] \times \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} + b \text{ ..... (5)}$$

La somme des v_k sera :

$$\forall k \in [1, m], \qquad v_k = (\sum_{i=1}^{n} w_{ik} + x_i) + b_k \text{ ..... (6)}$$

Et les sorties y_k seront :

$$\forall k \in [1, m], \qquad y_k = \varphi(v_k)$$

Soit n la dimension du vecteur X (nombre de données en entrée), et m le nombre de neurones dans une couche donnée, les expressions précédentes peuvent être écrite sous la forme matricielle, si W est la matrice des poids et N est le vecteur de sortie composé d'éléments v_m :

$$W = \begin{bmatrix} w_{11} & w_{12} & \dots & w_{1m} \\ w_{21} & w_{22} & \cdots & w_{2m} \\ \vdots & & \ddots & \vdots \\ w_{n1} & w_{n2} & \cdots & w_{nm} \end{bmatrix} \dots\dots\dots (7)$$

$$N = W \times X + B \dots\dots\dots (8)$$

$$\begin{bmatrix} \mathrm{v}_1 \\ \mathrm{v}_2 \\ \vdots \\ \mathrm{v}_m \end{bmatrix} = \begin{bmatrix} \mathrm{w}_{11} & \mathrm{w}_{12} & \dots & \mathrm{w}_{1m} \\ \mathrm{w}_{21} & \mathrm{w}_{22} & \cdots & \mathrm{w}_{2m} \\ \vdots & & \ddots & \vdots \\ \mathrm{w}_{n1} & \mathrm{w}_{n2} & \cdots & \mathrm{w}_{nm} \end{bmatrix} \times \begin{bmatrix} \mathrm{x}_1 \\ \mathrm{x}_2 \\ \vdots \\ \mathrm{x}_n \end{bmatrix} + \begin{bmatrix} \mathrm{b}_1 \\ \mathrm{b}_2 \\ \vdots \\ \mathrm{b}_m \end{bmatrix} \dots\dots\dots\dots (9)$$

Le vecteur de sortie Y sera exprimé ainsi :

$$Y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix} = \begin{bmatrix} \varphi_1(\mathrm{v}_1) \\ \varphi_2(\mathrm{v}_2) \\ \vdots \\ \varphi(\mathrm{v}_m) \end{bmatrix} \dots\dots\dots\dots (10)$$

Où n est le nombre de couches du réseau, k est la position de la couche. Le m est le nombre de nœuds d'une couche donnée, c.-à-d. m est le nombre de neurones de la couche k. Les majuscules sont des vecteurs ou matrices des variables en minuscules, c.-à-d. si $i \in \{1, \dots, m\}$ alors :

$$Y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix} \dots\dots\dots (11)$$

II.4.3 La rétro propagation

Dans le cas d'un apprentissage supervisé, des données sont présentées à l'entrée du réseau de neurones et celui-ci produit des sorties. La valeur des sorties dépend des paramètres liés à la structure du réseau de neurones : la connexion entre les neurones, fonctions d'agrégation et d'activation ainsi que les poids synaptiques.

Les différences entre ces sorties et les sorties désirées forment des erreurs qui sont corrigées via la rétro propagation, les poids du réseau de neurones sont alors changés. La manière de quantifier cette erreur peut varier selon le type d'apprentissage à effectuer**.** En appliquant cette étape plusieurs fois, l'erreur tend à diminuer et le réseau offre une meilleure prédiction. Il se peut toutefois qu'il ne parvienne pas à échapper à un minimum local, c'est pourquoi on ajoute en général un terme d'inertie (momentum) à la formule de la rétro propagation pour aider l'algorithme du gradient à sortir de ces minimums locaux.

Après la propagation directe, nous calculons l'erreur $d - y$ en fonction de gradient de critère d'erreur par rapport au poids synaptique nous corrigeons la valeur de ce dernier, selon la fonction suivante :

$$w = w + \alpha \frac{\partial J}{\partial W}$$

Avec J critère de l'erreur quadratique

$$J = \frac{1}{2}\sum e^2 = \frac{1}{2}\sum (d - y)^2$$

L'algorithme suivant nous détaille un peu plus les étapes utilisées pour faire un apprentissage automatique d'un réseau de neurones.

Algorithme 1 : Algorithme d'apprentissage avec rétro-propagation

Input : $learning\_Rate : \alpha, data\_Input\ ;\ D \leftarrow \{(\boldsymbol{x_p}, \boldsymbol{d_p})\}, number\_Of\_Epochs: epoch$
Output : $W^{(k)}$ //K *is the layer position*
for $k \leftarrow 1$ ***to*** n ***do***
  $initW\ eights(W^{(k)})$ //weights initialization
end
for $q \leftarrow$ **to** *epochs* **do**
  for (x, d) ***as*** $(\mathbf{x_p}, \mathbf{d_p})$ ***in*** $D = \{(\mathbf{x_p}, \mathbf{d_p})\}$ **do**
    $/*$ Forward propagation
    $Y^{(0)} = X$
    for $k \leftarrow 1$ **to** n **do**
      $Y^{(k)} = \varphi(W^{(k)} \times Y^{(k=1)})$
    end
    $/*$ *Backpropagation*
    for $i \leftarrow 1$ **to** $m^{(n)}$ **do**
      $e_i = d_i - y_i$
      $\delta_i = \varphi'_i$
    end
    for $k \leftarrow n - 1$ **to** 1 **do**
      for $j \leftarrow 1$ **to** $m^{(k)}$ **do**

$$e_j{}^{(k)} = \sum_{i=1}^{m^{(k+1)}} w_{ij} \times \delta_i{}^{(k+1)}$$

$$\delta_j{}^{(k)} = \varphi'_j \left(\sum_{i=1}^{m^{(k-1)}} w_{ij}{}^{(k)} \times y_i{}^{(k-1)} \right) \times e_j{}^{(k)}$$

      end
    end
    for $k \leftarrow 1$ **to** n **do**
      for $i \leftarrow 1$ **to** $m^{(k)}$ **do**
        for $j \leftarrow 1$ **to** m^k **do**
          $\Delta w_{ij}{}^{(k)} = \alpha \times \delta_i{}^{(k)} \times y_j{}^{(k-1)}$
          $w_{ij}{}^k = w_{ij}{}^{(k)} + \Delta w_{ij}{}^{(k)}$ //weights initialization
        end
      end
    end
  end
end

Parmi les grandes familles des réseaux de neurones profonds, nous avons les CNN (réseaux de neurones convolutifs ou réseaux de neurones à convolution) qui sont des modèles puissants permettant notamment la reconnaissance d'images en attribuant automatiquement à chaque image fournie en entrée, une étiquette correspondant à sa classe d'appartenance

Il existe quatre types de couches pour un réseau de neurones convolutif : la couche de convolution, la couche de pooling, la couche de correction ReLu et la couche fully-connected, voici un exemple explicatif de différentes couches :

II.5 Réseaux de neurones profonds

Un réseau de neurones est en général composé d'une succession de couches dont chacune prend ses entrées sur les sorties de la précédente. Chaque couche (k) est composée de m_k neurones, prenant leurs entrées sur les $m_k - 1$ neurones de la couche précédente.

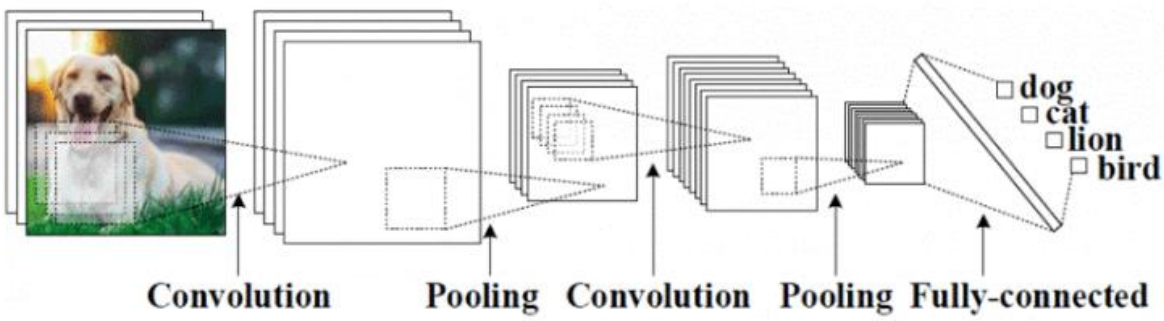


Figure II.6 les différentes couches d'un réseau convolutif.

II.5.1 Couche convolutive

La couche de convolution est la couche la plus importante des réseaux de neurone convolutif, Il est prévu d'appliquer un filtre de convolution à l'image pour découvrir les caractéristiques de l'image. Une image passe à travers une succession de filtres, créant de nouvelles images appelées cartes de caractéristiques ("Feature map output") selon un certain nombre de filtres ("Kernels"en anglais). Certains filtres intermédiaires réduisent la résolution de l'image par une opération de maximum local. La sortie est une matrice, chacun de ses éléments est calculé à partir de la somme des multiplications des éléments du filtre par les éléments de la matrice d'entrée (souvent une image), chaque élément du résultat est trouvé en fonction de la position du filtre [7].

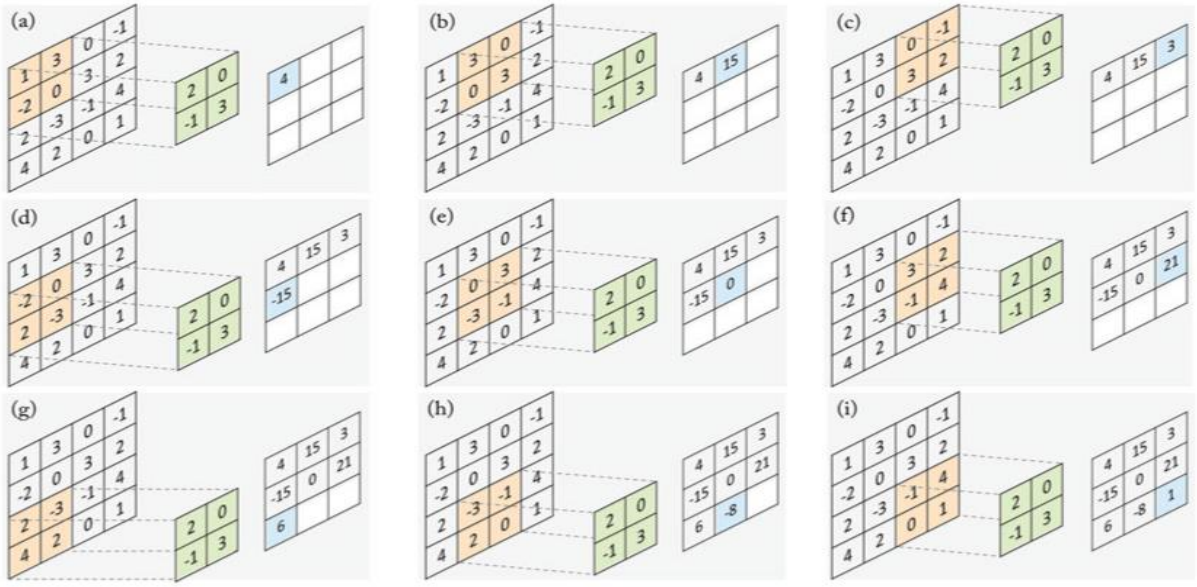


Figure II.8 L'opération de convolution avec filtre 2×2 stride 1

Il consiste à diminuer le nombre d'échantillons à traiter (réduire la dimension de la matrice), elle consiste à faire déplacer à chaque étape le filtre d'un certain nombre de crans, horizontalement et verticalement, ce nombre est appelé **Stride,** l'augmentation de stride fait réduire la dimension de la matrice de sortie. Sur la Figure précédente le Stride égale à 1 parce que le filtre est glissé d'un seul cran à chaque étape.

La figure suivante nous montre la différence entre un Stride de 1 et un Stride de 2

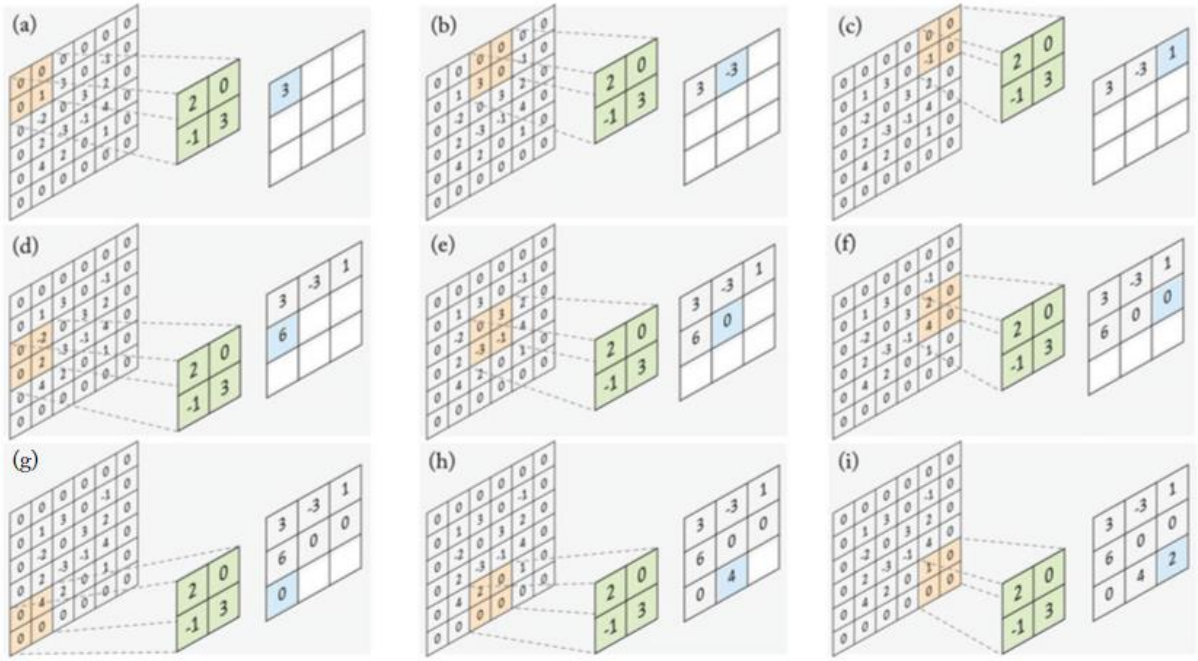


Figure II.9 L'opération de convolution avec filtre 2×2 stride 2

II.5.2 La couche de normalisation

La couche de Normalisation applique une normalisation à ces données. Pour être plus précis, cette couche va faire en sorte que les données aient la même répartition. On peut imaginer normaliser nos données pour qu'ils soient tous dans une échelle de 0 à 1. Appliquer une normalisation à nos données après chaque couche permet au modèle d'apprendre plus facilement en faisant moins d'erreur [2].

II.5.3La fonction ReLu

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

$$f_{ReLu(x)} = \begin{cases} 0 & x < 0 \\ x & x \geq 0 \end{cases}$$

II.5.4 La couche Pooling

La couche Pooling est une couche qui 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, et cette opération consiste à réduire la taille des images, tout en préservant leurs caractéristiques importantes.

Ou la taille de l'entrée est $w \times h$, la région de Pooling est$t \times t$, stride est s

$$w_{pooling} = \frac{w - t + s}{s}$$

$$h_{pooling} = \frac{h - t + s}{s}$$

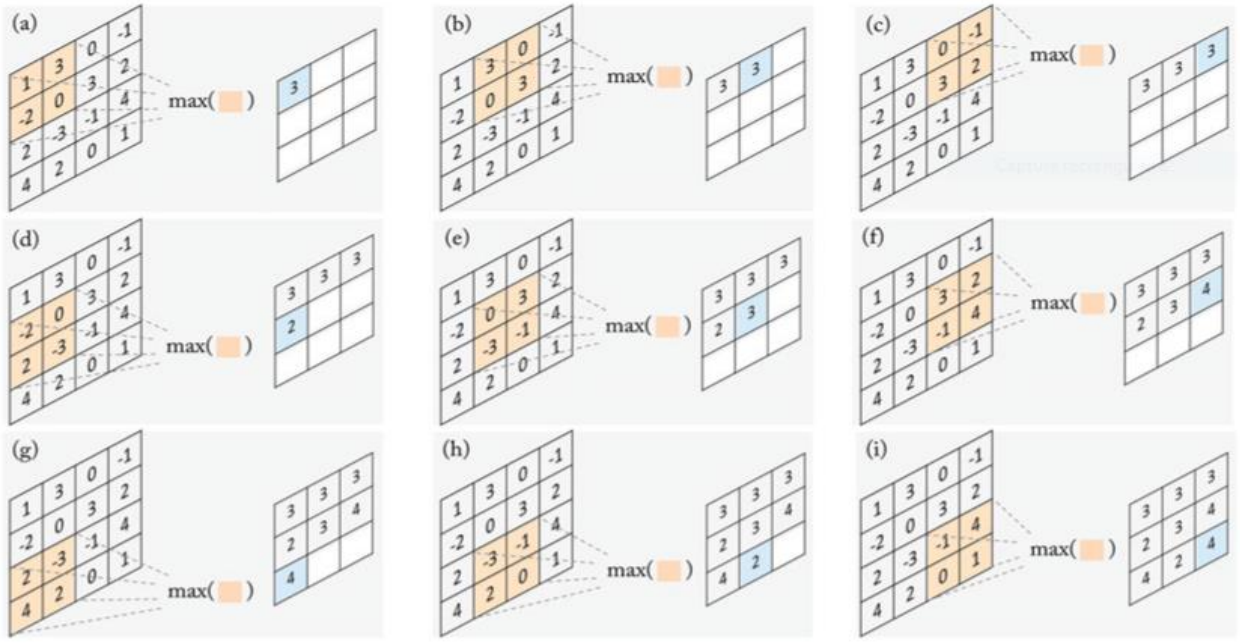


Figure II.10 L'opération de Max Pooling

II.5.5 La couche Fully Connected

Les couches Fully connected sont les dernières couches du réseau de neurones convolutif. Ce type de couche reçoit un vecteur en entrée et produit un nouveau vecteur en sortie. Pour cela, elle applique une combinaison linéaire puis éventuellement une fonction d'activation aux valeurs reçues en entrée [3].

Pour calculer les probabilités, la couche fully-connected multiplie donc chaque élément en entrée par un poids, fait la somme, puis applique une fonction. Ce traitement revient à multiplier le vecteur en entrée par la matrice contenant les poids. Le fait que chaque valeur en entrée soit connectée avec toutes les valeurs en sortie explique le terme *fully-connected*. [4].

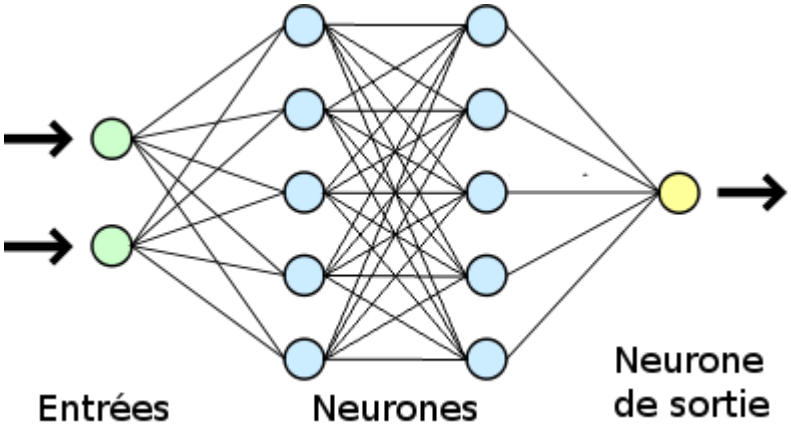


Figure II.11 Réseau multicouche profond avec une couche d'entrée, deux couches Fully Connected et une sortie

II.5.6 La couche de sortie Softmax

La fonction Softmax est un type de fonction d'activation, elle est utilisée pour calculer la distribution de probabilité d'un vecteur de valeurs réelles, et produit une sortie qui est un vecteur de valeurs entre 0 et 1, dont la somme est 1.

Nous utilisons la fonction Softmax quasiment dans toutes les couches de sortie des architectures de réseaux de neurones utilisés pour faire la classification.

II.6 Quelques architectures de réseaux de neurones convolutifs

II.6.1 L'architecture LeNet-5

LeNet-5, un réseau convolutionnel pionnier à 7 niveaux par LeCun et al en 1998, qui classe les chiffres, a été appliqué par plusieurs banques pour reconnaître les numéros manuscrits sur les chèques (chèques) numérisés en images d'entrée en niveaux de gris de 32x32 pixels. La capacité de traiter des images à plus haute résolution nécessite des couches plus grandes et plus convolutives, de sorte que cette technique est limitée par la disponibilité des ressources informatiques.

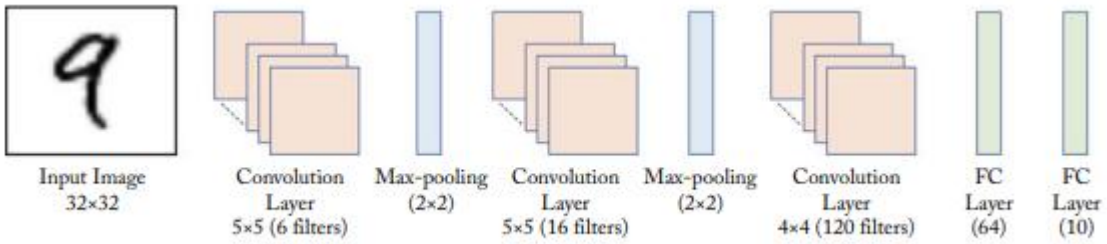


Figure II.12 Architecture de LeNet-5

II.6.2 L'architecture AlexNet

Le réseau a une architecture très similaire à LeNet mais il est plus profond, avec plus de filtres par couche et avec des couches convolutionnelles empilées et elle est la première architecture de CNN à grande échelle. Elle prend des images de $(24 \times 24 \times 3)$ en RGB et contient huit couches dont les cinq premières sont des couches de convolution et les trois dernières couches sont des couches de Fully Connected (FC). Cette architecture se présente comme suit :

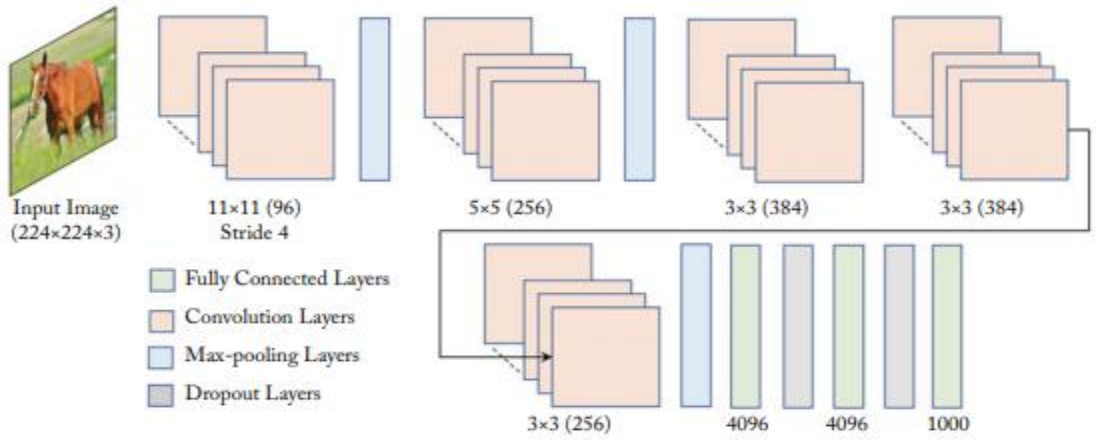


Figure II.13 Architecture d'AlexNet [5]

II.6.3 L'architecture VGGNet

Le finaliste du concours ILSVRC 2014 est surnommé VGGNet par la communauté et a été développé par K.Simonyan et A. Zisserman. VGGNet se compose de 16 couches convolutionnelles et est très attrayant en raison de son architecture très uniforme. Semblable à AlexNet, seulement des convolutions 3x3, mais beaucoup de filtres. Formé sur 4 GPU pendant 2 à 3 semaines. C'est actuellement le choix le plus préféré dans la communauté pour extraire des

fonctionnalités à partir d'images. La configuration de poids du VGGNet est accessible au public et a été utilisée dans de nombreuses autres applications et défis en tant qu'extracteur de fonctionnalités de base. Cependant, VGGNet se compose de 138 millions de paramètres, ce qui peut être un peu difficile à gérer [6].

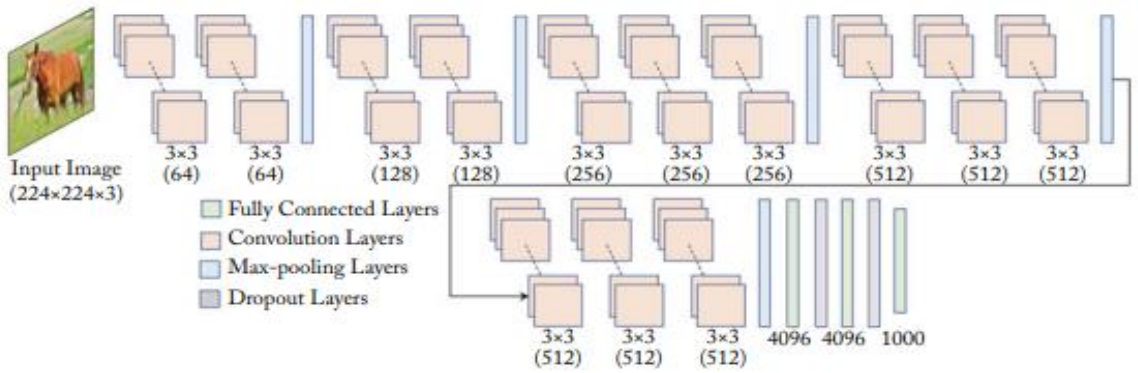


Figure II.14 L'architecture VGGNet

II.7 Conclusion

Dans ce chapitre, nous avons défini et décrit les réseaux de neurones convolutifs (CNN), les fonctions d'activation et les couches de CNN, et enfin nous avons exploré les différentes architectures les plus populaires et les plus largement utilisées. Dans le chapitre suivant nous expliquerons en détail le choix de l'architecture et la structure choisie.

III.1 Introduction

Ce chapitre est le chapitre le plus important, où nous aborderons des étapes spécifiques pour atteindre le réseau de neurones profond idéal dédié à la réalisation du système de lecture automatique des caractères berbères manuscrits par apprentissage profond.

Lors de chaque test, nous utiliserons deux bases de données AMHCD et BerbèreDataSet (BBDS). Pour chaque base de données, nous diviserons les tests en quatre parties, changement de paramètre, changement de structure, structure inspiré de LeNet, structure inspiré d'Alex Net, la bonne structure sera sélectionnée afin d'être capable de faire les tests sur la webcam, une application sera construite sur LabVIEW qui se compose de deux composants principaux : une face avant et un schéma, les résultats doivent être satisfaisants pour arriver à une bonne estimation.

III.2 Approche choisie pour l'apprentissage

III.2.1 Configuration utilisée dans l'implémentation

La configuration utilisée du matériel dans notre configuration est :

- PC portable HP-i5-PC
- Intel(R) Core(TM) i5-4310M à 2,70 GHz
- RAM de taille 4,00 Go
- Le système d'exploitation 64 bits, processeur x64
- Système d'exploitation windows 10 professional

III.2.2 Les bases de données utilisées

Pour chaque apprentissage nous avons besoin d'une base de données (Datasets), ces dernières sont généralement organisées sous forme d'images enregistrées sous format de fichiers spécifique pour des applications en vision artificielle ou vision par ordinateur. Dans le cas de l'apprentissage supervisé, ce sont les labels qui désignent la classe correspondante à l'image, chaque image est associé à son label (classe ou étiquette). Chaque image à sa capacité et son nombre de classes, les images généralement sont divisées en deux catégories : image d'apprentissage et images de tests.

III.2.2.3 Base de données berbère dataset (BBDS)

Le tifinagh ou alphabet touareg est l'écriture utilisée par les Berbères pour écrire leur langue, le tamazight. La base de données berbère dataset que nous avons réalisée est constituée d'un ensemble de 36 300 d'image de caractères manuscrits représentant les 33 lettres de l'alphabet berbères, en noir et blanc, normalisées, centrées, de dimension (28×28) pixels. 33 000 d'entre elles sont destinées à l'apprentissage du réseau de neurones, et 3300 pour le test [14].

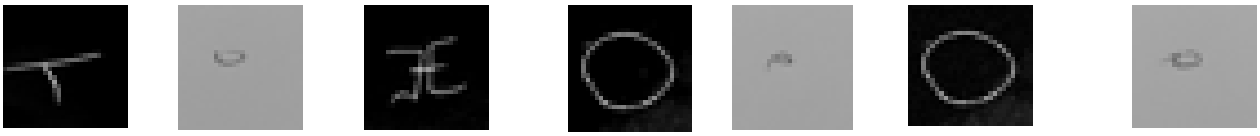


Figure III.3 Quelques échantillons de la base de données berbère dataset

III.2.2.4 Base de données AMHCD

La base de données AMHCD est constituée d'un ensemble de 25 740 d'images de caractères manuscrits écrits par 60 personnes (femme/homme) d'âge et de profession différents, représentant les 33 classes :**[ya, yab, yach, yad,yadd, yae, yaf, yag, yagh, yagw, yah, yahh, yaj, yak, yakw, yal, yam, yan, yaq, yar, yarr, yas, yass, yat, yatt, yaw, yax, yay, yaz, yazz, yey, yi, yu**], en noir et blanc, de dimension (28×28) pixels. La MHCD contient 780 images scannées de chaque caractère parmi les 33 caractères [15].

Figure III.4 Quelques échantillons de la base de données AMHCD

chiffres correspondance à la base BBDS	Lettres en tifinagh	Correspondance en Latin	Lettres correspondance à la base AMHCD	Transcription
01	ⴰ	A	Ya	A
02	ⴱ	B	Yab	B
03	ⴳ	C	Yach	CH
04	ⴷ	D	Yad	D
05	ⴹ	Ḍ	Yadd	DD
06	ⴻ	Ḓ	Yae	DDA
07	ⴼ	E	Yaf	E
08	ⴽ	F	Yag	F
09	ⵀ	G	Yagh	G
10	ⵃ	ɛ	Yagw	AA
11	ⵄ	H	Yah	H
12	ⵅ	Ḥ	Yahh	HH
13	ⵇ	I	Yaj	I
14	ⵉ	J	Yak	J
15	ⵊ	K	Yakw	K
16	ⵍ	L	Yal	L
17	ⵎ	M	Yam	M
18	ⵏ	N	Yan	N
19	ⵓ	P	Yaq	P
20	ⵔ	Q	Yar	Q
21	ⵕ	R	Yarr	R
22	ⵖ	Ṛ	Yas	RR
23	ⵗ	S	Yass	S
24	ⵘ	Ṣ	Yat	SS
25	ⵙ	T	Yatt	T
26	ⵚ	Ṭ	Yaw	TT
27	ⵛ	U	Yax	U
28	ⵜ	V	Yay	GH
29	ⵝ	W	Yaz	W
30	ⵞ	X	Yazz	X
31	ⵟ	Y	Yey	Y
32	ⵠ	Z	Yi	Z
33	ⵡ	Ẓ	Yu	ZZ

TableauIII.1 Classe associée à chaque lettre des deux bases de données.

III.3 Apprentissage du réseau de neurones convolutif

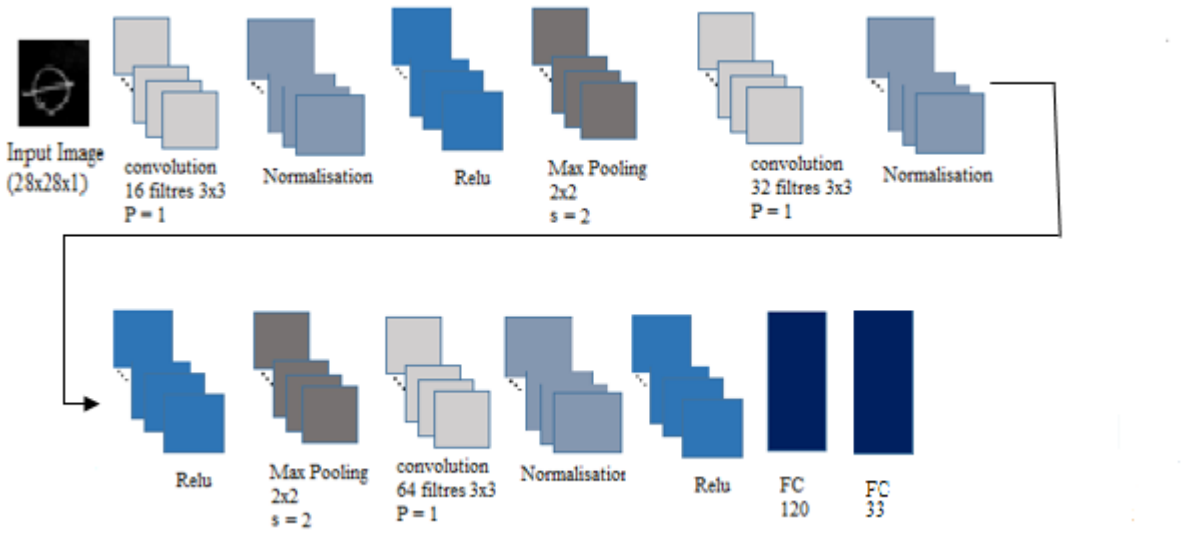
Dans cette section, nous allons présenter les résultats de l'apprentissage automatique et les réseaux de neurones (structures neuronales) utilisés. Le but principal est de découvrir et de déterminer quelle structure neuronale est la mieux adaptée pour reconnaitre les caractères tifinagh ainsi que les paramètres de l'apprentissage. Les deux bases de données BBDS et AMHCD ont été utilisées pour entrainer les CNNs.

III.3.1 Structure du réseau

Un CNN est simplement un empilement de plusieurs couches de convolution, pooling, la fonction ReLU et fully-connected. Chaque image reçue en entrée va être filtrée, réduite et corrigée plusieurs fois, pour finalement former un vecteur.

Tous les réseaux de neurones convolutifs doivent commencer par une couche de convolution et finir par une couche fully-connected. Les couches intermédiaires peuvent s'empiler de différentes manières, à condition que la sortie d'une couche ait la même structure que l'entrée de la suivante. Par exemple, une couche fully-connected, qui renvoie toujours un vecteur, ne peut pas être placée avant une couche de pooling, puisque cette dernière doit recevoir une matrice 3D. Plus il y a de couches, plus le réseau de neurones est "profond" : on est en plein dans le Deep Learning.

La structure du réseau CNN choisie contient 15 couches. 3 couches de Convolution, 2 couches de Max Pooling, 4 couches de Normalisation, 4 fonctions d'activation de type Relu et 2 couches de Fully Connected. Chaque couche de Convolution est différente à l'autre de nombre des filtres, la première contient 16 filtres de 3×3 , la deuxième a 32 filtres de 3×3, la dernière couche contient 64 filtres de 3×3 , et du stride de 1 pour les trois couches. Les deux couches de Max Pooling que nous avons utilisé sont de types 2×2 et les stride est 1. La couche de Fully Connected (FC) contient 120 neurones et la dernière couche de Fully Connected (FC) qui est la couche de sortie contient une fonction d'activation softmax cette dernière est utilisée pour calculer la distribution de probabilité d'un vecteur de valeurs réelles et elle contient 33 sorties



FigureIII.5 Structure du réseau CNN

III.3.1 choix des paramètres

L'appel de la fonction ***trainingOptions*** nous permet de faire le choix des paramètres, qui appartient au Toolbox des réseaux de neurones.

La fonction **`trainingOptions`** est appelée à travers le code suivant, ou X est le moment et Y le nombre des epochs :

```
function options = setOptions (imdsTest)

options = trainingOptions('sgdm',...
    'Momentum', X,...
    'MaxEpoch',Y, ...
    'Shuffle','every-epoch', ...
    'validationData',imdsTest, ...
    'Verbose',false,...
    'Plots','training-progress');

end
```

Listing III.1 Fonction de paramétrage du réseau

Les images de la base de données doivent être passées un certain nombre de fois, ce nombre est appelé **epoch**, il est choisi à partir de la paire : **`'MaxEpoch',28, ...`**

`'Shuffle','every-epoch', ...` demande à la fonction de mélanger la base de données c'est-à-dire l'ordre de passage des images dans le réseau durant l'apprentissage est différent pour chaque epoch.

Les données de validation sont définies par **`'validationData', imdsTest, ...`** les derniers paramètres sont pour visualiser l'évolution et résultats de l'apprentissage graphiquement et par du texte.

III.3.2.1 Le choix pour la base BBDS :

Nous avons fait le choix des paramètres X et Y d'après plusieurs essais pour les deux bases AMHCD et BBDS, le tableau suivant montre quelques tests pour la base BBDS, où Mem est le momentum

Numéro de réglage	Les paramètres	La précision	Temps
Test 1	Mem : 0.8 Max Epochs : 28	98.00%	17min 31sec
Test 2	Mem : 0.8 Max Epochs : 2	83.73%	2min 24sec
Test 3	Mem : 0.85 Max Epochs : 3	91.45%	3min 32sec

Tableau III.2 Tableau pour trois réglages

Nous prenons un exemple du **Test 1**

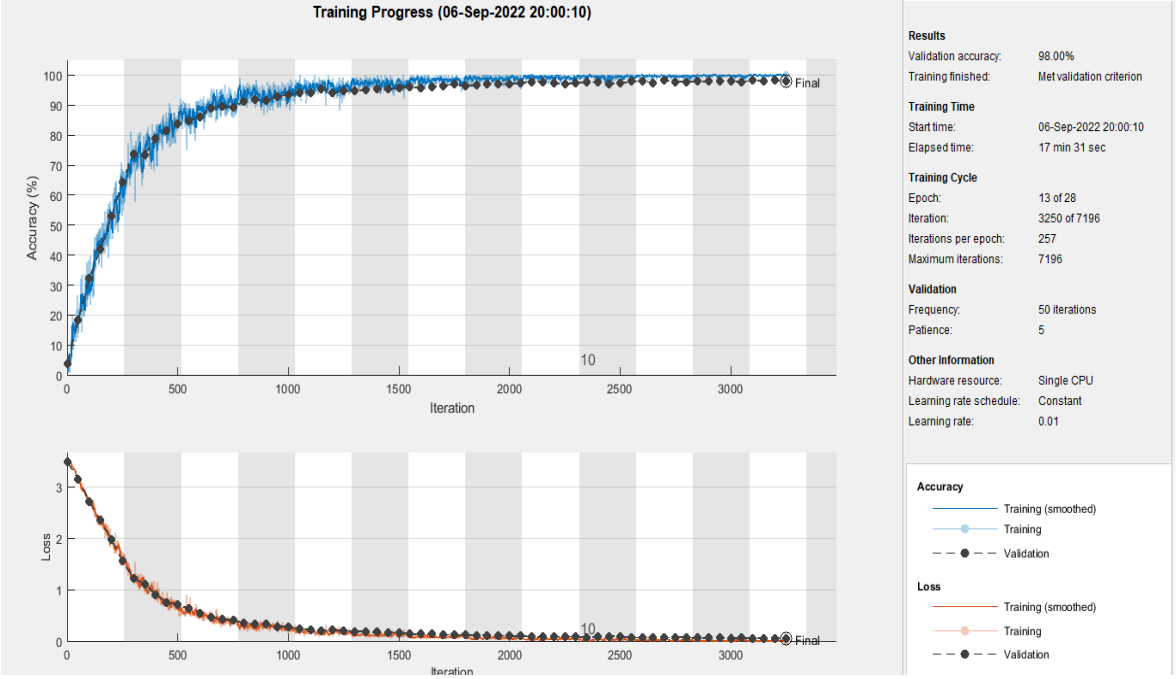


Figure III.6 Progression de l'apprentissage du modèle pour Test 1

Commentaire :

La Figure III.6 montre que l'apprentissage a pris un temps total de 17min 31sec, le modèle a atteint une précision de 98.00%. Nous remarquons que la précision du modèle augmente progressivement avec les épochs et l'erreur diminue.

III.3.2.2 Le choix pour la base AMHCD :

Nous avons fait le choix des paramètres X et Y d'après plusieurs essaies pour les deux bases AMHCD et BBDS, le tableau suivant montre quelques tests pour la base AMHCD :

Numéro de réglage	Les paramètres	La précision (Validation)	Temps
Test 4	Mem : 0.8 Max Epochs : 28	99.46%	11min 32sec
Test 5	Mem : 0.8 Max Epochs : 2	98.64%	1min 43sec
Test 6	Mem : 0.85	98.94%	2min 18sec

	Max Epochs : 3		

Tableau III.3 Tableau pour trois réglages.

Nous prenons un exemple du **Test 4**

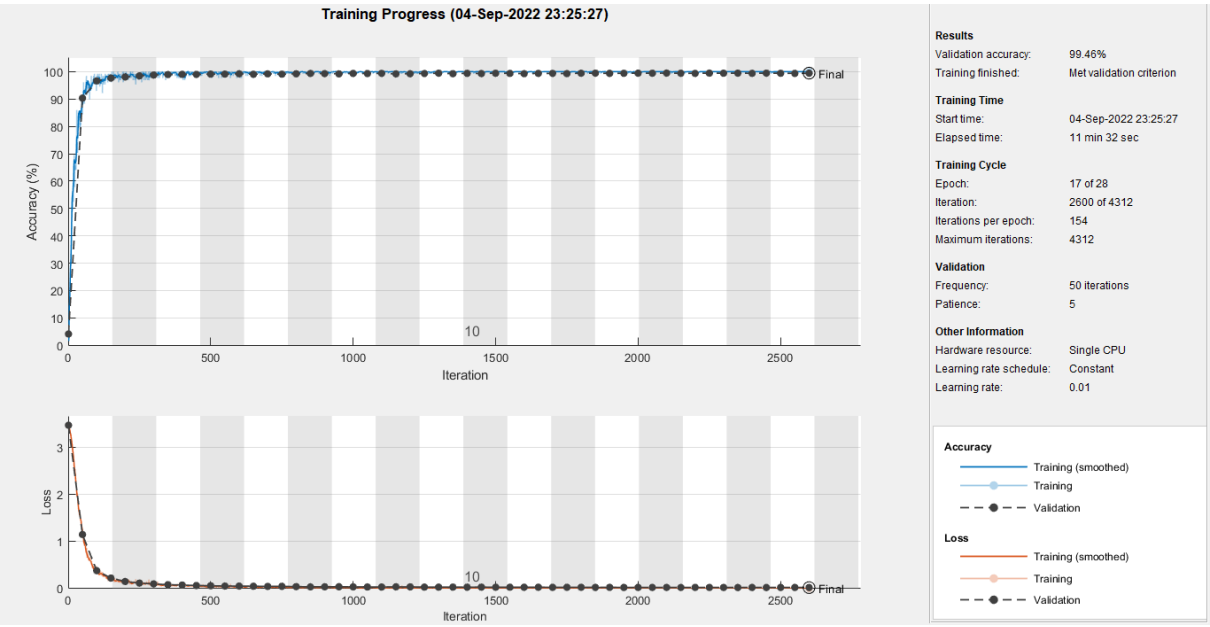


Figure III.7 Progression de l'apprentissage du modèle pour Test 4.

Commentaire :

La Figure III.7 montre que l'apprentissage a pris un temps total de 11min 32sec et le modèle a atteint une précision de 99,46%. Nous remarquons d'après les trois tests que nous avons effectués que la précision du modèle augmente progressivement avec les epochs et l'erreur diminue.

III.3.2 choix d'architecture de réseau de neurones

III.3.2.1 Changement de structure pour la BBDS

Dans cette section nous allons prendre quelques tests afin de pouvoir trouver la meilleure structure :

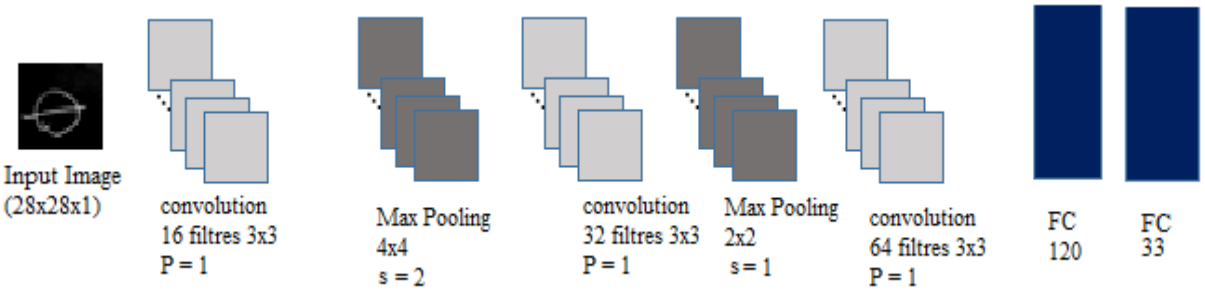
Test 7

La structure que nous avons utilisée dans cet essai contient trois couches de convolution, deux de Max Pooling et deux couches de Fully Connected, où la première couche de convolution contient 16 filtres 3 × 3 et de stride de 1, la deuxième est de 32 filtres 3 × 3 le stride 1, la dernière couche contient 64 filtres de 3 × 3 et de stride 1.

La première couche de Max Pooling est de 4 × 4 le stride 2, la deuxième est de 2 × 2 le stride 1.

La première couche de Fully Connected contient 120 neurones et la dernière couche qui est la couche de sortie contient 33 classes.

La précision que nous avons obtenue dans cet essai est 94.15% qui a duré 8min 40sec.

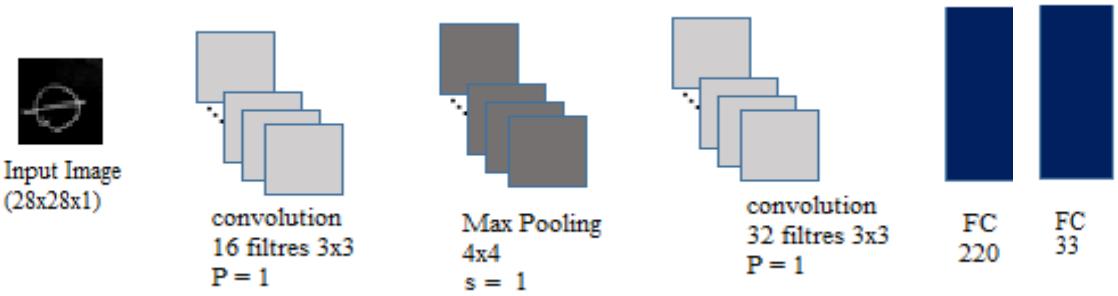


FigureIII.8 Architecture du modèle **Test 7**

Test 8

Dans cet essai nous avons utilisé une structure qui contient deux couches de convolution, une de Max Pooling et deux couches de Fully Connected, où la première couche de convolution contient 16 filtres 3 × 3 et de Padding de 1, le deuxième est de 32 filtres 3 × 3 le Padding 1. La couche de Max Pooling est de 4 × 4 et le stride 1. La première couche de Fully Connected contient 220 neurones et la dernière couche qui est la couche de sortie contient 33 classes.

La précision que nous avons obtenue dans cet essai est 85.27%qui a duré 4min 30sec.

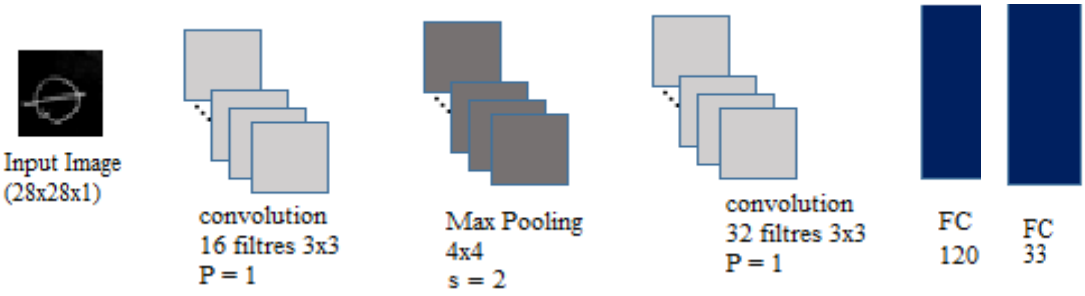


FigureIII.9 Architecture du modèle Test 8

Test 9

Ici notre structure contient deux couches de convolution, une de Max Pooling et deux couches de Fully Connected, la première couche de convolution contient 16 filtres 3×3 et de Padding de 1, le deuxième est de 32 filtres 3×3 le Padding 1. La couche de Max Pooling est de 4×4, le stride est 2. La première couche de Fully Connected contient 120 neurones et la dernière couche qui est la couche de sortie contient 33 classes.

La précision que nous avons obtenue dans cet essai est 61,52% qui ont duré 10min 19sec.

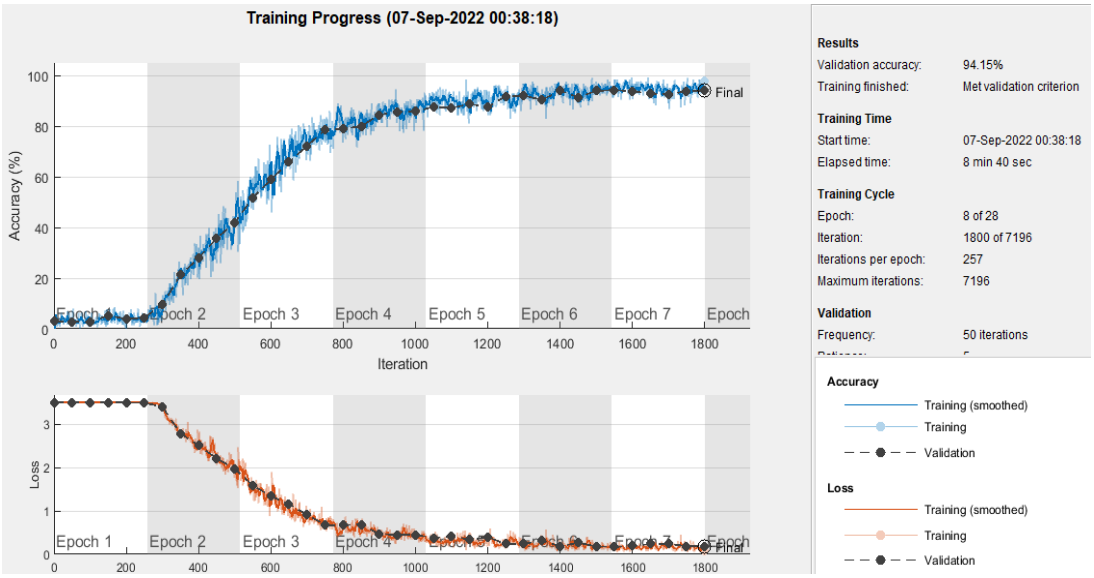


FigureIII.10 Architecture du modèle Test 9

Le tableau suivant présente les résultats de chaque structure

Numéro de réglage	La précision	Temps
Test 7	94.15%	8min 40sec
Test 8	85.27%	4min 30sec
Test 9	61,52%	10min 19sec

TableauIII.4 précision de quelques essaies pour la base BBDS



FigureIII.11 Progression de l'apprentissage du modèle Test 7

Commentaire :

La Figure III.7 montre que l'apprentissage a pris un temps total de 8 min et 40 sec et le modèle a atteint une précision de 94.15%. Nous remarquons d'après le teste 1 et 2 que plus le nombre de couches (convolution et max pooling) élevé plus la précision est elevé, d'après le Teste 2 et 3 on remarque que la précision augmente avec l'augmentation du nombre de neurones.

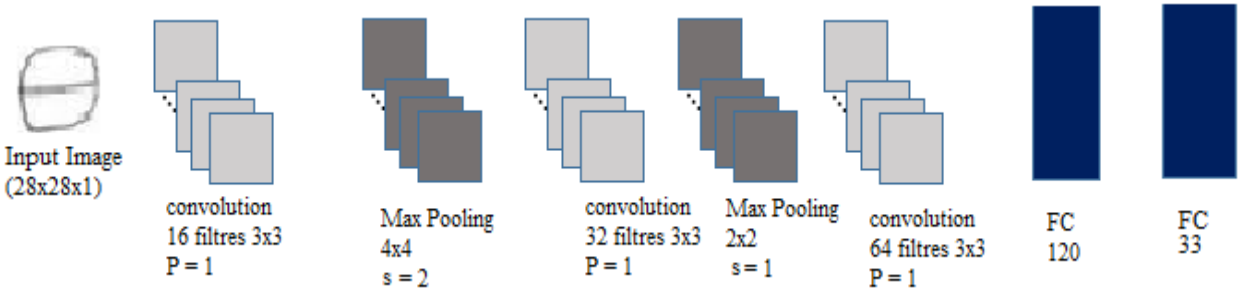
III.3.2.2 Changement de structure pour la AMHCD

Dans cette section nous allons prendre quelques tests afin de pouvoir trouver la meilleure structure :

Test 10

Dans cette section nous avons changé de la base de données, mais nous avons gardé la même structure que Test 7.

La précision que nous avons obtenue dans cet essai est 98.37% qui a duré 7min 30sec.

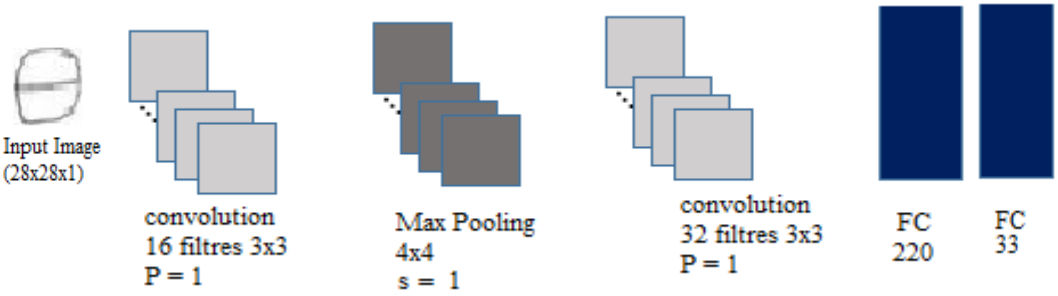


FigureIII.12 Architecture du modèle Test 10

Test 11

Nous gardons la même base AMHCD et la même structure que le Test 8.

La précision que nous avons obtenue dans cet essai est 97.83% qui a duré 3min 40sec.

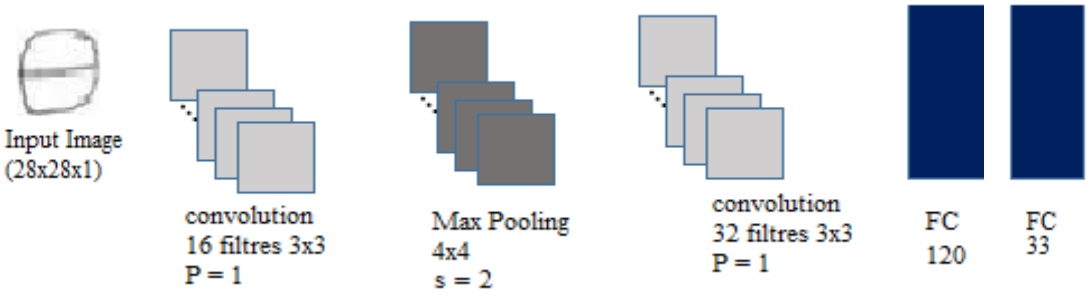


FigureIII.13 Architecture du modèle Test 11

Test 12 :

Le dernier test contient deux couches de convolution et une couche de Max Pooling et deux de Fully Connected, la première couche de convolution contient 16 filtres de 3 × 3 et de Padding 1 et la deuxième est de 32 filtres de 3 × 3 le Padding est 1. La couche de Max Pooling est de 4 × 4 et le stride 2. La couche Fully Connected contient 120 neurones et l'autre de 33 classes.

Nous avons obtenu une précision de 98.27% avec une durée de 3min 40sec



FigureIII.14 Architecture du modèle Test 12

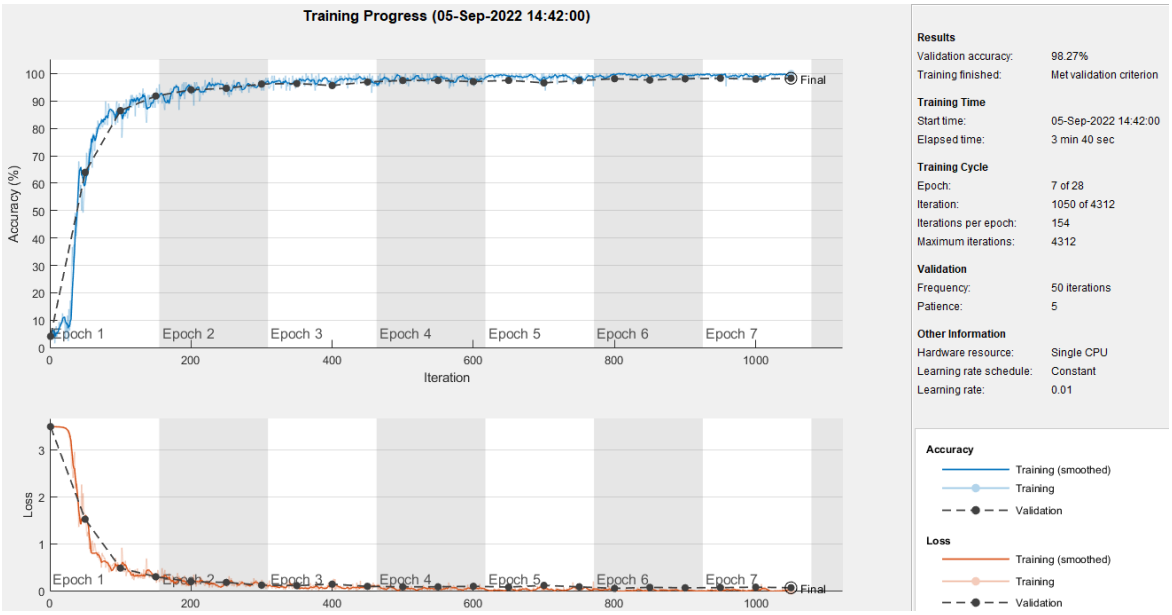
Le tableau suivant présente les résultats de chaque structure

Numéro du test	La précision	Temps
Test 10	98.03%	6min 29sec
Test 11	97.83%	8min 46sec
Test 12	98.27%	3min 40sec

TableauIII.5 précision de quelques essaies pour la base AMHCD

Commentaire :

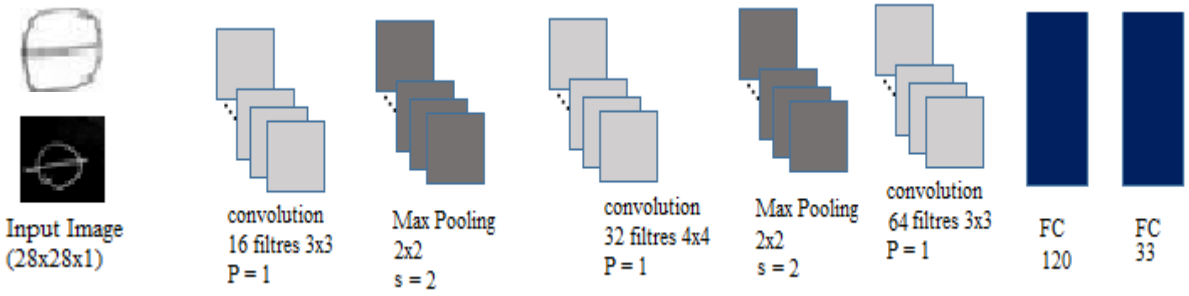
Après l'exécution de ces trois tests, on remarque que le meilleur résultat obtenu est dans le modèle de 3 couches de convolution ,2 couches de pooling et 2 couches de fully connected donc nous concluons que le nombre de couches (filtres, stride.) a une grande influence sur l'apprentissage tout changement de ce paramètre aura un impact sur les résultats finals présentés par une précision.



FigureIII.15 Progression de l'apprentissage du modèle Test 12

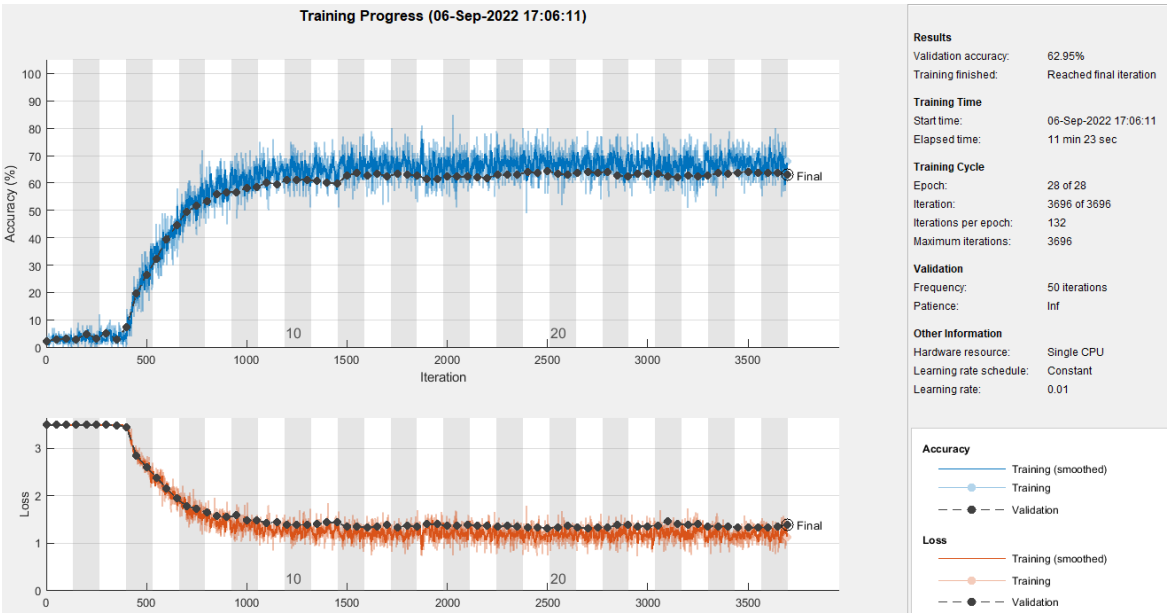
III.3.2.3 Structure inspirée de l'architecture LeNet_5

Cette architecture est proposée par LeCun et al en 1998 dans le but de reconnaître les numéros manuscrits. Dans cette partie nous avons inspiré et fait un test avec cette l'architecture, avec les deux bases, cette architecture contient 3 couches de convolution et 2 couches de Max Pooling de 2×2, et 2 couches de Fully Connected, une de 120 et l'autre de 33. Après la simulation nous avons eu une précision de 62.95%, le test a duré 11min 23sec pour la base BBDS, et pour la AMHCD nous avons eu 98.77%, le test a duré 16min 25sec [16].

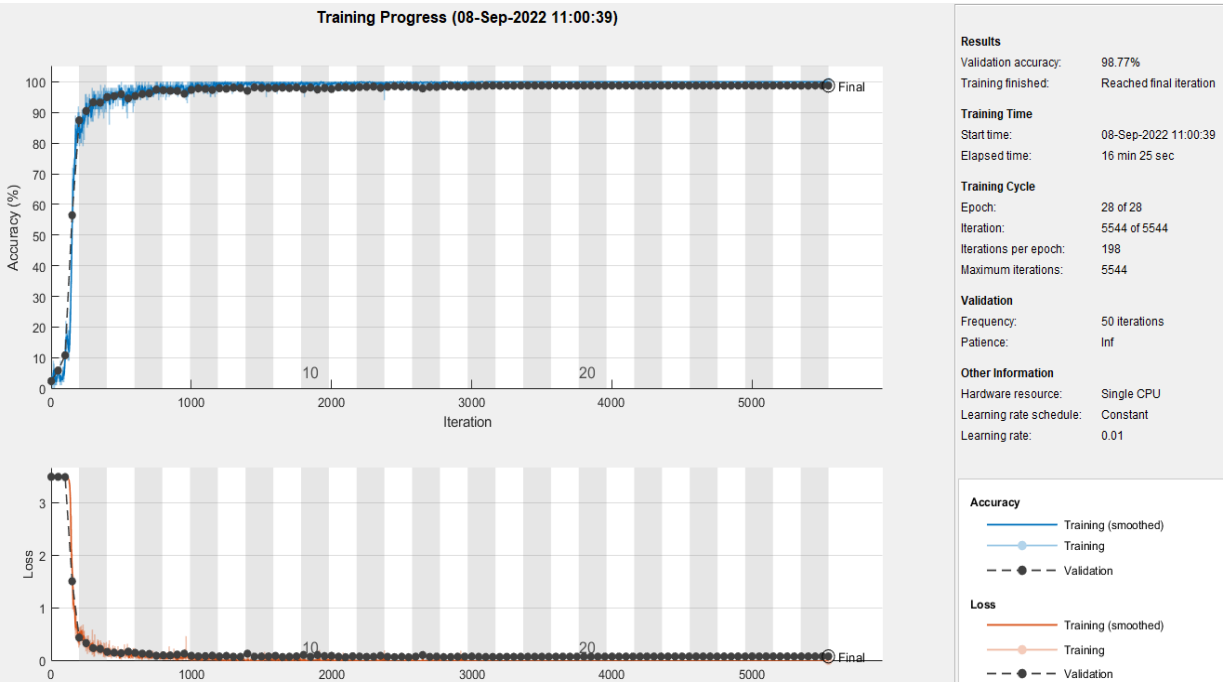


FigureIII.16 l'architecture de réseau LeNet_5

Les figures ces dessus montrent la précision que nous avons obtenue avec les deux bases :



FigureIII.17 Progression de l'apprentissage de LeNet_5 pour la BBDS



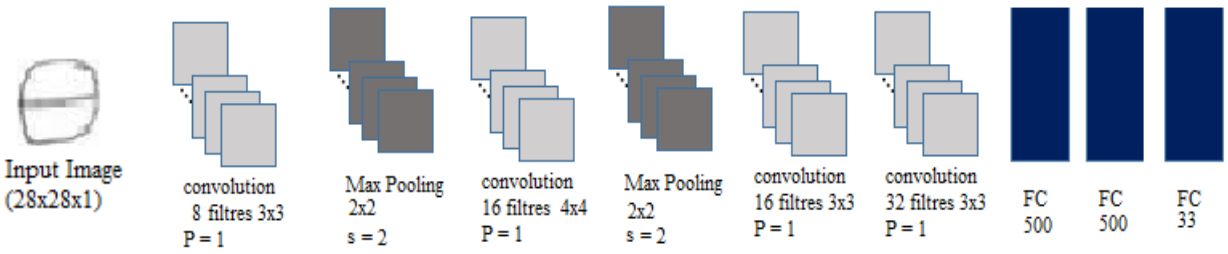
FiguireIII.18 Progression de l'apprentissage de LeNet_5 pour la AMHCD

Commentaire :

D'après les résultats obtenus, nous remarquons que la meilleure précision est celle que nous avons obtenue avec la base AMHCD, et ceci se justifie par le petit nombre d'images réelles que la base AMHCD contient et qui se ressemble trop avec un seul fond et cela permet au réseau de neurones de prendre un peu de temps pour comprendre, par contre la base BBDS contient des images bruitées avec toutes les catégories (petit, grand) et toutes les couleurs.

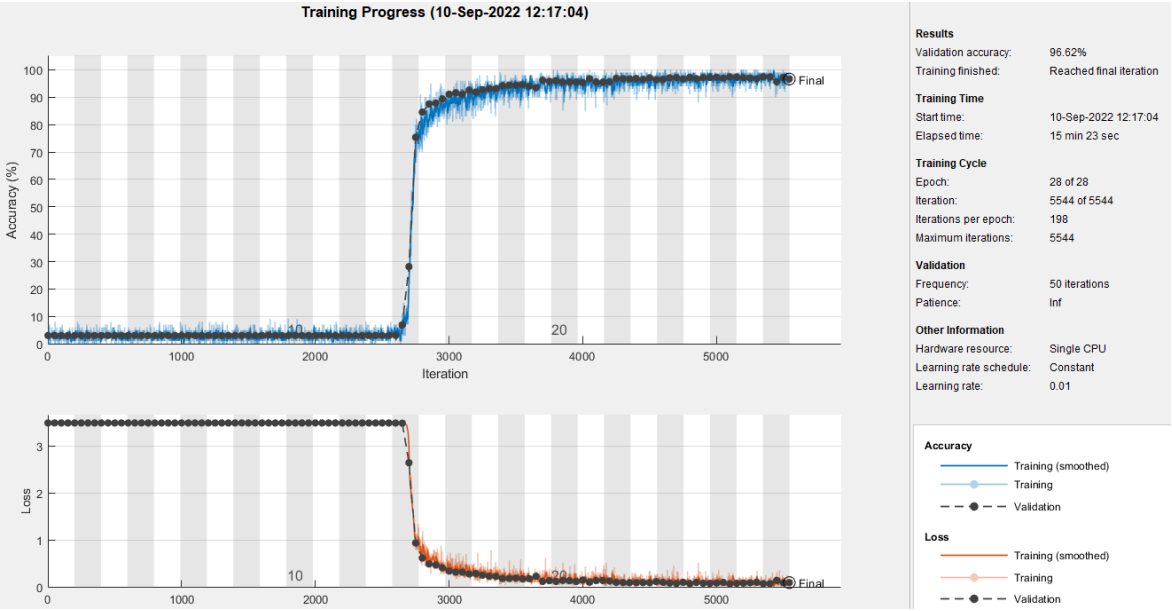
III.3.2.4 Structure inspirée d'Alex_Net

Dans cette partie nous avons fait un test avec l'architecture Alex_Net avec la base AMHCD, cette architecture contient 4 couches de convolution et 2 couches de Max Pooling de 2×2 avec Stride = 2, et 3 couches de Fully Connected, deux de 500 neurones et l'autre contient une fonction softmax qui a 33 sorties. Après la simulation nous avons eu une précision de 95.62%, le test a duré 15min et 23sec [17].



FigureIII.19 l'architecture de réseau AlexNet

La figure ces dessus montre la précision que nous avons obtenue :



FigureIII.20 Progression de l'apprentissage de AlexNet

Commentaire :

FigureIII.20 montre que le programme mit beaucoup de temps pour converger et ça à cause de la complexité de réseau de neurone.

Conclusion des tests

D'après tous ces tests, nous remarquons que les réseaux entrainés par la base AMHCD convergent mieux que ceux entrainés par BBDS, donc les réseaux de neurones profonds apprennent plus vite de la base AMHCD. Cette dernière n'est pas volumineuse et ses images ne sont pas bruitées. Par contre, la base BBDS est plus volumineuse et la diversité de son contenu contraint le réseau de neurones. Ce signifie que pour l'application réelle, la base BBDS est plus recommandée. C'est ce que nous allons présenter dans la section suivante.

III.4 mise en œuvre pratique

Après plusieurs tests, nous avons choisi la meilleure structure et les meilleurs paramètres. La structure choisie sera utilisée pour faire la reconnaissance en temps réel des caractères manuscrits. Cette section est consacrée à la réalisation d'une application sur LabView.

III.4.1 Architecture du réseau de neurones convolutif utilise

Nous présentons dans cette section l'architecture choisie. Elle contient une couche d'entrée acceptant des données de 28×28 avec une seule chaine de couleur (niveau de gris), trois couche de convolution, deux couches Fully connected et enfin une couche de sortie Softmax avec 33 sorties (33 classes).

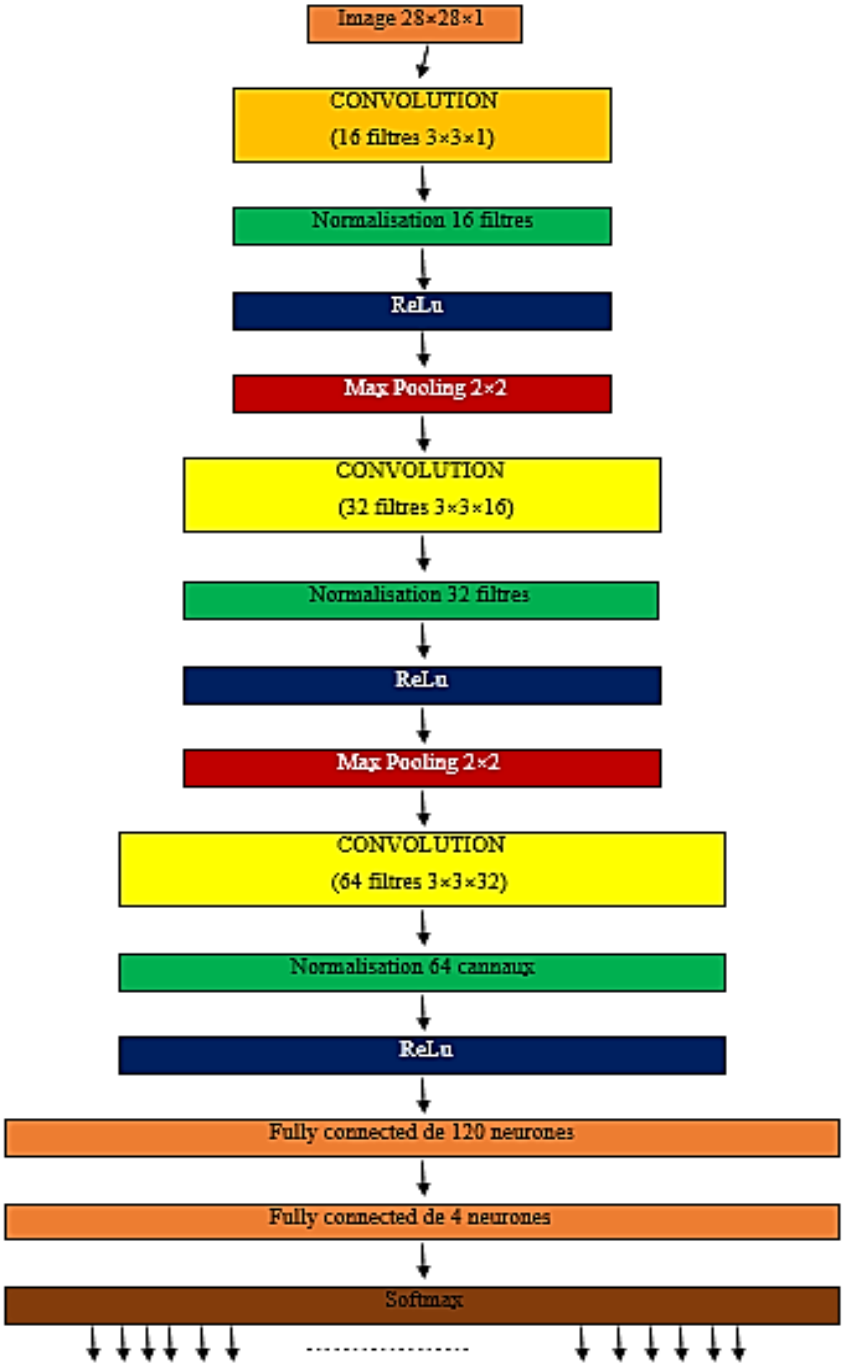


Figure III.21 L'architecture du CNN choisie pour la reconnaissance des caractères manuscrits

Cette architecture est implémentée sous Matlab sous type Layer :

```
function layers = setlayer()
layers = [ ...
      imageInputLayer([28 28 1])
      convolution2dLayer(3,16,'padding',1)
      batchNormalizationLayer
      reluLayer

      maxPooling2dLayer(2,'Stride',2)

      convolution2dLayer(3,32,'padding',1)
      batchNormalizationLayer
      reluLayer

      maxPooling2dLayer(2, 'Stride', 2)

      convolution2dLayer(3,64,'padding',1)
      batchNormalizationLayer
      reluLayer

      fullyConnectedLayer(120)
      batchNormalizationLayer
      reluLayer

      fullyConnectedLayer(33)
      softmaxLayer
      classificationLayer ];
end
```

Listing III.2 Implémentation de l'architecture CNN sous MATLAB

III.4.2 Implémentation de l'application sur LabView

LabView propose une variété de fonctionnalités et d'outils allant des assistants interactifs aux interfaces configurable définies par l'utilisateur, il se différencie par son langage de programmation graphique à usage générale appelé G qui est effectué en câblant des icônes graphiques sur un diagramme, qui est ensuite compilé directement en code machine afin que les processeurs informatiques puissent l'exécuter bien que représenter graphiquement au lieu de texte.

Le code G est généralement plus facile et plus simple à comprendre dans un diagramme LabView, il a tous les outils, plus facile à utiliser, il est connecté à l'entrée USB de notre appareil qui lui permet d'utiliser une caméra.

Un programme LABVIEW est appelé « instrument virtuel » (VI).

LABVIEW permet de réaliser des interfaces graphiques personnalisées et conviviales, et ainsi de créer un instrument virtuel spécifique à vos besoins.

Un programme LabVIEW comporte 2 éléments principaux :

- Une face-avant
- Un diagramme

III.4.2.1 Interface graphique

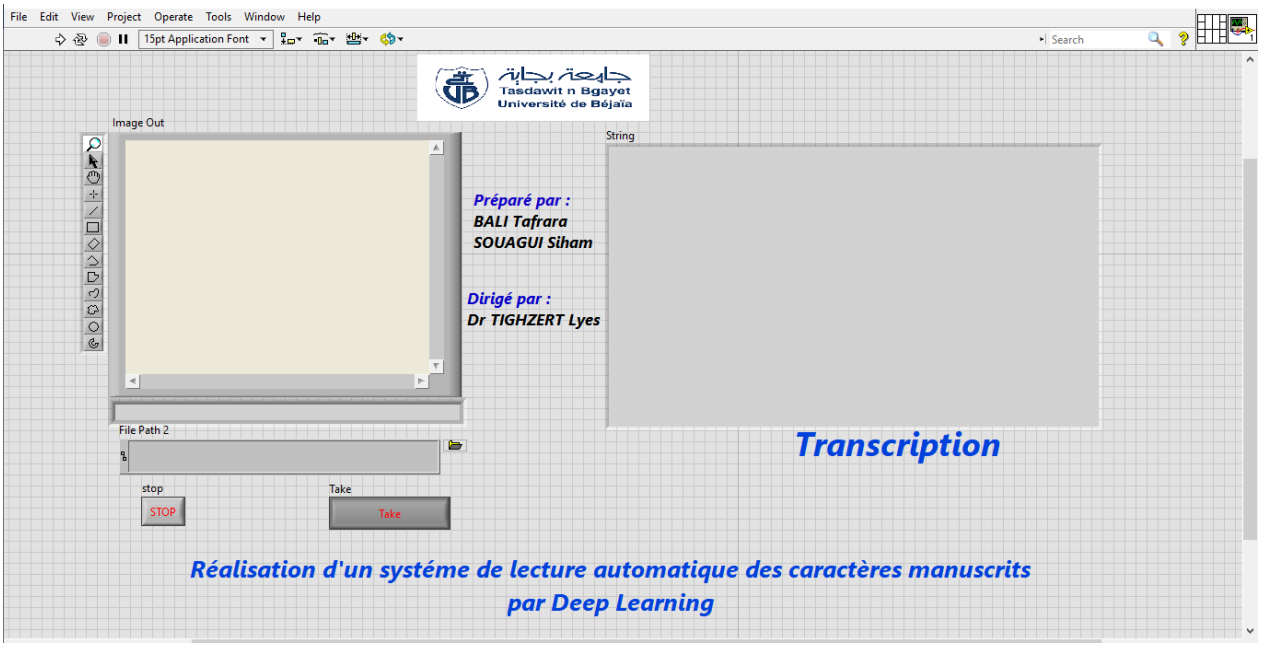
Une face avant du programme est l'interface utilisateur du VI contenants des entrées (les commandes) et des sorties (les indicateurs) du programme. Les commandes et indicateurs sont : image out, string, take, stop.

Image out : affichage l'image capturée.

String : affichage la transcription.

Take : un bouton poussoir pour prendre une capture.

Stop : un poussoir qui sert à arrêter le programme.

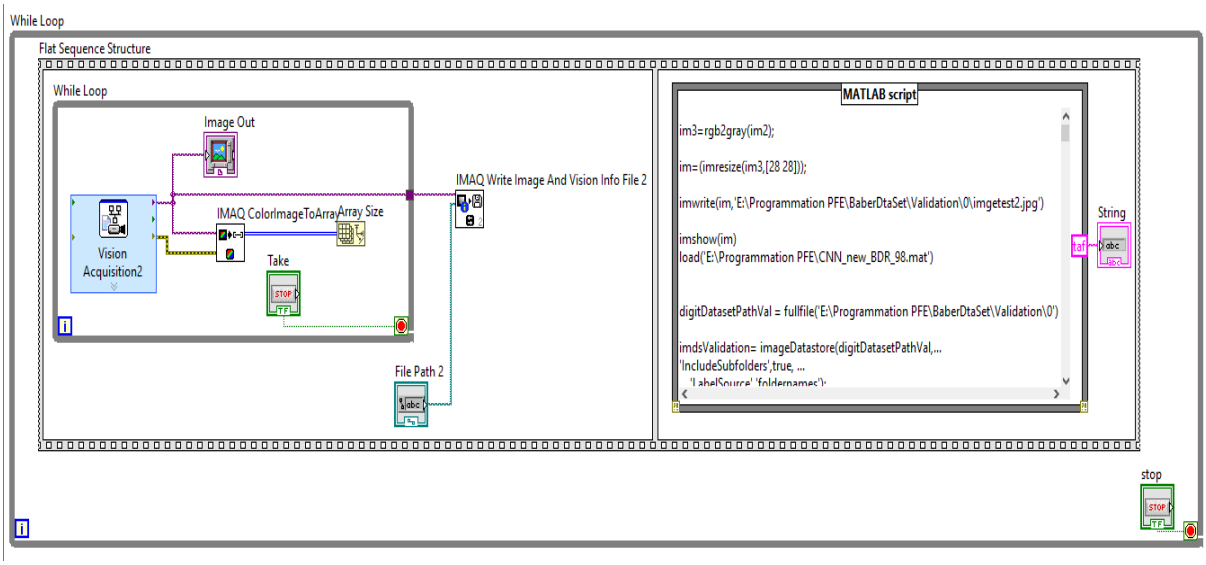


FigureIII.22 La face-avant du programme

III.4.2.2 Code diagramme LabView

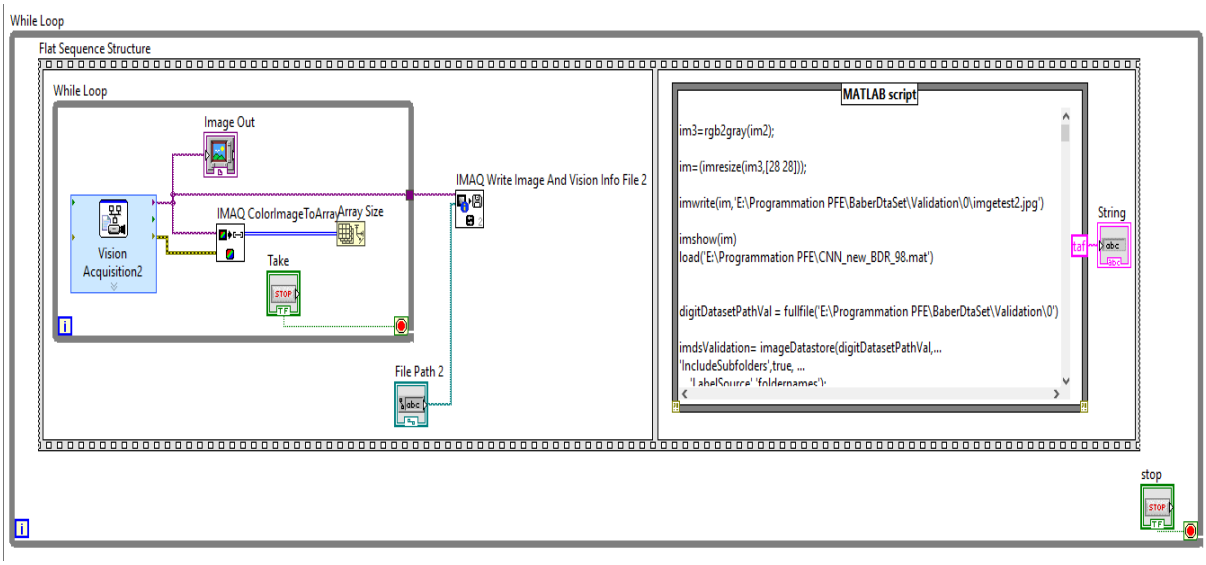
Contient le code graphique du programme LabVIEW (VI).

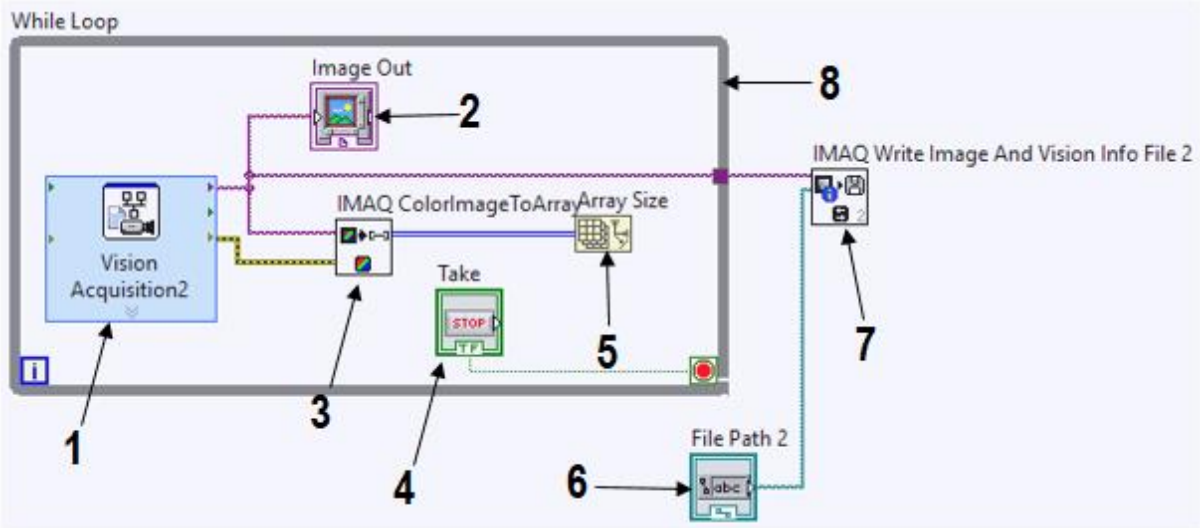
La programmation est graphique selon une logique de flux de données



FigureIII.23 Réalisation d'un diagramme sur LabView

III.4.2.3 Exécution en code G





FigureIII.24 l'exécution en code G sur LabView

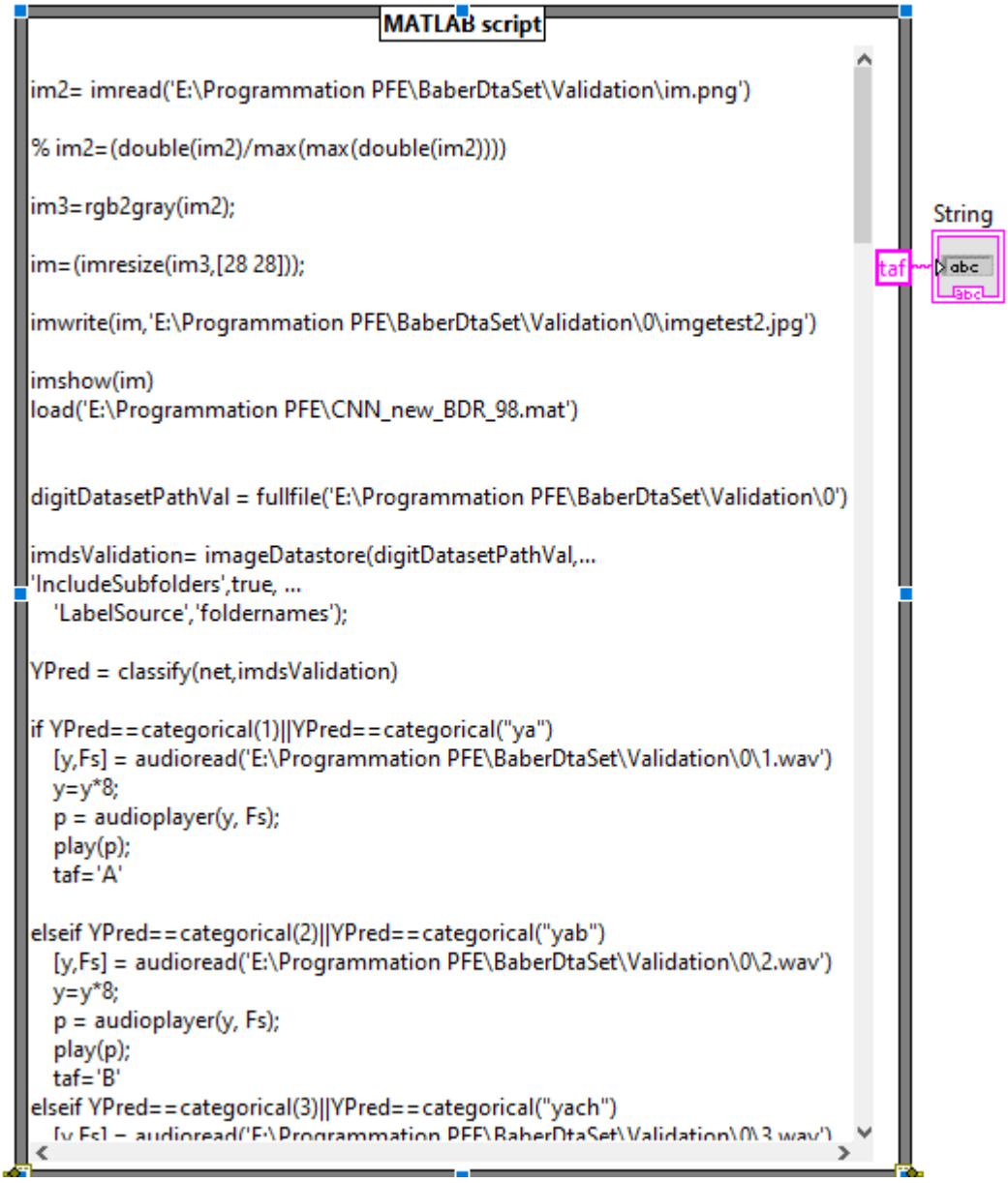
Numéro de fonction	Le nom de la fonction	Définition
1	**Vision aquisition2**	crée et édite l'acquisition à l'aide du VI Express Acquisition de vision NI.
2	**Image out**	Image out est la référence à l'image capturée.
3	**IMAQ ColorImage To Array**	Extrait les pixels d'une image couleur ou d'une partie d'une image coulée dans un tableau 2D.
4	**Take**	Bouton pour prendre une image.
5	**Array Size**	Renvoie le nombre d'éléments dans chaque dimension du tableau.
6	**File Path2**	C'est le chemin d'accès complet, y compris le lecteur la répertoire et le nom du fichier lu
7	**IMAQ Write Image And Vision Info File2**	Ecrit une image avec des informations de vision supplémentaires associées à l'image ,dans un fichier PNG.
8	**While Loop**	Répète le code dans un sous- diagramme jusqu' à ce qu'une condition spécifique se produise. Une boucle while s'exécute toujours au moins une fois.

TableauIII.6 Définition pour les fonctions utilisées

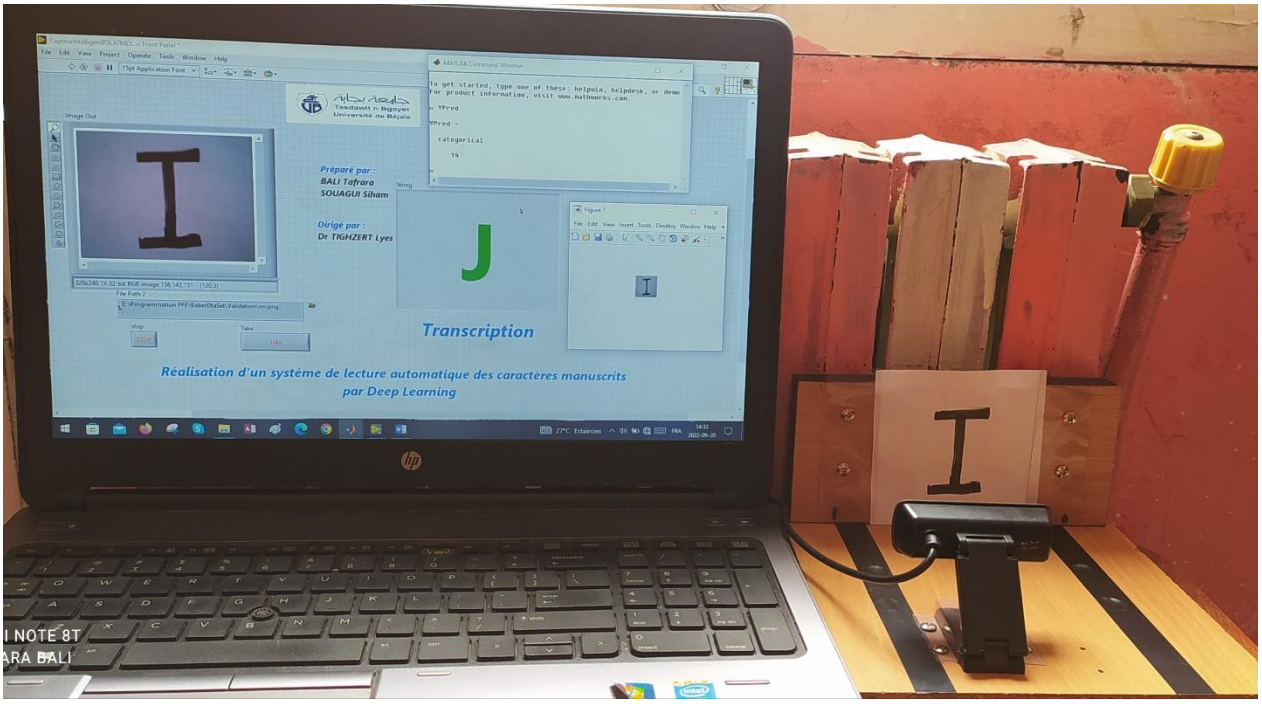
Algorithme2 : Pseudocode du programme labVIEW réalisé

```
While stop = =0 do
    Flat Sequence
        Sequence 1
            While Take = 0
                Acquisition d'image
                Image out (Afficher l'image)
                Convertir l'image en tableau
                Calculer sa dimension
            End
            Enregistrer l'image (file_path)
        End Sequence1
        Sequence2
            Call Matlab do
                Télécharger l'image enregistrée
```

Convertir en niveau de gris
La classe par le réseau de neurone
Prononciation de la lettre
End Call Matlab
Afficher la transcription
End Sequence2
End Flat Sequence
End while stop

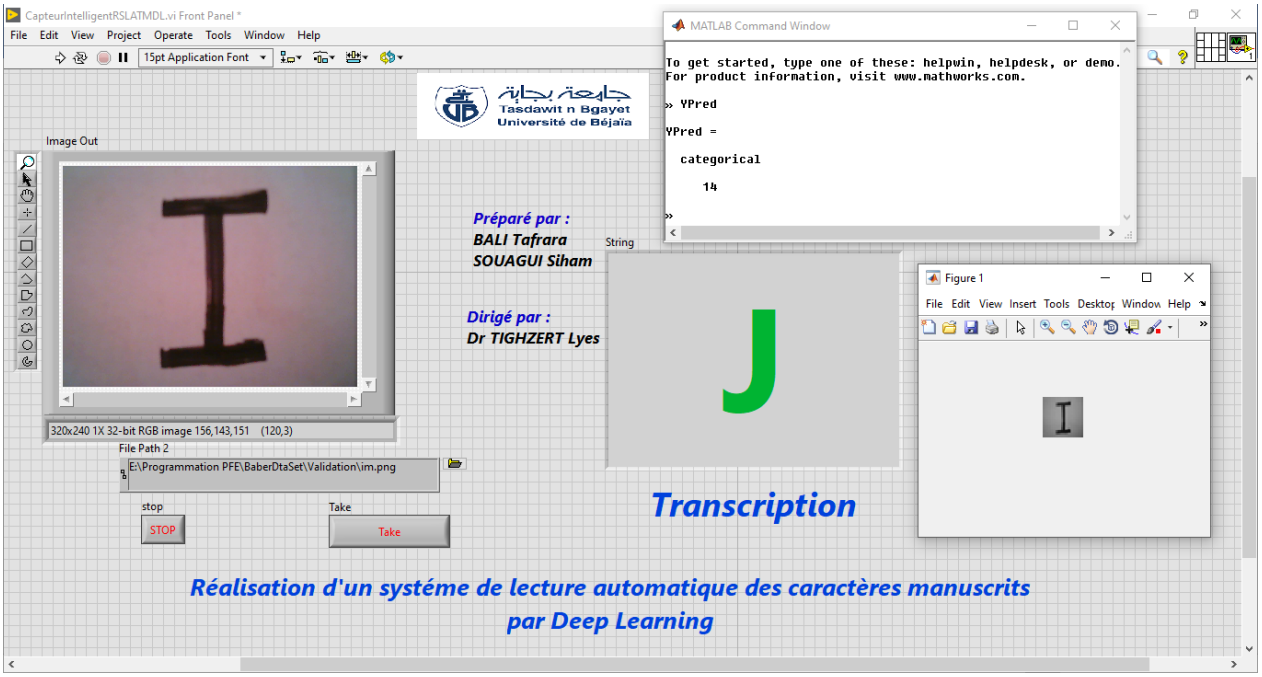


FigureIII.25 MATLAB Script node



FigureIII.26 Image du système réel

Notre étude est concentrée sur la reconnaissance des caractères manuscrits, donc pour tester notre système nous devons mettre une lettre manuscrite en face la caméra, et lancer le programme manuellement en cliquant sur Take pour prendre une capture, après la reconnaissance notre programme affiche la lettre transcrire et la prononcer, pour effectuer cet essai nous avons utilisé une caméra et un PC HP-i5.



FigureIII.27 une capture d'écran pour l'interface graphique

Pour la mise en œuvre réelle, les réseaux entrainés par AMHCD ne donnent pas de bons résultats ; Le réseau utilisé dans l'application réelle est entrainé sur la base de données BBDS.

Matrice de confusion

III.5 Conclusion

Ce chapitre est consacré à la réalisation pratique de notre système et faire plusieurs tests pour trouver le réseau idéal.

Nous avons utilisé deux bases de données AMHCD et BerbèreDataSet (BBDS) pour l'apprentissage du réseau de neurones. Pour trouver la bonne structure du CNN, nous avons divisé les tests en quatre parties, changement des paramètres d'apprentissage, changement de structure, structure inspiré de LeNet, structure inspiré d'Alex Net. L'étude comparative nous a permis de trouver les meilleurs paramètres, le réseau choisi est entrainé sur la BBDS et il est utilisé pour faire la reconnaissance en temps réel des caractères manuscrits à l'aide d'une caméra sur l'application LabView et les résultats étaient satisfaisants.

Conclusion générale

Ce mémoire est un projet de fin de cycle de master réalisé à l'université de Bejaïa Abderrahmane Mira. Ce travail est consacré à la réalisation d'un système de reconnaissance automatique des caractères de la langue berbères manuscrits par Deep Learning. Ce projet nous a permet de faire appel au Deep Learning, et en particulier aux réseaux de neurones de convolution. Ces derniers démontrent de nos jours et dans beaucoup de travaux leurs efficacités dans la reconnaissance des images. Pour l'apprentissage, nous avons utilisé la méthode de la Descente de Gradient avec Momentum SGDM, les réseaux CNN et deux bases de données BBDS et AMHCD.

Le premier chapitre est consacré à la présentation des généralités et enrichir nos connaissances sur les réseaux de neurones, le machine Learning et le Deep Learning. Il nous a permis d'approfondir nos connaissances en apprentissage des réseaux de neurones artificiels profonds.

Le deuxième chapitre est dédié à la présentation détaillée des réseaux de neurones convolutifs (ConvNet ou CNN). Nous avons présenté les différentes couches rencontrées en Deep Learning, les fonctions d'activation, la propagation directe et la rétro propagation. A la fin, nous avons exploré les différentes architectures les plus populaires et les plus largement utilisées en Deeplearning.

Le troisième chapitre est la partie la plus importante dans ce mémoire, il est consacré à la réalisation pratique et faire plusieurs tests afin de trouver les meilleurs paramètres puis faire des autres tests pour trouver la bonne structure. Le réseau est entrainé sur un PC portale HP-i5, ensuit nous avons choisi le meilleur réseau et l'utiliser pour la reconnaissance des lettres manuscrits, les essais sont effectués à l'aide d'une caméra et d'une application sur LabView

Les résultats obtenus dans ce document sont satisfaisants, par conséquent, l'objectif principal fixé dans la problématique est atteint, l'apprentissage du réseau de neurones convolutif choisi est accompli avec 98.00% de précision du modèle. Son fonctionnement en temps réel est aussi

satisfaisant. Les réseaux entrainés par BBDS présentent des résultats plus compétitifs que ceux entrainés par AMHCD quand il s'agit d'image réelle.

En perspective, notre système peut être amélioré par plusieurs moyens à savoir :

1. Utilisation d'une base de données plus riche en quantité et en qualité ou combiner les deux bases BBDS et AMHCD.
2. Explorer d'autres architectures neuronales.
3. Améliorer le programme en rajoutant une phase de segmentation pour reconnaitre des paragraphes entiers.

Bibliographie

[1] Balacheff, N., 1994. Didactique et intelligence artificielle. *Recherches en didactique des mathématiques , 14* , pp.9-42.

[2] Visetti, YM, 1991. Des systèmes experts aux systèmes à base de connaissances : à la recherche d'un nouveau schéma régulateur. *Intellectica , 12* (2), pp.221-279.

[3] Alloula, K., 2007. *Modèle de coopération entre calcul formel et calcul numérique pour la simulation et l'optimisation des systèmes* (Thèse de doctorat).

[4] Mitchell, T., Buchanan, B., DeJong, G., Dietterich, T., Rosenbloom, P. et Waibel, A., 1990. Apprentissage automatique. *Revue annuelle d'informatique , 4* (1), pp.417-433.

[5] Kennedy, J., 2006. Intelligence d'essaim. Dans *Manuel d'informatique inspirée de la nature et innovante* (pp. 187-219). Springer, Boston, MA.

[6] Bougrine, A., Ledee, R., Canals, R., Harba, R. et Jabloun, M., Segmentation d'images thermiques de la voûte plantaire par Deep Learning.

[7] Berthelier, A., Yan, Y., Chateau, T., Blanc, C., Duffner, S. et Garcia, C., 2020, juin. Compression de réseaux convolutifs par utilisation d'un terme de clarté l1/l2 sur les noyaux. Dans *RFIAP* .

[8] Cleuziou, G., 2004. *Une méthode de classification non-supervisée pour l'apprentissage de règles et la recherche d'information* (Doctoral dissertation, Université d'Orléans).

[9] Loesdau, M., 2016. *Towards a Computer Vision Based Quality Assessment of Tahitian Pearls* (Thèse de doctorat, Université de la Polynésie française).

[10] Bouafia, N., 2020. *Classification efficace des vêtements de mode basée sur les approches : apprentissage automatique ML et apprentissage profond DL* (Doctorat, FACULTE MATHEMATIQUES ET INFORMATIQUE-DEPARTEMENT INFORMATIQUE-OPTION : Informatique Décisionnelle Et Optimisation).

[11] Hainaut, JL, 2009. Bases de données. *Concepts, utilisation et développement, Vottem, Belgique, Dunod* .

[12] Khan, S., Rahmani, H., Shah, S.A.A. and Bennamoun, M., 2018. A guide to convolutional neural networks for computer vision. *Synthesis lectures on computer vision*, *8*(1), pp.1-207

[13] Menasria, A. et Zemouli, S., 2016. *Reconnaissance hors ligne des chiffres manuscrits isolés (Base de donnée MNIST)* (Doctorat, Universite laarbi tebessi tebessa).

[14] Hassairi, S., Ejbali, R. et Zaied, M., 2015, décembre. Un réseau d'ondelettes neuronales à convolution profonde pour la classification supervisée d'images de lettres arabes. En *2015, 15e Conférence internationale sur la conception et les applications de systèmes intelligents (ISDA)* (pp. 207-212). IEEE.

[15] Par, BBBV, Parameter, C., Latitude, E., Magnetic-Field, HLIP, Baume, B., Beatus, B. et Belgium, BB, 2001. A. Heck.

[16] Haidar, A., Fakir, M. et Bencharef, O., 2012. Hybridation des modèles de Markov cachés et de la logique floue pour la reconnaissance des caractères Tifinagh manuscrits. Dans *5ème conférence internationale sur les TIC pour l'"amazighe*.

[17] Sarraf, S. et Tofighi, G., 2016. Classification des données d'IRM structurelles de la maladie d'Alzheimer par réseaux de neurones convolutifs à apprentissage profond. *préimpression arXiv arXiv:1607.06583* .

[18] Singh, I., Goyal, G. et Chandel, A., 2022. Réseau neuronal convolutif basé sur l'architecture AlexNet pour la classification des commentaires toxiques. *Journal de l'Université King Saud - Informatique et sciences de l'information.*

Résumé

Dans notre travail, nous avons utilisé le Deep Learning, plus précisément, les réseaux des neurones convolutionnels (CNN) pour la reconnaissance des caractères manuscrits de la langue Berbère. Les CNN sont des réseaux de neuronaux multicouches spécifiquement conçus pour les tâches de reconnaissance. Dans ce projet nous avons réalisé un système de lecture des caractères manuscrits par Deep Learning. Nous avons fait plusieurs tests pour choisir les meilleurs paramètres et la bonne structure, d'après une étude comparative nous avons choisi la bonne structure. Ensuite nous avons passé à la mise en œuvre réelle, le réseau utilisé dans l'application réelle est entrainé sur la base de données BBDS. Pour effectuer les tests en temps réels nous avons créé une application sur LabView et à l'aide d'une caméra notre système a pu identifier les lettres tifinagh sans difficulté.

Mot clé : Intelligence Artificielle, Machine Learning, Deep Learning, Réseaux de neurones convolutionnels (CNN), caractères manuscrits, Base de données, LabVIEW.

في عملنا، استخدمنا التعلم العميق، وبشكل أكثر دقة، الشبكات العصبية التلافيفية (CNN) للتعرف على الأحرف المكتوبة بخط اليد للغة البربرية. شبكات CNN عبارة عن شبكات عصبية متعددة الطبقات مصممة خصيصًا لمهام التعرف. في هذا المشروع، أدركنا نظامًا لقراءة الأحرف المكتوبة بخط اليد من خلال التعلم العميق. قمنا بإجراء عدة اختبارات لاختيار أفضل المعايير والبنية الجيدة، وفقًا لدراسة مقارنة اخترنا الهيكل الجيد. ثم انتقلنا إلى التنفيذ الحقيقي، حيث تم تدريب الشبكة المستخدمة في التطبيق الحقيقي على قاعدة بيانات BBDS. لإجراء الاختبارات في الوقت الفعلي، أنشأنا تطبيقًا على LabView وباستخدام الكاميرا، تمكن نظامنا من التعرف على أحرف Tifinagh دون صعوبة.

الكلمات الرئيسية: الذكاء الاصطناعي، التعلم الآلي، التعلم العميق، الشبكات العصبية التلافيفية (CNN)، الأحرف المكتوبة بخط اليد، قاعدة البيانات، LabVIEW

In our work, we used Deep Learning, more precisely, convolutional neural networks (CNN) for the recognition of handwritten characters of the Berber language. CNNs are multi-layered neural networks specifically designed for recognition tasks. In this project we realized a system for reading handwritten characters by Deep Learning. We made several tests to choose the best parameters and the good structure according to a comparative study, we chose the good structure. Then we moved to the real implementation, the network used in the real application is trained on the BBDS database. To perform the tests in real time we created an application on LabView and using a camera our system was able to identify the Tifinagh letters without difficulty.

Keyword: Artificial Intelligence, Machine Learning, Deep Learning, Convolutional Neural Networks (CNN), handwritten characters, Database, LabVIEW