République Algérienne Démocratique et Populaire
Ministère de l'Enseignement et de la Recherche Scientifique

UNIVERSITE AMAR TELIDJI
LAGHOUAT

FACULTE DES SCIENCES ET DE L'INGENIERIE

DEPARTEMENT DE GENIE INFORMATIQUE

PROJET DE FIN D'ETUDES
Pour l'Obtention Du Diplôme

D'INGENIEUR D'ETAT EN INFORMATIQUE

option : intelligence artificielle

Thème

Le filtrage d'images par les réseaux de neurones d'ordre supérieur

Présenté Par :

1- Bensalem Boulerbah

2- Dada Mustapha

Encadré Par :

Mr Lagraa Nacer Elddine

N° d'ordre **: …../2003-PFE/DGI**

Couche l

Remerciements

Nous exprimons nos sincères remerciements à notre promoteur Mr Nacer Eddine Lagraa, pour avoir bien voulu nous aider à réaliser ce travail. Ses critiques et ses conseils nous ont été d'une aide très précieuse.

Nos sincères remerciements vont aux membres du jury :

Dr Ouinten Youcef et Mr Tahari Abdel Karim

Nous exprimons un remerciement particulier à nos parents qui nous ont aidé et soutenu patiemment.

Enfin, nos remerciements vont à toute personne qui nous a aidé, de prés ou de loin, à la réalisation de ce travail.

Table des figures

1. Première partie Théorique

Chapitre I : Introduction aux réseaux de neurones

Chapitre II : Les réseaux de neurones multi couche

Chapitre III : Le filtrage d'image

2. deuxième partie Pratique

I. Simulation

Sommaire

1. Première partie Théorique

Chapitre I : Introduction aux réseaux de neurones

Chapitre II: Les réseaux de neurones multi couche

Chapitre III : Le filtrage des images

2. deuxième partie Pratique

I. Simulation

Introduction

Le traitement d'image est apparu dans les années 60.Depuis, plusieurs applications ont déjà été développées dans différents domaines, comme le militaire, l'aérospatiale, le médical, l'industrie ainsi que la robotique.

Le traitement d'images est principalement séparé en deux catégories.

Une première, nommée le traitement d'images bas niveau, qui consiste à préparer une image pour les traitements de niveau supérieur. Elle consiste principalement à améliorer la qualité visuelle d'une image (amélioration du contraste, élimination du bruit, ...) qui a pu être perturbée par les capteurs d'acquisitions, mais peut aussi permettre l'extraction de mesures comme par exemple le contour.

La deuxième étape, nommée traitement d'image haut niveau, permet, avec les données issues de l'étape précédente, d'avoir une analyse « intelligente » de l'image. C'est lors de cette étape que l'on effectue la reconnaissance de formes (objets, caractères, ...) ou de scènes.

Il existe différentes techniques pour effectuer ce type de traitement d'images. Une première consiste à travailler avec l'image dans son domaine fréquentiel, pour se faire, elle utilise les outils issus de la théorie du signal, comme par exemple la transformée de Fourrier. Cette technique est principalement utilisée pour le filtrage des images.

Une autre technique consiste à travailler avec l'image dans son domaine spatial. On travail sur des ensembles de pixels issues de l'image avec des opérateurs de morphologie mathématique, ou en effectuant des opérations avec des masques. Ces types de traitement peuvent par exemple être l'amincissement de contours en utilisant les opérateurs morphologiques de base, l'érosion et la dilatation, ou le filtrage d'image à l'aide de masques permettant par exemple le lissage d'une image.

Une nouvelle technique est apparue. Elle consiste à utiliser l'image comme un exemple ou un ensemble d'exemples qui servent d'entrée à un réseau de neurones artificiels.

Un réseau de neurones est une approche inspirée du système nerveux humain, et les études récentes ont montré que les cellules nerveuses "neurones" sont des cellules relativement simples, et les performances dont se montre capable le système nerveux étant dues aux comportement d'un très grande nombre de ces neurones connectés entre eux.

: Neurone d'entrée
Couche l

Les chercheurs relèvent le défi de construire des circuits électronique imitant la capacité exceptionnelle dont font preuve les neurones.

Les neurones artificiels sont des éléments de base qui vont être assemblés entre eux de manière à construire un système de grande taille capable de résoudre de nombreuses tâches, le choix du réseau pour une tâche est très essentiel pour un système non linéaire en "robotique", les réseaux dynamiques ont donné de meilleurs résultats par rapport aux réseaux statiques.

Dans le traitement d'images, les réseaux multicouches ont donné des résultats satisfaisants.

Dans ce travail, on va comparer deux types de réseaux multicouches qui sont les réseaux de neurone d'ordre un qui est linéaire et les réseaux de neurone d'ordre supérieur qui sont non linéaires.

Cette thèse est divisée en deux parties, une partie théorique qui contient trois chapitres :

Le premier chapitre contient des généralités sur les réseaux de neurones (introduction, point de vue biologique, modèle mathématique) ainsi que les déférents types de neurones (statique, dynamique), dans le deuxième chapitre nous présenterons un algorithme d'apprentissage des réseaux multicouches qui est l'algorithme de rétro propagation basée sur la méthode du gradient.

Dans le troisième chapitre, différentes techniques de filtrage (linéaires et non linéaires) ainsi que le filtrage neuronal (architecture et apprentissage) sont étudiées.

Dans la deuxième partie les algorithmes d'apprentissage des réseaux MLP ont été appliqué tout d'abord sur un exemple de base très utilisé "Xor" puis sur le filtrage d'image, et nous terminerons notre thèse par une conclusion.

.

Chapitre I :

Introduction aux Réseaux de Neurones

I.1 Introduction:

Une manière très simple de concevoir un réseau de neurones est de considérer qu'il s'agit d'un système de traitement de l'information composé d'un grand nombres de processeurs interconnectés (cellules). Chaque cellule calcule sa sortie sur la base des informations reçues des autres cellules qui lui sont connectées et des poids de ces connexions.

L'architecture du réseau est entièrement spécifiée par :

- *Le nombre de cellules (entrée, cachée ou sortie).*
- *La nature des cellules (la fonction d'activation est généralement la même pour toutes les cellules).*

Les parties de ce chapitre qui suivront seront consacrées, dans un premier temps à un petit historique et à quelques notions neurophysiologique, après cela nous présenterons le modèle mathématique et comment les réseaux de neurones font leur traitement ? Pour cela nous avons besoin de données de notions d'apprentissage et de reconnaissance, nous pourrons voire à la fin de ce chapitre qu'il existe deux grands modèles de réseaux de neurones (statique et dynamique).

I.2 Historique:

1943 : ***J. MC Culloch*** et ***W. Pitts*** sont les premiers à montrer que les réseaux de neurones formels (Caractérisés par son état binaire (actif ou inactif) et par les connexions qui le relient à d'autres unités) peuvent réaliser des fonctions logiques, arithmétiques et toute fonction calculable. Ils ont construit une machine capable de reconnaître des formes déduites les unes des autres par des transformations simples.

1943-1982 : des recherches ont été effectuées durant cette période, mais elles étaient moins importantes.

1982 : ***J. J. Hopfield*** est un physicien reconnu, à qui l'on doit le renouveau d'intérêt pour les réseaux de neurones artificiels. Il présente une théorie du fonctionnement et des possibilités des réseaux de neurones**.** Il fixe préalablement le comportement à atteindre pour son modèle et construit à partir de là, la structure et la loi d'apprentissage correspondant au résultat escompté**[2].**

I.3 Le modèle neurophysiologique :

Le neurone est une cellule composée d'un corps cellulaire. Le corps cellulaire se ramifie pour former ce que l'on nomme les dendrites. Celles-ci sont parfois si nombreuses que l'on parle alors de chevelure dendritique ou d'arborisation dendritique. C'est par les dendrites que l'information est acheminée de l'extérieur vers le corps du neurone, figure (I.1).

L'information traitée par le neurone chemine ensuite le long de l'axone pour être transmise aux autres neurones. La transmission entre deux neurones n'est pas directe. En fait, il existe un espace intercellulaire de quelques dizaines d'Angstrom (10^{-9} m) entre l'axone du neurone afférent et les dendrites efférent **[3]**.

Un neurone est composé de :

I.3.1 Le corps cellulaire : qui contient le noyau du neurone et effectue les transformations biochimiques nécessaires à la vie du neurone.

I.3.2 Les dendrites : sont de fins tubes de quelques dixièmes de microns de diamètre et d'une longueur de quelques dizaines de microns. Ce sont des petites branches qui se ramifient à leur extrémité. Elles forment une sorte d'arborescence autour du corps cellulaire. Ce sont elles qui permettent au neurone de capter les signaux qui parviennent de l'extérieur.

I.3.3 L'axone est la fibre nerveuse qui permet de transporter les signaux émis par le neurone. Sa membrane externe possède certaines propriétés, il est plus long que les dendrites (un

millimètre à plus d'un mètre) et se ramifie à son extrémité là où il communique avec les autres neurones.

I.3.4 Les Synapses : Deux neurones sont connectés entre eux dans des endroits qui s'appellent les synapses. Ils ne sont pas directement reliés l'un à l'autre, mais sont séparés par un petit espace, appelé espace synaptique.

Les synapses jouent un rôle fondamental pour permettre aux neurones de communiquer. Certains traitements sont effectués à leurs niveaux.

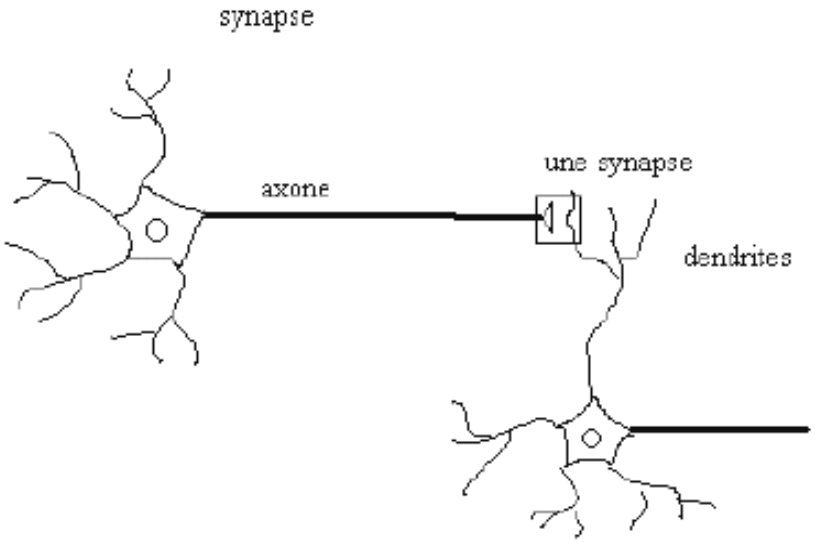


Figure (I.1): *Connexion entre deux neurones.*

I.4 Le modèle mathématique

I.4.1 Le neurone artificiel

Chaque neurone artificiel est un processeur élémentaire, il reçoit un nombre variable d'entrée, a chacune de ces entrées est associé un poids représentatif de la force de la connexion. Chaque neurone est doté d'une sortie unique, qui se ramifie ensuite pour alimenter un nombre variable de neurones aval, figure (I.2)**[1].**

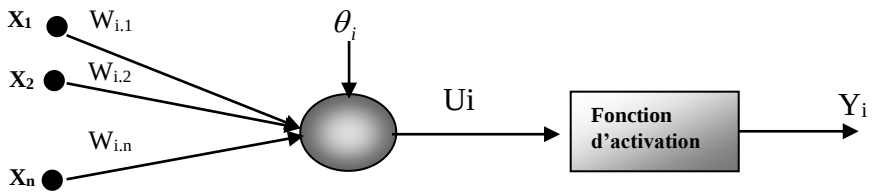


Figure (I.2): *Modèle de base d'un neurone artificiel.*

Le premier modèle de neurone est caractérisé par :

- Les entrées [x_1,x_2,…..x_n]

- Le seuil θi: est une valeur utilisée pour activer ou inactiver la sortie de neurone.
- La fonction d'activation :

$$F(x) \begin{cases} -1 \text{ si } x \leq 0 \\ +1 \text{ si } x > 0 \end{cases}$$

- La sortie du neurone :

$$Yi \begin{cases} -1 \text{ si } U_i(w,x) \leq 0 \\ +1 \text{ si } U_i(w,x) > 0 \end{cases}$$

- L'état du neurone :

$$U_i(w,x) = \sum_{j=1}^{N} w_{i.j} x_j - \theta_i$$

Wij sont les poids synaptiques.

En générale Un neurone artificiel est donc caractérisé par :

- Son état (fonction de base).
- Sa fonction d'activation.

I.4.1.1 La fonction de base :

Les connexions d'un réseau de neurones sont représentées analytiquement par la fonction Ui(w.x) avec :

W *:* étant la matrice des poids synaptiques.

X : étant le vecteur des entrées.

U_i(w.x) : C'est une fonction qui décrit l'état du réseau.

La fonction de base peut prendre plusieurs formes dont :

a). La fonction L.B.F (Linear Basis Function) qui représente une combinaison linéaire des entrées :

$$U_i(w,x)=\sum_{j=1}^{n} w_{j.i}\,x_j$$

b). La fonction N.L.B.F (No-Linear Basis Function) qui représente une combinaison non-linéaire des entrées et peut prendre plusieurs formes, la plus utilisée est la fonction **R.B.F** (Radial basis function) :

$$U_i(w,x)=\left[\sum_{j=1}^{n}\left(x_j-w_{j.i}\right)^2\right]^{1/2}$$

I.4.1.2 La fonction d'activation :

La fonction d'activation ou de transfert, permet d'introduire un seuil et une saturation. Il est intéressant d'utiliser des fonctions dont la dérivée est facile à calculer.
Les fonctions les plus courantes sont :

Sigmoïde unipolaire :

$$f_1(x)=\frac{1}{1+e^{-x}}$$
$$f_1'(x)=f_1(x)\left[1-f_1(x)\right]$$

Sigmoïde bipolaire :

$$f_2(x)=\frac{1-e^{-x}}{1+e^{-x}}=2\left[f_1(x)-0.5\right]$$
$$f_2'(x)=\frac{1}{2}\left[1+f_2(x)\right]\left[1-f_2(x)\right]$$

Tangente hyperbolique :

$$f_3(x)=\frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$$
$$f_3'(x)=\frac{4}{\left(e^{x}+e^{-x}\right)^2}$$

I.5 Traitement de l'information par les réseaux de neurones :

Dans le traitement de l'information par les réseaux de neurones, il existe deux phases :

- La phase d'apprentissage
- la phase de reconnaissance.

I.5.1 La phase d'apprentissage [2]:

L'apprentissage est une phase du développement d'un réseau de neurones durant laquelle le comportement du réseau est modifié jusqu'à l'obtention du comportement désiré. L'apprentissage neuronal fait appel à des exemples de comportement.

Dans le cas des réseaux de neurones artificiels, on ajoute souvent à la description du modèle l'algorithme d'apprentissage. Le modèle sans apprentissage présente en effet peu d'intérêt. Dans la majorité des algorithmes actuels, les variables modifiées pendant l'apprentissage sont les poids des connexions.

L'apprentissage est donc la modification des poids du réseau dans l'optique d'accorder la réponse du réseau aux exemples et à l'expérience. Il est impossible de décider à priori des valeurs des poids des connexions d'un réseau pour une application donnée. A l'issu de l'apprentissage, les poids sont fixés.

Au niveau des algorithmes d'apprentissage, il existe trois grandes classes d'apprentissage.

I.5.1.a Apprentissage supervisé [1] :

L'apprentissage supervisé, figure (I.3) est basé sur des exemples d'apprentissage de type entrée/sortie (X_i,Y_i) le problème consiste à construire un réseau de neurones tel que :

$$R_n(X_i) = Y_i \quad \forall_i$$

R_n : équation de sortie d'un réseau de neurones.

Pour apprendre le réseau, il doit savoir qu'il a commis une erreur et doit connaître la réponse qu'il aurait dû donner. La règle d'apprentissage est locale dans le sens que chaque cellule de sortie apprend sans avoir besoin de connaître la réponse des autres cellules.

La cellule ne modifie la valeur de ses synapses que lorsqu'elle se trompe.

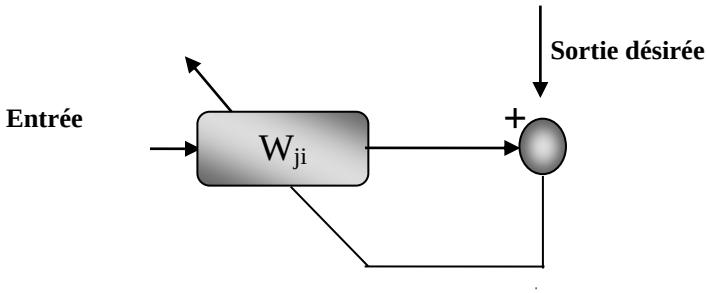


Figure (I.3) : *apprentissage supervisé*.

I.5.1.b Apprentissage semi-supervisé :

II est basé sur des exemples d'apprentissage de la forme (entrée). La seule information disponible en sortie du réseau de neurones est un signal d'échec.

I.5.1.c Apprentissage non supervisé :

II est basé sur des exemples d'apprentissage de type (entrées), figure (I.4). Le réseau organise ses entrées de telle sorte à optimiser un critère de performance donné

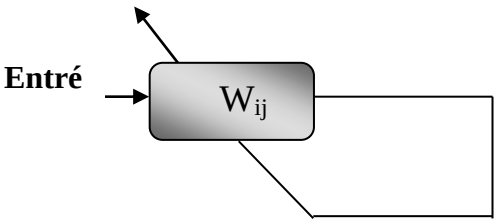


Figure (I.4) : *apprentissage non supervisé.*

I.5.2 La phase de reconnaissance :

C'est la phase utilisation du réseau après les testes réalisé dans l'apprentissage.

I.6 Architecture des réseaux de neurones :

Les réseaux de neurones sont divisés en deux classes selon la topologie :

- Réseaux de neurones statiques.
- Réseaux de neurones dynamiques.

I.6.1 Réseaux de neurones statiques :

Dans le cas des réseaux statiques, la sortie actuelle d'un neurone n'a aucune influence sur les sorties future des autres neurones. La sortie actuelle d'un neurone n'est injectée ni directement ni indirectement à son entrée.

I.6.2 Réseaux de neurones dynamiques :

Dans les réseaux dynamiques, appelés aussi réseaux récurrents, l'influence entre les neurones s'exerce dans les deux sens. L'état global du réseau dépend de ses états précédents, c'est à dire que chaque neurone reçoit une partie, ou la totalité des sorties précédentes des autres neurones.

La classification des réseaux en réseaux dynamiques et statiques n'est pas unique, les réseaux de neurones peuvent être aussi divisés en classes suivant l'ordre de l'entrée, il existe, les réseaux de neurones dynamiques ou statiques d'ordre un, deux... etc. Figure (I.5) et (I.6).
Les réseaux dont l'ordre de l'entrée est supérieur ou égal à 2 sont appelés les réseaux de neurones d'ordre supérieur.

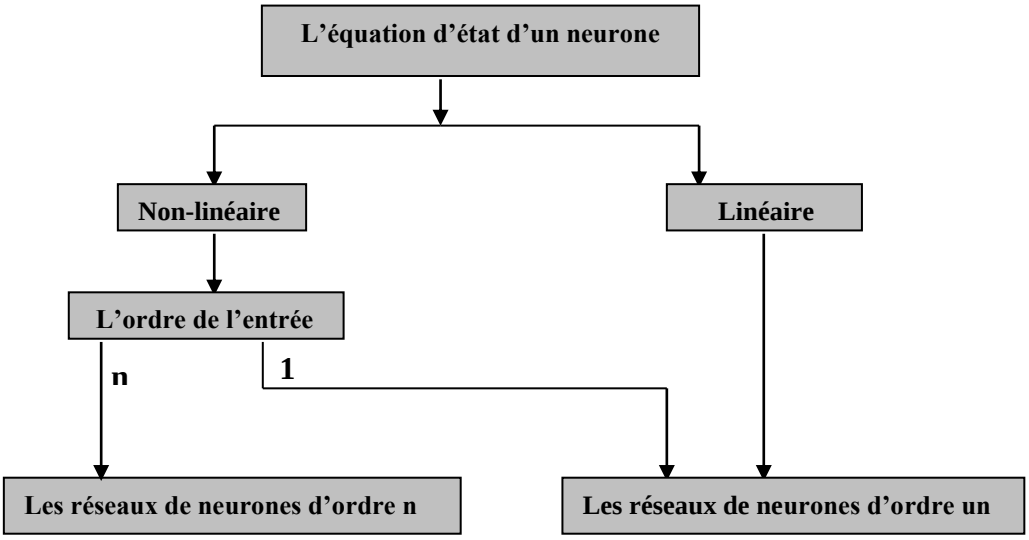


Figure (I.5) : *Classification des réseaux de neurones suivant l'ordre de l'entrée.*

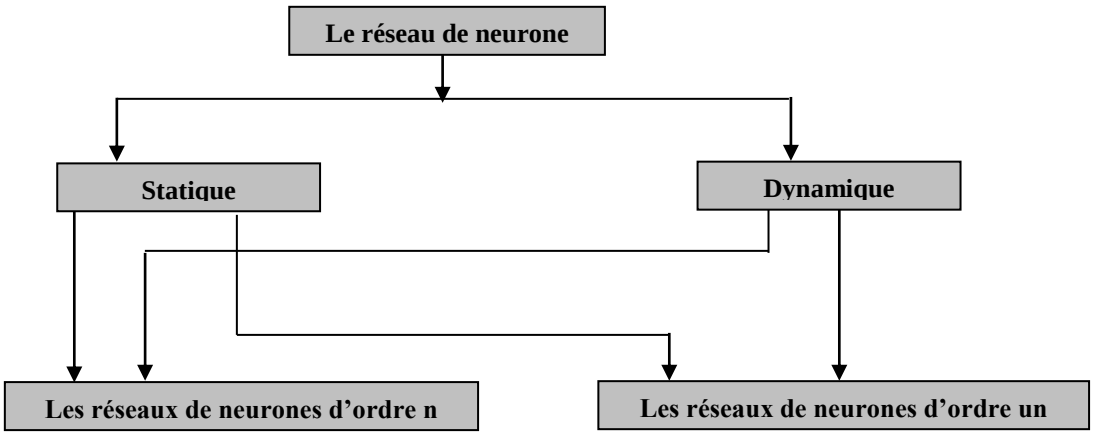


Figure (I.6) : *Classification des réseaux de neurones suivant le type de réseau.*

: Neurone d'entrée
Couche l

Après avoir étudié les généralités sur les réseaux de neurones, nous passerons au deuxième chapitre pour étudier des notions sur les réseaux multicouches ainsi que l'algorithme de rétro propagation le plus répandu qui joue un rôle très important dans la phase d'apprentissage.

Chapitre II :

Les Réseaux de neurones Multicouches

II.1 Introduction

Un réseau de neurone d'ordre un est défini comme étant un réseau dont les entrées sont d'ordre un, c'est à dire que dans les équations des sorties ou des états des neurones, les multiplications entre les entrées ou les sorties des neurones n'existent pas, tandis que dans les réseaux de neurones d'ordre supérieur nous trouvons les multiplications entre les entrées ou les sorties des neurones.

Dans le présent chapitre, nous commençons par définir les réseaux de neurones multicouches d'ordre un et d'ordre deux, ainsi que l'algorithme de rétro propagation et nous terminons par une généralisation aux réseaux de neurones multicouches d'ordre supérieur.

II.2 Les réseaux de neurones multicouches d'ordre un MLP:

Comme son nom l'indique, la structure de ce réseau de neurones est organisée en plusieurs couches de neurones, couche d'entrée (la couche de perception) et la couche de sortie (couche de décision), Une ou plusieurs couches sont insérées entre ces deux couches. Ces couches intermédiaires sont appelées les couches cachées, figure (II.1) **[4].**

Chaque couche contient un nombre défini de neurones, chaque neurone de la couche l est connecté à tous les neurones de la couche l+1.

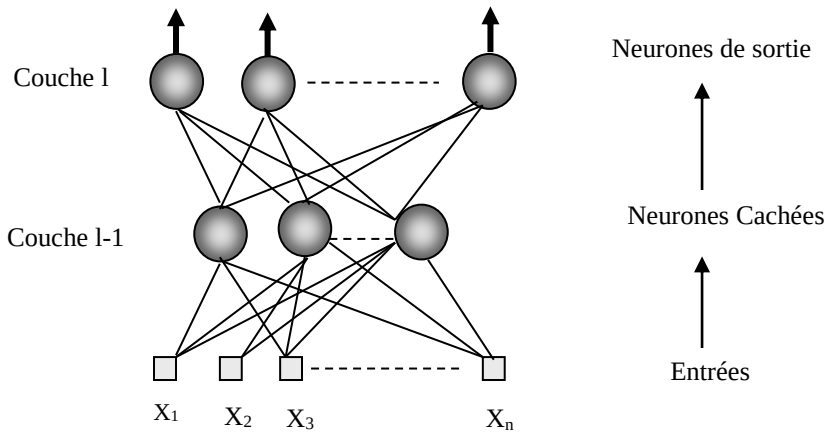


Figure (II.1) : *Réseaux de neurone multicouches d'ordre un.*

Pour les réseaux multicouches d'ordre n on respecte les notions suivantes **[1]** :

$U_{i,j}$	La sortie du neurone j de la couche l.
$U_{0,i}$	La i^{eme} entrée.
$d_j(x_p)$	La sortie désirée de la j^{eme} sortie.
N	Nombre de neurones de la couche l.
L	Nombre de couches.
P	Nombre de donnée d'apprentissage
$W_{l,j,i}$	Le poids entre la i^{eme} neurone de la couche l-1 et la j^{eme} neurone de la couche
$W_{l,j,i,k}$	Le poids entre le produit du i^{eme} et k^{eme} neurone de la couche l-1 et la j^{eme} neurone de la couche l
$W_{l,j1,j2,..;jn}$	Le poids entre le produit des $j_1,j_2 \ldots j_n$ neurones de la couche l-1 et la j^{eme} neurone de la couche l.

II.2.1 L'algorithme de rétro propagation :

La sortie d'un neurone de la couche l est donnée par :

$$u_{l,j} = f\left[\sum_{i=0}^{N_{l-1}} w_{l,j,i} u_{l-1,i}\right] \tag{1.1}$$

Où $f(\bullet)$ est une fonction sigmoïdale dont la dérivée est :

$$f'(\alpha) = \frac{df(\alpha)}{d\alpha} = f(\alpha)(1 - f(\alpha)) \tag{1.2}$$

L'algorithme d'apprentissage utilisé pour les réseaux MLP est basé sur la méthode du gradient dont le but est de trouver les poids synaptiques qui minimisent une fonction de critère donnée. La fonction de critère à minimiser dans notre cas est :

$$j(w) = \sum_{p=1}^{p} j_p(w) \tag{1.3}$$

P est le nombre de données d'apprentissage.

Et
$$j_p(w) = \frac{1}{2}\sum_{q=1}^{N_l}\left(u_{l,q}(x_p) - d_q(x_p)\right)^2 \tag{1.4}$$

Les poids seront ajustés de la façon suivante :

$$\begin{aligned} w_{l,j,i}(k+1) &= w_{l,j,i}(k) - \mu \frac{\partial j(w)}{\partial w_{l,j,i}}\Big|_{w(K)} \\ &= w_{l,j,i}(k) - \mu \sum_{p=1}^{p} \frac{\partial j_p(w)}{\partial w_{l,j,i}}\Big|_{w(K)} \end{aligned} \tag{1.5}$$

Où μ est le taux d'apprentissage.

Pour construire l'algorithme, il faut développer l'expression de la dérivée partielle de j_p en respectant chaque poids dans le réseau.

$$\frac{\partial j_p(w)}{\partial w_{l\cdot j\cdot i}} = \frac{\partial j_p(w)}{\partial u_{l\cdot j}} \frac{\partial u_{l\cdot j}(w)}{\partial w_{l\cdot j\cdot i}} \tag{1.6}$$

$$\frac{\partial u_{l,j}}{\partial w_{l\cdot j\cdot i}} = \frac{\partial}{\partial w_{l\cdot j\cdot i}} \left[f\left(\sum_{m=0}^{N_{l-1}} w_{l,j,m} u_{l-1,m} \right) \right]$$

$$= f'\left(\sum_{m=0}^{N_{l-1}} w_{l,j,m} u_{l-1,m} \right) \frac{\partial}{\partial w_{l,j,i}} \left(\sum_{m=0}^{N_{l-1}} w_{l,j,m} u_{l-1,m} \right) \tag{1.7}$$

$$= f'\left(\sum_{m=0}^{N_{l-1}} w_{l,j,m} u_{l-1,m} \right) u_{l-1,i}$$

En remplaçant la valeur de f' dans l'équation (1.2) on trouve :

$$\frac{\partial u_{l,j}(w)}{\partial w_{l,j,i}} = u_{l,j}\left(1 - u_{l,j}\right) u_{l-1,i} \tag{1.8}$$

Donc l'équation (1.6) sera :

$$\frac{\partial j_p(w)}{\partial w_{l,j,i}} = \frac{\partial j_p(w)}{\partial u_{l,j}} u_{l,j}\left(1 - u_{l,j}\right) u_{l-1,i} \tag{1.9}$$

- Le terme $\frac{\partial j_p(w)}{\partial u_{l\cdot j}}$ représente la sensibilité de $j_p(w)$ par rapport à la sortie de neurone $u_{l,j}$
  - Le terme $u_{l,j}$ exerce son influence sur j_p sur tous les neurones des couches suivantes.
  - $\frac{\partial j_p(w)}{\partial u_{l,j}}$ est une fonction de sensibilité du terme $j_p(w)$ à la sortie du neurone $u_{l,j}$ et peut être représentée en fonction des sensibilités par rapport aux sorties des neurones de la couche suivante.

$$\frac{\partial j_p(w)}{\partial u_{l,j}} = \sum_{m=1}^{N_{l+1}} \frac{\partial j_p(w)}{\partial u_{l+1,m}} \frac{\partial u_{l+1,m}}{\partial u_{l,j}} \tag{1.10}$$

$$\frac{\partial j_p(w)}{\partial u_{l,j}} = \sum_{m=1}^{N_{l+1}} \frac{\partial j_p(w)}{\partial u_{l+1,m}} \frac{\partial}{\partial u_{l,j}} \left[f\left(\sum_{q=0}^{N_{l_1}} w_{l+1,m,q} u_{l,q} \right) \right]$$

$$= \sum_{m=1}^{N_{3+1}} \frac{\partial j_p(w)}{\partial u_{l+1,m}} f'\left(\sum_{q=0}^{N_l} w_{l+1,m,q} u_{l,q} \right) \frac{\partial}{\partial u_{l,j}} \left[\sum_{q=0}^{N_l} w_{l+1,m,q} u_{l,q} \right]$$

$$= \sum_{m=1}^{N_{l+1}} \frac{\partial j_p(w)}{\partial u_{l+1,m}} u_{l+1,m} \left(1 - u_{l+1,m}\right) w_{L+1,m,j}$$

On continue de la même façon jusqu'à la couche de sortie. On trouve pour la couche de sortie :

$$\frac{\partial j_p(w)}{\partial u_{l,j}} = u_{l,j}\left(x_p\right) - d_j\left(x_p\right) \tag{1.11}$$

II.2.2 Résumé de l'algorithme d'apprentissage de rétro propagation du gradient:

1. Initialisation des poids synaptiques avec des petites valeurs aléatoires entre 0 et 1
2. Présentation des données d'apprentissage (entrée et sortie désirée)
3. Calcul de la sortie de neurone *équation* (1.1).
4. Calcul du gradient :

   Equation (1.11) pour la couche de sortie.

   Equation (1.10) pour la (les) couche(s) cachée(s).
5. Ajustement des poids *équation* (1.5)
6. Répétition des étapes (2) jusqu'à (5) pour toutes les données et ce, jusqu'à convergence de l'algorithme.

II.3 Les réseaux Multicouches MLP d'ordre supérieur :

Un réseau multicouche d'ordre supérieur contient une couche d'entrées, une couche de sorties, et une ou plusieurs couche(s) cachée(s) figure (II.2). Chaque couche contient un nombre défini de neurones, est chaque neurone de la couche l est connecté à tous les neurones de la couche l+1, chaque neurone de la couche l+1 reçoit toutes les sorties des neurones de la couche l ainsi que leurs multiplication .Ils sont caractérisés par la présence de termes non- linéaires dans les équations qui définissent la dynamique du réseau **[1]**.

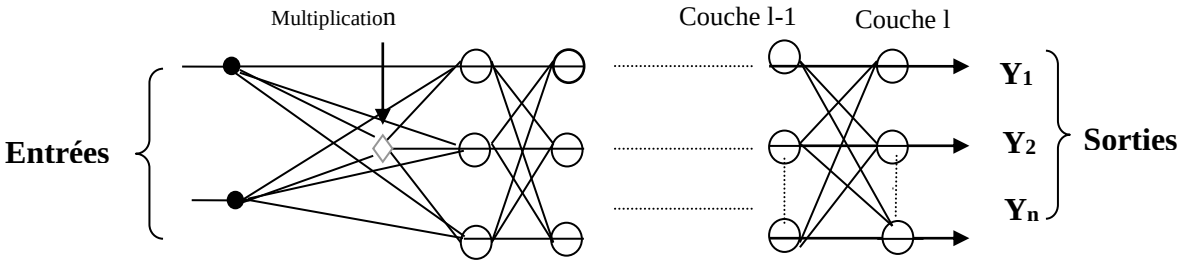


Figure (II.2) : *Réseaux de neurones multicouches d'ordre supérieur.*

$f(x)$ peut être :

- Une fonction linéaire :

$$f(x) = x^T \qquad x^T = [x_1, x_2, \Lambda, x_n]$$

- Une fonction non-linéaire :

$$f(x) = x_{i1} + x_{i1}x_{i2} + x_{i1}x_{i2}\Lambda x_{ir} \qquad \text{Pour } il, i2\Lambda ir = 1\Lambda p$$

II.3.1 Les réseaux d'ordre 2 :

Dans ce réseau l'équation de sortie d'un neurone n'est plus linéaire et a la forme suivante :

$$u_{l,j} = f\left[\sum_{i=0}^{N_{l-1}} w_{l,j,i}u_{l-1,i} + \sum_{i=0}^{N_{l-1}}\sum_{K=i}^{N_{l-1}} w_{l,j,i,k}u_{l-1,i}u_{l-1,K}\right] \qquad (2.1)$$

Le terme $\sum_{i=0}^{N_{l-1}}\sum_{K=i}^{N_{l-1}} w_{l,j,i,k}u_{l-1,i}u_{l-1,k}$ dans l'équation (2.1) représente la non linéarité.

L'algorithme d'apprentissage se base sur le principe du gradient et les poids seront ajustés de la façon suivante :

$$w_{l,j,i}(n+1) = w_{l,j,i}(n) - \mu \frac{\partial j(w)}{\partial w_{l,j,i}}\Big|_{w(n)} \tag{2.2}$$

$$w_{l,j,i,K}(n+1) = w_{l,j,i,K}(n) - \mu \frac{\partial j(w)}{\partial w_{l,j,i,K}}\Big|_{w(n)} \tag{2.3}$$

L'algorithme d'apprentissage pour ce cas, il contient les mêmes étapes que l'algorithme d'apprentissage du réseau linéaire.

Donc on a :

$$\frac{\partial j_p(w)}{\partial w_{l\cdot j\cdot i}} = \frac{\partial j_p(w)}{\partial u_{l\cdot j}} \frac{\partial u_{l\cdot j}(w)}{\partial w_{l\cdot j\cdot i}} \tag{2.4}$$

$$\frac{\partial j_p(w)}{\partial w_{l\cdot j\cdot i}} = \frac{\partial j_p(w)}{\partial u_{l\cdot j}} u_{l,j}\left(1-u_{l,j}\right)u_{l-1,i} \tag{2.5}$$

Où

$$\frac{\partial j_p(w)}{\partial u_{l,j}} = \sum_{m=1}^{N_{l+1}} \frac{\partial j_p(w)}{\partial u_{l+1,m}} \frac{\partial u_{l+1,m}}{\partial u_{l,j}} \tag{2.6}$$

$$\frac{\partial j_p(w)}{\partial u_{l,j}} = \sum_{m=1}^{N_{l+1}} \frac{\partial j_p(w)}{\partial u_{l+1,m}} \frac{\partial}{\partial u_{l,j}} \left[f\left(\sum_{q=0}^{N_{l1}} w_{l+1,m,q} u_{l,q} + \sum_{q=0}^{N_l} \sum_{q=0}^{N_l} w_{l+1,m,q,r} u_{l,q} u_{l,r} \right) \right]$$

$$\frac{\partial j_p(w)}{\partial u_{l,j}} = \sum_{m=1}^{N_{l+1}} \frac{\partial j_p(w)}{\partial u_{l+1,m}} u_{l+1,m}\left(1-u_{l+1,m}\right) \frac{\partial}{\partial u_{l,j}}$$

$$= \sum_{0}^{N_l} w_{l+1,m,q} u_{l,q} + \sum_{q=0}^{N_l} \sum_{r=q}^{N_l} w_{l+1,m,q,r} u_{l,q} u_{l,r}$$

$$= \sum_{m=1}^{N_{l+1}} \frac{\partial j_p(w)}{\partial u_{l+1,m}} u_{l+1,m}\left(1-u_{l+1,m}\right)$$

$$\bullet \left[w_{l+1,m,j} + \sum_{q=0}^{N_l} w_{l+1,m,q,j}\ u_{l,q} + \sum_{r=j}^{N_l} w_{l+1,m,j,r} u_{l,r} \right]$$

Pour la couche de sortie on trouve :

$$\frac{\partial j_p(w)}{\partial u_{l,j}} = u_{l,j}(x_p) - d_j(x_p) \qquad (2.7)$$

Pour développer un algorithme d'apprentissage on doit déterminer le terme $\frac{\partial j(w)}{\partial w_{l,j,i,K}}$ dans l'équation (2.3)

Donc :

$$\frac{\partial j_p(w)}{\partial w_{l,j,i,K}} = \frac{\partial j_p(w)}{\partial u_{l,j}} \frac{\partial u_{l,j}(w)}{\partial w_{l,j,i,K}} \qquad (2.8)$$

$\frac{\partial j_p(w)}{\partial u_{l,j}}$ est donnée par l'équation (2.6) ou (2.7) et le terme $\frac{\partial w_{l,j}}{\partial w_{l,j,i,K}}$ comme suite :

$$\frac{\partial u_{l,j}}{\partial w_{l,j,i,K}} = \frac{\partial}{\partial w_{l,j,i,K}} \left[f\left(\sum_{i=0}^{N_{l-1}} w_{l,j,i} u_{l-1,i} \sum_{i=0}^{N_{l-1}} \sum_{K=i}^{N_{l-1}} w_{l,j,i,K} u_{l-1,K} u_{l-1,i} \right) \right]$$

$$\frac{\partial u_{l,j}}{\partial w_{l,j,i,K}} = u_{l,j}(1 - u_{l,j}) u_{l-1,i} u_{-1,K} \qquad (2.9)$$

II.3.1.1 Résume de l'algorithme d'apprentissage MLP d'ordre 2 :

1. Initialisation des poids synaptiques avec des petites valeurs aléatoires.
2. Présentation des données d'apprentissage (entrées et sorties désirées)
3. Calcul de la sortie de neurone *équation (2.*1).
4. Calcul du gradient
5. Ajustement des poids comme suit :

$$W_{l,j,i}=W_{l,j,i} - \mu\ g_{l,j,I}$$
$$W_{l,j,i,k}=W_{l,j,i,k} - \mu\ g_{l,j,i,k}$$

Pour la couche de sortie :

$$g_{l,j,i} =(U_{l,j}-d_j)U_{l,j}(1-U_{l,j})U_{l-1,i}$$
$$g_{l,j,i,k} =(U_{l,j}-d_j)U_{l,j}(1-U_{l,j})U_{l-1,i}\ U_{l-1,k}$$

pour la (les) couche (s) cachée (s) :

$$g_{l,j,i} = \sum_{m=1}^{N_{l+1}} \frac{\partial j_p}{\partial u_{l+1,m}} [S] u_{l+1,m}(1-u_{l+1,m}) u_{l,j}(1-u_{l,j}) u_{l-1,i}$$

$$g_{l,j,i,k} = \sum_{m=1}^{N_{l+1}} \frac{\partial j_p}{\partial u_{l+1,m}} [S] u_{l+1,m}(1-u_{l+1,m}) u_{l,j}(1-u_{l,j}) u_{l-1,i} u_{l-1,K}$$

6. Répétition des étapes (2) jusqu'à (5) pour toutes les données et ce. Jusqu'à convergence de l'algorithme.

II.3.2 Les Réseaux d'ordre n :

La sortie d'un neurone pour un réseau d'ordre n est :

$$u_{l,j} = f(t) = f\left[\sum_{i=0}^{N_{l-1}} w_{l,j_1,j_2} u_{l-1,j_2} + \sum_{j_2=0}^{N_{l-1}} \sum_{j_3=j_2}^{N_{l-1}} w_{l,j_1,j_2,j_3} u_{l-1,j_2} u_{l-1,j_3}\right.$$

$$\left. + \Lambda + \sum_{j_2=0}^{N_{l-1}} \Lambda + \sum_{j_n=j_{n-1}}^{N_{l-1}} w_{l,j_1,\Lambda,j_n} u_{l-1,j_2} \Lambda u_{l-1,j_n} \right] \quad (2.10)$$

de la même façon que le MLP d'ordre 2 on doit déterminer seulement les termes suivants :

$$w_{l,j_1,\Lambda,j_l}(n+1) = w_{l,j_1,\Lambda,j_2}(n) - \mu \frac{\partial j(w)}{\partial w_{l,j_1,\Lambda,j_2}}\Big|_{w(n)} \quad l = 2 \Lambda n \quad (2.11)$$

$$\frac{\partial j_p(w)}{\partial u_{l\cdot j}} \quad (2.12)$$

$$\frac{\partial u_{l,j_1,\Lambda,j_2}}{\partial w_{l\cdot j_1\cdot\Lambda,j_n}} \quad (2.13)$$

$$\frac{\partial u_{l,j}}{\partial w_{l\cdot j_1\cdot\Lambda,j_n}} = u_{l,j}(1-u_{l,j}) u_{l-1,j_2} \Lambda u_{l-1,j_n} \quad (2.14)$$

$$\frac{\partial j_p(w)}{\partial u_{l,j}} = \sum_{m=1}^{N_{l+1}} \frac{\partial j_p(w)}{\partial u_{l+1,m}} \frac{\partial u_{l+1,m}}{\partial u_{l,j}} \quad (2.15)$$

$$\frac{\partial u_{l+1,m}}{\partial u_{l,j}} = \frac{\partial}{\partial u_{l,j}} f(t) \quad (2.16)$$

$$\frac{\partial j_p(w)}{\partial u_{l,j}} = \sum_{m=1}^{N_{l+1}} \frac{\partial j_p(w)}{\partial u_{l+1,m}} u_{l+1,m}(1-u_{l+1,m})$$

$$* \left[w_{l+1,j_1,j_2} + \sum_{q=0}^{N_l} w_{l+1,j_1,j_2,j_3} u_{l,j_2} + \sum_{r=j}^{N_l} w_{l+1,j_1,j_2,j_3} u_{l,j_3} \right]$$

: Neurone d'entrée
Couche l

$$\Lambda + \left[\sum_{j_3=j_2}^{N_l} \Lambda \sum_{j_n=j_{n-1}}^{N_l} w_{l+1,j_1,\Lambda,j_n} u_{l,j_2} \Lambda u_{l,j_n} \right]$$

$$\Lambda + \left[\sum_{j_3=j_2}^{N_l} \Lambda \sum_{j_n=j_{n-1}}^{N_l} w_{l+1,j_1,\Lambda,j_n} u_{l,j_2} \Lambda u_{l,j_{n-1}} \right]$$

II.4 Conclusion du chapitre :

En générale les réseaux de neurones d'ordre un et d'ordre supérieur sont apparus dans le but de réduire la taille des réseaux et d'améliorer la vitesse d'apprentissage.

Dans le chapitre suivant nous traiterons différentes techniques de filtrage d'image, ainsi que l'étude du filtre neuronal.

Chapitre III :

LE FILTRAGE DES IMAGES

III.I Introduction :

Les techniques de filtrage d'images sont destinées à l'exploitation des informations contenues dans les images, ceci dans le but d'améliorer leur qualité et de les rendre plus facilement interprétables, soit par un observateur soit par une machine.

Plusieurs méthodes ont été utilisées pour le filtrage d'images, l'une de ces méthodes s'appelle le filtrage spatial, qui consiste à remplacer la luminosité d'un pixel de l'image par une valeur qui à été déterminée grâce au voisinage de ce pixel. Les filtres peuvent être séparés en deux catégories, filtres linéaires et filtres non linéaires.

III.2 Filtre linéaire [5],[8]:

Un filtre est dit linéaire s'il peut s'écrire de la manière suivante :

Filtre (c1.image1+c2.image2) = c1Filtre (image1) +c2Filtre (image2)

Les filtres linéaires les plus connus sont les filtres *passe-haut, passe-bas, Sobel,Prewitt* et *Laplacien,* ils sont décrits plus en détails dans **[6],[7]** Dans tous les cas, ces filtres sont représentés sous forme de masque auquel est associé un coefficient diviseur qui permet de ramener les valeurs calculées entre 0 et 255 pour une image à 256 niveaux de gris.

III.2.1 Filtre Passe-Bas (lissage) :[5],[8]

Ce filtre n'affecte pas les composants de basse fréquence dans les données d'une image, mais doit atténuer les composants de haute fréquence. L'opération de lissage est souvent utilisée pour atténuer le bruit et les régularités de l'image, elle peut être répétée plusieurs fois, ce qui crée un effet de flou.

En pratique, il faut choisir un compromis entre l'amélioration du bruit et la conservation des détails et contours significatifs.

Un exemple de coefficients d'un filtre Passe-Bas pour un masque de matrice de convolution 3x3 est :

$$PB=1/9\begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{bmatrix}$$

Les règles concernant un filtre Passe-Bas sont :

1) Tous les coefficients doivent être positifs.

2) La somme de tous les coefficients doit être égale à 1.

Exemple : Le filtre binominal bidimensionnel :

$$BIN = 1/16\begin{bmatrix}1 & 2 & 1\\ 2 & 4 & 2\\ 1 & 2 & 1\end{bmatrix}$$

Image d'origine *Image lissée*

Figure (III.1): *Image filtrée avec un filtre passe-bas.*

III.2.2 Filtre Passe-haut :

Le filtre digital Passe-haut a les caractéristiques inverses du filtre Passe-Bas. Ce filtre n'affecte pas les composants de haute fréquence d'un signal, mais atténuer les composants de basse fréquence.

Un exemple de coefficients d'un filtre Passe-haut pour un masque de matrice de convolution 2x2 est :

$$PH = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

Les règles concernant un filtre Passe-haut sont :

- Les coefficients peuvent être positifs ou négatifs.
- La somme de tous les coefficients doit être égale à 0.

Un exemple de filtrage d'image qui utilise le filtre passe-haut.

Image d'origine *Filtrage passe-haut*

Figure (III.2): *Image filtrée avec un filtre passe-haut.*

III.2.3 Filtre de Sobel:

Il est utilisé pour la détection de contours. Les contours correspondent en général à des changements brusques de propriétés physiques ou géométriques de la scène

Ce filtre comporte deux masques de convolution, un pour la détection des contours verticaux et un autre pour la détection des contours horizontaux. On obtient après convolution une image des gradients. Cette image pourra par la suite être modifiée dans une phase de seuillage pour ne contenir que des pixels à 1 ou 0 correspondant respectivement à point de contour ou à un non point de contour.

Masques de convolution :

1	0	-1
2	0	-2
1	0	-1

Diviseur 1
Vertical

1	2	1
0	0	0
-1	-2	-1

Diviseur 1
Horizontal

$$Gv(x,y) = I(x,y) * V(x,y) \quad (3.1) \qquad Gh(x,y) = I(x,y) * H(x,y) \quad (3.2)$$

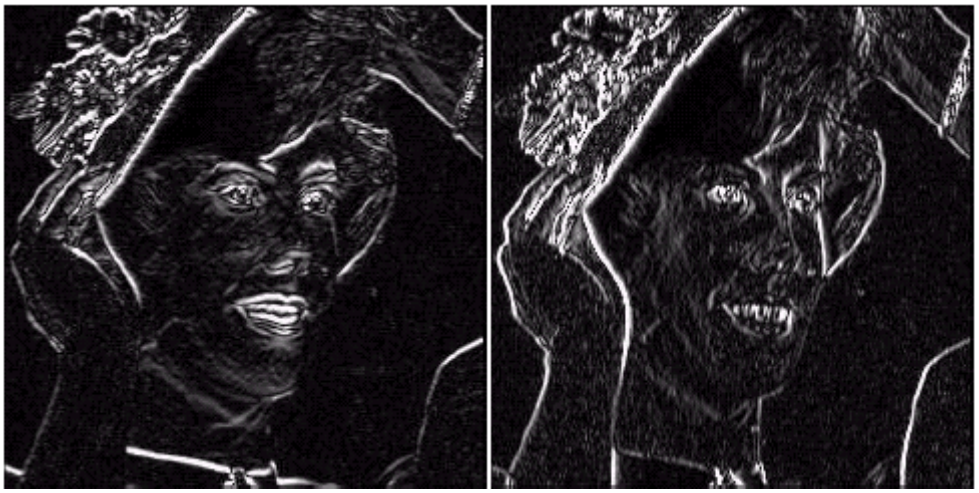
Le gradient final vaut :

$$G(x,y) = \sqrt{Gv(x,y)^2 + Gh(x,y)^2} \quad (3.3)$$

Souvent, pour diminuer le temps de calcul, on remplace la racine carrée par des valeurs absolues, l'équation devient donc :

$$G(x,y) = |Gv(x,y)| + |Gh(x,y)| \tag{3.4}$$

De plus, on applique souvent un coefficient diviseur différent de 1 pour ramener les valeurs des pixels entre 0 et 255.



Sobel Horizontal *Sobel Vertical*

Figure (III.3): *Image filtrée avec le filtre de Sobel*

III.2.4Le filtre de Deriche:

Le filtre de Deriche, opérateur optimal sous forme de filtre de réponse impulsionnelle infinie (RII), s'écrit de la manière suivante :

$$f(x) = Sxe^{-|x|} \tag{3.5}$$

Où S correspond à la résolution à laquelle les contours sont à détecter. Ses indices de localisation et de détection sont simplement exprimés en fonction de ce paramètre. Une diminution de S a permet de favoriser la détection au détriment de la localisation et vice versa.
Pour le cas à 2 dimensions, on utilise la fonction de lissage suivante :

$$h(x) = K(|x|+1)e^{-|x|} \tag{3.6}$$

L'avantage de ce filtre, est qu'il peut être implémenté de manière récursive et qu'il nécessite très peut d'opérations par pixel. Pour un cas 2D, on utilise les équations suivantes :

Pour la détection horizontale :

-Convolution dans la direction horizontale avec l'opérateur de dérivation f(x) :

$$Y^+(i,j) = \mathrm{I}(i,j-1) - b_1 Y^+(i,j-1) - b_2 Y^+(i,j-2) \quad (3.7)$$

$$j = l,....,NCL \qquad i = l,....,NLG$$

$$Y^-(i,j) = \mathrm{I}(i,j+1) - b_1 Y^-(i,j+1) - b_2 Y^-(i,j+2) \quad (3.8)$$

$$j = NCL,..,l \qquad i = l,....,NLG$$

$$S(i,j) = a\left(Y^+(i,j) - Y^-(i,j)\right) \quad (3.9)$$

$$j = l,....,NCL \qquad i = l,....,NLG$$

-On applique ensuite sur l'image S(i,j) le filtre de lissage h(n) dans la direction verticale :

$$Grx^+(i,j) = a_0 S(i,j) - a_1 S(i-1,j) - b_1 Grx^+(i-1,j) - b_2 Grx^+(i-2,j) \quad (3.10)$$

$$i = l,....,NLG \qquad j = l,....,NCL$$

$$Grx^-(i,j) = a_2 S(i+1,j) + a_3 S(i+2,j) - b_1 Grx^-(i+1,j) - b_2 Grx^-(i+2,j) \quad (3.11)$$

$$i = NLG,..,l \qquad j = l,....,NCL$$

$$Grx(i,j) = Grx^+(i,j) + Grx^-(i,j) \quad (3.12)$$

$$i = l,....,NLG \qquad j = l,....,NCL$$

avec:

$$a = Se^- \qquad S = -\left(\frac{\left(1-e^-\right)^2}{e^-}\right) \qquad K = \left(\frac{\left(1-e^-\right)^2}{1+2\cdot e^- - e^{-2}}\right) \quad (3.13)$$

$$a_0 = K \qquad a_1 = K(-1)e^- \qquad a_2 = a_1 - Kb_1 \qquad a_3 = -Kb_2$$

$$b_1 = -2e^- \qquad b2 = e^{-2}$$

Pour la détection verticale :

Il suffit d'échanger h(x) et f(x) et d'appliquer le même procédé.

L'algorithme pour extraire les gradients est le suivant :

a) Mise en œuvre récursive du produit de convolution I(i,j)*X(m,n) pour obtenir l'image gradient directionnel Grx(i,j) en x

b) Mise en œuvre récursive du produit de convolution I(i,j)*Y(m,n) pour obtenir l'image gradient directionnel Gry(i,j) en Y

c) Calcul de la norme A(i,j) et de la direction D(i,j) du gradient en fonction de Grx(i,j) et Gry(i,j)

$$A(i,j) = \sqrt{Grx(i,j)^{e^-} + Gry(i,j)^2} \tag{3.14}$$

et

$$D(i,j) = \arctan\left(\frac{Gry(i,j)}{Grx(i,j)}\right) \tag{3.15}$$

Un exemple de filtrage d'image qui utilise le filtre de deriche.

Figure (III.4)
Image des gradients Verticaux

Figure (III.5)
Image des gradients horizontaux

III.3 La convolution :[5]

Pour mettre en œuvre un filtrage avec des filtres linéaires, on utilise un opérateur mathématique nommé convolution (noté $\otimes$).

La convolution consiste donc à prendre un voisinage du pixel que l'on veut modifier, le plus souvent ce voisinage est compris dans une fenêtre carrée de côté de longueur impair, de multiplier chaque valeur du voisinage par des coefficients appartenant à un masque de convolution, dans faire la somme qui deviendra la nouvelle valeur du pixel. Ce traitement est effectué pour chaque pixel de l'image, mis à part les bords de l'image pour lesquels, soit on met les pixels à 0, soit on modifie le masque et le diviseur. Dans la plupart des cas, les bords où la taille du masque de convolution est (2d+1)*(2d+1).

Par la suite, les filtres seront toujours représentés sous forme de fenêtre et appelés masque de convolution.

La règle de convolution est définit par :

$$\mathrm{I}'(x,y) = \mathrm{I}(x,y) \otimes w(x,y) = \sum_{m=-d}^{d} \sum_{m=-d}^{d} w(m,n) \cdot \mathrm{I}(x+m, y+n) \tag{3.16}$$

III.4 Filtre non-linéaire: [5],[8]

III.4.1Filtre médian :

Ce filtre est très utilisé pour éliminer le bruit sur une image qui peut être de différentes origines (poussières, petits nuages, baisse momentanée de l'intensité électrique sur les capteurs, ...) et qui se traduit par des taches de faible dimension dont la distribution sur l'image est aléatoire. L'avantage de ce filtre réside dans le fait qu'il conserve les contours, alors que les autres types de filtres ont tendance à les adoucir.

Pour une image digitale (numérique). Ce filtre permet de prendre toutes les valeurs de niveaux de gris des pixels formant la valeur du pixel considéré et son voisinage, puis trie pour mettre la cinquième valeur de la liste obtenue (médiane) à la place du pixel concerné**.**

32	33	34	35	**36**	36	36	37	98
1	**2**	**3**	**4**	**(5)**				

III.4.2 Filtre maximum :

Il en est de même que le filtre médian toutefois la valeur maximale prend la place du pixel concerné.

Ligne tirée :

32	33	34	35	36	36	36	37	**98**

III.4.3 Filtre minimum :

Pour ce cas la valeur minimal prend la place du pixel concerné.

Ligne tirée :

32	33	34	35	36	36	36	37	98

III.5 Filtre neuronal :[5]

L'opération du Filtre neuronal Ψ sur une image I,

$$I\Psi I'(x,y) = \Psi[I(x,y)] \tag{3.17}$$

Oυ

$$\Psi = K \circ V \circ K^{1}$$

K : représente une fonction qui effectue un changement d'échelle sur les données de la fenêtre pour qu'elles puissent être utilisables par le réseau (données comprise entre 0 et 1), on obtient alors l'exemple noté p qui servira d'entrée au réseau.

$$P = K[I(x,y)] \tag{3.18}$$

On note ensuite v comme étant la fonction qui détermine la sortie du réseau en fonction d*e*

$$V[p] = 0 \tag{3.19}$$

La fonction K^{-1} effectue le travail inverse de la fonction k, convertir la sortie du réseau en valeur pour l'image résultat, donc en niveaux de gris,

$$K^{-1} = [K[I(x,y)]] = (x,y) \tag{3.20}$$

III.5.1 Architecture du réseau :

L'idée du filtre neuronal , figure (III.6) est très simple, elle est basée sur un réseau multicouche. L'architecture représentée, figure (III.7) est celle d'un filtre neuronal pour lequel la fenêtre sur l'image est de taille 3*3. Cette taille peut varier suivant le filtre que l'on veut réaliser, donc suivant le nombre de pixels du voisinage qui va être pris en compte.

Le nombre de cellules pour la couche cachée est variable et va être définit durant l'apprentissage du réseau suivant les résultats obtenus.

Pour le traitement d'image à 256 niveaux de gris, la fonction K est de la forme :

$K = \frac{1}{255}$ et la fonction K^{-1} de la forme $K^{-1} = 255$.

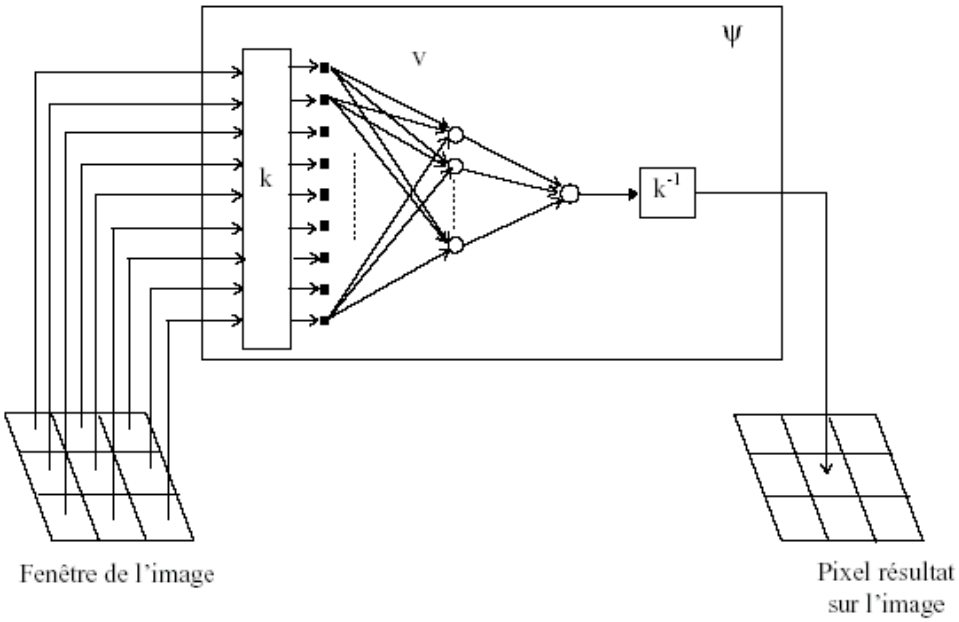


Figure (III.6) : *architecture du réseau*

Lors de l'apprentissage et de la phase de filtrage d'une image, la fenêtre du filtre neuronal est déplacée pixel par pixel sur l'image. Cette architecture permet de filtrer des images de tailles différentes, les images ne nécessitent donc pas de phase de normalisation avant la phase de filtrage. La figure (III.6) présente le parcours de l'image par les réseaux de neurones.

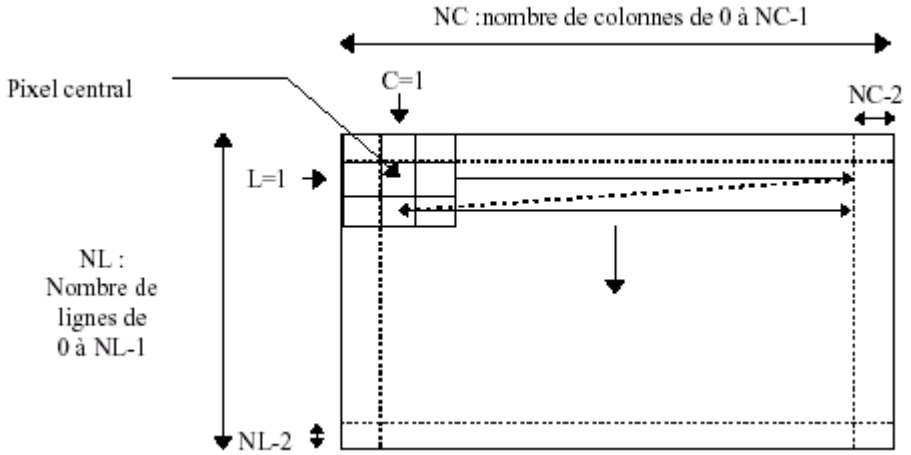


Figure (III.7) : *Parcours de l'image.*

: Neurone d'entrée
Couche l

Pour un filtre ayant une taille 3*3, le début et l'arrêt du balayage de l'image ce fait comme sur la figure (III.7).
Pour un filtre ayant une taille supérieure, les lignes de départ et d'arrivée ainsi que les colonnes changent. Les bords de l'image ne peuvent pas être modifiés par le filtre neuronal, la fenêtre de balayage devant être identique pour la totalité de l'image. Lors de la création de l'image filtrée, ses bords sont mis à zéro.

III.5.2 L'apprentissage :

Pour notre réseau, nous avons choisi un apprentissage supervisé qui consiste à présenter un exemple en entrée du réseau, puis de comparer la sortie obtenue avec notre sortie désirée.
Pour que le réseau puisse apprendre une fonction désirée, il faut lui présenter une série d'exemples appropriés. Dans notre cas, ces exemples seront issus d'images puisque notre réseau de neurones doit apprendre à filtrer des images.
Une fois l'apprentissage effectué, on passe à une étape de généralisation qui consiste à comparer l'image obtenue par le filtre neuronal avec l'image théorique qui à servit à l'apprentissage, puis d'effectuer le même travail avec une image quelconque.
L'avantage principal de ces filtres neuronaux est qu'il pourra filtrer des images pour lesquelles on ne connaît que l'image perturbée et l'image sans perturbations.
On pourrait par exemple créer un filtre neuronal devant supprimer le bruit sur une image. Pour cela on va prendre une image nette sur laquelle on va ajouter du bruit de façon aléatoire. Les données d'apprentissage seront donc l'image bruitée en entrée et l'image d'origine en sortie. Un tel filtre peut être utilisé par exemple pour déparasiter les images qui ont été transmises par ondes hertziennes.

Petit glossaire

Les définitions fournies ici ont pour objectif d'éclairer la lecture de cet ouvrage. Leurs validités est, a priori, restreintes.

Apprentissage : L'apprentissage est une phase du développement d'un réseau de neurones durant laquelle le comportement du réseau est modifié jusqu'à l'obtention du comportement désiré. L'apprentissage neuronal fait appel à des exemples de comportement.

Connexionnisme : Discipline définie par l'utilisation des réseaux de neurones artificiels pour l'ingénieur.

Dendrite : Ceux sont elles qui permettent au neurone de capter les signaux qui parviennent de l'extérieur.

Gradient : Le gradient pour un neurone est l'erreur relative à ce neurone.

Neuromimétique : Discipline définie par l'utilisation des réseaux de neurones artificiels pour valider des hypothèses biologiques par le neuroscientiste.

Neurone : Unité de traitement de l'information dans le cerveau, au nombre mille milliards environ.

Neurosciences : Ensemble des sciences ayant pour objectif l'étude du cerveau, depuis le niveau moléculaire jusqu'aux comportements sociaux en passant par les réseaux de neurones (Histologie, Neuropsychopharmacologie, Neurophysiologie, Ethologie,
Psychologie, etc.).

Processeur élémentaire : Allusion au processeur de calcul (en beaucoup plus simple), aussi appelé neurone ou cellule.

Perceptron : Un seul neurone.

Réseaux de neurones artificiels : Réseaux fortement connectés de processeurs élémentaires fonctionnant en parallèle. Chaque processeur élémentaire calcule une sortie unique sur la base des informations qu'il reçoit. Toute structure hiérarchique de réseaux est évidemment un réseau.

Supervisé : Les poids sont modifiés en fonction de la sortie désirée.
Non supervisé : Les sorties désirées sont inconnues, les entrées sont projetées sur l'espace du réseau.

Non supervisé : Les sorties désirées sont inconnues, les entrées sont projetées sur l'espace du réseau.

Synapse : Jonction entre deux neurones, très complexe au niveau des mécanismes chimiques mis en oeuvre ; outrageusement simplifiée dans les modélisations.

Bibliographie

[1]: Identification des systèmes non linéaire par les réseaux d e neurones d'ordre supérieur, application à une ruche apicole, université de Blida, institut d'électronique, Fev 2000.

[2]: Claude Touzet "Les réseaux de neurones artificiel, Introduction au connexionnisme, cours exercices et travaux pratiques, juillet 1992.

[3]: Frédéric Devaux "Filtrage d'images par réseaux de neurones, Septembre 1997.

[4]: http://www.neurones.espci.fr**,** Gérard Dreyfus" Introduction réseaux de neurones" Espci, Laboratoire d'Électronique.

[5]: http://www.mime.univ-paris8.fr/memoires/devaux.pdf

[6]*:* Anil K. Jain, Fundamentals of digital image processing, Prentice Hall, 1989.

[7]: Radu Honraud, Olivier Monga *Vision par ordinateur, outils fondamentaux,* 2e édition, HERMES, 1995.

[8]: Alfoldi T.T " Introduction aux images numériques et aux techniques d'analyse numérique" Centre canadien de télédétection, Energie, Mineset Ressources, Ottawa, note technique, 1978.

[9] : I.Bellido and E.Fiesler, Institut Dalle Molle d'intelligence Artificielle Perceptive (IDIAP).

[10] : Georg Thimm and Emil Fiesler ,Member, IEEE,"High_Order and Multilayer Perceptron Initialization"

Ainsi que d'autres site :

http:// www.Irde.epita.fr

http://www.enst.fr/~chaps

http://www.barth.netliberte.org/

La deuxième partie :

Simulation

I.1 Introduction :

Dans ce chapitre, nous allons présenter des résultats d'apprentissage et de reconnaissance des réseaux de neurones d'ordre un et d'ordre supérieur

Tout d'abord on a commencé par un exemple simple qui est le xor, et on a essayé de voir les influences sur le résultat de l'apprentissage des différents cas possibles (Initialisation des poids, taille du réseau, taux d'apprentissage, nombre des itérations,……etc).

Une fois l'étude détaillée de l'exemple achevée, les réseaux de neurones d'ordre un et d'ordre supérieure seront appliqués sur le filtrage d'image.

I.2 Le XOR:

I.2.1 L'étude de différents paramètres:

I.2.1.1 Poids Initial:

L'initialisation des poids avec des valeurs aléatoires joue un rôle très important dans la phase d'apprentissage pour cela on a fait 03 tests dont les valeurs initiales des poids ne sont pas les mêmes.

L'architecture du réseau est la suivante :

Réseau d'ordre 1

- Taille des couches : $N^3_{2\,4\,1}$
- Taux d'apprentissage : 0.99
- Nombre des itérations : 10^5

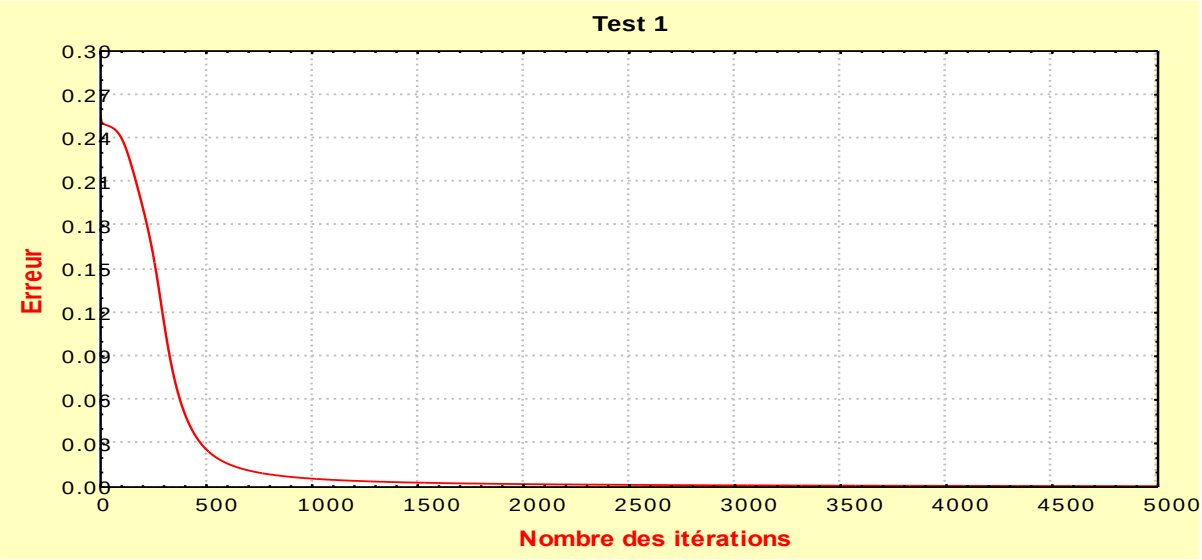
Réseau d'ordre 2

- Taille des couches : $N^3_{2\,2\,1}$
- Taux d'apprentissage : 0.99
- Nombre des itérations : 5000

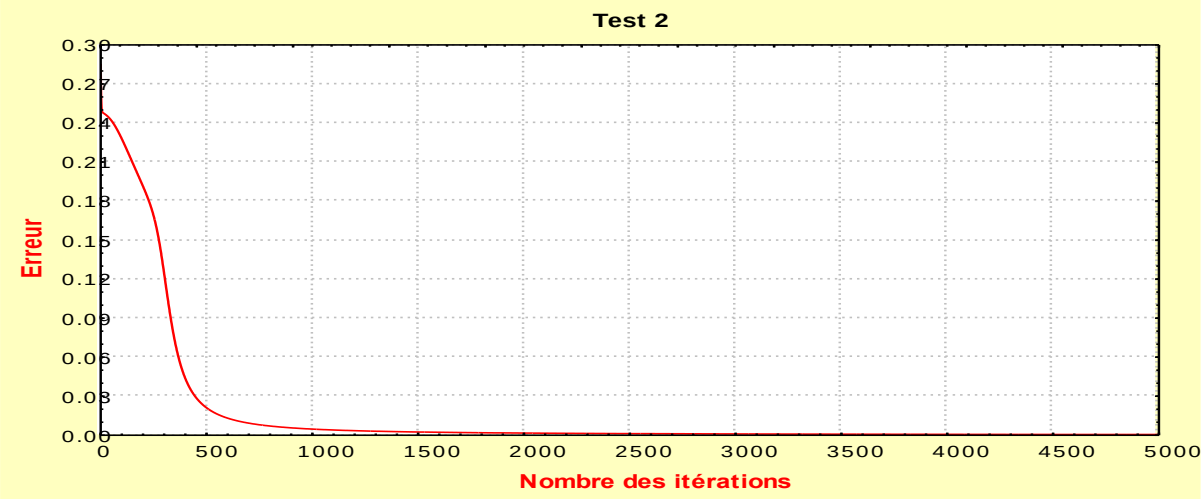
RESEAU D'ORDRE 1					RESEAU D'ORDRE 2			
Sortie désirée	**Test 1**	**Test 2**	**Test 3**		**Sortie désirée**	**Test 1**	**Test 2**	**Test 3**
0	0.0306	0.0267	0.0240		0	0.0530	0.1340	0.0284
1	0.9735	0.9811	0.9573		1	0.9669	0.9317	0.9827
1	0.9753	0.9756	0.9575		1	0.9736	0.5167	0.9759
0	0.0191	0.0215	0.5180		0	0.0300	0.0387	0.0143

Tableaux (I.1): *poids initial.*

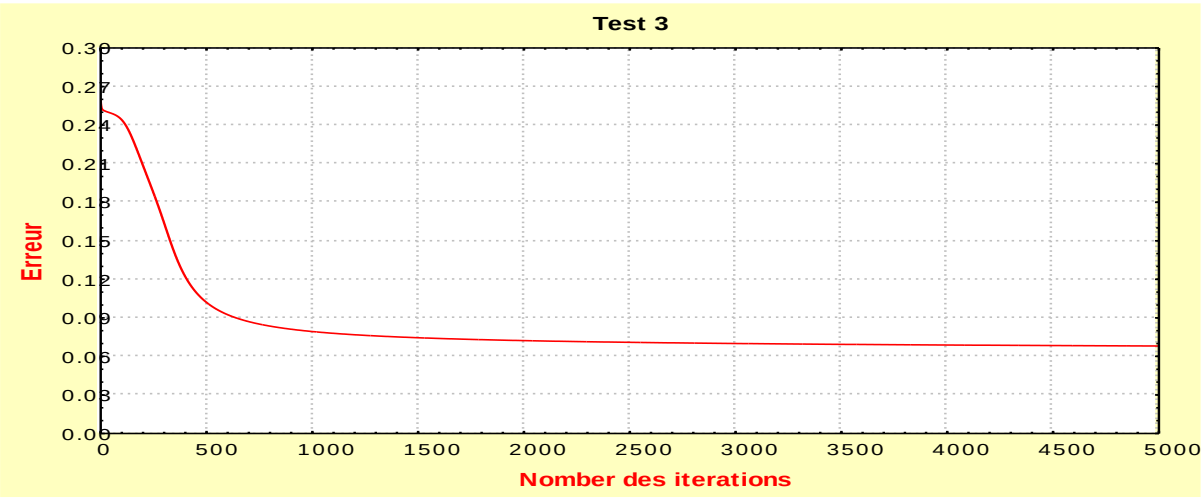
D'après les figures (I.1), (I.2) et les résultats du Tableau (I.1) et avec les mêmes caractéristiques pour réseaux d'ordre 1 et 2**,** et vu que les résultats d'apprentissage ne sont pas les mêmes on peut conclure que le choix des valeurs initiales des poids peut influer le comportement général d'un réseau **[10]**



A

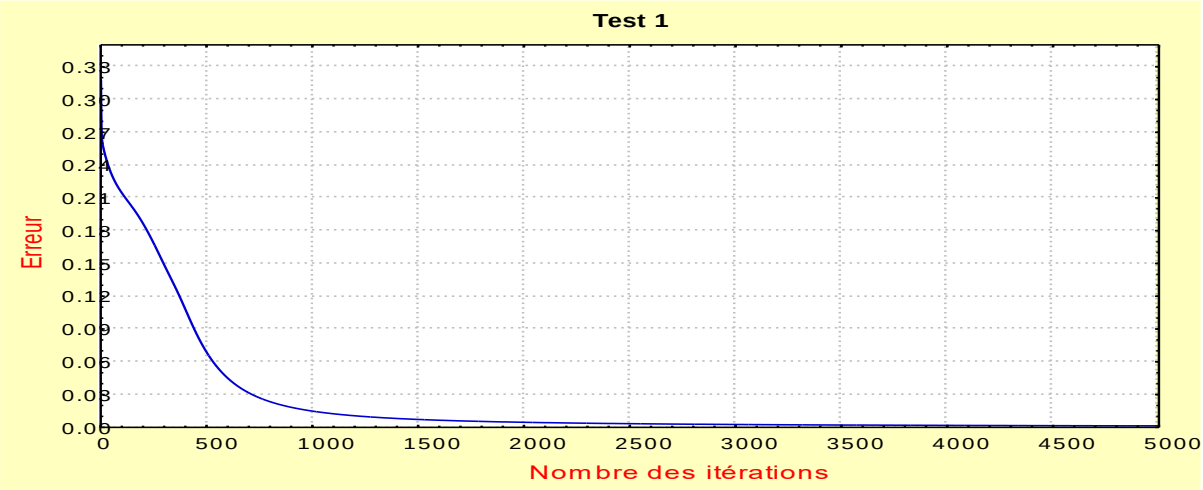


B



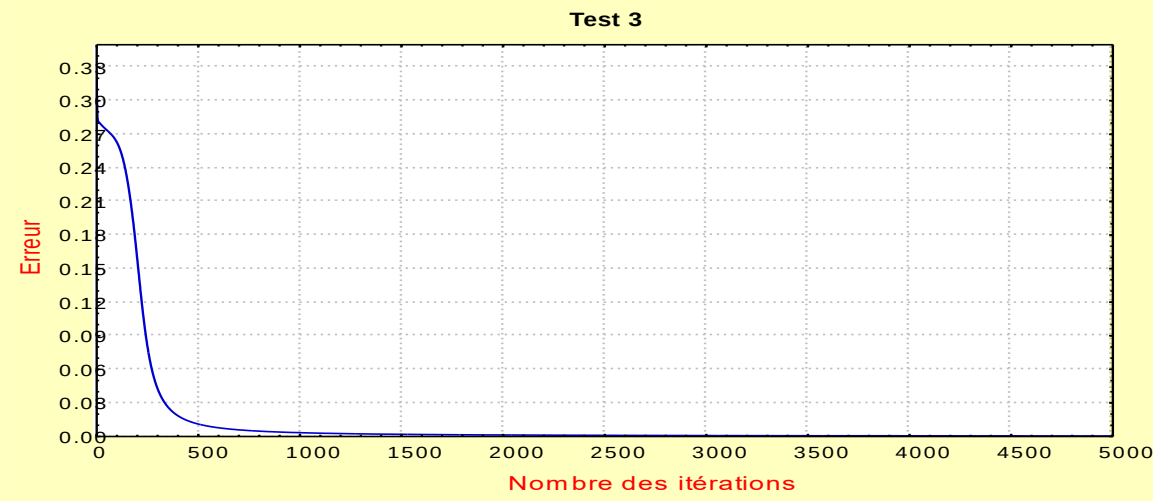
C

Figure (I.1): *poids Initial (réseaux d'ordre un)*



A

B



C

Figure (I.2): *poids Initial (réseaux d'ordre deux)*

I.2.1.2 Nombre des itérations :

L'architecture du réseau est la suivante :

Réseau d'ordre 1

- Taille des couches : $N^3_{2\,4\,1}$
- Taux d'apprentissage : 0.99

Réseau d'ordre 2

- Taille des couches : $N^3_{2\,2\,1}$
- Taux d'apprentissage : 0.99

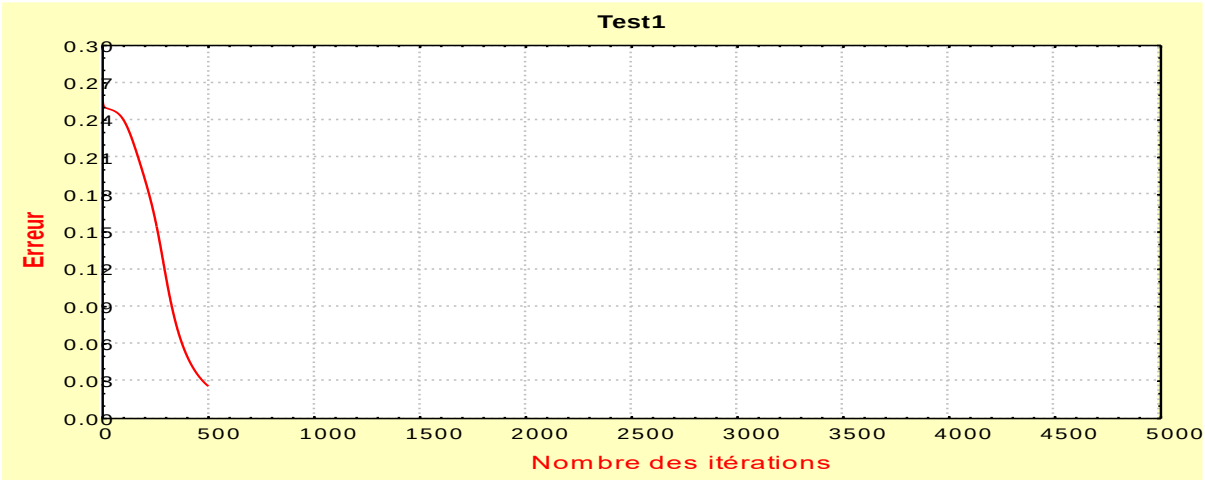
On a fait 3 tests différents, chaque fois on modifie le nombre des itération comme suit :

Test	Nbre_d'itération
1	500
2	2000
3	5000

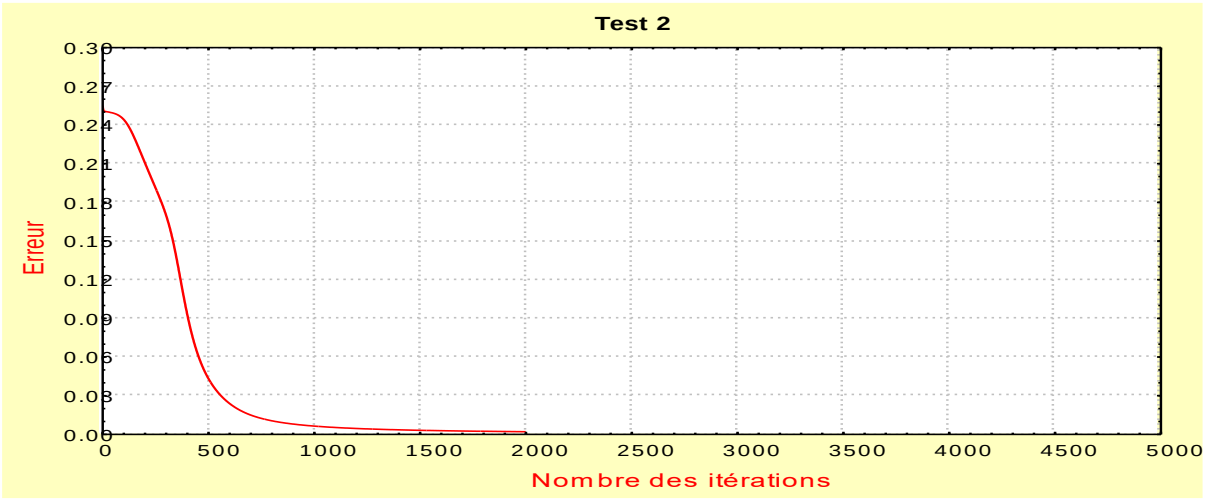
RESEAU D'ORDRE 1					RESEAU D'ORDRE 2			
Sortie désirée	**Test 1**	**Test 2**	**Test 3**		**Sortie désirée**	**Test 1**	**Test 2**	**Test 3**
0	0.1814	0.0535	0.0273		0	0.1377	0.0556	0.0232
1	0.8336	0.9571	0.9806		1	0.9162	0.9682	0.9761
1	0.8413	0.9527	0.9750		1	0.8855	0.9516	0.9864
0	0.1325	0.0365	0.0221		0	0.0697	0.0290	0.0252

Tableaux (I.2) : *Nombre des itérations.*

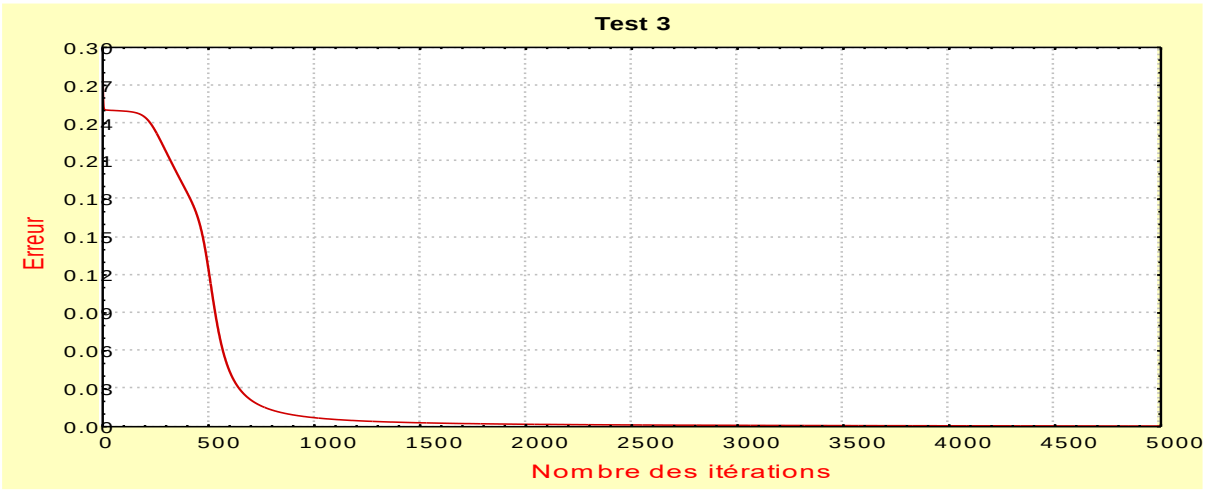
D'après les résultat obtenus dans les figures (1.3) et (I.4). Le nombre d'itération est un facteur très important, il peut influer l'apprentissage positivement ou négativement (cas de sur apprentissage) **[10]**



A

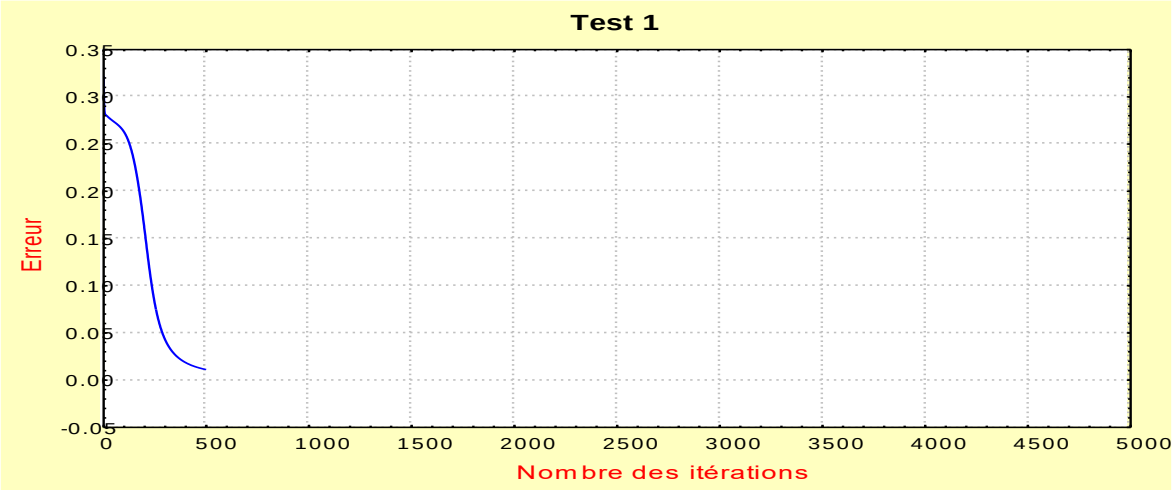


B

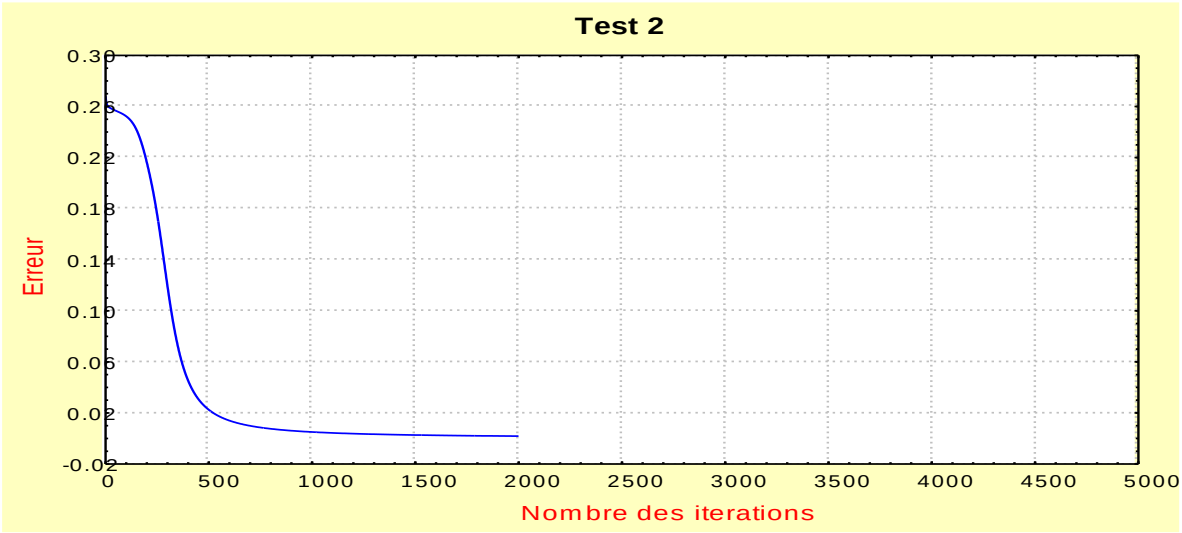


C

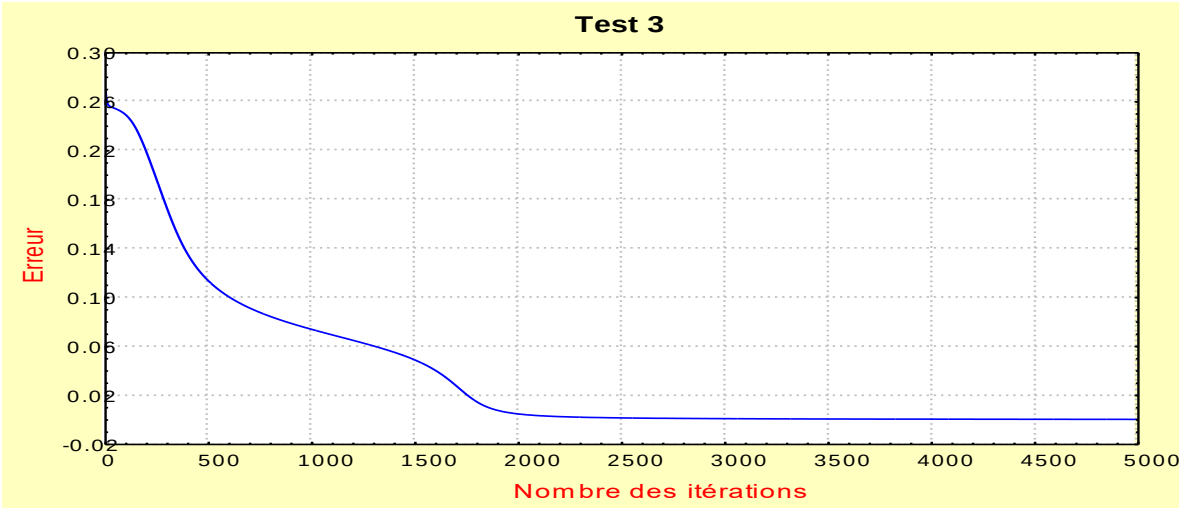
Figure (I.3): *Nombre des itérations (réseaux d'ordre un).*



A



B



C

Figure (I.4): *Nombre des itérations (réseaux d'ordre deux).*

I.2.1.3 Taux d'apprentissage :

L'architecture du réseau est la suivante :

Réseau d'ordre 1

- Taille des couches : $N^3_{2\,4\,1}$
- Nombre des itérations : 10^5

Réseau d'ordre 2

- Taille des couches : $N^3_{2\,2\,1}$
- Nombre des itérations : 5000

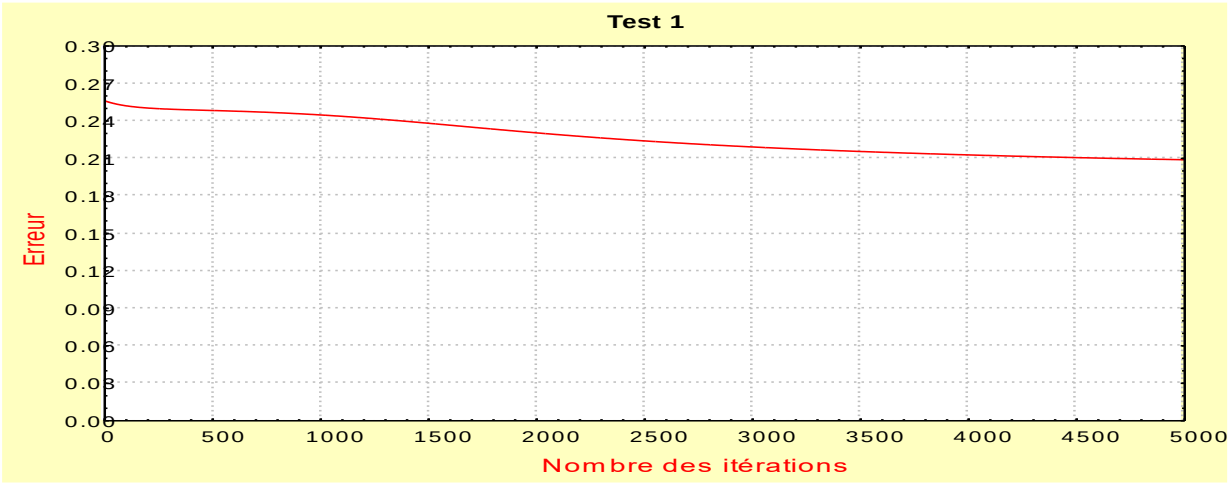
On a fait 3 tests différents, chaque fois on modifie le taux d'apprentissage comme suit :

Test	*Taux 'apprentissage d'ordre un*	*Taux 'apprentissage d'ordre un*
1	0.1	0.1
2	0.8	0.5
3	0.9	0.8

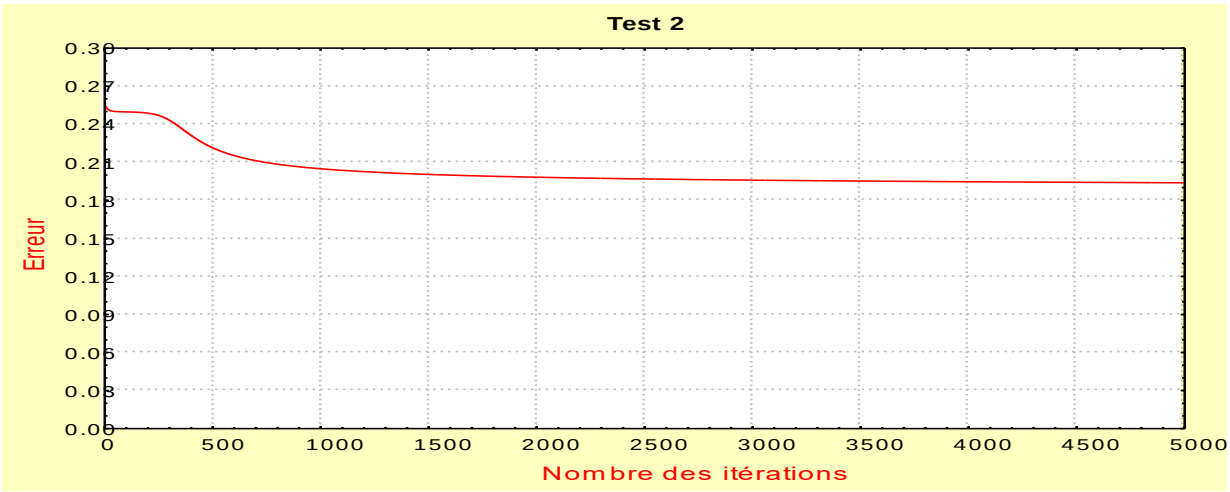
RESEAU D'ORDRE 1					**RESEAU D'ORDRE 2**			
Sortie désirée	**Test 1**	**Test 2**	**Test 3**		**Sortie désirée**	**Test 1**	**Test 2**	**Test 3**
0	0.1706	0.0685	0.0630		0	0.2693	0.2693	0.2990
1	0.4718	0.4897	0.4908		1	0.7046	0.7046	0.7402
1	0.4720	0.4897	0.4908		1	0.8316	0.8316	0.9989
0	0.4989	0.4999	0.4999		0	0.0002	0.0002	0.0873

Tableaux (I.3) : *Taux d'apprentissage.*

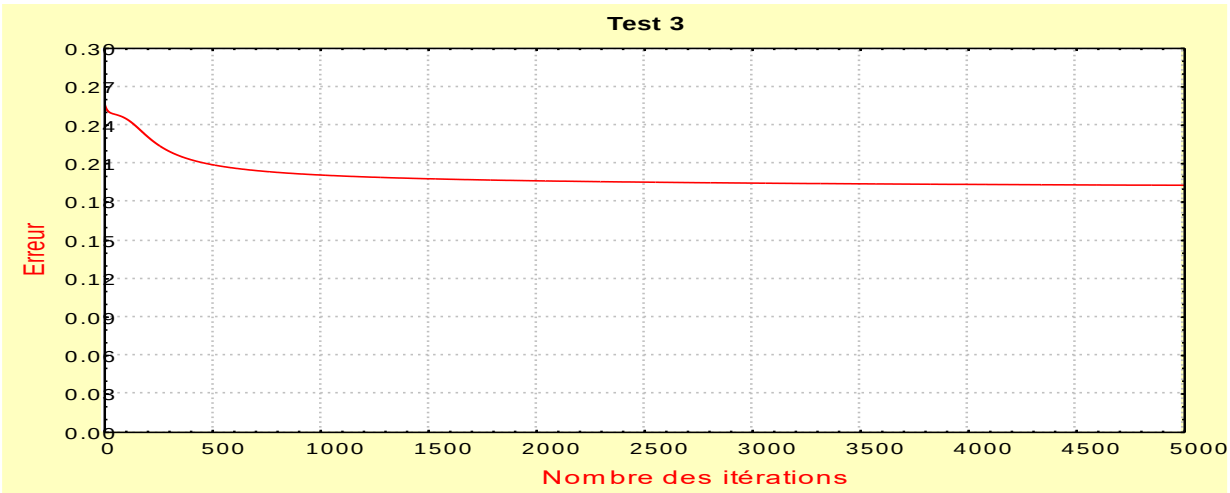
Comme pour le nombre d'itérations et les poids initiaux, le choix du taux d'apprentissage est très délicat, car il n'y a pas une méthode précise pour fixer ce taux d'apprentissage **[9],** ainsi qu'un bon choix de résultats, augmente la rapidité d'apprentissage.



A

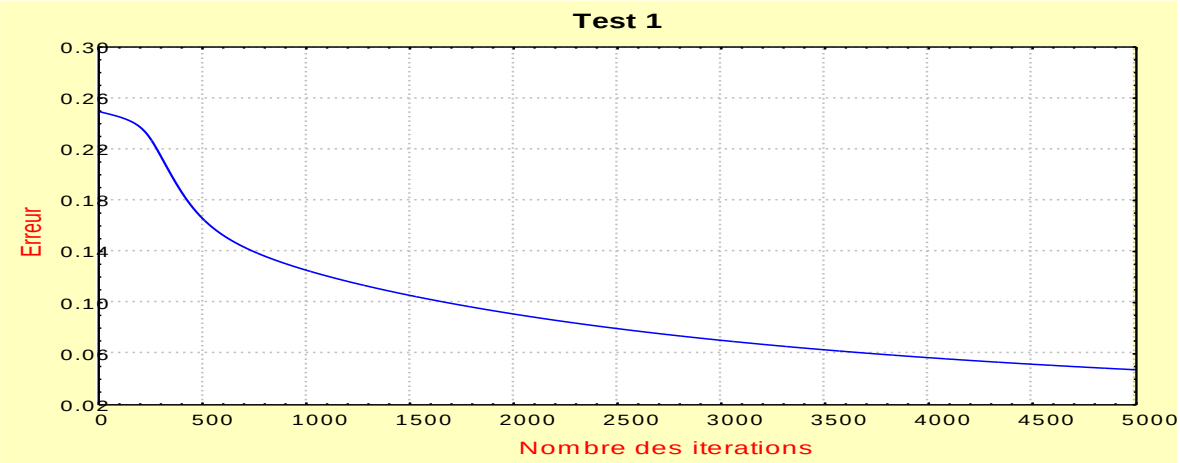


B

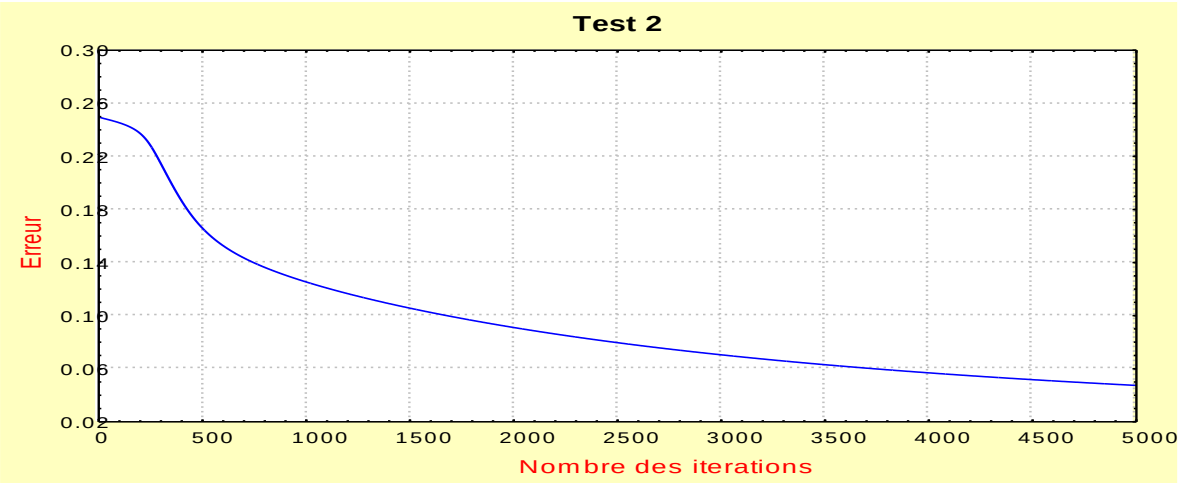


C

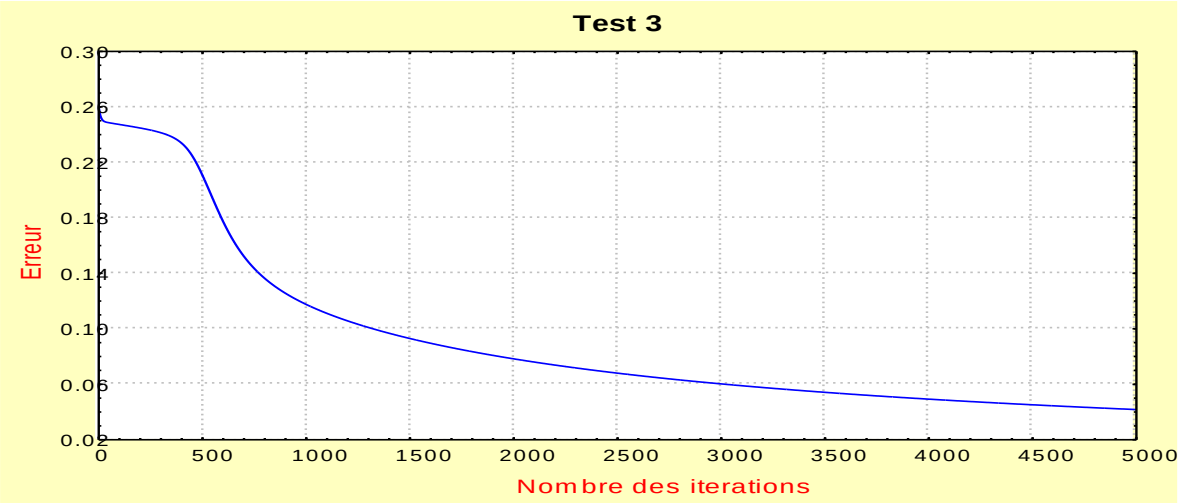
Figure (I.5): *Taux* d'apprentissage (Réseaux d'ordre un)



A



B



C

Figure (I.6): *Taux* d'apprentissage (Réseaux d'ordre deux)

I.2.1.4 La taille du réseau :

L'architecture du réseau est la suivante :

Réseau d'ordre 1

- Taux d'apprentissage : 0.99
- Nombre des itérations : 10^5

Réseau d'ordre 2

- Taux d'apprentissage : 0.99
- Nombre des itérations : 5000

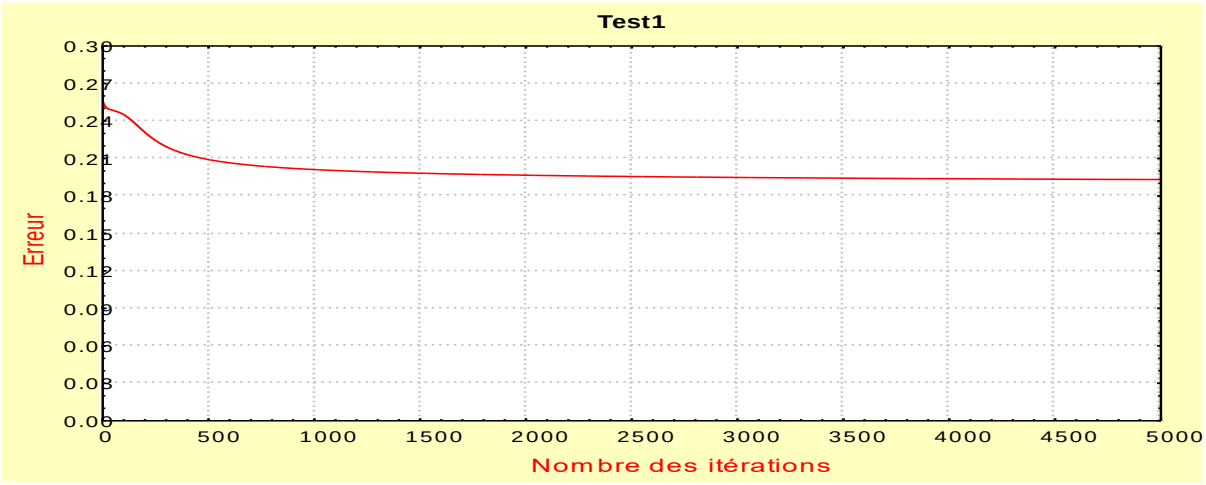
On a fait 3 tests différents, chaque fois on modifie le nombre de neurone dans la couche cachée comme suite :

Test	*Nombre de neurones dans la couche cachée (d'ordre 1)*	*Nombre de neurones dans la couche cachée (d'ordre 2)*
1	1	1
2	2	2
3	3	3

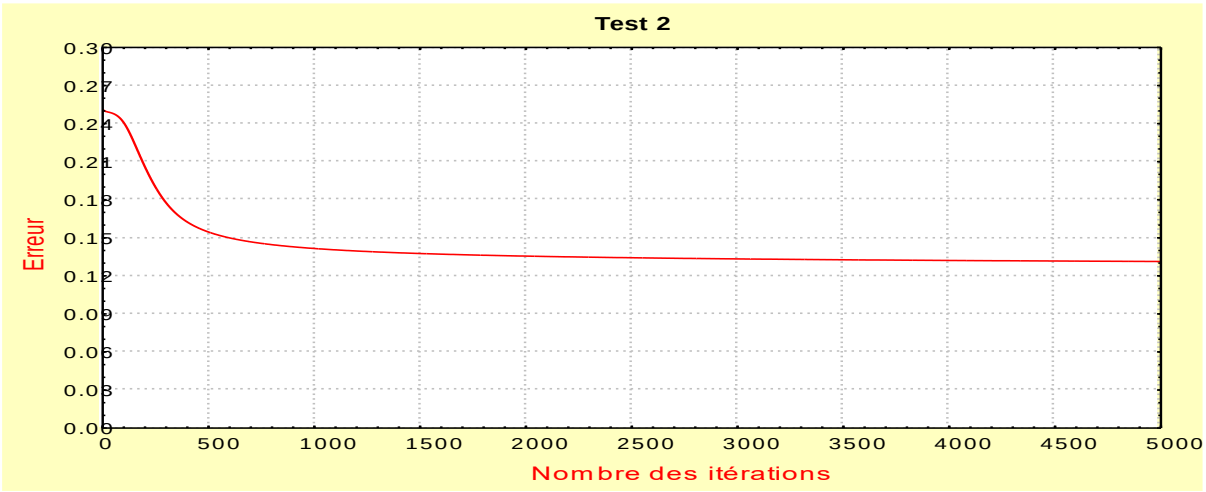
RESEAU D'ORDRE 1					**RESEAU D'ORDRE 2**			
Sortie désirée	**Test 1**	**Test 2**	**Test 3**		**Sortie désirée**	**Test 1**	**Test 2**	**Test 3**
0	0.0630	0.0431	0.0491		0	0.2990	0.0319	0.0226
1	0.4908	0.9488	0.9708		1	0.7402	0.9817	0.9803
1	0.4908	0.4885	0.9708		1	0.9989	0.9721	0.9821
0	0.4999	0.5091	0.0063		0	0.0873	0.0166	0.0152

Tableaux (I.4) : *Taux d'apprentissage.*

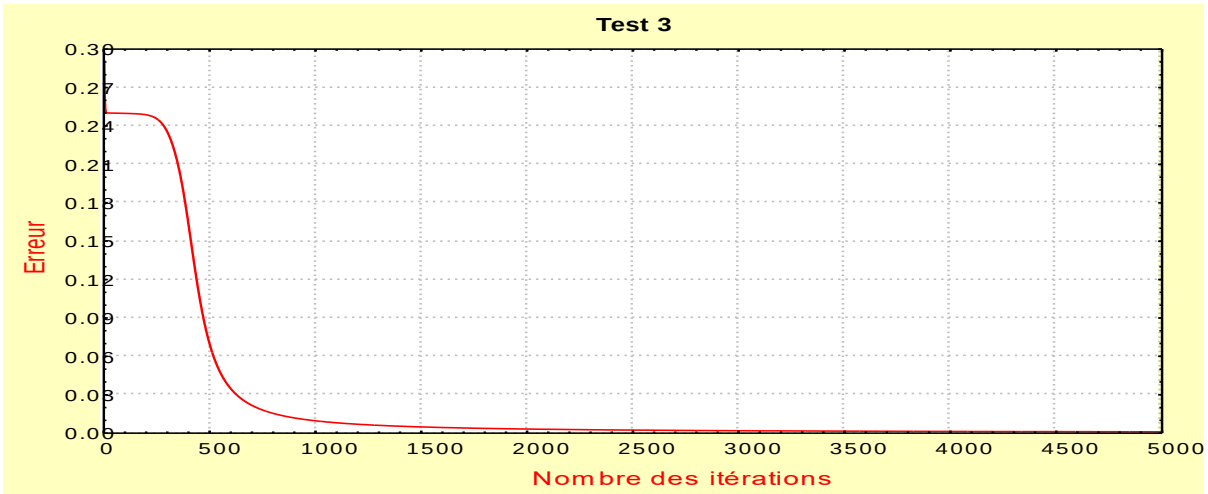
Les études sur le choix du nombre de couches cachées et du nombre de neurones par couches démontrent qu'il n'existe aucune méthode entièrement automatisable qui permette de trouver le nombres de couches cachées et de neurones par couche **[9]**.



A

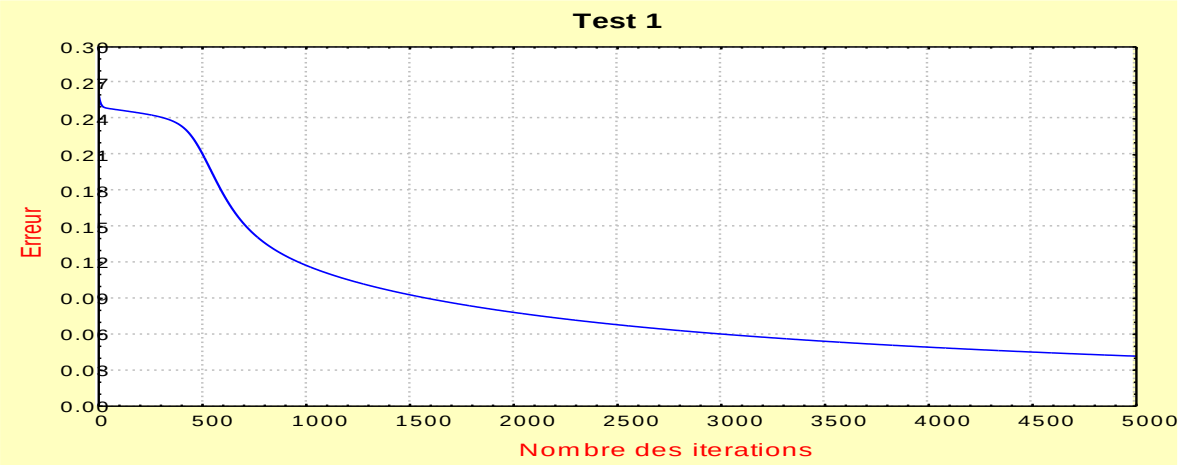


B

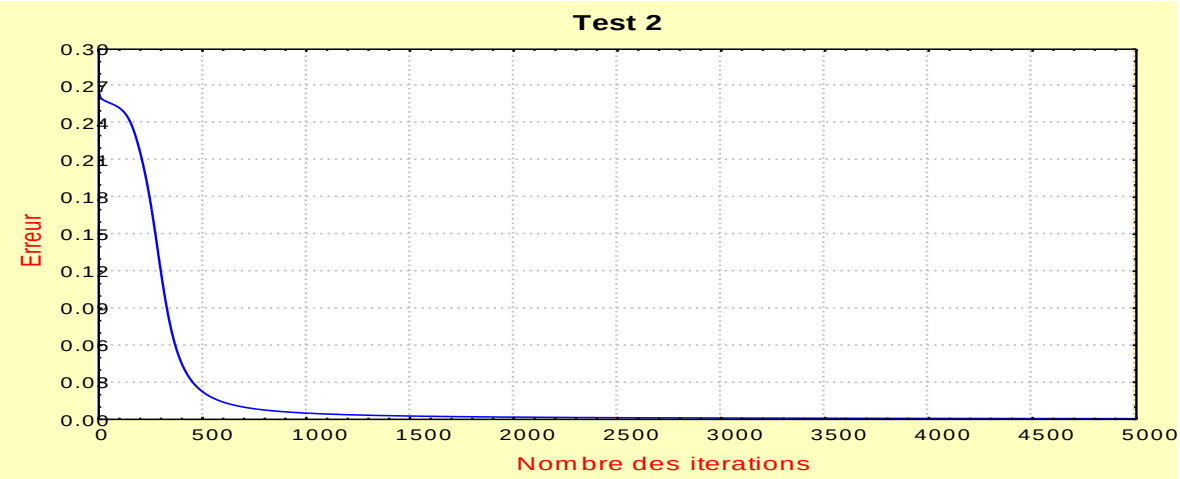


C

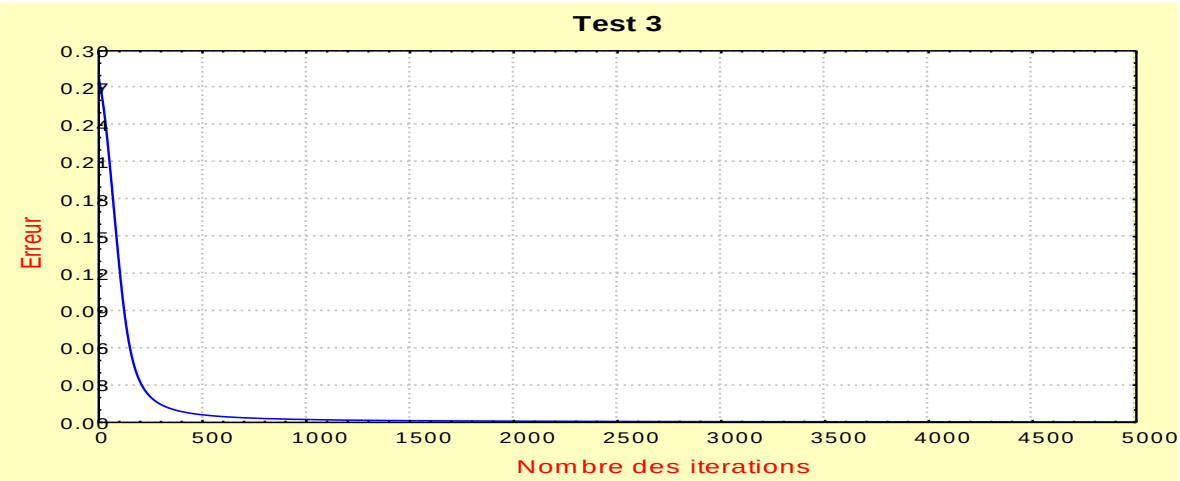
Figure (I.7): *Taille du Réseau (Réseau d'ordre un).*



A



B



C

Figure (I.8): *Taille du Réseau (Réseau d'ordre deux).*

I.2.2 Tests par les différents réseaux de neurones :

L'ordre du réseau	Taille du réseau	Taux d'apprentissage	Nbr_d'iteration	Résultat d'apprentissage
Un	N^3 2 2 1	0.99	50000	Test 1
Deux	N^3 2 2 1	0.99	50000	Test 2

Tableau (I.5): *les caractéristiques d'apprentissage*

SORTIE DESIREE	TEST 1	TEST 2
0	0.0198	0.0151
1	0.9751	0.9907
1	0.9460	0.9925
0	0.5030	0.0081

Tableaux (I.6): *les résultats d'apprentissage par les différents réseaux de neurones*

D'après ces résultats, on pourra conclure que les réseaux statiques d'ordre deux donne des bons résultats par rapport au réseau statique d'ordre un.

I.3 Filtrage d'images :

Les résultats obtenue avec les réseaux de neurones d'ordre deux.

Réseau d'ordre 2

- Taille des couches : $N^3_{1\,6\,1}$
- Nombre d'itération : 500000
- Taux d'apprentissage : 0.99

Image d'origine

Image Identique

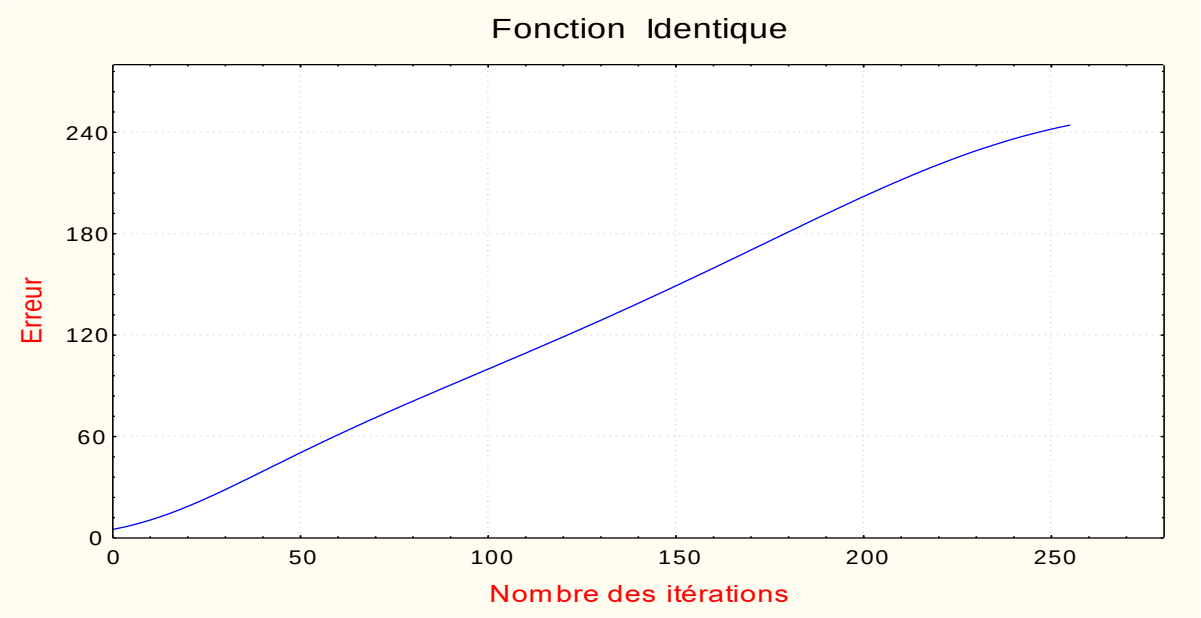


Figure (I.9): Fonction Identique (Réseaux d'ordre deux).

Image avec 5% de bruit

Image Filtrer

: Neurone d'entrée
Couche l

Conclusion :

Les réseaux de neurones peuvent être très avantageux dans le traitement d'images avec les problèmes pour lesquels on n'a pas d'algorithmes permettant de les résoudre ou pour lesquels le temps de traitement est trop long. Leur utilisation sur une machine neuronale permettrait d'effectuer du traitement d'images en temps réel. Ils sont utilisés pour la segmentation, le filtrage, la reconnaissance de forme... etc

Dans notre projet, ils sont utilisés pour le traitement bas niveau (filtrage d'images).

Lors de ce projet, on a vu que les paramètres d'initialisation comme le Nombre d'itération, le taux d'apprentissage, initialisation des poids, la taille du réseau jouent un rôle très important dans la vitesse d'apprentissage et sa réussite mais le problème c'est qu'il n'existe une méthode précise pour fixer ces paramètres, à la fin on a conclu que les réseaux de neurones multicouches donnent des résultats satisfaisants dans le filtrage d'images , et que les réseaux d'ordre supérieur donnent des meilleurs résultats par rapport au réseaux de neurones d'ordre un.

On peut citer quelques avantages et quelques inconvénients :

Avantage :

- On peut les appliquer à n'importe quelle application.
- Ils sont les plus pertinents pour fournir une approximation efficace de fonctions non linéaires sans connaissance préalable des fonctions qui modélisent les relations étudiées

Inconvénient :

- Ce sont les techniques les plus coûteuses en termes de temps de traitements.
- Il n'y a pas des méthodes précises pour fixer les paramètres d'initialisation.

En fin nous conseillons les étudiants des prochaines promotions qui s'intéressent au domaine de réseaux de neurones qu'il serait intéressant de reprendre dans le cadre des recherches d'étudier d'autre modèles de réseaux de neurones (dynamique, modulaire..) de les évoluer et de les comparer.