REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR

ET DE LA RECHERCHE SCIENTIFIQUE

THESE PRESENTEE

POUR L'OBTENTION DU GRADE DE

MAGISTER

A L'UNIVERSITE DE BLIDA

SPECIALITE: Electronique

OPTION: Communication

PAR Mr. KHORISSI NASR-EDDINE

SUJET DE LA THESE

SIMULATION DE RESEAUX DE NEURONES APPLICATION A LA RECONNAISSANCE DES CARACTERES ARABES

SOUTENUE LE: 29 Juin 1995

Devant le jury composé de:

Mr. BELKHAMZA N.	(M.C)	Inst. d'électronique	Président
Mr. GUESSOUM A.	(M.C)	Inst. d'électronique	Rapporteur
Mr. MELIANI H.	(M.C)	Inst. d'électronique	Examinateur
Mr. NAMANE A.	(C.C)	Inst. d'éléctronique	Examinateur
Mr. MANSEUR S.	(C.C)	Inst. de mathématique	Examinateur

JUIN 1995

REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR

ET DE LA RECHERCHE SCIENTIFIQUE

THESE PRESENTEE

POUR L'OBTENTION DU GRADE DE

MAGISTER

A L'UNIVERSITE DE BLIDA

SPECIALITE: Electronique

OPTION: Communication

PAR Mr. KHORISSI NASR-EDDINE

SUJET DE LA THESE

SIMULATION DE RESEAUX DE NEURONES APPLICATION A LA RECONNAISSANCE DES CARACTERES ARABES

SOUTENUE LE: 29 Juin 1995

Devant le jury composé de:

Mr. BELKHAMZA	N.	(M.C)	Inst. d'électronique	Président
Mr. GUESSOUM	A.	(M.C)	Inst. d'électronique	Rapporteur
Mr. MELIANI	H.	(M.C)	Inst. d'électronique	Examinateur
Mr. NAMANE	A.	(C.C)	Inst. d'éléctronique	Examinateur
Mr. MANSEUR	S.	(C.C)	Inst. de mathématique	Examinateur

JUIN 1995

REMERCIEMENTS

Je tiens à formuler en ces lignes l'expression de ma profonde et sincère gratitude à mon directeur de thèse MONSIEUR Abderrezak GUESSOUM pour toute sa disponibilité, sa tolérance et sa grande contribution à ma formation.

J'exprime ma gratitude à MONSIEUR Nour-Eddine BELKHAMZA pour l'honneur qu'il me fait d'accepter la présidence du jury de cette thèse.

Je suis sensible à l'honneur que me font MONSIEUR Hamza MELIANI, MONSIEUR Abderrahmane NAMANE, MONSIEUR Salah MANSEUR, pour avoir accepter de lire, d'examiner ce travail et de participer à ce jury.

Je tiens également à remercier MONSIEUR Samir BOUBEKEUR pour ses encouragements et ses aides précieuses.

Je dédie ce travail à tous les membres de ma famille qui m'ont soutenu.

SOMMAIRE

INTRODUCTION

Sous le terme de réseaux de neurones, on regroupe aujourd'hui un certain nombre de modèles dont l'intention est d'imiter quelques fonctions du cerveau humain en reproduisant certaines de ses structures de base [11]. Historiquement, les origines de cette discipline sont très diversifiées.

En 1943,Mc Culloch et Pitts étudièrent un ensemble de neurones formels interconnectés , et montrèrent leurs capacités à calculer certaines fonctions logiques .

En 1949 , Hebb ,dans une perspective psycho-physiologique , souligna l'importance du couplage synaptique dans les processus d'apprentissage

En 1958 , Rosenblatt décrivit le premier modèle opérationnel de réseaux de neurones, mettant en oeuvre les idées de Hebb ,Mc Culloch et Pitts. Il introduisait le Perceptron inspiré du système visuel, capable d'apprendre à calculer des fonctions logiques en modifiant les connexions synaptiques.

Ce modèle suscita beaucoup de recherches et sans doute trop d'espoirs. Lorsque deux mathématiciens, Minsky et Papert, démontrèrent en 1969 les limites théoriques du perceptron. Les chercheurs se désintéressent des réseaux de neurones pour se tourner vers l'approche symbolique de l'intelligence artificielle, qui semblait beaucoup plus prometteuse.

Le renouveau actuel des réseaux de neurones est du à des contributions originales, comme celle de Hopfield en 1982, qui en montrant l'analogie des réseaux de neurones avec certains systèmes physiques, a permis de leur appliquer un formalisme riche et bien maîtrisé.

En 1985, des nouveaux modèles mathématiques, tel que la rétropropagation du gradient, ont permis de dépasser les limites du perceptron.

Actuellement les premières applications pratiques des réseaux de neurones commencent à voir le jour. Cette discipline concerne un public de plus en plus large d'étudiants, de chercheurs, d'ingénieurs et d'industriels. Plusieurs articles ont été publiés sur ce sujet, dans des revues de biologie, de mathématique, de physique et d'électronique. Chacun abordant le problème dans le cadre de sa spécialité.

L'intérêt porté aujourd'hui aux réseaux de neurones tient sa justification dans les quelques propriétés qu'ils possèdent et qui devraient permettre de dépasser les limitations de l'informatique traditionnelle.

Ce nouveau mode de traitement de l'information parait très prometteur pour résoudre un certain nombre d'applications jusqu'alors très difficiles. Parmi les propriétés, il faut citer:

- Le parallélisme qui permet d'obtenir des temps de réponses extrêmement rapides.
- L'apprentissage à partir d'éxemples et la généralisation aux cas non appris.
- La résistance aux bruits dans l'information reçue.
- La tolérance aux défauts.

Bien que le passage de la simulation à l'implantation sur hardware n'en est encore qu'à ses débuts, les réseaux de neurones commencent à être réalisés en VLSI [21].

Les domaines d'applications privilégiés sont la reconnaissance des formes (vision, caractères manuscrits), le traitement du signal, la classification, la robotique et la reconnaissance de la parole.

Mon travail s'inscrit dans le cadre du projet de recherche: "Développement d'un système de réseaux de neurones appliqué au traitement des signaux sismiques " [N°:J0901/01/05/92]. Il consiste à simuler deux modèles de réseaux de neurones:

- La mémoire associative (modèle de Hopfield).
- Le perceptron multicouches (réseau MLP).

Dans notre application, les réseaux subissent l'apprentissage de quelques caractères arabes représentés numériquement, soit directement par les états de tous les pixels de l'image du caractère (codage rétinien), soit par un codage de l'image par la méthode des moments. Les expériences sont organisées en deux parties: La première sera consacrée aux mémoires associatives, la deuxième au réseau MLP.

Dans le chapitre 1 on essayera de présenter les modélisations basées sur les fondements biologiques.

Le chapitre 2 sera consacré à la présentation des deux modèles étudiés avec leurs algorithmes d'apprentissage et de reconnaissance.

Dans le chapitre 3 on présentera le codage par les moments invariants, et les moments de Zernike.

Tout le protocole éxpérimental ainsi que les résultats de la classification seront discutés dans le chapitre 4.

Enfin, le chapitre 5 sera consacré à la conclusion générale de notre étude. Le lecteur intéréssé trouvera aussi en ANNEXE A quelques applications déja réalisées avec les réseaux de neurones.

CHAPITRE 1
MODELISATIONS

1.1. LE NEURONE BIOLOGIQUE

Les cellules nerveuses, appelées neurones, sont les éléments de base du système nerveux central. Celui-ci en possède environ cent milliards. On estime le nombre des interconnections à 10^{15} [1]. Les neurones reçoivent, propagent et transmettent des signaux. Ils sont le siège d'excitations électriques, appelées potentiel d'action ou influx nerveux qui se propage à 100 m/s.

1.1.1. STRUCTURE DES NEURONES

Le neurone se compose de trois parties: (fig.1.1)

Le corps cellulaire: c'est le centre de la synthèse biologique. Ayant un diamètre de quelques microns, le corps cellulaire contient le noyau du neurone et effectue les transformations biochimiques nécessaires à la synthèse des enzymes et des autres molécules qui assurent la vie du neurone.

Les dendrites: Chaque neurone possède une chevelure de dendrites. Celles-ci sont de fines extentions tibulaires de quelques dizièmes de microns de diamètre et d'une longueur de quelques dizaines de microns. Elles sont les récepteurs principaux du neurone pour capter les signaux qui lui parviennent.

L'axone: c'est une fibre nerveuse qui sert de moyen de transport pour les signaux ou les potentiels d'action émis par le neurone (fig.1.2). Il est généralement plus long que les dendrites. Sa longueur varie d'un millimètre à plus d'un mètre. Il se ramifie à son extremité là où il communique avec d'autres neurones.

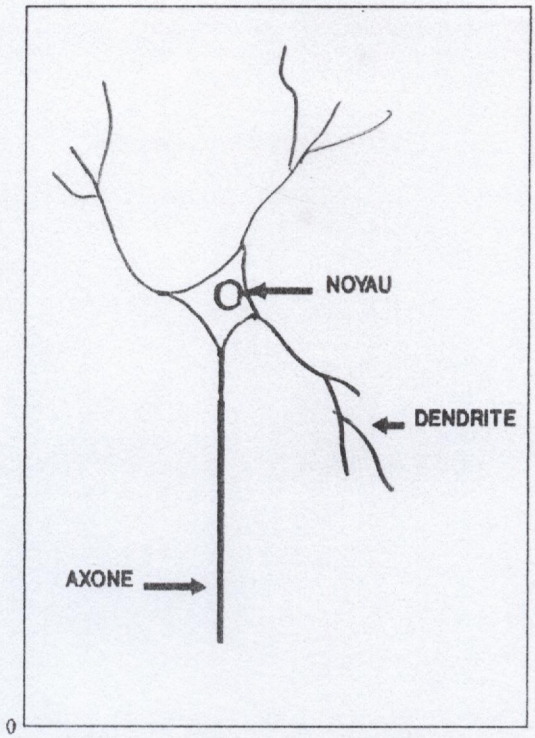


Fig.1.1: schéma d'un neurone

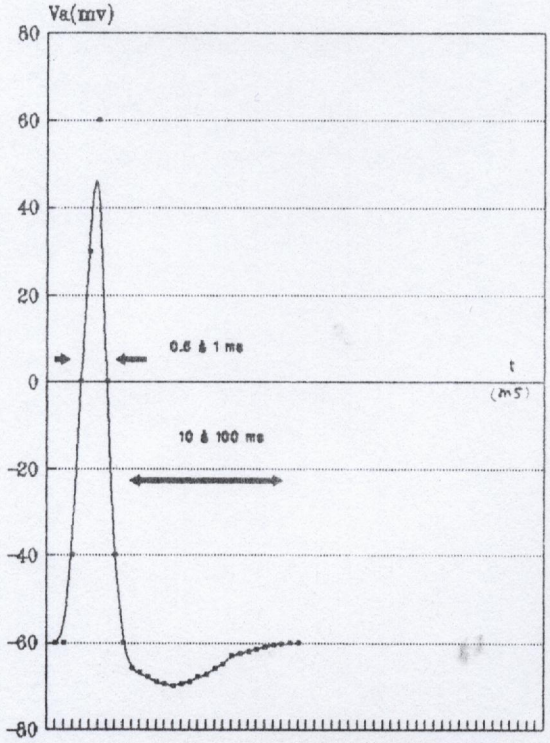


Fig.1.2: Le potentiel d'action.

Pour former le système nerveux, les neurones sont connectés les uns aux autres suivant des répartitions spatiales complexes. Les connections entre deux neurones se font en des endroits appelés synapses (fig.1.3).

1.1.2. FONCTIONNEMENT DES NEURONES

Toutes les fonctions réalisées par un neurone dépendent essentiellement des propriétés de sa membrane externe. Cette membrane sert à propager les impulsions électriques tout au long de l'axone et des dendrites. Elle permet aussi de libérer les médiateurs à l'extémité de l'axone, comme elle réagit à ces médiateurs au niveau des dendrites. Au niveau du corps cellulaire, la membrane réagit aux impulsions électriques que lui transmettent les dendrites pour générer ou non une nouvelle impulsion. Au cours de la formation du cerveau, la membrane externe permet au neurone d'identifier son environnement afin qu'il puisse trouver les neurones auxquels il doit être connecté.
Le schéma classique du corps cellulaire présenté par les biologistes est celui d'un soma effectuant une sommation des influx nerveux transmis par ses dendrites (fig.1.4). Si cette sommation dépasse un seuil, le neurone répond par un influx nerveux ou potentiel d'action qui se propage le long de son axone. Si cette sommation est inférieure à ce seuil, le neurone reste inactif. L'influx nerveux qui se propage entre différents neurones est un phénomène électrique.

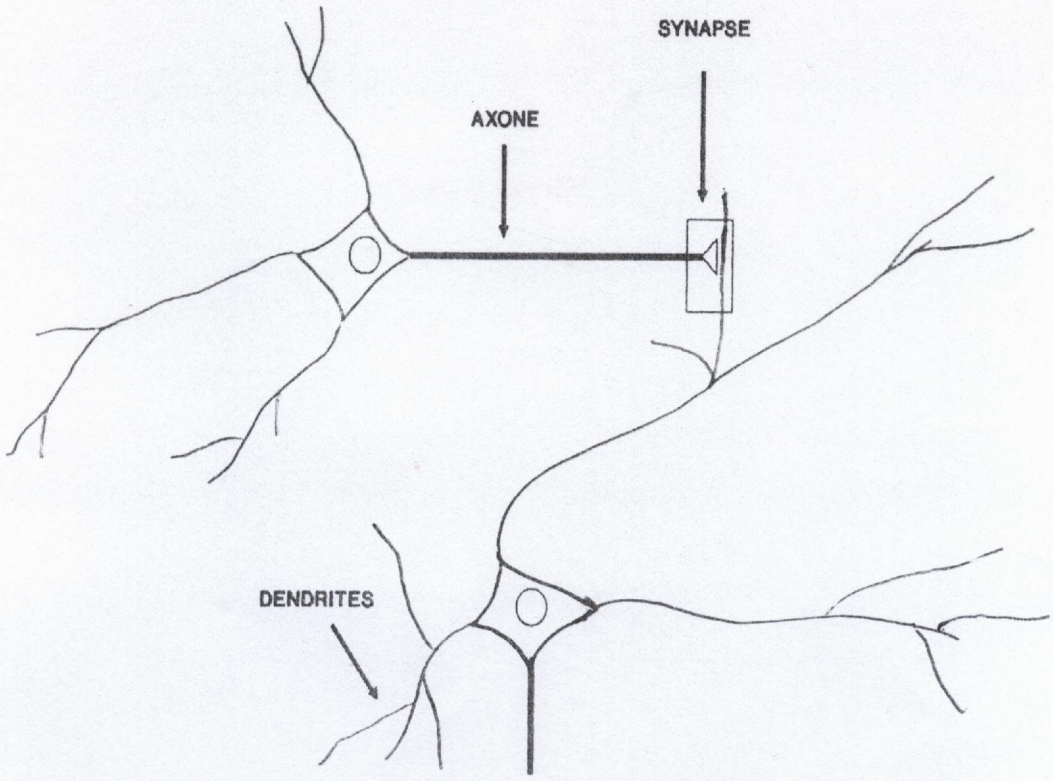


Fig.1.3: Schéma d'une synapse

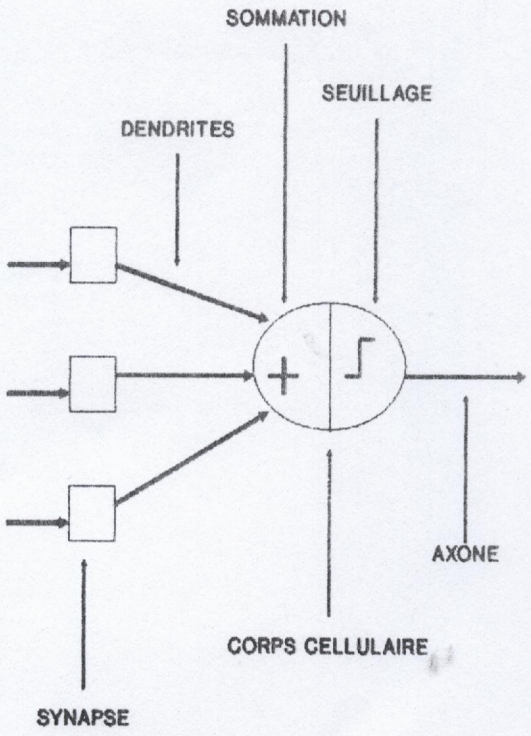


Fig.1.4: Neurone sommateur à seuil

1.1.3. MEMOIRE ET APPRENTISSAGE

On peut définir l'apprentissage comme l'acquisition de propriétés associatives stables considérées comme des modifications du comportement que l'on peut attribuer à l'expérience sensorielle passée de l'organisme [7], et la mémorisation comme l'engrangement et le rappel de ces propriétés ou modifications.

La découverte du stockage de notre code génétique dans les molécules d'ADN, a laissé les neuro-biologistes établir une analogie entre le mode de stockage de notre mémoire innée et celui de notre mémoire acquise. L'idée qui domine est celle d'un stockage dans les macro-molécules du cerveau; les protéines ou l'ARN.

L'évolution du système nerveux est due à l'interaction entre l'environnement extérieur et le programme génétique. Cette évolution se traduit par l'évolution des synapses qui soit dégénèrent, soit se stabilisent d'une manière plus ou moins réversible.

L'apprentissage, comme la mémorisation, ne peut se caractériser au niveau biochimique que par l'évolution des connexions entre les neurones. Si les connexions n'ont pas une stabilisation définitive, cela peut être grave pour un système qui ne crée plus de nouvelles connexions comme c'est le cas chez l'adulte.

1.2. LES MODELISATIONS

1.2.1. LE NEURONE FORMEL DE Mac de Culloch et Pitts.[20]

Dans ce modèle le neurone formèl effectue la somme pondérée des potentiels d'actions qui lui parviennent. Si cette somme dépasse un certain seuil, le neurone est activé et transmet une réponse dont la valeur est celle de son activation, sinon le neurone n'est pas activé et ne transmet rien (Fig.1.5).

Pour une modélisation générale, on peut définir le neurone formel par les éléments suivants: la nature de ses entrées, la fonction d'entrée totale qui constitue un prétraitement effectué sur les entrées, la fonction d'activation (d'état ou de transfert) du neurone qui définit son état interne en fonction de son entrée totale, la fonction de sortie qui calcule la sortie du neurone en fonction de son activation. Les entrées et la sortie peuvent être:

binaires: (-1,1) ou (0,1).

réelles.

La fonction d'entrée totale est en générale linéaire:

$$E = \sum_i W_i . e_i \qquad (1.1)$$

avec;

W_i: poids de la ième connexion.

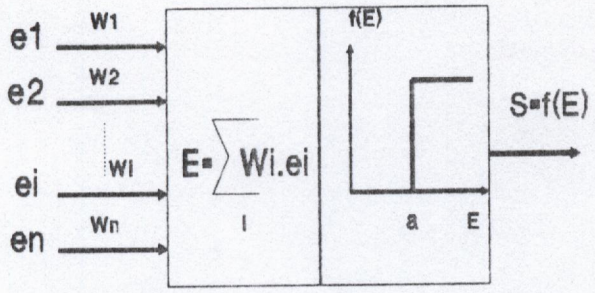
e_i: ième entrée.

On utilise parfois une fonction non linéaire, polynomiale de degré supérieur à deux.

La fonction d'activation est choisie généralement croissante et impaire. Dans la (fig.1.6), nous avons représenté les fonctions d'activation les plus utilisées:

binaire à seuil : Soit la fonction Heaviside, soit la fonction signe.

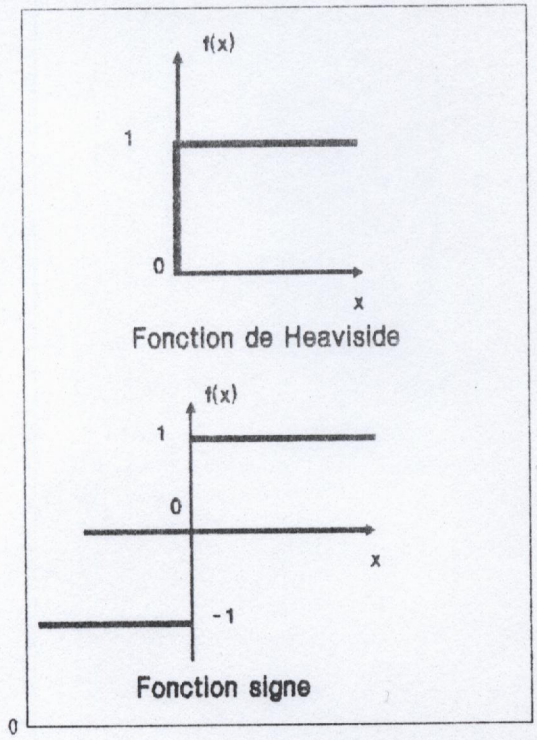
fonction sigmoïde: C'est une fonction non linéaire et surtout dérivable ce qui facilite le calcul de la variation des poids synaptiques lors de l'apprentissage. La fonction sigmoïde est très utilisée dans les algorithmes de la rétropropagation du gradient.



a:seuil

Fig.1.5: Le neurone formel.

$$f(x) = a \frac{\exp(kx) - 1}{\exp(kx) + 1}$$



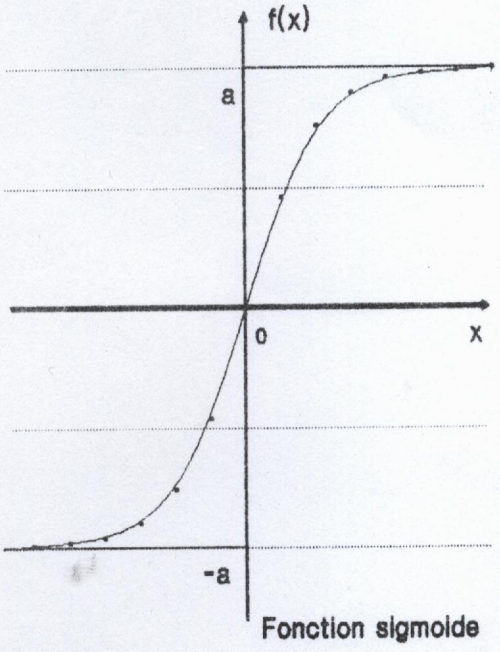


Fig.1.6: Fonctions d'activation

1.2.2. LE PERCEPTRON

Le perceptron [5],[19],[24] est le premier modèle solide présenté par Rosenblatt qui semble en accord avec les faits biologiques, puisqu'il est inspiré du système visuel. Le perceptron comprend trois éléments principaux: (Fig.1.7)

- La rétine:

Elle est Constituée de cellules sur lesquelles s'inscrit le stimulus. La réponse fournie par ces cellules est modulée suivant l'intensité du stimulus. Pour notre cas, on peut assimiler ces cellules à des photodiodes.

- La couche de cellules d'association:

Chacune de ces cellules peut être connectée à des cellules de la rétine et aussi à des cellules de décision. Ces cellules fonctionnent exactement comme les modèles de neurones formels. Le sens des connections se fait de la rétine vers les cellules d'association.

- La couche de cellules de décision:

Les cellules de décision reçoivent leurs entrées des cellules d'association ou d'autres cellules de décision. Elles représentent la couche de sortie du perceptron. Le sens des connections avec les cellules d'association est bi-directionnel, ce qui permet un feed-back de la sortie sur le réseau.

Pour faciliter l'étude, on utilise le perceptron simple (Fig.1.8) pour pour lequel:

-Les valeurs de sorties de toutes les cellules sont binaires (0,1).

-Les fonctions réalisées par les cellules d'association sont booléenne

-Les cellules de décision sont des neurones à seuil.

-Les connections sont à sens unique.

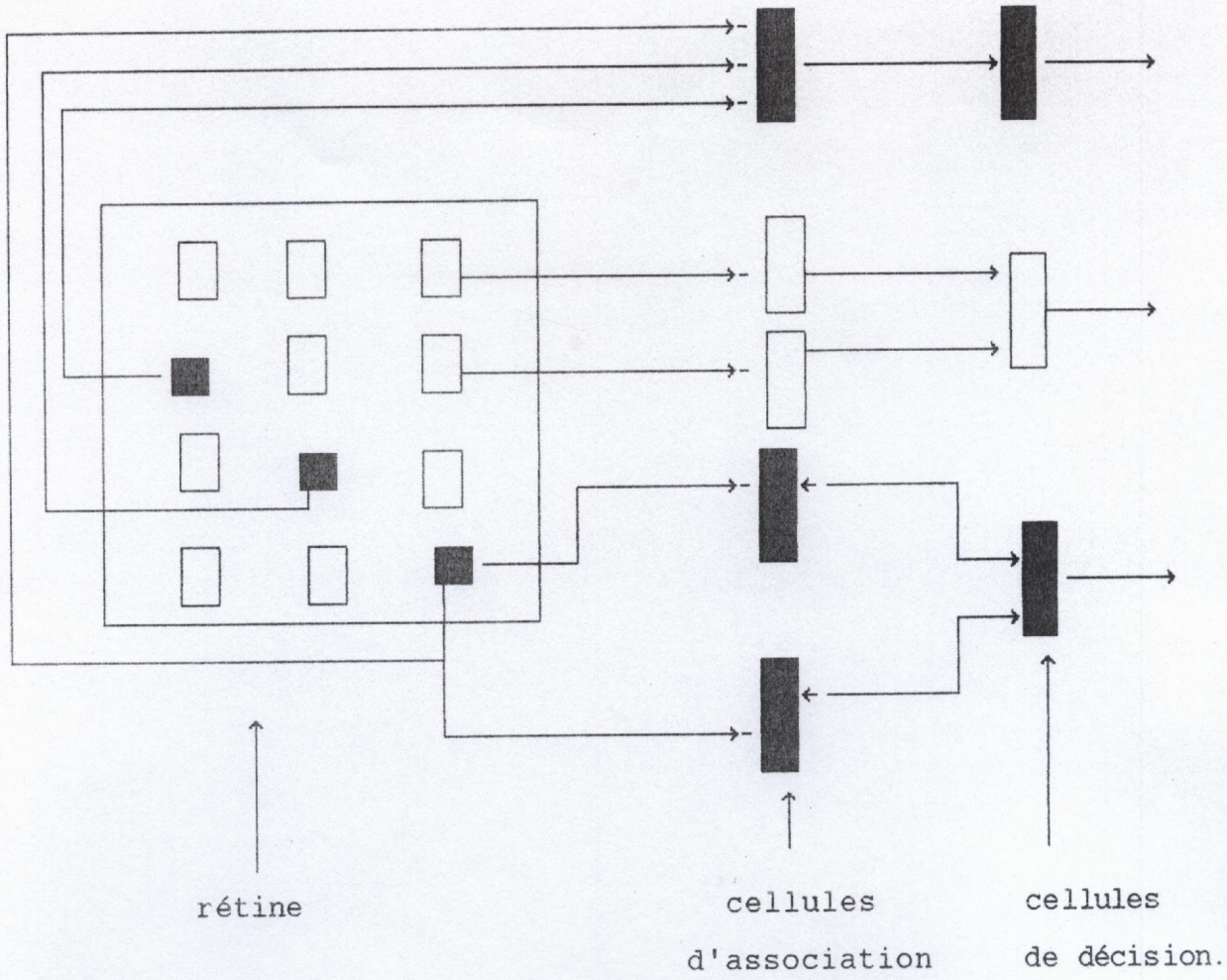


Fig.1.7: Schéma du Perceptron.

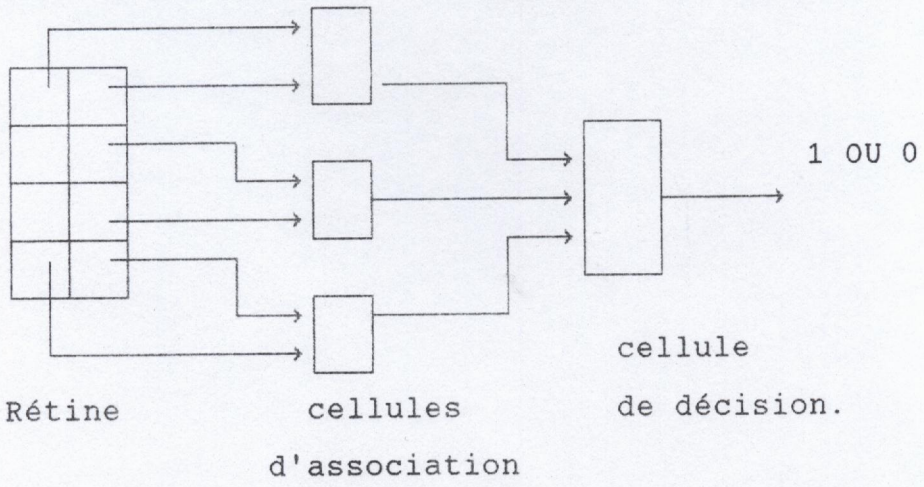


Fig.1.8: Perceptron simple.

1.2.3. LA DYNAMIQUE D'UN RESEAU DE NEURONES

La dynamique des états s'intéresse à l'évolution des états des différents neurones. Elle cherche à déterminer l'existence d'états stables ou de cycles stables pour un ensembles de neurones ou pour le réseau entier. Ceci s'éxplique par la stabilité des relations entre certains stimumus auquels est soumis le cerveau et certains comportements. Ce qui nous permet de dire que le réseau de neurones aboutit à des configurations associatives stables.

D'autre part, les connexions entre neurones(synapses) sont modélisées par des poids qui pondèrent les signaux transmis. On parle alors de dynamique des connexions pour les réseaux dont les poids sont évolutifs.

1.2.4. ALGORITHMES D'APPRENTISSAGE

Si on prend comme exemple le problème de la classification de p objets représentés par leurs formes f1 ,f2 ,....., fp. La forme fp peut être le vecteur code de l'image de l'objet p. Pour classer les p formes dans deux classes différentes C^+ et C^-, on peut utiliser un neurone formel à seuil ayant un nombre d'entrées égale au nombre de composantes du vecteur forme. Le neurone doit répondre par +1 si la forme $\in$ à C^+ et par -1 si la forme $\in$ à C^-. A chaque présentation d'une forme à l'entrée du réseau on doit eventuellement corriger les poids des connexions pour obtenir la bonne réponse à la sortie du neurone.

Ce type d'apprentissage par erreur/correction est appelé apprentissage supervisé. Si on considère que la modification des poids est le principe même de l'apprentissage, alors les règles utilisées sont variées et diffèrent par leur source d'inspiration.

a) La règle de Hebb:

La règle de Hebb [9] est le premier mécanisme d'évolution proposé pour les synapses. Qualitativement, l'apprentissage consiste à renforcer la connexion entre deux neurones activés au même moment. Pour les autres cas la connexion reste inchangée.

Si $W_{ij}(t)$ est le poids de la connexion entre le neurone j et le neurone i à l'instant t. Et soient A_j et A_i leurs activations à l'instant t. La règle de Hebb qui décrit la dynamique de cette connexion s'écrit:

$$W_{ij}(t + dt) = W_{ij}(t) + k.A_i.A_j \qquad (1.2)$$

avec:

k; le coefficient qui caractérise la vitesse d'apprentissage (k>0)

A_i , A_j booléens (0 , 1).

D'une façon générale on excite le réseau par un signal à l'entrée (un vecteur d'entrée) et on observe les activations de tous les neurones (leurs sorties). On modifie les poids de toutes les connexions en utilisant la relation (1.2).

b) La règle du Perceptron (La règle delta):

L'apprentissage du perceptron est un apprentissage supervisé.

Prenons comme exemple le perceptron simple. Pour une forme donnée sur la rétine (une image par exemple), le neurone de sortie (cellule de décision) répond par "1" ou "0".

Si nous désignons par S la sortie actuelle du neurone et par d la sortie désirée par le superviseur "le maître", on a les cas suivants:

d = 0 et S = 0 (sortie désirée satisfaite)

d = 1 et S = 1 (" " ")

d = 0 et S = 1 (sortie désirée non satisfaite)

d = 1 et s = 0 (" " " ")

Pour le premier et deuxième cas nous n'avons pas besoin de modifier les poids des connexions.

Pour le troisième cas il faut diminuer la sortie. Pour cela il faut diminuer les poids des connexions pour lesquelles les entrées sont positives et augmenter les poids des connexions pour lesquelles les entrées sont négatives. Faire l'opération inverse pour le 4ème cas.

On peut maintenant écrire la règle de modification des poids:

$$W_i \longrightarrow W_i + k.(d - S).e_i \qquad (1.3)$$

avec: k: coefficient d'apprentissage (k>0)

d: sortie désirée.

S: sortie actuelle.

e_i:entrée de la connexion de poids W_i.

D'après Le Cun [19], s'il existe un ensemble de poids W_i pouvant discriminer les p formes en deux classes. Autrement dit si la séparation est linéaire, alors la procédure (1.3) converge vers la première solution sans l'améliorer. Dans la figure (1.9). nous avons représenté un cas de non séparation linéaire et deux cas de séparation linéaire dont l'une est améliorée.

Dans le cas ou la solution existe, Widrow [31] propose une règle qui justement permet de l'améliorer.

c) La règle de Widrow-Hoff :

La solution [31] est améliorée en faisant une séparation des deux classes par un hyperplan (figure.1.10) qui ne soit pas trop proche de l'une d'elles. Pour cela Widrow modifie les poids avant le seuillage.

$$W_i \longrightarrow W_i + k.(d - \sum_j W_j.e_j).e_i \qquad (1.4)$$

Avec cette règle on peut obtenir une solution approchée lorsque le problème n'est pas linéairement séparable.

1.2.5. LIMITE DU PERCEPTRON SIMPLE MONOCOUCHE

Dans le perceptron simple, la seule couche de poids modifiables est celle qui se trouve entre les cellules d'association et la cellule de décision qui fait le véritable travail de classification. Cette cellule ne peut pas réaliser toutes les fonctions booléennes, en particulier l'exemple classique du XOR illustré dans la figure (1.11). En effet, dans la réprésentation de X, Y, et XOR(X,Y) sur un plan, on voit bien que la cellule ne peut pas séparer linéairement les classes: { (0,0), (1,1) } et { (0,1) , (1,0) }.

1.3. CONCLUSION

Le modèle de neurones formels du perceptron est sujet à une limitation intrinsèque mis en évidence par Minsky et Papert [22]. Il ne peut résoudre que les problèmes linéairement séparable. Ce qui a laissé les chercheurs abondonner cette nouvelle discipline.
Nous allons montrer dans le chapitre qui suit que le problème peut être contourné par l'utilisation du perceptron multicouches dont les poids sont modifiables.

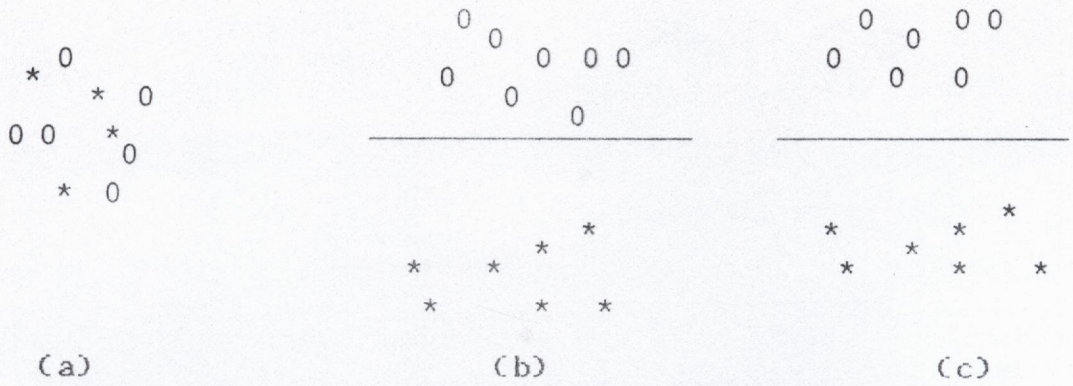


Fig.1.9: Séparation de deux classes

(a): 2 classes non linéairement séparables.

(b): 2 classes linéairement séparables.

(c): 2 classes linéairement séparables (solution améliorée)

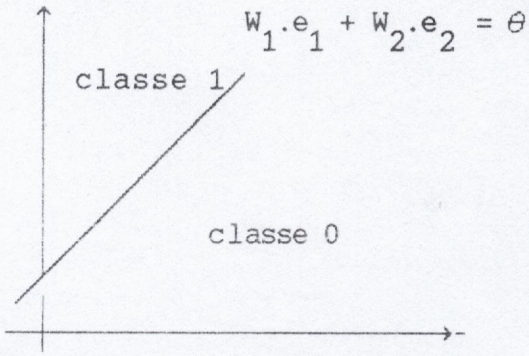


Fig.1.10: Séparation linéaire de 2 classes.

hyperplan à 2 dimensions. θ: seuil du neurone

X	Y	XOR(X,Y)
1	1	0
1	0	1
0	1	1
0	0	0

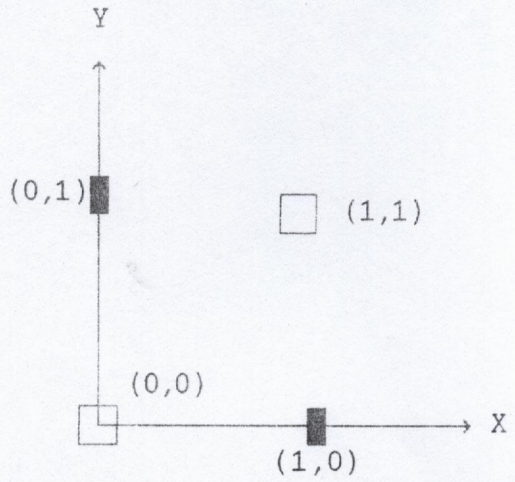


Fig.1.11: Représentation du XOR à 2 dimensions.

En réalité, bien avant le modèle multicouches, John Hopfield a motivé le départ des réseaux de neurones grâce à son article publié en 1982 [12], dans lequel il montre l'analogie des réseaux de neurones avec certains systèmes physique.

Pour une démarche conforme à la réalité historique, nous commencerons par présenter dans le chapitre 2, le modèle de Hopfield puis le modèle multicouches.

CHAPITRE 2

LES ALGORITHMES DE LA MEMOIRE ASSOCIATIVE ET DU MODELE MULTICOUCHES

2.1 LA MEMOIRE ASSOCIATIVE (Modèle de Hopfield).

L'approche de Hopfield [12] est relativement nouvelle, bien qu'elle fasse appel à des résultats connus. Selon lui, le système nerveux recherche des états stables attracteurs dans son éspace d'états. Les états voisins tendent à se rapprocher d'un état stable, ce qui permet la correction des erreurs et la capacité de completer les informations manquantes

Le réseau de Hopfield est donc une mémoire adressable par son contenu. Une forme mémorisée est retrouvée par une stabilisation du réseau s'il a été éxcité par une partie de cette forme.

Hopfield propose un modèle capable de réaliser ces propriétés. Le modèle proposé est un ensemble de neurones de Mc Culloch et Pitts tous interconnectés. La règle utilisée pour l'apprentissage est tout simplement la règle de Hebb.

Une mémoire associative est un système qui permet de réaliser une association entre deux ensembles de formes X et Y.

Exemple si on regarde uniquement les yeux d'une personne qu'on a mémorisé dans notre tête, on reconnaitra tout de suite son visage. On parle alors de mémoire auto-associative.

Dans une mémoire associative l'information n'est pas codée à un endroit précis, elle est distribuée sur tout le réseau.

2.1.1. DESCRIPTION DU MODELE

Le modèle est composé de neurones à seuil totalement interconnectés, fig.(2.1). Les poids d'interconnections vérifient les conditions suivantes:

$$W_{ij} = W_{ji} \quad \text{et} \quad W_{ii} = 0 \qquad (2.1)$$

Les sorties sont calculées de la même manière que pour le neurone de Mc Culloch.

$$S_i = F(a_i) = F\left(\sum_j W_{ij}.S_j\right) = \begin{cases} 1 \text{ si } a_i > 0 \\ -1 \text{ si non} \end{cases} \qquad (2.2)$$

Un autre apport essentièl de Hopfield au domaine des réseaux de neurones est l'analogie avec la physique statistique. Il montre que durant l'évolution du réseau qui tend à se rapprocher d'un état stable une fonction d'énergie, analogue à celle des verre de spin d'ising, décroit vers un minimum local.

Hopfield introduit une fonction d'énergie H qui caractérise un état du réseau [12],donnée par:

$$H(S) = -1/2 \left(\sum_i \sum_j W_{ij}.S_i.S_j \right) \qquad (2.3)$$

L'état du réseau correspond aux valeurs de sorties des neurones.

La relation (2.3) peut s'écrire sous la forme suivante:

$$H(S) = -1/2 \left(S^t.W.S \right) \qquad (2.4)$$

Avec S l'état du réseau et W, la matrice des poids W_{ij}.

Pour n neurones, n entrées et n sorties, la matrice W est une matrice carrée (nxn). S^t étant le vectuer transposé de S.

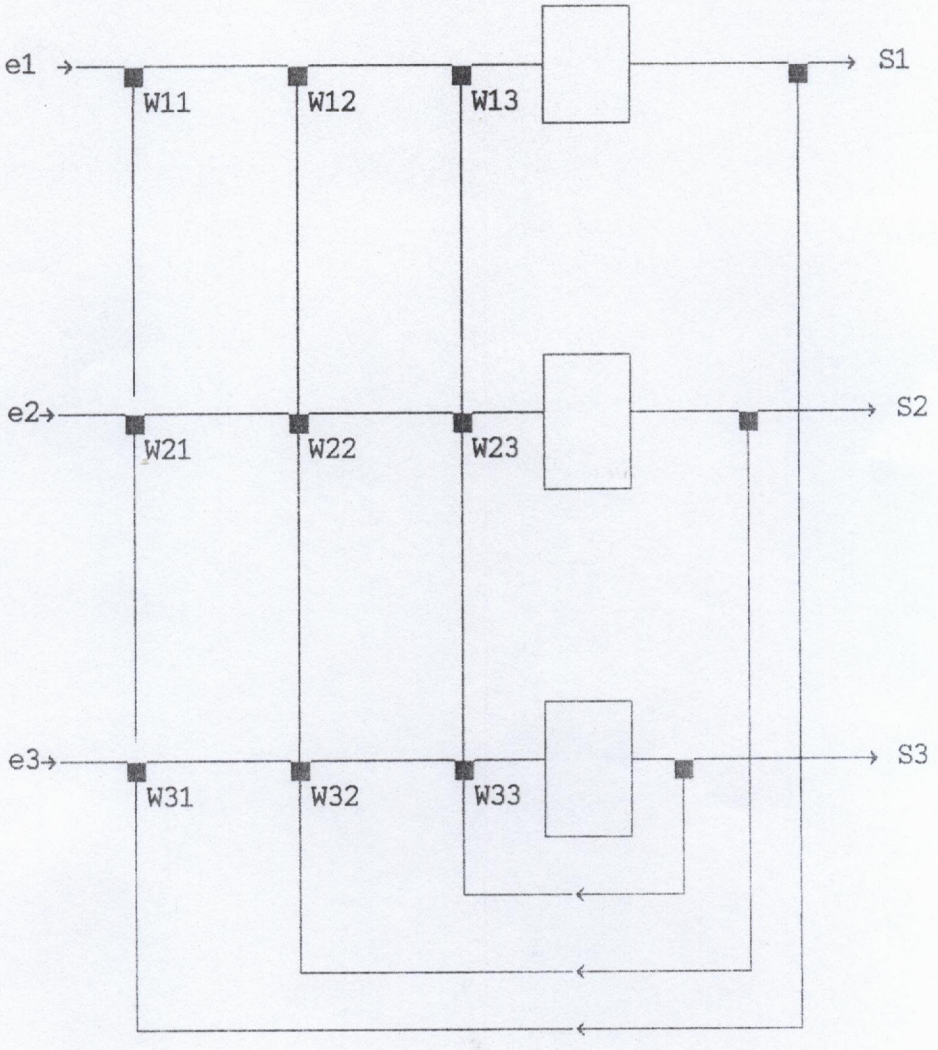


Fig (2.1): Le réseau de Hopfield.

2.1.2. APPRENTISSAGE

Mémoriser p formes représentées par p vecteurs d'états $S^1, S^2, S^3, \ldots S^p$, consiste à les imposer comme des minima locaux d'énergies. Ces états attireront tous ceux situés dans leur bassin d'attraction, fig (2.2).

Les poids sont calculés à partir de la règle de Hebb vue au chapitre 1.

$$W_{ij} = 1/n \left(\sum_{h=1}^{p} s_i^h \, . \, s_j^h \right) \qquad (2.5)$$

Le calcul de W_{ij} est local et simple puisqu'il ne dépend que des sorties des neurones i et j. D'autre part il faut faire le calcul de W_{ij} pour tous les prototypes présentés au réseau.

La relation (2.5) est une des conséquences de l'analogie faite par Hopfield avec la physique statistique. n est un facteur de normalisation des poids. Il correspond au nombre de neurones du réseau.

On peur vérifier la règle de Hebb pour un prototype m présenté au réseau:

$$W_{ij} = \begin{cases} W_{ij} + 1/n \ \text{(poids renforcé)} & \text{si } s_i^m = s_j^m = 1 \\ W_{ij} - 1/n \ \text{(non renforcé)} & \text{si } s_i^m \neq s_j^m \end{cases} \qquad (2.6)$$

Soit:

$$\Delta W_{ij} = 1/n \left(s_i^m s_j^m \right) \qquad (2.7)$$

L'oubli du prototype m correspond à une modification de poids:

$$\Delta W_{ij} = - 1/n \left(s_i^m \, s_j^m \right) \qquad (2.8)$$

L'oubli sera catastrophique si cette modification se fait pour tous les prototypes.

Un autre résultat qui découle de l'analogie faite par Hopfield, nous permet d'établir la matrice des poids à partir des vecteurs prototypes [33]:

$$W = \sum_{i=1}^{p} \frac{S_i . S_i^t}{S_i^t . S_i} \qquad (2.9)$$

avec;

S_i le ième vecteur prototype présenté au réseau et S^t le vecteur transposé. Le produit scalaire $S_i^t . S_i$ correspond au nombre de neurones en bipolaire (-1 et +1).

EX: soit le prototype S donné par $\begin{bmatrix} -1 \\ 1 \\ -1 \end{bmatrix}$ pour un réseau de 3 neurones

Le produit scalaire vaut: $(-1)^2 + (1)^2 + (-1)^2 = 3$

La matrice des poids pour ce prototype sera:

$$\begin{bmatrix} -1 \\ 1 \\ -1 \end{bmatrix} \begin{bmatrix} -1 & 1 & -1 \end{bmatrix} = \begin{bmatrix} 1 & -1 & 1 \\ -1 & 1 & -1 \\ 1 & -1 & 1 \end{bmatrix} / 3 = \begin{bmatrix} 1/3 & -1/3 & 1/3 \\ -1/3 & 1/3 & -1/3 \\ 1/3 & -1/3 & 1/3 \end{bmatrix}$$

Pour d'autre prototypes il faut répeter la même opération et faire la somme des matrices d'apprentissage.

La matrice est symétrique, tout évolution libre du réseau en mode séquentièl conduit à un état stable.

Le calcul de la matrice synaptique introduit dans le réseau.des états stables autres que ceux des prototypes. Ces états sont appelés états parasites. Ils correspondent à une combinaison des prototypes fig (2.3).

2.1.3 DISCUSSION

la stabilité des vecteurs mémorisés n'est assurée que s'il sont orthogonaux ce qui limite la capacité de mémorisation.
Les expériences [33] montrent que pour un nombre de neurones égale à N,la capacité de stockage est de 0,15 N. Si cette limite est dépassé le réseau éxecute le désapprentissage qui entraine parfois l'oubli catastrophique.

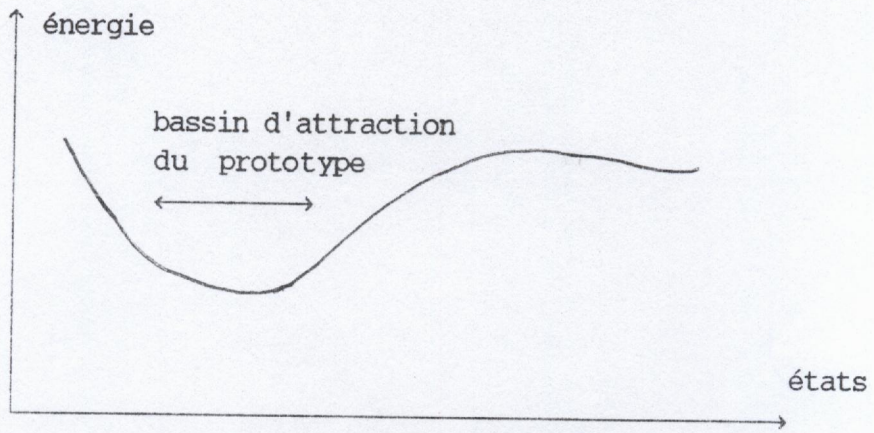


Fig.2.2: Les bassins d'attraction.

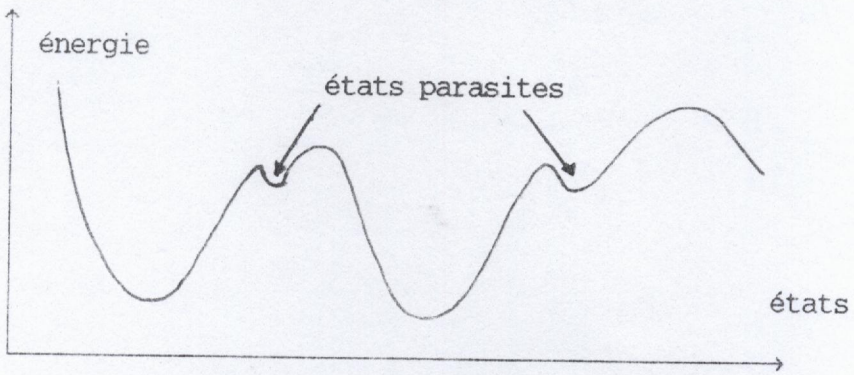


Fig.2.3: Les états parasites

2.2 LE MODELE MULTICOUCHE

Les réseaux à couches actuels sont directement issues du perceptron vu au chapitre 1 .En raison de la diversité des applications déja opérationnelles, ce sont probablement les plus populaires.
on parle souvent de perceptron multicouches (M.L.P), rappelant ainsi leur origine.
Minsky et Papert [12] ont bien montré que la solution au problème du perceptron passait par la multiplication des couches. A cette époque aucune règle d'apprentissage ne pouvait fonctionner sur plusieurs couches. La situation a été débloqué grâce à l'algorithme de la rétropropagation du gradient [19],[25].

2.2.1 DESCRIPTION DU MODELE

Le réseau utilisé est un réseau à couches, comportant une couche d'entrée, qui correspond à la rétine, une couche de sortie qui correspond à la décision et un certain nombre de couches dites cachées. Chaque neurone est connecté à l'ensemble des neurones de la couche suivante. par des connexions dont les poids sont des nombres réels quelconques Fig.(2.4).
Le neurone utilisé est fondamentalement de même nature que le neurone à seuil. La fonction d'activation est une version lissée de la fonction seuil. On utilise, en général, une fonction sigmoide (fonction dérivable)

$$f(x)= a \frac{e^{kx} - 1}{e^{kx} + 1} \qquad (2.10)$$

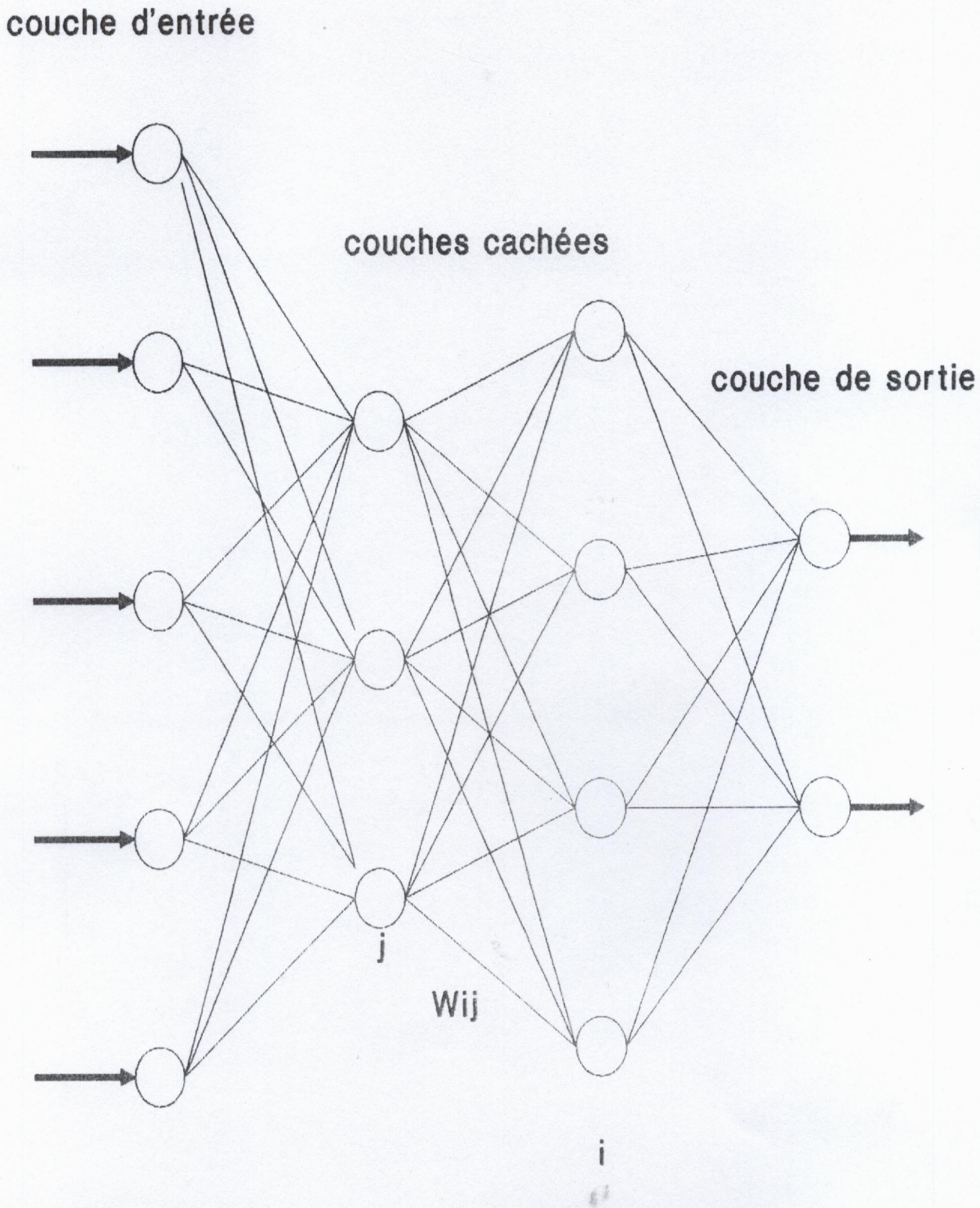


Fig.2.4: Réseau MLP à 2 couches cachées

2.2.2 APPRENTISSAGE

L'apprentissage fonctionne sur le même principe que les règles simples du perceptron. On dispose d'un ensemble d'éxemples appelé ensemble d'apprentissage. Ces éxemples sont des couples (entrées, sorties désirées).

A chaque étape un exemple est présenté à l'entrée du réseau. Une sortie réelle est, alors, calculée. Ce calcul est effectué de proche en proche de la couche d'entrée à la couche de sortie. Cette phase est appelée propagation avant. Ensuite l'erreur égale à la somme quadratique des erreurs sur chaque cellule de sortie est calculée. Cette erreur est ensuite rétropropagée dans le réseau donnant lieu à une modification de chaque poids. Ce processus est répété en présentant successivement chaque éxemple de l'ensemble d'apprentissage.

Si pour tous les exemples l'erreur est inférieure à un seuil choisi, on dit alors que le réseau a convergé.

L'apprentissage consiste à minimiser l'erreur quadratique commise sur l'ensemble des exemples. Cette erreur considérée comme une fonction des poids des connexions est minimisée par une déscente du gradient.

Toute la difficulté pour effectuer cette déscente dans un réseau multicouches, était de pouvoir calculer la dérivée de l'erreur quadratique par rapport à un poids donné, d'où le choix d'une fonction d'activation dérivable telle que la sigmoide.

Algorithme de la rétropropagation du gradient [16]

Supposons qu'on ait un problème de classification d'objets dans m classes. On utilisera pour cela un réseau MLP avec une couche de sortie ayant m neurones. Chaque neurone représente une classe. Pour qu'un objet soit bien classé dans sa classe m, il faut que la sortie

du neurone m soit égale à 1 et le reste des neurones à zéro. On utilisera cette méthode de classification dans le chapitre 4.

étape 1: Initialiser les poids W_{ij} par de faibles valeurs aléatoires.

étape 2: Présenter un exemple de la classe m à l'entrée du réseau et spécifier la sortie désirée: elle sera nulle pour tous les neurones de la couche de sortie sauf le neurone m pour lequel la sortie vaut 1.

étape 3: Ccalculer les sorties actuelles de tous les neurones

étape 4: Calculer la somme des erreurs quadratiques au niveau des neurones de la couche de sortie:

$$E = \sum_{i=1}^{i=m} (d_i - y_i)^2 \qquad (2.11)$$

avec

d_i: sortie désirée

y_i: sortie actuelle.

Si E est inférieure à la valeur fixée au départ

aller à l'étape 2 .

sinon aller à l'étape 5.

étape 5: Calculer l'érreur $\delta_j = (d_j - y_j)\, y_j\, (1-y_j)$ (2.12)

pour un neurone de la couche de sortie

et l'érreur $\delta_j^{(L)} = y_j^{(L)} (1- y_j^{(L)}) . \sum_k \delta_k^{(L+1)} . W_{jk}$ (2.13)

pour un neurone de la L ème couche cachée.

k porte sur les neurones sur lesquels le neurone j envoie des connexions.

étape 6: Ajuster les poids des connexions par:

$$W_{ij}(k+1) = W_{ij}(k) + \alpha\,\delta_j.y_i + \zeta\,[W_{ij}(k) - W_{ij}(k-1)] \qquad (2.14)$$

avec:

α le coefficient d'apprentissage qui caractérise la vitesse d'apprentissage.(compris entre 0 et 1)

et

ζ le coefficient de viscosité qui permet d'éliminer les oscillations pendant la phase d'apprentissage

étape 7: aller à l'étape 2.

2.3 CONCLUSION

Le modèle en couche qui donne,maintenant, une solution au problème du perceptron, est généralement le plus utilisé. Ce modèle n'est pas limité en matière de capacité de mémorisation, au contraire plus l'ensemble d'apprentissage est grand plus les performances à la reconnaissance sont meilleures. Le problème reste posé au niveau du choix des paramètres d'apprentissage qui permettent une convergence sure et rapide. Les performances à la reconnaissance dépendent aussi de la représentation numérique des formes appliquées à l'entrée du réseau. Si cette forme est codée, il faut que le choix du codage vérifie un certains nombre de critères. Dans le chapitre qui suit nous allons présenter un type de codage par les moments qui sera utilisé par la suite dans notre application.

CHAPITRE 3

CODAGE DES IMAGES

3.1. GENERALITES

Dans le domaine de la reconnaissance des formes, la représentation numérique de la forme est une étape très importante, voir décisive pour la reconnaissance. Un codage s'avère indispensable vu le nombre important de pixels.

Il existe un grand nombre de codes. Cependant, ils présentent des caractéristiques qui conditionnent le choix selon le traitement qu'on veut faire. Citons quelques caractéristiques:

- Bonne représentation de la forme

  Pour deux formes différentes il faut avoir deux représentations numériques différentes.

- Une perte d'information minime au codage et à la reconnaissance

-Invariance de l'image

  la représentation doit être la même quelque soit la position de l'objet dans le champ de vision d'une caméra, par exemple. Il faut donc que l'image soit invariante à la rotation, à la translation et au changement d'échelle de l'objet.

- La reconstruction de l'image à partir du code

Le codage rétinien pour lequel chaque cellule de la couche d'entrée correspond à un pixel de l'image Fig.(3.1) devient inadapté lorsque la dimension de l'image est importante. Il faut donc rechercher les codes qui vérifient les caractéristiques vues précèdemment et qui soient aussi traitables.

Parmis ces codes, nous allons présenter les moments invariants issus des moments géométriques réguliers, et les moments de Zernike.

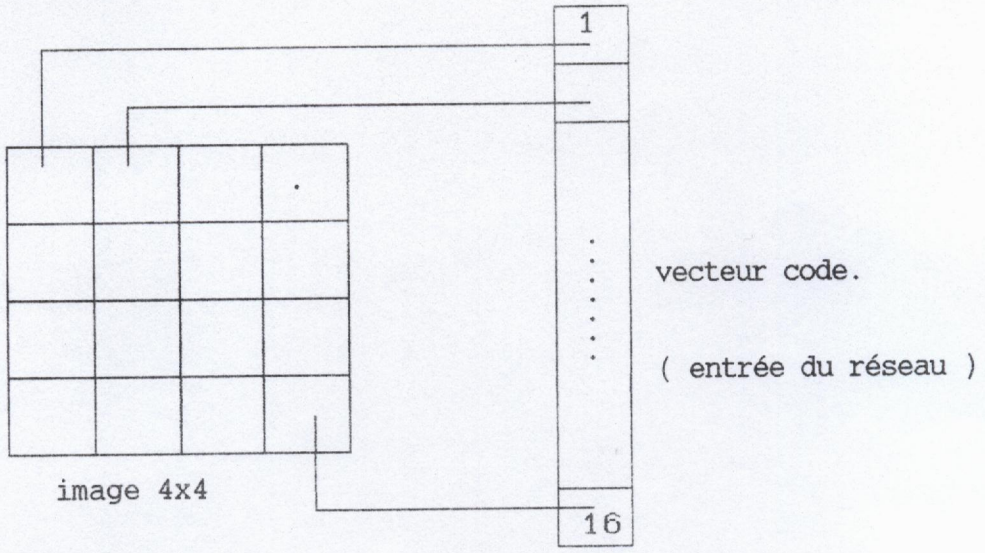


Fig.3.1: Codage rétinien

3.2. CODAGE PAR LES MOMENTS INVARIANTS

Les moments invariants [16], [28], [29] introduits par Hu [34] sont des fonctions non linéaires invariantes à la translation, à la rotation, et au changement d'échelle d'une image donnée. Ces moments découlent des moments géométriques réguliers.

Pour une image digitale à deux dimensions (2D) f(x,y), le moment géométrique régulier d'ordre p+q s'écrit:

$$m_{p,q} = \sum_{x=0}^{M-1} \sum_{y=0}^{M-1} x^p \, y^q \, f(x,y) \qquad (3.1)$$

Avec p = q = 0,1,2,...

Pour obtenir l'invariance de l'image vis à vis de la translation, on introduit le moment central:

$$\mu_{p,q} = \sum_{x=0}^{M-1} \sum_{y=0}^{M-1} (x - \bar{x})^p \, (y - \bar{y})^q \, f(x,y) \qquad (3.2)$$

$$\text{avec} \quad \bar{x} = \frac{m_{10}}{m_{00}} \quad \text{et} \quad \bar{y} = \frac{m_{01}}{m_{00}} \qquad (3.3)$$

$\bar{x}$ et $\bar{y}$ représentent les coordonnées du centre de masse de l'image.

Pour obtenir l'invariance en changement d'échelle le moment central sera normalisé par la relation suivante:

$$\eta_{pq} = \frac{\mu_{pq}}{\mu_{00}^{\gamma}} \qquad (3.4)$$

Avec;

$$\gamma = \frac{p + q}{2} + 1 \qquad (3.5)$$

Pour satisfaire l'invariance en translation, en changement d'échelle, et en rotation Hu [16], utilise une série de fonctions non linéaires définies sur η_{pq}.

$$\phi_1 = \eta_{20} + \eta_{02} \qquad (3.6)$$

$$\phi_2 = (\eta_{20} - \eta_{02})^2 + 4\eta_{11}^2 \qquad (3.7)$$

$$\phi_3 = (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} - \eta_{03})^2 \qquad (3.8)$$

$$\phi_4 = (\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2 \qquad (3.9)$$

$$\phi_5 = (\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12}).[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] +$$

$$(3.\eta_{21} - \eta_{03})(\eta_{21} + \eta_{03}).[3.(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2]. \qquad (3.10)$$

$$\phi_6 = (\eta_{20} - \eta_{02}).[(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] + 4.\eta_{11}(\eta_{30} + \eta_{12})(\eta_{21} + \eta_{03}) \qquad (3.11)$$

$$\phi_7 = (3.\eta_{21} - \eta_{03})(\eta_{30} + \eta_{12}).[(\eta_{30} + \eta_{12})^2 - 3.(\eta_{21} + \eta_{03})^2] +$$

$$(3.\eta_{12} - \eta_{30})(\eta_{21} + \eta_{03}).[3.(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2]. \qquad (3.12)$$

D'après leur définition les moments géométriques réguliers correspondent à la projection de l'image à 2D f(x,y) sur une base $x^p y^q$. Cette base n'est pas orthogonale, on s'attend donc à ce que la

reconstruction de l'image à partir des moments soit difficile et coûteuse.

Teague suggère alors l'utilisation des moments orthogonaux basés sur la théorie des polynomes orthogonaux [28]. Parmi ces moments, il existe une classe qui vérifie l'invariance en rotation, entre autres, les moments de Zernike.

3. 3. CODAGE PAR LES MOMENTS DE ZERNIKE

Zernike [32] introduit un ensemble de polynomes complexes orthogonaux dans le cercle unité ($x^2 + y^2 = 1$).

L'ensemble de ces polynômes est noté:{ $V_{nm}(x,y)$ }

$$V_{nm}(x,y) = V_{nm}(\rho,\theta) = R_{nm}(\rho).e^{jm\theta} \qquad (3.13)$$

avec:

$n \geq 0$, $n - |m|$ pair et $|m| \leq n$. ρ et θ sont les coordonnées polaires.

$R_{nm}(\rho)$ est le polynome radial défini par:

$$R_{nm}(\rho) = \sum_{s=0}^{n-|m|/2} \frac{(-1)^s . (n - s)!}{s!(\frac{n + |m|}{2} - s)!(\frac{n - |m|}{2} - s)!} . \rho^{n - 2s} \qquad (3.14)$$

Ces polynomes sont orthogonaux et vérifient la relation suivante:

$$\iint_{x^2+y^2 \leq 1} [v_{nm}(x,y)]^* . v_{pq}(x,y).dxdy = \frac{\pi}{n + 1} \delta_{np} \delta_{mq} \qquad (3.15)$$

$$\text{avec: } \delta_{ab} = \begin{cases} 1 & a=b \\ 0 & \text{ailleurs.} \end{cases} \qquad (3.16)$$

Les moments de Zernike correspondent à la projection de l'image f(x,y) sur cette base de fonctions orthogonales.

Le moment de Zernike d'ordre n avec m répétitions pour une image digitale f(x,y) est donné par la relation suivante:

$$A^{*}_{nm} = \frac{n+1}{\Pi} \sum_{X} \sum_{Y} f(x,y).V_{nm}(\rho,\theta) \qquad x^2 + Y^2 \leq 1 \qquad (3.17)$$

Pour calculer les moments de Zernike d'une image, l'origine des coordonnées sera prise au centre de l'image. Les coordonnées des pixels seront placées à l'intérieur du cercle unité. Les points se trouvant à l'éxtérieur ne seront pas pris en compte durant le calcul.

Le tableau de la figure (3.2) donne le nombre de moments utilisé suivant l'ordre.

Si on considère une rotation de l'image d'un angle α , la relation entre l'image originale et celle qui subit la rotation s'écrit:

$$f_r(\rho,\theta) = f(\rho,\theta-\alpha) \qquad (3.18)$$

Aprés transformation en coordonnées polaires, L'éxpression des moments de Zernike s'écrit:

$$A_{nm} = \frac{n+1}{\Pi} \int_{0}^{2\Pi} \int_{0}^{1} f(\rho,\theta).V^{*}_{nm}(\rho,\theta)\ \rho\ d\rho\ d\theta \qquad (3.19)$$

Pour une image ayant subi une rotation α , le moment de Zernike s'ecrit:

$$A^{r}_{nm} = \frac{n+1}{\pi} \int_{0}^{2\pi} \int_{0}^{1} f(\rho, \theta-\alpha).R_{nm}(\rho).\, e^{-jm\theta}\, .\, \rho\, d\rho\, d\theta \qquad (3.20)$$

Soit le changement de varible: $\theta_1 = \theta - \alpha$. On obtient en remplaçant par θ_1 :

$$A^{r}_{nm} = A_{nm}.\, e^{(-jm\alpha)} \,. \qquad (3.21)$$

On constate que:

$$|A^{r}_{nm}| = |A_{nm}| \qquad (3.22)$$

Par conséquent, lors du codage on ne considère que les modules des moments de Zernike.

Les moments de Zernike sont invariants seulement en rotation. Pour les rendre invariant en translation et en changement d'échelle, il faut procéder à la normalisation de l'image par l'utilisation des moments géométriques réguliers.

Pour la translation, il faut transformer l'image f(x,y) en $f(x+\overline{x}, y+\overline{y})$ où $\overline{x}$ et $\overline{y}$ représentent les coordonnées du centre de masse de l'image originale (relation donnée en 3.3).

Autrement dit, l'origine est déplacée vers le centre de masse avant de faire le calcul des moments.

Pour le changement d'échelle, il s'agit d'augmenter ou de réduire l'objet de l'image de telle façon que le moment d'ordre o; m_{oo} soit égale à une valeur prédéterminée β.

Il est important de remarquer que pour une image digitale m_{oo} représente le nombre de pixels de l'objet.

Il s'agit maintenant de trouver la relation entre les moments géométriques réguliers d'une image et ceux de l'image obtenue par changement d'échelle

Soit f(x/a, y/b) l'image obtenue par changement d'echelle à partir de f(x,y).

Le moment régulier de f(x/a, y/b) est donnée par la relation:

$$m'_{pq} = \int_x \int_y x^p . y^q . f\left(\frac{x}{a} , \frac{y}{a} \right) dx\, dy \qquad (3.23)$$

Soit,

$$m'_{pq} = \int \int a^p x^p a^q y^q \, f(x,y) a^2 \, dx\, dy \qquad (3.24)$$

ou bien,

$$m'_{pq} = a^{(p+q+2)} . \int_x \int_y x^p . y^q \, f(x,y) \, dx\, dy \qquad (3.25)$$

Finalement;

$$m'_{pq} = a^{(p+q+2)} . m_{pq} \qquad (3.26)$$

Pour un changement d'echelle donné par:

$$\beta = m'_{00} \qquad (3.27)$$

On prendra:

$$a = (\beta / m_{00})^{1/2} \qquad (3.28)$$

L'invariance en changement d'échelle et en translation sera, donc, obtenue en transformant l'image f(x,y) en g(x,y) avec:

$$g(x,y) = f\left(\overline{x} + \frac{X}{a} , \overline{Y} + \frac{Y}{a} \right) \qquad (3.29)$$

ordre	MOMENTS DE ZERNIKE	NBR DE MOMENTS
0	A_{00}	1
1	A_{11}	1
2	A_{20} A_{22}	2
3	A_{31} A_{33}	2
4	A_{40} A_{42} A_{44}	3
5	A_{51} A_{53} A_{55}	3
6	A_{60} A_{62} A_{64} A_{66}	4
7	A_{71} A_{73} A_{75} A_{77}	4
8	A_{80} ; A_{82} ; ... A_{84} A_{86} ; A_{88} ;	5
9	A_{91} ; A_{93} ; ... A_{95} A_{97} ; A_{99}	5
10	$A_{10,0}$; $A_{10,2}$; $A_{10,4}$ $A_{10,6}$; $A_{10,8}$; $A_{10,10}$	6
11	$A_{11,1}$; $A_{11,3}$; $A_{11,5}$ $A_{11,7}$; $A_{11,9}$; $A_{11,11}$	6
12	$A_{12,0}$; $A_{12,2}$; $A_{12,4}$ $A_{12,6}$; $A_{12,8}$; $A_{12,10}$; $A_{12,12}$	7

Fig.3.2: Moments de Zernike jusqu'à l'ordre 12

CHAPITRE 4

APPLICATION A LA CLASSIFICATION DES CARACTERES ARABES

4.1. INTRODUCTION

Dans ce chapitre, on se fixe comme objectif la vérification des performances des deux modèles étudiés précédemment , sur des exemples de classification de caractères arabes. Les deux réseaux de neurones (Mémoire associative et réseau MLP) seront programmés sur un IBM PC avec le langage de programmation turbo pascal. La structure générale de notre système de classification est organisée comme suit:

- Acquisition de l'image d'un caractère: Cette acquisition se fait à partir du clavier, sur une fenêtre, préalablement ouverte sur l'écran. L'écriture du caractère se fait en mode texte.
- Codage de l'image: Le codage sera différent pour les deux modéles.On adoptera le codage rétinien pour la mémoire associative et le codage par les moments pour le réseau MLP.
- Apprentissage: Les algorithmes de la mémoire associative et de la rétropropagation seront implémentés.
- Classification: Les deux réseaux seront considérés comme classifieurs. Pour la mémoire associative, les images des prototypes appris seront bruitées puis reconnues et classées.
  Pour le réseau MLP , on utilisera un ensemble d'images test différent de l'ensemble d'apprentissage. Ces images seront bruitées puis reconnues et classées.

4.2. MEMOIRE ASSOCIATIVE

4.2.1. ARCHITECTURE DU RESEAU

Le réseau utilisé est un réseau de Hopfield, vu au chapitre 2, composé de 64 neurones complètement interconnectés.

4.2.2. CODAGE

Le prototype à mémoriser est un vecteur dont les composantes représentent les états de tous les pixels de l'image du caractère en question: 1 pour pixel allumé et -1 pour pixel éteint. Cette image est de taille 8 x 8, elle sera représentée par un vecteur de 64 composantes.

4.2.3. APPRENSISSAGE

La règle d'apprentissage utilisée est la règle de HEBB vue au chapitre 1, pour laquelle la matrice d'apprentissage normalisée est donnée par l'equation (2.9).

4.2.4. RECONNAISSANCE

Le vecteur représentant l'image d'un caractère à reconnaitre, donc à classer, sera appliqué à l'entrée du réseau. Il en résulte un vecteur de sortie qui sera à son tour appliqué à l'entrée. Après un certain nombre d'itérations, on obtient en sortie, soit un des p prototypes mémorisés, soit une sortie indésirable correspondant à une combinaison des prototypes. Le vecteur de sortie est donné par la relation:

$$S(k+1) = F[W.S(k)] \qquad (4.2)$$

avec:

S(k): sortie à la k ème itération .

S(k+1): sortie à la k+1 ème itération .

W : matrice d'apprentissage.

F: fonction d'activation (fonction seuil:-1 ou 1).

Les prototypes appris seront bruités puis classés par le réseau. Pour introduire le bruit, nous utiliserons une méthode proposée par Khotanzad et J.H. Lu [9] pour laquelle le rapport signal/bruit exprimé en dB est donné par la relation:

$$(S/B)_{dB} = 20.\log[(N - H)/H] \qquad (4.3)$$

avec: N: le nombre de pixels de l'image.

H: le nombre de pixels affectés par le bruit (distance de Hamming).

Le choix de H pixels parmi N est aléatoire. Les états des H pixels choisis seront inversés.

4.2.5. ORGANIGRAMME D'APPRENTISSAGE ET DE RECONNAISSANCE

Dans la figure (4.1), On a représenté l'organigramme général du programme de la mémoire associative . Le listing du programme se trouve dans l'annexe.B. Dans la figure (4.2). on a représenté le menu

qui apparait juste après l'éxecution du programme.

L'action de la touche de fonction F1 permet l'ouverture d'une fenêtre dans laquelle sera déssiné le caractère à l'aide de la touche (*).

Une fois saisi, le caractère sera appris donc mémorisé à l'aide de la touche (ECHAP). Un méssage de fin d'apprentissage apparait sur l'écran. on répète la même opération pour mémoriser d'autres caractères. Une commande (ALT W) permet à l'opérateur d'éffacer un caractère au moment de son écriture.

L'action de la touche de fonction F2 permet l'ouverture d'une fenêtre dans laquelle sera déssiné le caractère à reconnaitre. Une fois introduit, le caractère sera reconnu à l'aide de la touche (ECHAP) qui permet d'afficher dans une fenêtre située à droite de la précédente, un des caractères mémorisés ou une forme correspondant à un état parasite.

La reconnaissance peut se faire en présence du bruit. Pour cela on agit sur la touche de fonction F4 qui entraine l'affichage d'un méssage de spécification du bruit. Une fois la valeur S/B introduite, une fenêtre s'affiche pour l'introduction du caractère à reconnaitre.

A la reconnaissance le caractère bruité et le prototype reconnu s'affichent en même temps.

Pour initialiser la mémoire associative, on agit sur la touche de fonction F3 ce qui permet la remise à zero de la matrice d'apprentissage.

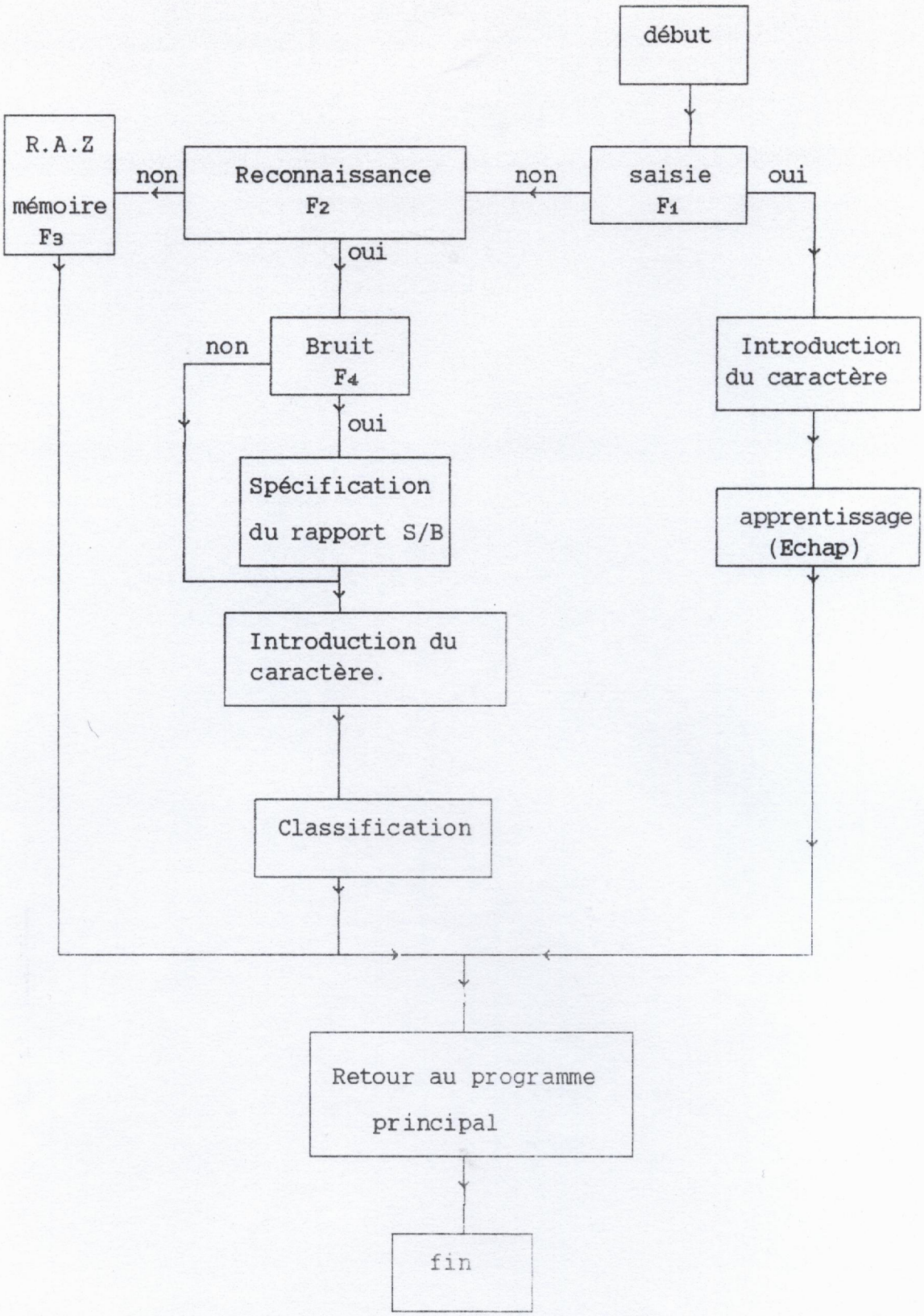


Fig.(4.1): Organigramme du programme d'apprentissage et de classification de la mémoire associative.

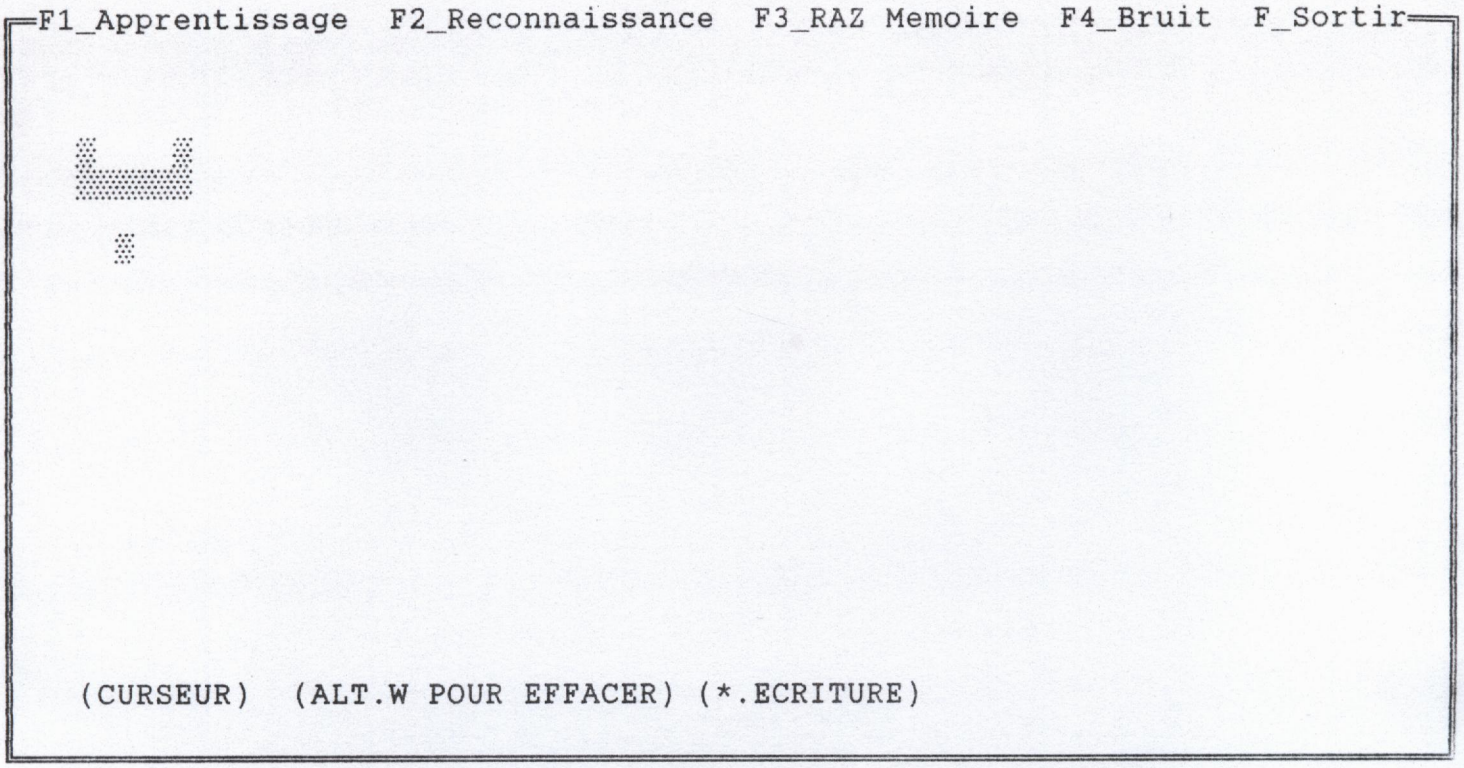


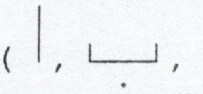
Fig.(4.2): Menu d'éxecution du programme de la mémoire associative.

4.2.6 COMMENTAIRE DES RESULTATS OBTENUS

Dans la figure.(4.3) On a présenté sous forme d'histogrammes, les taux de réussites de la classification en fonction du nombre de caractères mémorisés pour des niveaux de bruit différents.

a) Apprentissage

Durant la phase d'apprentissage, on peut introduire tous les caractères qu'on veut. Mais en réalité, le réseau n'a pu mémoriser que

3 caractères (). En effet, on s'est rendu compte que le taux de réussite s'annule à partir du 4 ème caractère appris. En d'autres termes, aucun des 4 caractères ne sera reconnu même en l'absence du bruit.

Les expériences de simulation montrent que le nombre de prototypes mémorisés se situe autour de 0,15 N. (voir chapitre 2).

Pour notre cas N=64 neurones , ce qui correspond à 9 prototypes mémorisés. La différence s'éxplique par le fait que l'orthogonalité des prototypes n'est pas vérifiée à partir du 4 ème caractère appris, ce qui n'assure pas leur stabilité.

b) Reconnaissance

Après avoir spécifier le bruit, on réalise 20 tests sur chaque caractère appris.Dans la figure (4.3) On a représenté 3 exemples de reconnaissance:

- Pour un caractère mémorisé, une réussite à 1.5 dB, Fig.(4.3 a).
- Pour deux caractères mémorisés, une réussite à 10 dB, Fig.(4.3.b).
- Pour trois caractères mémorisés, un echec à 25 dB. Dans ce cas le réseau converge vers un état parasite indésirable, Fig.(4.3.c).

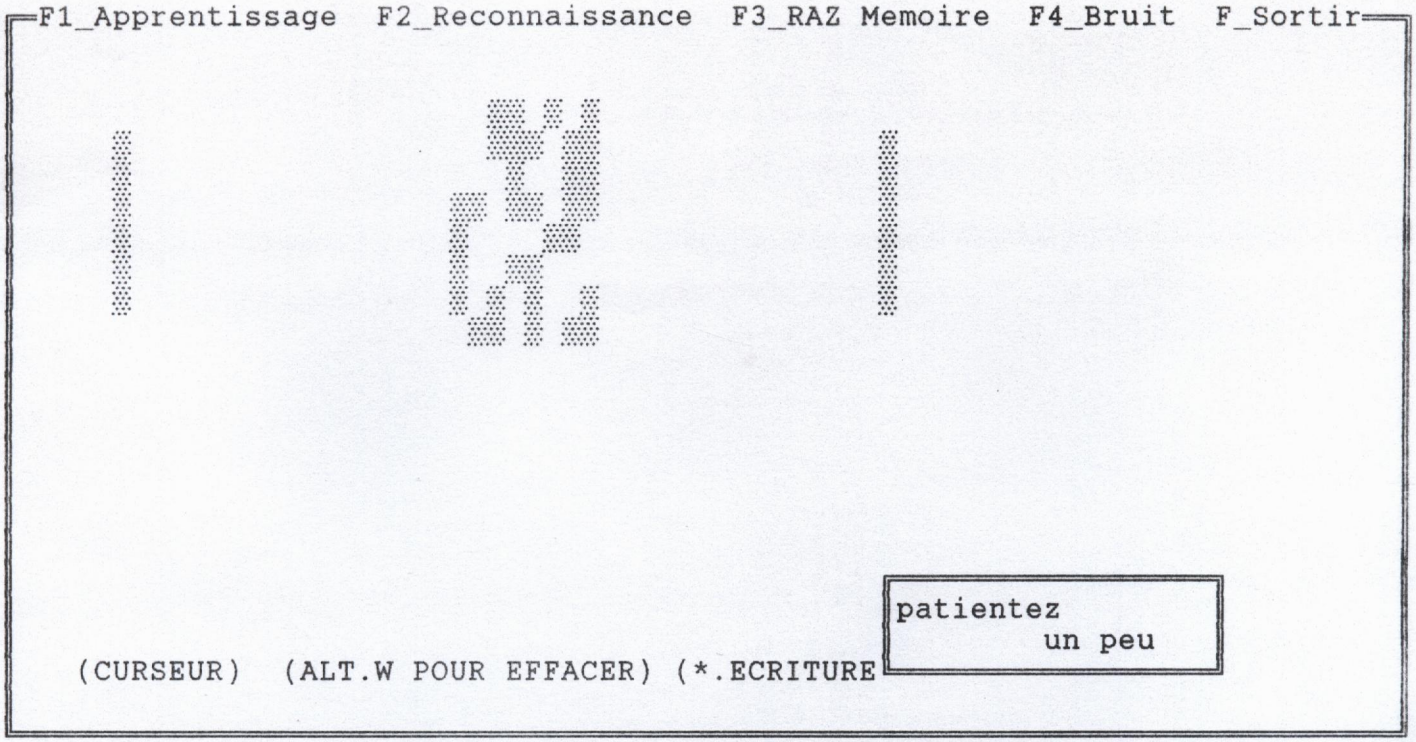


Fig.(4.3 a): Reconnaissance du caractère " ALIF " à 1,5 dB.

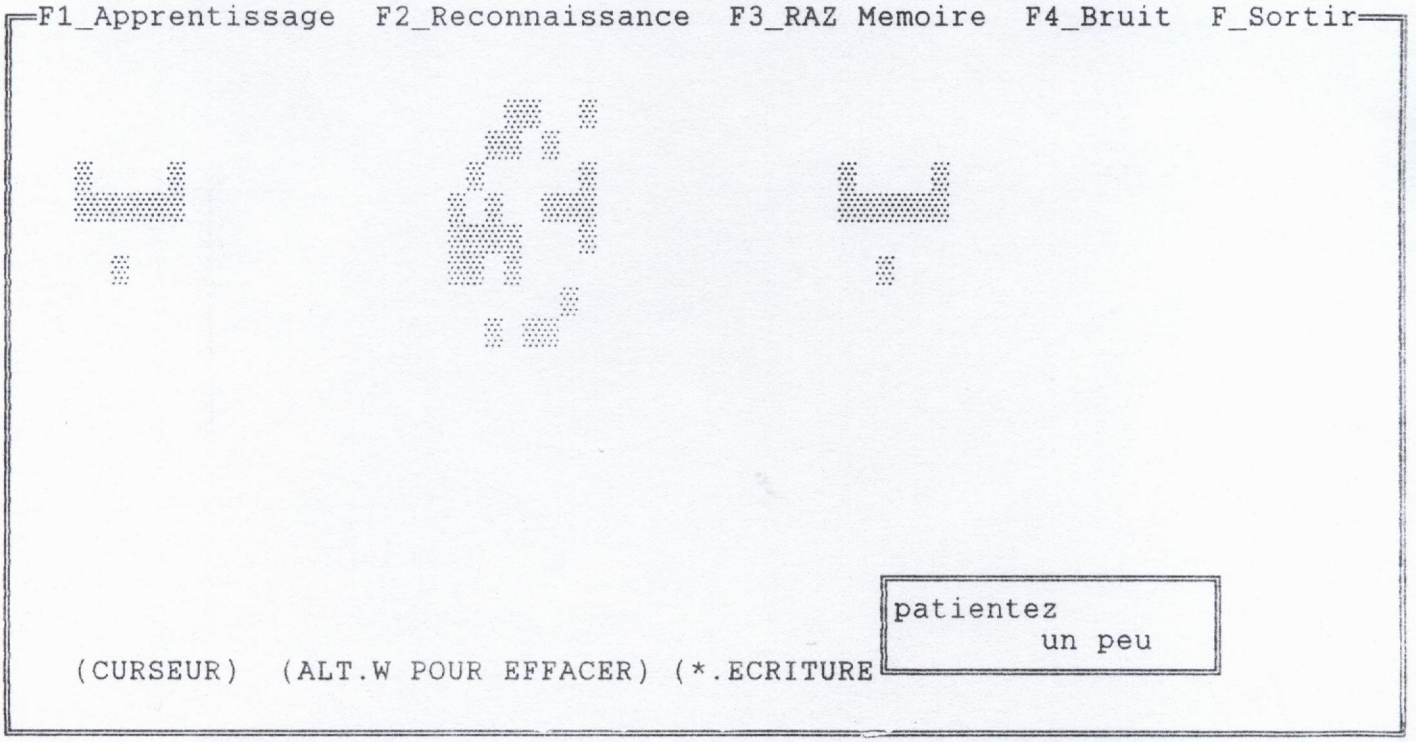


Fig.(4.3.b): Reconnaissance du caractère " BA " à 10 dB.

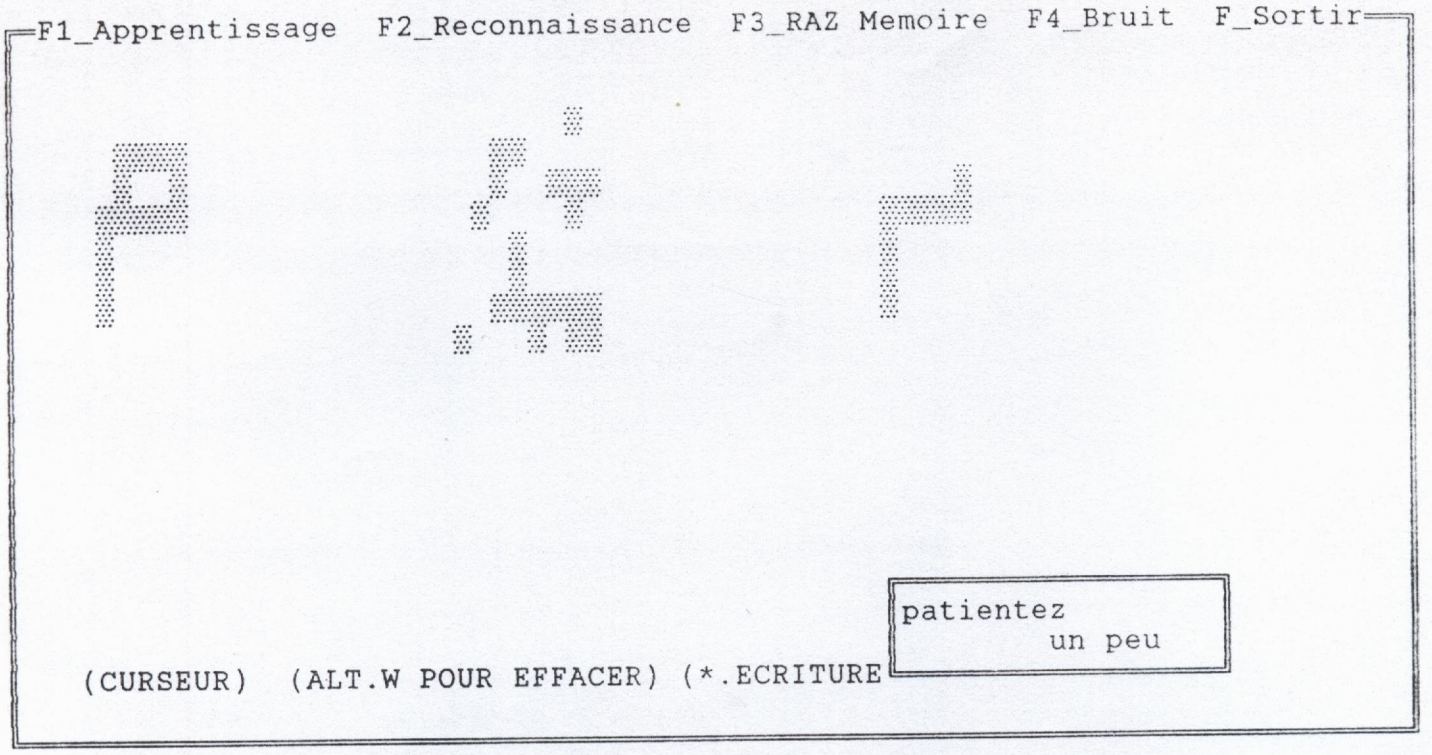


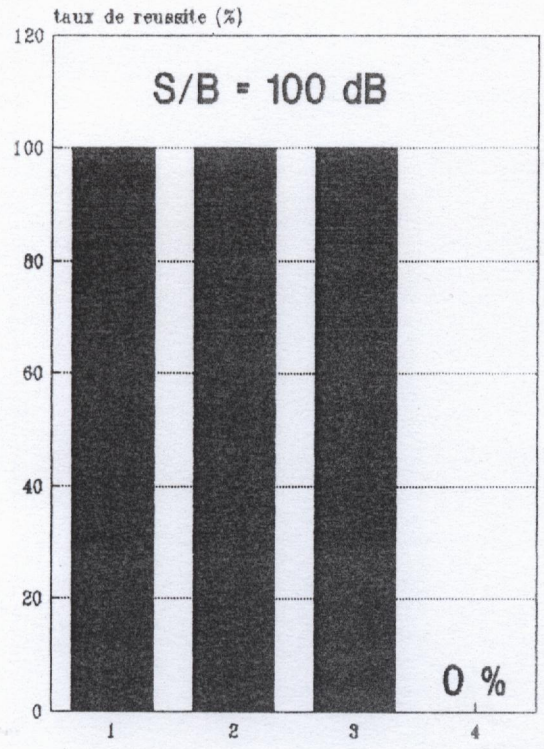
Fig.4.3.(c): Reconnaissance du caractère " MIM " à 25 dB

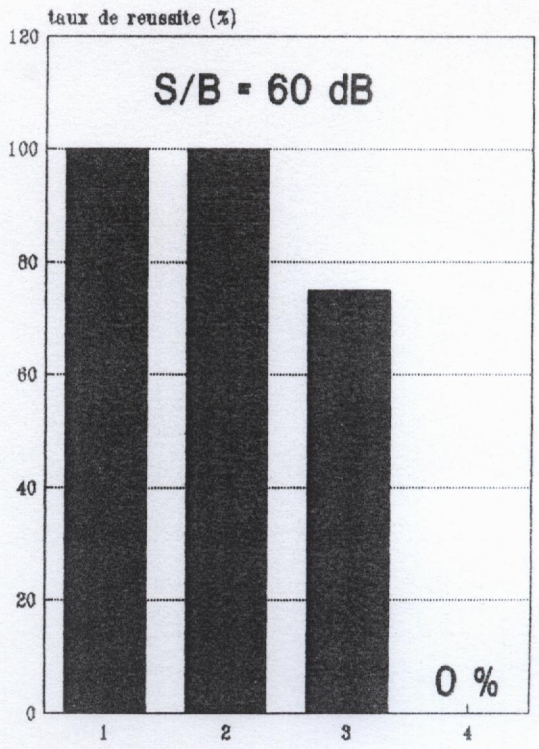
Dans la figure (4.4), on a représenté tous les résultats de la classification.

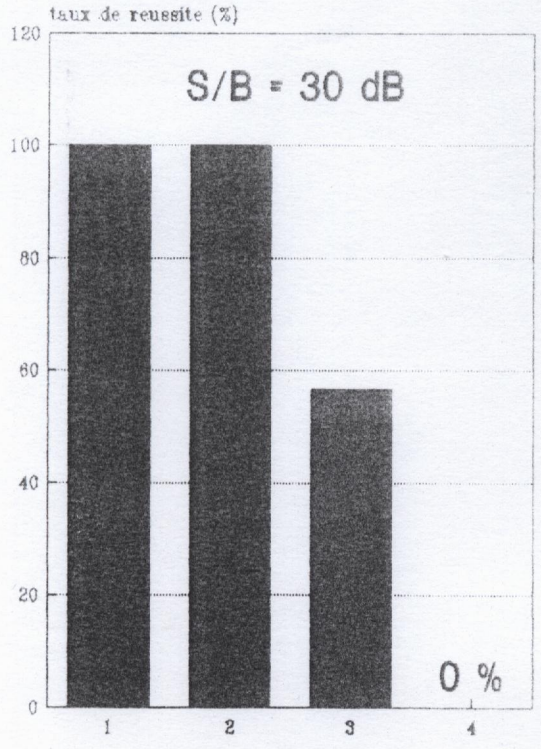
Pour un caractère mémorisé, le taux de réussite est de 100 % jusqu'à 1.5 dB et s'annule à 0 dB. Ceci s'éxplique par le fait qu'il éxiste un seul état stable très attracteur pour le réseau.

Pour deux caractères mémorisés, le taux est de 100 % jusqu'à 30 dB. Sa valeur est de 27.5 % à 1.5 dB . Il s'annule pour 0 dB.

Pour trois caractères mémorisés, le taux de réussite est de 100 % en l'absence du bruit. Comparé aux deux cas précédent, il diminue rapidement pour s'annuler à 1.5 dB. Ceci s'éxplique par l'éxistance d'états parasites non désirables mémorisés par le réseau. Un état parasite qui apparait lors de la reconnaissance est considéré comme un échec.


taux de reussite (%)
S/B = 100 dB
120
100
80
60
40
20
0
0 %
1
2
3
4
nbre de caractères mémorisés


taux de reussite (%)
S/B = 60 dB
120
100
80
60
40
20
0
0 %
1
2
3
4
nbre de caractères mémorisés


taux de reussite (%)
S/B = 30 dB
120
100
80
60
40
20
0
0 %
1
2
3
4
nbre de caractères mémorisés

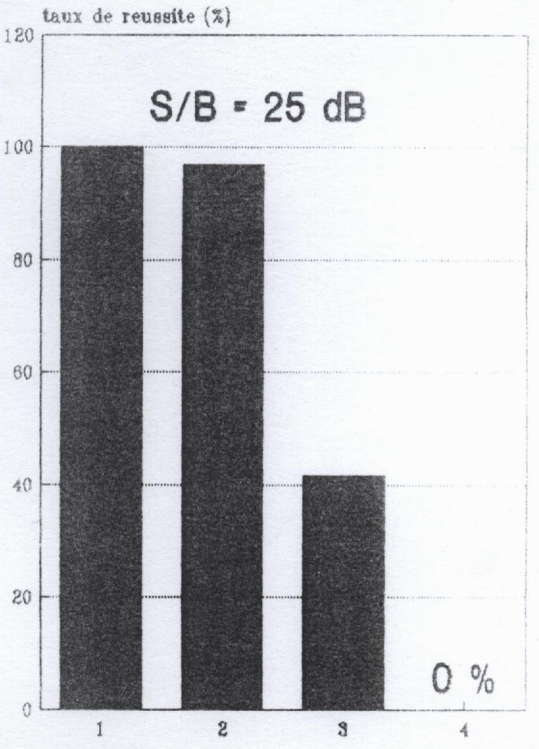

taux de reussite (%)
S/B = 25 dB
120
100
80
60
40
20
0
0 %
1
2
3
4
nbre de caractères mémorisés

Fig.4.4 (a)

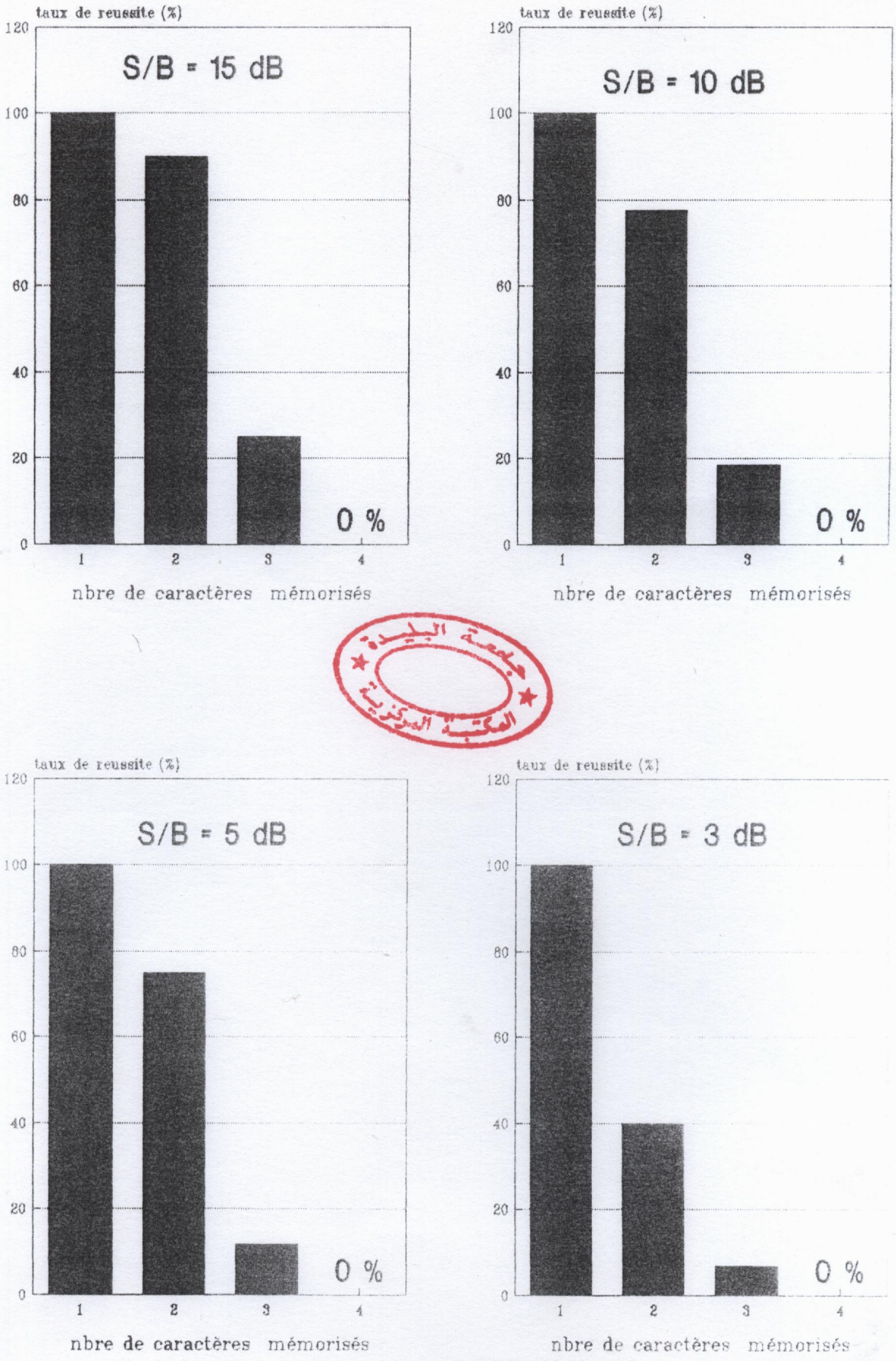
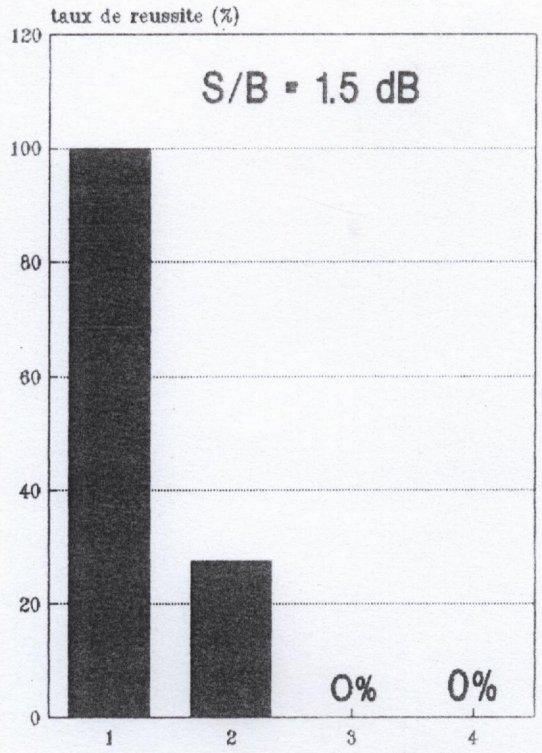


Fig.4.4(b)


taux de reussite (%)
S/B = 1.5 dB
120
100
80
60
40
20
0
0%
0%
1
2
3
4
nbre de caractères mémorisés

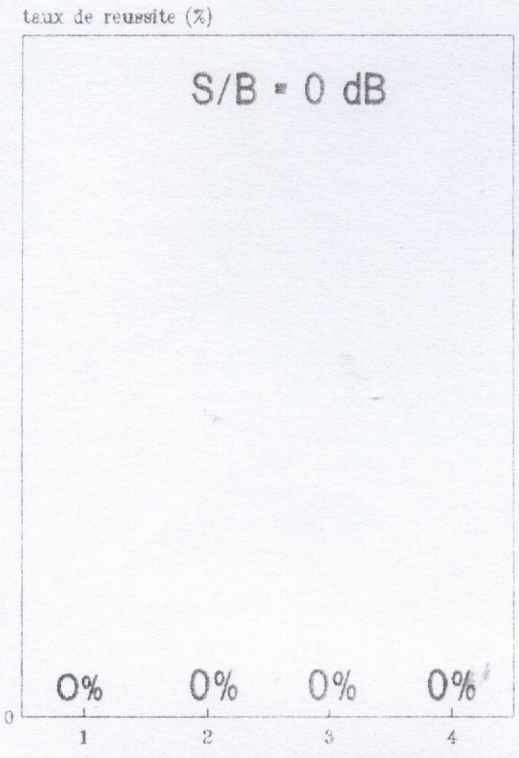

taux de reussite (%)
S/B = 0 dB
0%
0%
0%
0%
0
1
2
3
4
nbre de caractères mémorisés

Fig.4.4(c)

4.3. RESEAU M.L.P

4.3.1. ARCHITECTURE DU RESEAU

Le réseau classifieur utilisé est un réseau du type multicouche à une seule couche cachée.

La couche d'entrée correspond au vecteur code de l'image. Pour cela on utilise soit 7 entrées pour les moments invariants, soit un vecteur d'entrée dont les composantes sont variables entre 7 et 28 suivant l'ordre des moments de Zernike utilisés.

Le nombre de neurones de la couche cachée est variable entre 2 et 12.

La couche de sortie est composée de 10 neurones correspondant à 10 classes de caractères.

4.3.2. CODAGE

On utilisera une image (19x19), qui sera codée par la méthode des moments invariants , puis par les moments de Zernike.

Dans le calcul des moments de Zernike, On aura besoin des coordonnées du centre de l'image, ce qui explique le choix d'une dimension impaire (19). Dans notre cas le centre aura pour coordonnées (10,10).

4.3.3. APPRENTISSAGE

Pour la dynamique des connexions, on utilise l'algorithme de la rétropropagation du gradient vu au chapitre 2.

Les paramètres d'apprentissage sont choisis d'une manière empirique. Pour cela on a utilisé les valeurs suivantes:

Le pas du gradient: $\alpha = 0.25$.

Le coefficient de viscosité: $\zeta = 0.75$

Les poids initiaux: aléatoires entre -0.5 et 0.5.

La fonction d'activation: $f(x) = 1/ (1 + e^{-x})$ est une sigmoide qui varie entre 0 et 1, voir Fig.(4.4.d), ce qui correspond aux sorties désirées des neurones de la couche de sortie (soit 0 soit 1).

Vu les problèmes de convergence du perceptron multicouche, à savoir le temps d'apprentissage qui se chiffre à des journées, on s'est limité à un ensemble d'apprentissage de 10 classes de 4 caractères chacune, voir figure.(4.5 .a). On a, donc, un ensemble d'apprentissage de 40 caractères codés et stockés dans un fichier.

L'opération d'apprentissage est lancée à chaque fois que l'on change le nombre de neurones de la couche cachée.

4.3.4. RECONNAISSANCE

Pour la reconnaissance, on fait subir à chaque caractère de l'ensemble d'apprentissage 5 opérations de translation, rotation et changement d'échelle. On obtient alors un ensemble test de 200 caractères codés et stockés dans un fichier. Dans la figure.(4.5.b) On a représenté 20 caractères de l'ensemble test.

La reconnaissance se fait en présence du bruit que l'on introduit exactement de la même façon que la mémoire associative. Le rapport signal sur bruit est donné par la relation (4.3).

Enfin nous classons les caractères de l'ensemble test suivant les valeurs du vecteur de sortie (neurones de la couche de sortie).

(1,0,0,0,0,0,0,0,0,0) ⟶ ALIF

(0,1,0,0,0,0,0,0,0,0) ⟶ BA

(0,0,1,0,0,0,0,0,0,0) ⟶ TA

(0,0,0,1,0,0,0,0,0,0) ⟶ THA

(0,0,0,0,1,0,0,0,0,0) ⟶ SA

(0,0,0,0,0,1,0,0,0,0) ⟶ CHA

(0,0,0,0,0,0,1,0,0,0) ⟶ LA

(0,0,0,0,0,0,0,1,0,0) ⟶ MIM

(0,0,0,0,0,0,0,0,1,0) ⟶ NOUN

(0,0,0,0,0,0,0,0,0,1) ⟶ YA

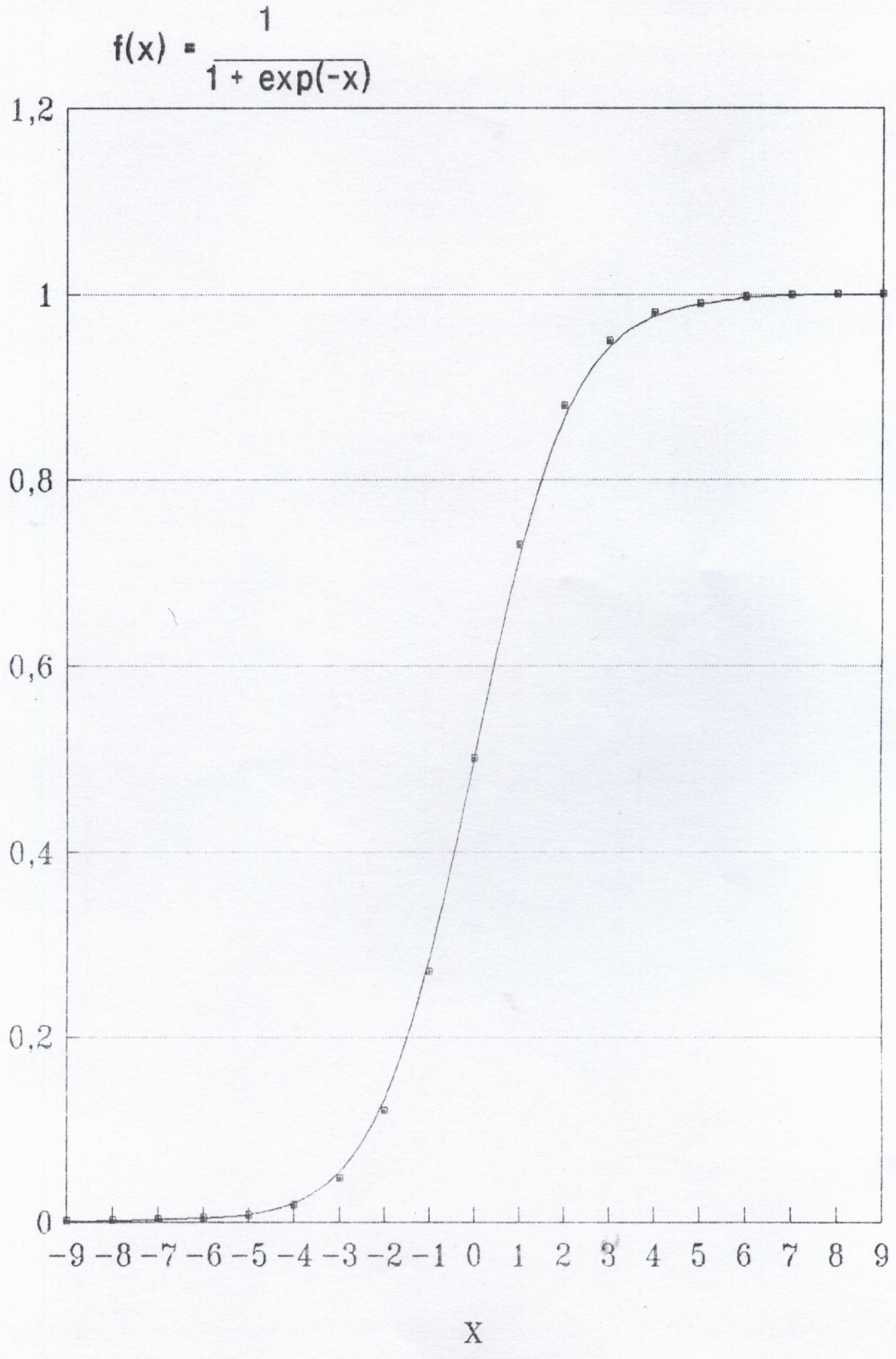


fig.4.4.d: fonction sigmoide

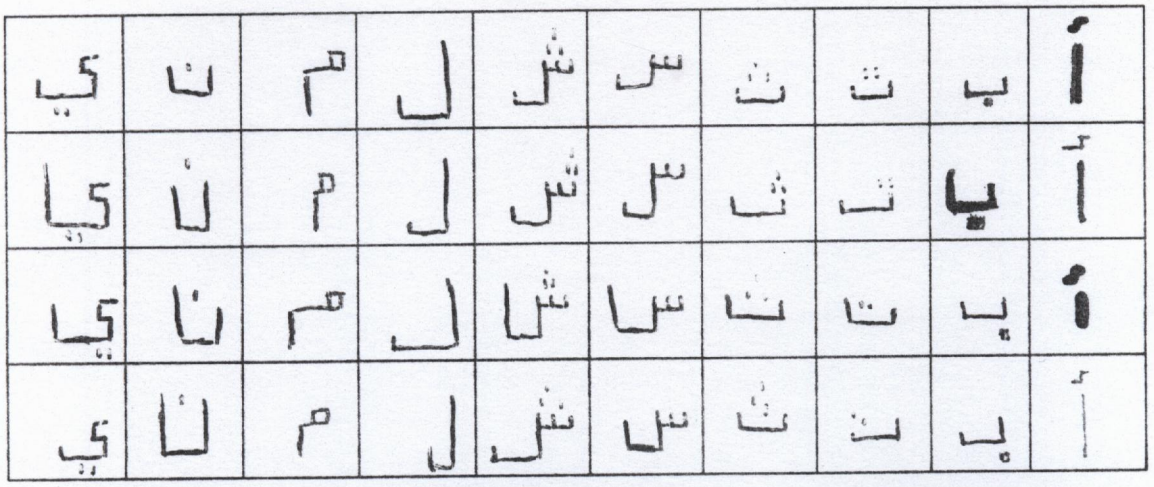


Fig.(4.5 a): Ensemble d'apprentissage.

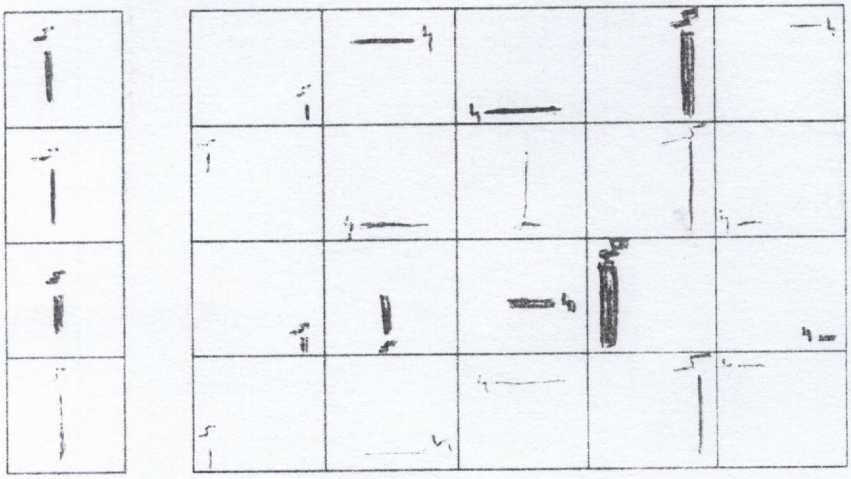


Fig.(4.5 b): Une partie de l'ensemble test

4.3.5. ORGANIGRAMME D'APPRENTISSAGE ET DE RECONNAISSANCE

Dans la figure (4.6), on a représenté l'organigramme général d'apprentissage et de reconnaissance pour le réseau M.L.P. Le listing du programme se trouve dans l'Annexe.C. Dans la figure (4.7),on a représenté le menu qui apparait juste après l'éxecution du programme.

L'action de la touche de fonction F1 permet l'ouverture d'une fenêtre dans laquelle sera dessiné le caractère à l'aide de la touche (*).En agissant sur (ECHAP) l'image sera codée puis stockée dans un fichier. Cette saisie concerne l'ensemble d'apprentissage et l'ensemble test, il suffit pour cela d'utiliser deux fichiers différents.

La touche de fonction F2 permet l'éxecution de l'apprentissage. A la fin de l'apprentissage les poids seront stockés dans un fichier.

La touche de fonction F3 nous permet de faire la reconnaissance des caractères stockés dans un fichier test.

La touche de fonction F4 nous permet de faire la reconnaissance des caractères du fichier test (image) mais cette fois-ci en présence du bruit.

Pour revenir au programme principal il faut agir sur la touche F.

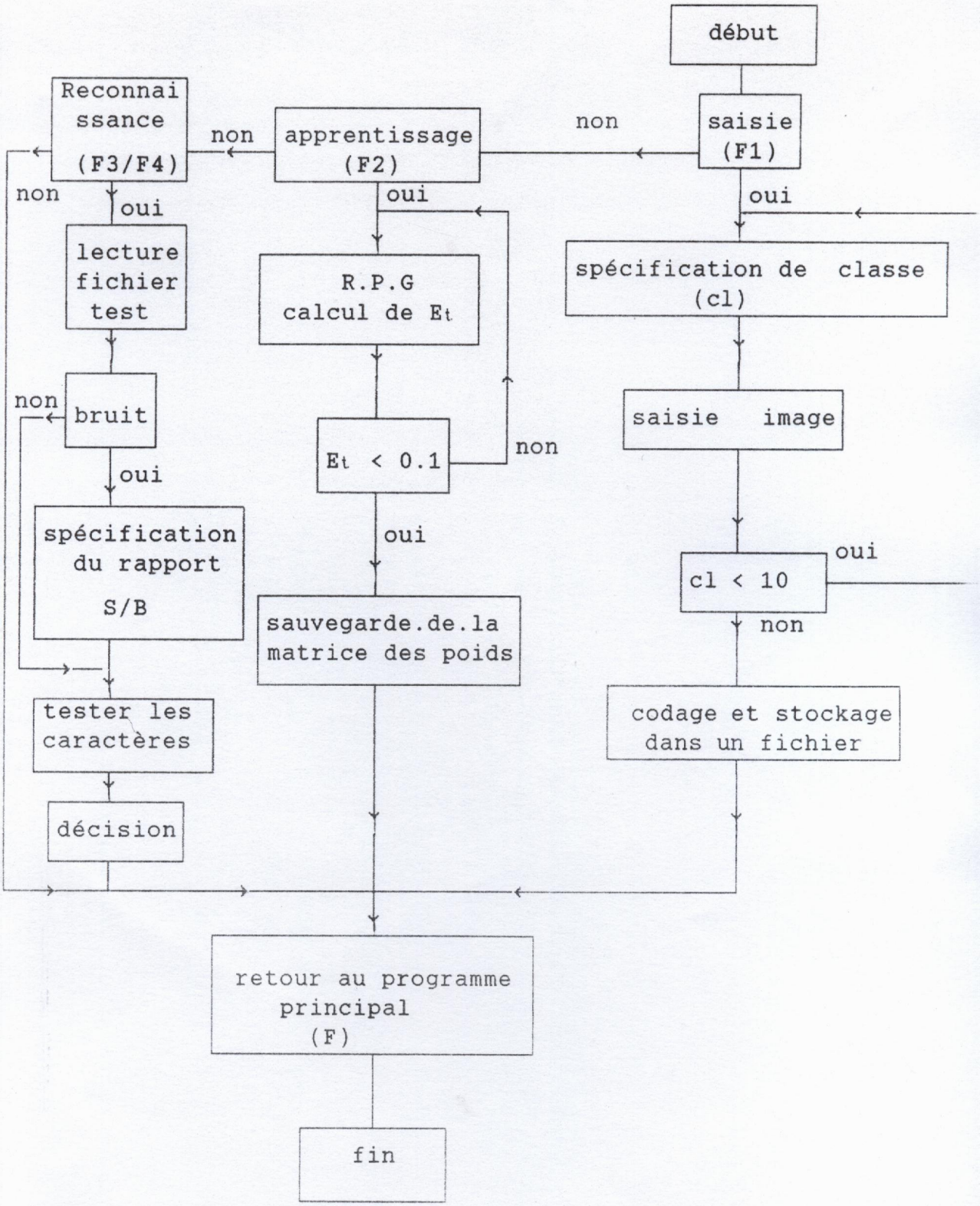


Fig.(4.6): organigramme d'apprentissage et de classification pour le réseau M.L.P

Figure (4.7): Menu du programme du réseau multicouche

F1_SAUVEGARDE DES CARACTERES DANS FICHIE

F2_APPRENTISSAGE

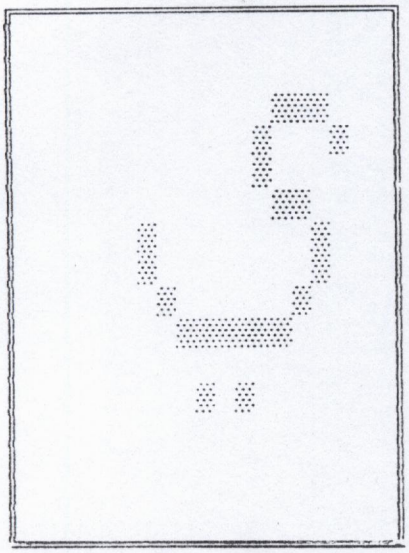
F3_RECONNAISSANCE

F4-RECONNAISSANCE AVEC BRUIT

F_RETOUR AU PROGRAMME PRINCIPAL

donner la classe du caractère à sauvegarder:10

Introduire 1 caractères de cette classe



LA SAUVEGARDE EST FINIE

4.3.6. COMMENTAIRE DES RESULTATS OBTENUS

Dans tous les cas, on a représenté les taux de réussites sous forme de courbes.

Dans la figure.(4.7.a,b,c,d), on a représenté les variations du taux de réussite en fonction du nombre de neurones de la couche cachée, pour des niveaux de bruit différents, et un codage par la méthode des moments invariants.

Dans la figure.(4.8. a,b,c,d,e,f), on a représenté les variations du taux de réussite mais cette fois-ci, avec un codage par la méthode des moments de Zernike.

Dans la figure.(4.9.a,b,c,d,f), on a représenté les variations du taux de réussite en fonction de l'ordre des moments de Zernike, pour des niveaux de bruit différents, mais en conservant cette fois-ci le nombre de neurones de la couche cachée.

Dans la figure (4.10), on a représenté le taux de réussite en fonction du nombre de neurones deffectueux de la couche cachée, et en l'absence du bruit.

a) Apprentissage

Les problèmes rencontrés lors de l'apprentissage sont les suivants:

- On ne peut savoir si le réseau allait converger ou pas.
- Il est inutile de continuer l'apprentissage si le réseau oscille autour d'un minimum local. Il faut parfois le relancer et espérer qu'il converge.
- Si le réseau converge, le temps d'apprentissage dans ce cas est énorme. (plus de 4 jours avec un PC 386 équipé d'un coprocesseur), sans tenir compte du temps de codage qui lui aussi est élevé.

b) Reconnaissance

codage par les moments invariants:

On remarque dans la figure (4.7) que:

- le taux de réussite en l'absence de bruit, augmente avec le nombre de neurones de la couche cachée.
- Plus de 90 % de réussite pour 8 neurones et plus.
- 42 % de réussite pour 2 neurones.
- Les taux sont moins interessants lorsque le rapport S/B diminue
- moins de 5 % de réussite avec 6 neurones et plus pour S/B = 12 dB.

En conclusion on dira que le codage par les moments invariants est sensible au bruit. il donne de bon résultats en l'absence du bruit.

codage par les moments de Zernike:

Ce codage améliore le taux de réussite en l'absence du bruit. En effet, dans la figure (4.8.a) le taux atteint les 98 % .

Le taux est aussi amélioré en présence du bruit, en effet:

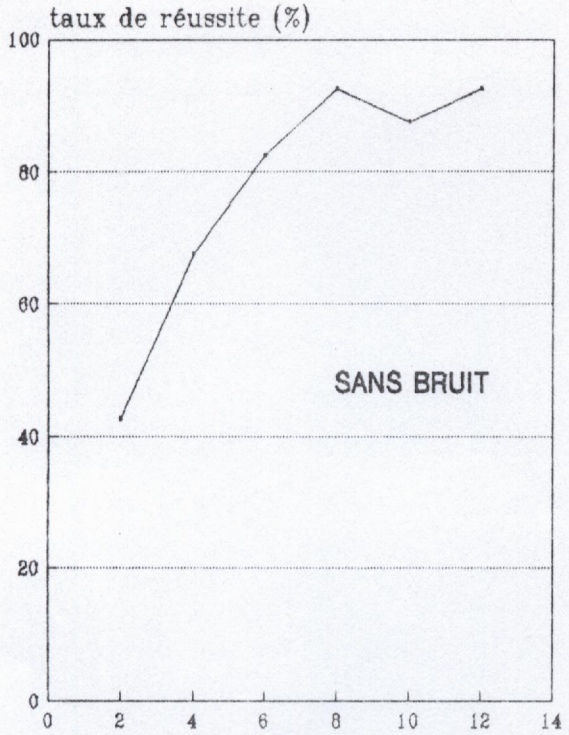
- Plus de 60 % de réussite à partir de 6 neurones pour un S/B = 12 dB.

  Par rapport aux moments invariants, ce résultat est nettement meilleur, voir figure.(4.8.d).

- Plus de 40 % de réussite à 12 neurones pour S/B = 5 dB, fig.(4.8.f)

Dans la figure.(4.9), on remarque que l'augmentation de l'ordre des moments n'a pas d'effet sur le taux de réussite jusqu'à S/B = 25 dB. L'effet devient significatif lorsque S/B diminue. En effet, dans la figure.(4.9.f), on observe une amélioration de 16 % lorsqu'on passe de l'ordre 5 à l'ordre 9.

Dans la figure.(4.10),on constate que le taux de réussite n'est pas affecté par les pannes au niveau de 3 neurones de la couche cachée.
La panne au niveau d'un neurone correspond à une sortie forcée à 0, ceci bien sûr à la reconnaissance.
Par contre le taux diminue de 53 % pour 8 neurones deffectueux.



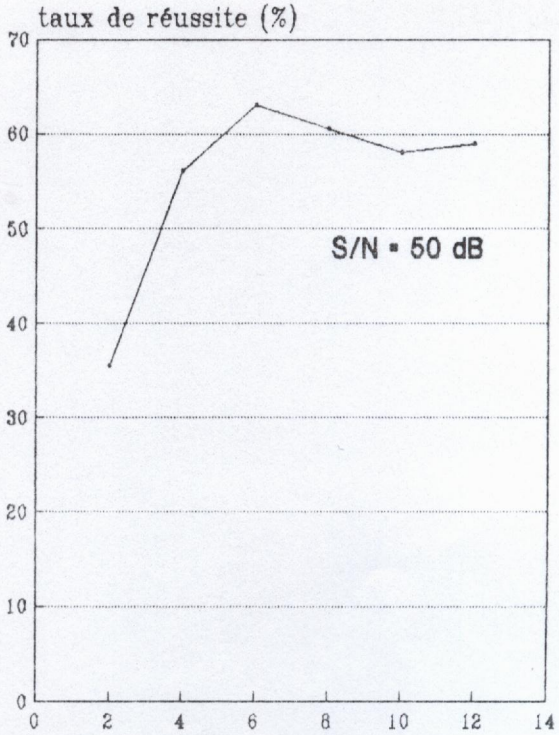
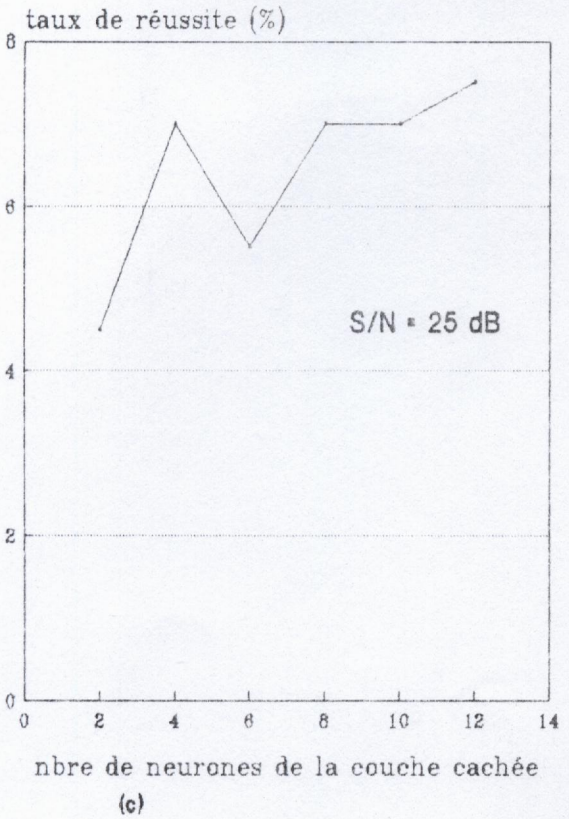
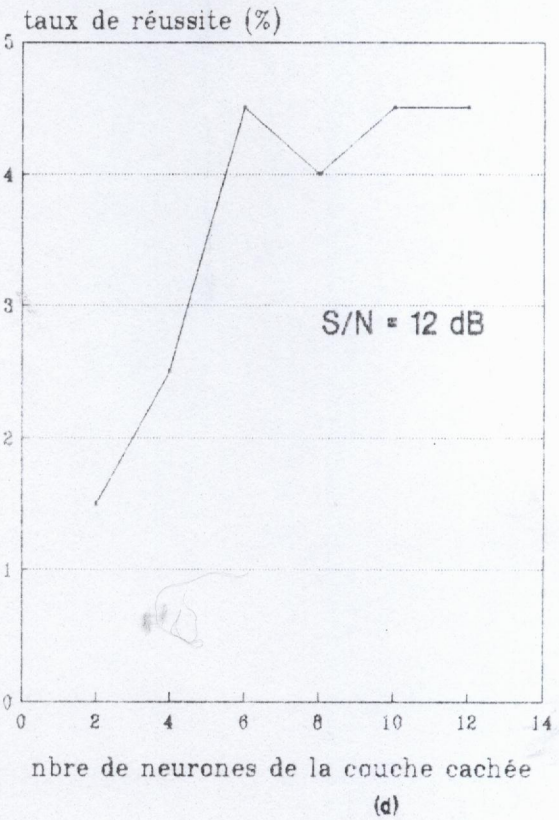


Fig.4.7: codage par moments invariants





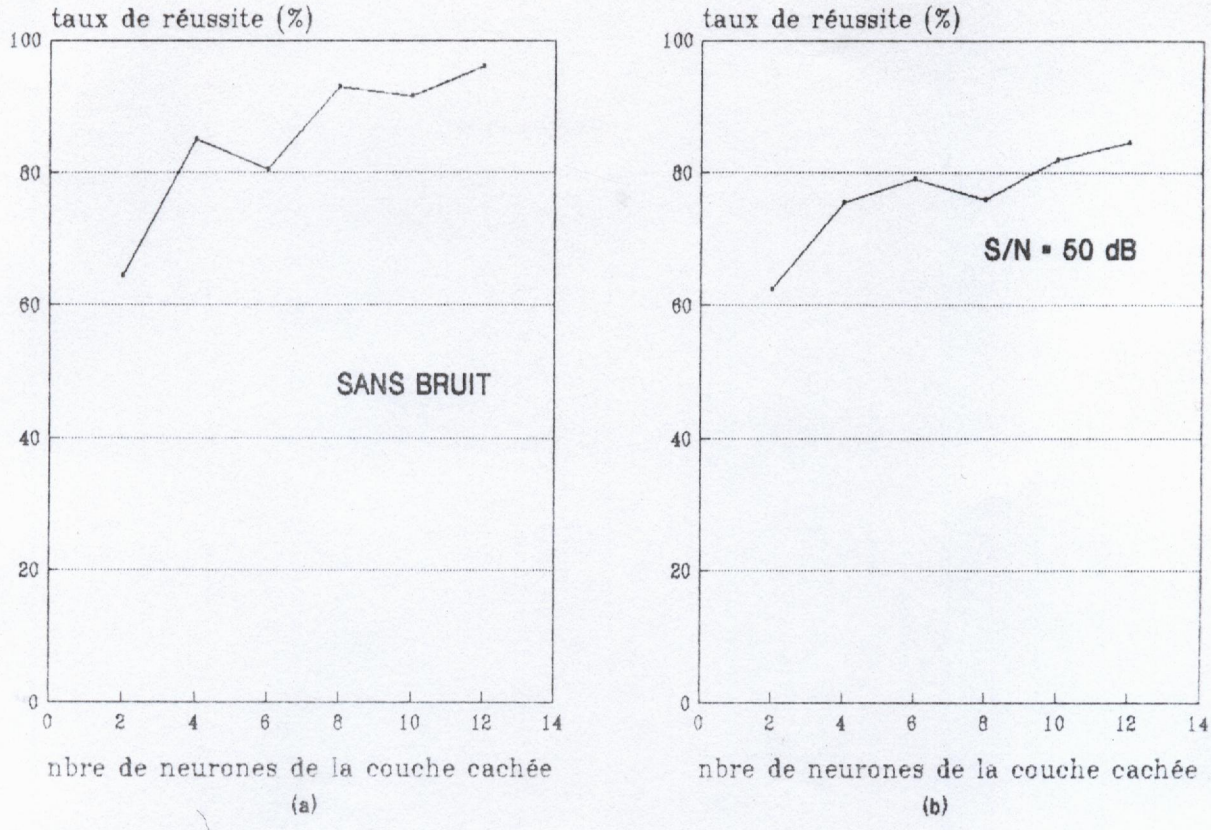
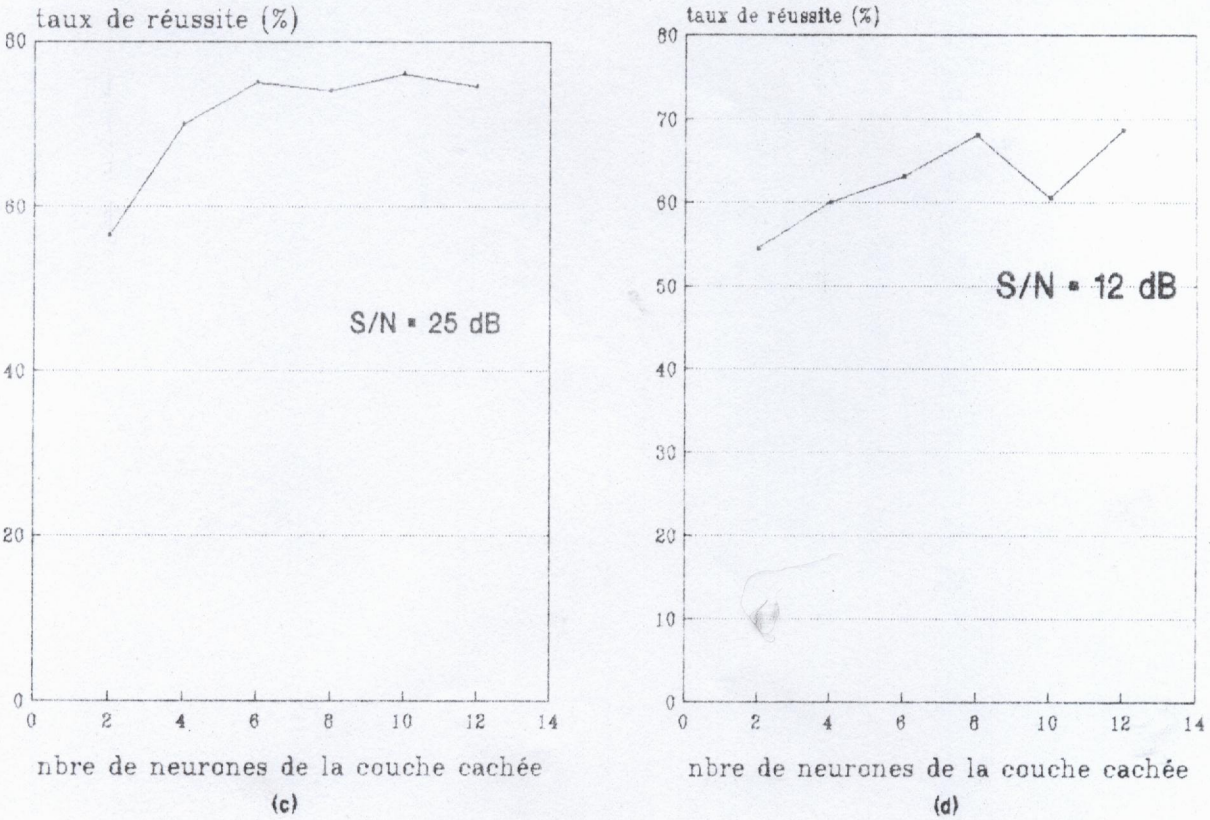


Fig.4.8: codage par les moments de Zernike (5ème ordre)



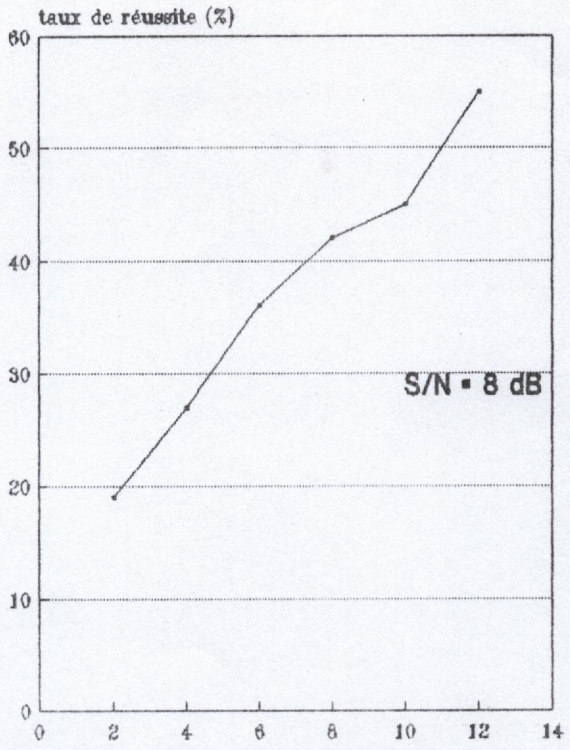
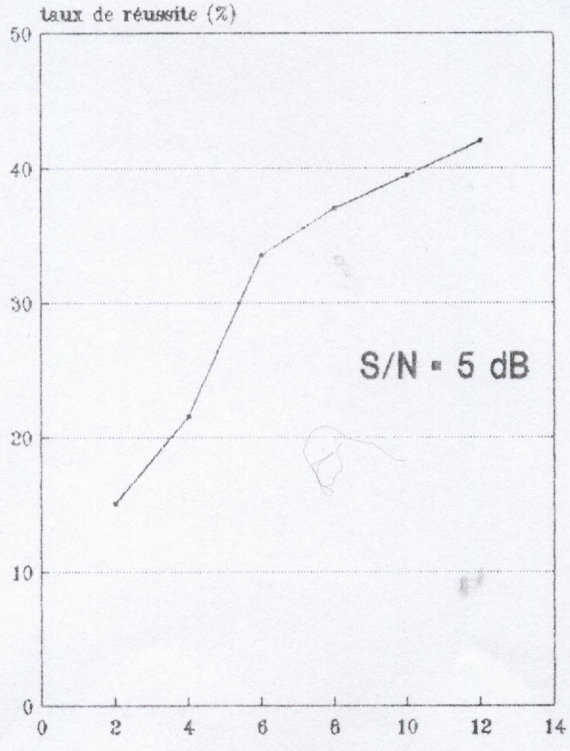


Fig.4.8: codage par les moments de Zernike (5ème ordre)



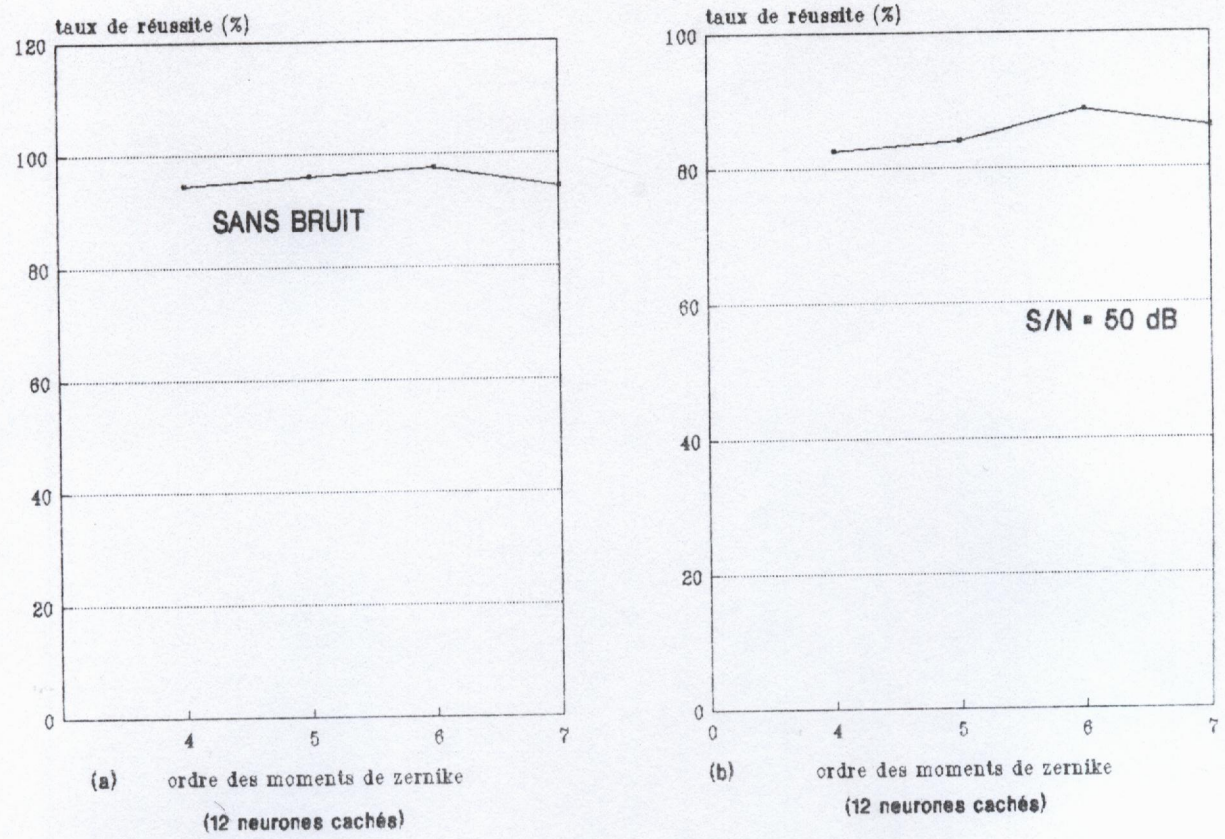
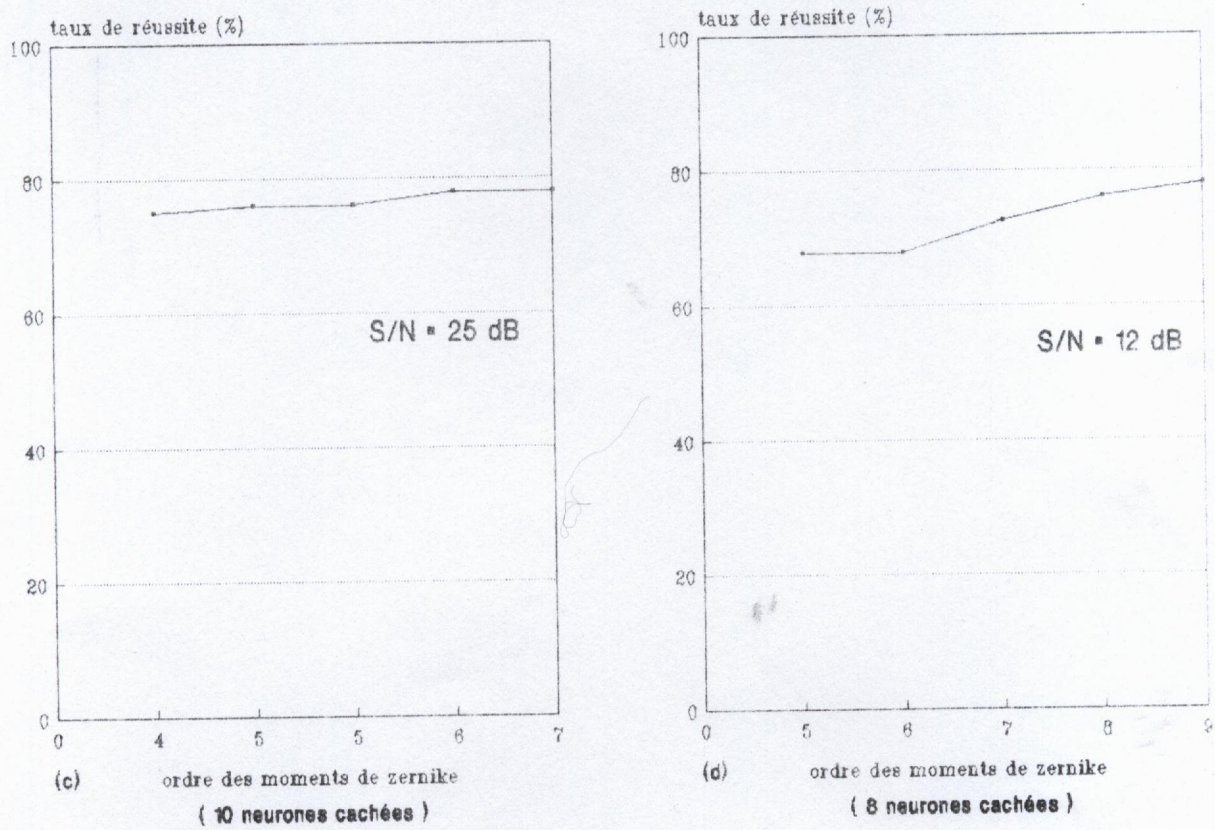
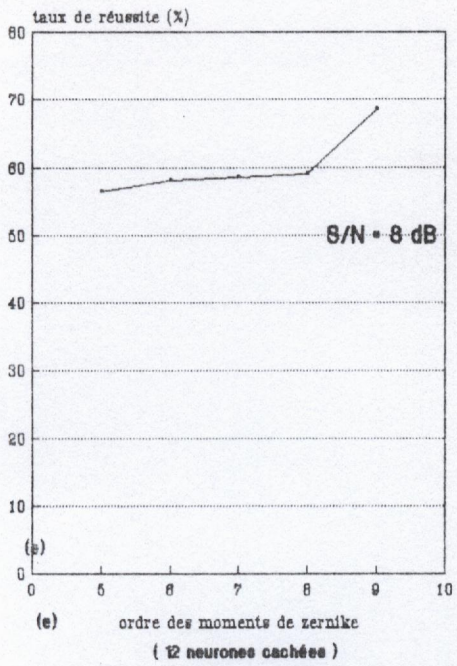


Fig.4.9: Effet de l'ordre des moments sur la classification





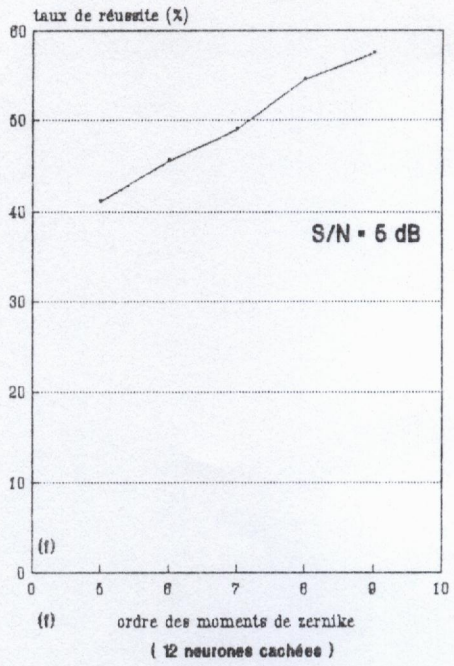


Fig.4.9

apprentissage avec 12 neurones cachés (6ème ordre)

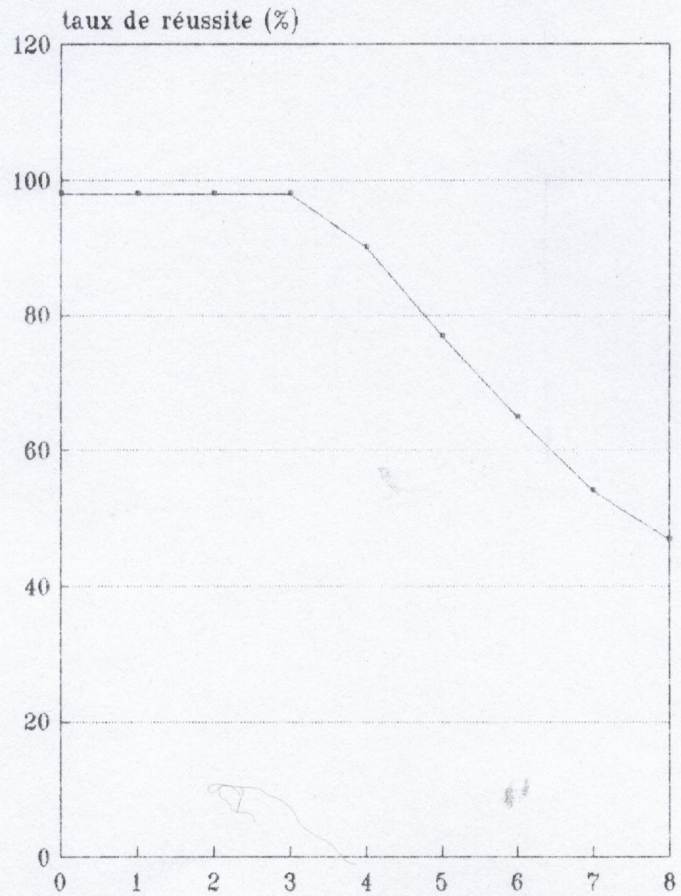


Fig.4.10

4.4. CONCLUSION

L'exemple de classification présenté dans ce chapitre nous permet de faire quelques remarques concernant les deux modèles de réseaux de neurones. En effet, si l'apprentissage était simple et rapide pour le modèle de Hopfield, il ne l'ai pas pour le cas du modèle multicouche, qui parfois n'abouti pas lorsqu'il s'agit d'un problème de grande taille.

On était limité par les possibilités du turbo-pascal en matière de mémoire, mais en réalité on peut augmenter la capacité de mémorisation de la mémoire associative en augmentant le nombre de neurones. Mais il ne faut pas oublier que plusieurs états stables non orthogonaux peuvent créer d'autres états stables attracteurs non désirables: cas de la reconnaissance du "MIM" de notre éxemple..

Pour le cas du modèle multicouche, et dans le but d'améliorer ses performances en reconnaissance, il est possible d'augmenter le nombre de neurones comme on l'a vu dans l'éxemple traité dans ce chapitre, mais au depend du temps d'apprentissage.
Quant au type de codage utilisé, nous avons remarqué que le temps de calcul des moments de Zernike d'ordre supérieur était trop élevé. Cette méthode devient inadapté pour une reconnaissance en temps réel.

Enfin , il est très important de signaler que pour une application réelle qui utilise un modèle tel que le modèle multicouche, il faut des moyens de traitement robuste tel que les ordinateurs parallèles.

CHAPITRE 5

CONCLUSION GENERALE

A la lumière de l'étude précédente, on peut dire que les raisons d'utiliser les réseaux de neurones trouvent leurs justifications dans les quelques propriétés fascinantes comme:

Le parallélisme qui constitue la base de l'architecture des réseaux de neurones considérés comme des entités élémentaires travaillant simultanément. L'intérêt d'une telle conception du traitement des données a été mis en évidence par l'échec des méthodes séquentielles.
Le neurone biologique est relativement lent (100 Hz) par rapport à un modeste processeur (10 MHz). De ce fait, le parallélisme permet une rapidité de calcul supérieur.
La plupart des applications sont simulées sur des ordinateurs séquentiels, ce qui est encore possible pour des réseaux de petites tailles, mais lorsque la taille du réseau atteint les 10^5 neurones et plus, il ne sera possible de traiter l'explosion du nombre de connexions que par des méthodes parallèles.

La capacité d'adaptation qui se manifeste dans les réseaux de neurones par leur capacité d'apprentissage qui permet au réseau de tenir compte de nouvelles données du monde extérieur. Cette adaptabilité se caractérise dans certains réseaux par leur capacité d'auto-organisation qui assure leur stabilité en tant que système dynamique.

La mémoire distribuée qui correspond à une carte d'activation des neurones. Cette carte est en quelque sorte un codage de la forme mémorisée. Cette forme mémorisée, ne sera pas perdue si un élément du réseau est détruit. D'autre part, on peut, à partir d'une donnée bruitée, faire émerger la carte d'activation neuronale de la donnée sans bruit.

La simplicité relative de la programmation d'une simulation d'un réseau, qui peut se compliquer lorsqu'on s'attaque à des problèmes necessitant de gros réseaux. Dans ce cas, il faut envisager la parallélisation ou la réalisation d'une carte coprocesseur neuronale.

Cependant, il faut noter les principales limites dans l'utilisation des réseaux de neurones à savoir:

* Le temps de calcul important pour la plupart des réseaux de grande taille simulés sur des machines séquentielles. C'est le cas du réseau multicouche qui utilise l'algorithme de la rétro-propagation vu dans notre application.

* Seul, le passage de la simulation à l'implantation sur hardware permettera l'éxploitation réelle de leur parallélisme.

* Les performances des réseaux de neurones sont sensibles à la qualité du prétraitement effectué sur les données utilisées.

*.Enfin, il faut citer l'incapacité des réseaux de neurones à expliquer les résultats qu'ils fournissent. La qualité de leurs performances ne peut être mesurée que par des méthodes statistiques.

Néanmoins, il est toujours intéressant de citer les applications connues et les quelques réalisations très encouragentes, qui ouvrent des perspectives très prometteuses dans le domaine des réseaux de neurones. Quelques applications et réalisations seront présentées en Annexe A.

L'approche connexioniste sera envisagée pour des applications où l'on dispose d'une grande quantité d'exemples avec les réponses attendues (cas des réseaux classifieurs),et aussi, là où les solutions statistiques ont donnés des résultats intéressants.

Dans le but de tester la capacité d'adaptation et de généralisation du réseau, nous avons directement coder les images des caractères arabes isolés sans tenir compte des opérations classiques de prétraitement et d'extraction des primitives des caractères utilisés [18].

Les chercheurs dans le domaine de la reconnaissance de l'écriture cursive arabe comme Amin [3] Talaat [18] et Almuallim [2], utilisent des méthodes de classifications en recherchant dans un dictionnaire arborescent préalablement rempli lors de l'apprentissage.

Ce qui m'intéresserait dans ce domaine, c'est de remplacer la multiplication des niveaux consécutifs de reconnaissance par un réseau de neuronesdu type MLP capable de faire une classification des caractères à partir de leurs primitives (boucles , branches ,noeud,ponctuation), et aussi l'amélioration des performances d'apprentissage des réseaux multicouches par des algorithmes rapides.

ANNEXE(A)

LES APPLICATIONS ET LES REALISATIONS

1. Reconnaissance des formes:

Le Néocognitron [8]: C'est un modèle en couches (MLP) avec deux couches cachées et une couche de sortie de 10 neurones correspondant aux dix formes numériques. Les performances du Néocognitron de Fukushima pour la reconnaissance des chiffres manuscrits dépassent les 98 % .

Le simulateur NDS: Une réalisation de la société NESTOR [23] où le réseau fait la reconnaissance des codes postaux après un apprentisage de 7200 caractères manuscrits prétraités puis codés suivant 9 caractéristiques. Le test se fait sur un ensemble de 1800 caractères avec un taux de réussite de 97.7 % .

Le même produit a été utilisé pour la reconnaissance des caractères japonais Kandji. Le réseau organisé en couches utilise au total 1100 neurones. Après l'apprentissage de 5600 motifs, le taux de réussite sur un ensemble de 1400 motifs est de 95 % .

Zip code recognition [19]: Un modèle en couches qui permet la reconnaissance des codes postaux, réalisé par Yann Le Cun pour la société ATT-BELL à Buffalo (USA). Le réseau est formé d'une couche d'entrée correspondant à l'image (28x28) prétraitée, du chiffre manuscrit, et de quatre couches cachées. Après l'apprentissage de 7291 chiffres, le taux de réussite à la reconnaissance est de 99 % .

2.Le traitement du signal:

La reconnaissance d'une cible:

Les applications réalisées concernent essentiellement la reconnaissance des signaux radar ou sonar.
En utilisant le même produit de base vu précédemment, la société NESTOR [23] a développée un système d'identification d'avions par signaux radar. Le résultat obtenu est:

100 % de réussite (cible identifiée à coup sûr)

95 % de réussite en environnement bruité.

La séparation des sources:

Comme exemple de séparation de sources, Jutten [13] de l'I.N.P.G (Grenoble) a développé une méthode basée sur un réseau de neurones récursif complètement interconnectés, dont les poids des connections évoluent suivant une règle d'adaptation, qui opère un test d'indépendance des sorties du réseau.
L'idée est d'extraire une source $X_j(t)$ parmi les n sources présentes dans le mélange $E_i(t)$, en éliminant dans ce mélange les (n-1) autres sources par des soustractions pondérées. La sortie $S_i(t)$ d'un neurone N_i est connectée aux entrées de tous les autres neurones $N_j(j \neq i)$ avec des poids C_{ij} ($C_{jj}=0$).

$$S_i(t) = E_i(t) - \sum_{\substack{j=1 \\ j \neq i}}^{n} C_{ij} . S_j(t) \qquad \text{avec: } E_i(t) = \sum_{j=1}^{n} a_{ij}(t) * X_j(t)$$

3. Reconnaissance et synthèse de la parole:

Le système NETTALK développé par Terence J.SEJNOWSKI et Charles R. ROSENBERG [26]. Ce réseau de neurones apprend à lire à haute voix des textes écrits en anglais. Autrement dit les chaînes de caractères lu par NETTALK seront convertit en chaînes de phonèmes qui servent d'entrées à un synthétiseur vocal. Le réseau utilise trois couches avec une couche d'entrée de 29x7 soit 203 neurones.

29 correspond à 26 caractères plus l'espace, la virgule et le point.

7 correspond à une fenêtre de 7 cases de caractéristiques qui se déplace le long du texte.

La couche cachée comporte 80 neurones, et la couche de sortie en comporte 26 pour les différents phonèmes.

Avec l'algorithme de la rétro-propagation du gradient, le temps d'apprentissage sur un DEC VAX est de 12 heures de mise à jour de connexions. Le réseau peut produire des phonèmes du texte d'apprentissage avec un taux de réussite de 95 % .Pour d'autres exemples non appris, le taux peut atteindre les 80 % ..

4. LA VISION:

Le système ALVINN " Autonomous Land Vehicule Neural Networks " [30], une réalisation de l'université de Carnegie Mellon .Pittsburgh. Le système permet le guidage d'un véhicule le long d'une route.

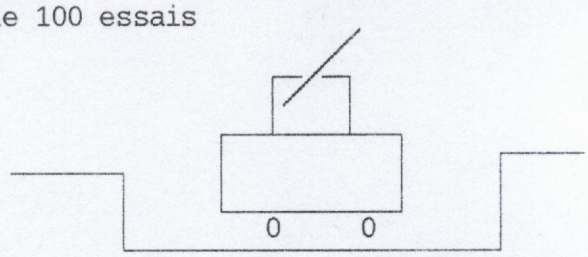
Il utilise un réseau en couches.

- La couche d'entrée correspond à l'image vidéo (30x32) de la route et au télémètre (8x32) .
- La couche cachée comporte 29 neurones.
- La couche de sortie est composée de 45 neurones représentatifs de la direction à prendre.

30 minutes d'apprentissage de 1200 routes présentés 40 fois chacune,simulées sur le super ordinateur "Carnegie Mellon's Warp Systolic Array qui peut effectuer 100 millions d'opération/s.
Le système ALVINN est capable de piloter le NAVLAB à une vitesse de 6 km/h.

5. le contrôle adaptatif

Comme exemple, un réseau mis au point par A.BARTO [4] qui peut contrôler la stabilité d'un balancier sur un petit mobile.
Les données à présenter à la couche d'entrée du réseau sont:
La position du mobile, l'angle du balancier avec la verticale, la vitesse du mobile, et la vitesse angulaire du balancier. Le réseau répond en ordonnant d'exercer une force sur le mobile soit vers la droite soit vers la gauche. Le réseau parvient à maintenir le balancier droit en moins de 100 essais



6. Les réalisations commerciales:

La plupart des simulateurs de réseaux de neurones sont distribués avec une extension matérielle représentée en général par un processeur spécialisé doté d'une quantité importante de mémoire à accès rapide.
Comme exemple de simulateurs existants, on peut citer [6]:

SIGMA 1 : Un simulateur pour IBM-PC mis au point par la société SAIC qui utilise une carte avec processeur spécialisé pour l'exécution des multiplications et des additions en virgule flottante, et une mémoire additionnelle de 12 Méga. Le simulateur implémente la plupart des modèles de réseaux de neurones existants avec un langage de spécification de réseaux: ANSPEC.

ANZA ET ANZA PLUS: Développés par les société Hecht-Nielsen et NeuroComputer, les deux simulateurs pour IBM-PC sont fourni avec une carte co-processeur basée sur un Motorola M68020 et M68881, une mémoire de 4 Méga, et un langage de spécification: AXON.

LES PROTOTYPES EN VLSI:

Pour l'implantation des réseaux de neurones en hardware, le problème posé est celui de la réalisation d'un circuit à très forte connectivité en VLSI. Citons quelques prototypes:

LE DENDROS [33] de Syntonic Systems Inc, qui contient 8 neurones analogiques avec certaines connections fixes.

LE NEURAL BIT SILICE [33] de MicroDevices, qui contient lui aussi 8 neurones à connexions programmables.

UNE PUCE développée par AT & T [6] qui contient 54 neurones analogiques et 2916 connexions programmables. Le circuit réalisé implémente le modèle de Hopfield avec un temps de convergence vers un état stable de l'ordre de 100 ns.

LE N1000 de Intel et Nestor Inc [17], qui contient 1000 neurones et qui peut traiter jusqu'à 150.10^9 CUPS (connection update per second)

```
{*****************************************************************}

 {******* M E M O I R E    A S S O C I A T I V E *********}
 {*****************************************************************}
 uses crt,dos,graph;
{$i c:cadrant.prc}
 const
  q=10;
  dim=8;
  nbrimax=1;


 var
    origmode,lastcol,lastrow,n,k,l,nbri,fi,erc,gd,gm      :integer;
    s,p,enr,db,max  :real;
    mat:array[1..dim,1..dim] of real;
    si:array[1..dim*dim] of real;
    sau:array[1..dim,1..dim] of real;
    br :array[1..dim,1..dim] of real;
    z  :array[1..dim,1..dim] of real;
    g  :array[1..dim*dim,1..dim*dim] of real;
    x  :array[1..dim*dim] of real;
    v  :array[1..64] of real;
    h  :array[1..dim*dim] of real;
    b  :array[1..dim,1..dim+1] of real;
    a  :array[1..dim*dim,1..dim*dim] of real;
    vec:array[1..dim,1..dim] of real;
    mtf:array[1..dim*dim] of real;
    mtf1:array[1..dim*dim] of real;
    namef:string[20];
    i,j,m,p1,w:integer;
    f:text;
    done:boolean;
    ch,kh  :char;
  {*****************************************************************}


  {********************* I N I T I A L I S A T I O N ***************}

   {*****************************************************************}
 procedure initialize;
   begin
       checkbreak:=false;
       origmode:=lastmode;
       textmode(lo(lastmode)+font8x8);
       lastcol:=lo(windmax)+1;
       lastrow:=hi(windmax)+1;
       gotoxy(2,lastrow-28);
       textbackground(black{blUE});
       textcolor(white{+RED});
       window(1,25,80,24);
       textcolor(black);
       textbackground(white);
write('   (CURSEUR)',' ','(ALT.W POUR EFFACER)',' ','(*.ECRITURE)   ' );
encadre(1,1,76,24);
window(1,1,80,4);
gotoxy(3,1);
textcolor(black);
textbackground(white);
write('F1_Apprentissage',' ','F2_Reconnaissance',' ','F3_RAZ Memoire'
,' ','F4_Bruit',' ','F_Sortir');
dec(lastrow,80 div lastcol);
```

```
n:=dim*dim;
    begin
         for k:=1 to dim*dim do
         for l:=1 to dim*dim do
         begin
          a[k,l]:=0;g[k,l]:=0;
         end;
    end;
  m:=0;
end;
procedure initialize1;
begin
  directvideo:=false;
  gd:=detect;
  initgraph(gd,gm,'a:');
  erc:=graphresult;
   if erc<>grok then
  begin
  writeln('grapherreur',grapherrormsg(erc));
  highvideo;writeln('n''est pas initialisé',gd,'    ',gm);readln;halt(1);
      end;
end;
procedure iimage;
begin
settextstyle(1,0,7);
outtextxy(1,50,'SIMULATION DES');
settextstyle(1,0,7);
outtextxy(1,125,' DES RESEAUX');
settextstyle(1,0,7);
outtextxy(1,200,'NEURONAUX');
delay(2000);
cleardevice;
settextstyle(1,0,7);
outtextxy(30,50,'MEMOIRE ');
settextstyle(1,0,7);
outtextxy(30,200,'ASSOCIATIVE ');
delay(2000);
closegraph;
end;

procedure vecteur_initiale;
begin
   for i:=1 to dim*dim do
   begin
       mtf1[i]:=mtf[i];
   end;
end;
{*************************************************************************}
  {*********************  S A I S A I E  **********************}
 {*************************************************************************}

procedure makewindow;
begin
  window(4,4,dim+3,dim+3);
     textbackground(white);
     textcolor(black);
     clrscr;

end; {make window}

procedure affiche;
begin
    window(25,4,40,20);
    textbackground(0);
    textcolor(1);
    for i:=1 to dim do
```

```
    begin
        for j:=1 to dim do
        begin
              if z[i,j]=+1 then write(#178)
            else  write(' ');
        end;writeln;
    end;
end;
procedure affiche2;
begin
    window(25,4,40,20);
    textbackground(black);
    textcolor(white);
     for i:=1 to dim do
       begin
          for j:=1 to dim do
           begin
               if br[i,j]=+1 then write(#178)
           end;writeln;
      end;
end;

procedure affiche1;
begin
window(45,4,60,20);
textbackground(black);
textcolor(white);
for i:=1 to dim do
  begin
    for j:=1 to dim do
        begin
          if sau[i,j]=1 then write(#178)
              else write(' ');
        end;writeln;
  end;
end;
procedure vecteur_bruit;
begin
k:=1;

  begin
      for i:=1 to dim do
        begin
          for j:=1 to dim do
                begin
                     mtf1[k]:=br[i,j];
                     k:=k+1;
                end;
         end;
   end;

end;


procedure generalise;
 begin
     for i:=1 to dim do
           begin
              for j:=1 to dim do
                  begin
                    if mat[i,j]=1 then
                           sau[i,j]:=mat[i,j]
                           else
                           sau[i,j]:=vec[i,j];
                   end;
           end;
```

```
  end;
procedure vecteur_referance;
begin
    for i:=1 to dim do
      begin
          for j:=1 to dim do
          vec[i,j]:=-1;
      end;
end;
procedure aleatoire;
begin
 for i:=1 to dim do
   begin
     for j:=1 to dim do
         mat[i,j]:=random(15)+3;
    end;
end;
procedure vecteur;
begin
k:=1;

   begin
       for i:=1 to dim do
         begin
            for j:=1 to dim do
                  begin
                       mtf[k]:=sau[i,j];
                       k:=k+1;
                  end;
          end;
    end;

end;
procedure mat1;
begin
    k:=0;
   for i:=1 to dim do
     begin
      for j:=1 to dim do
       begin
        k:=k+1;
        br[i,j]:=mtf1[k];
       end;
     end;
 end;
procedure saisie;
begin
    randomize;
    aleatoire;
    makewindow;
    done:=false;
    i:=1;
    j:=1;
    repeat
    ch:=readkey ;
    case ch of
    #0:
    begin
    ch:=readkey;
    case ch of
    #17:begin  makewindow;
    for i:=1 to dim do
    for j:=1 to dim do
    mat[i,j]:=-1;
    end;
    #45:done:=true;
```

```
    #72:begin gotoxy(wherex,wherey-1);j:=j;i:=i-1; if i<2 then  begin
    i:=1; mat[i,j]:=mat[i,j];end  else mat[i,j]:=mat[i,j] end;
    #75:begin gotoxy(wherex-1,wherey);i:=i;j:=j-1; if j<2 then begin
    j:=1;mat[i,j]:=mat[i,j];end else mat[i,j]:=mat[i,j];end;
    #77:begin  gotoxy(wherex+1,wherey);i:=i;j:=j+1;if j>dim then begin
    j:=dim;mat[i,j]:=mat[i,j];end else mat[i,j]:=mat[i,j];end;
    #80:begin gotoxy(wherex,wherey+1);i:=i+1;j:=j;if i>dim then  begin
    i:=dim;mat[i,j]:=mat[i,j];end  else  mat[i,j]:=mat[i,j];end;
    #83:delline;
    end;  { case #0}
    end;{ begin #0 }
    #3:done:=true;

    #13:begin writeln; i:=i+1;j:=1;mat[i,j]:=random(15)+3;end;
    #27:done:=true;
    else begin

    write(#178);  j:=j+1;if i*j=dim*dim then gotoxy(wherex,wherey);


    if ch='*' then
    begin
    b[i,j]:=1;mat[i,j-1]:=b[i,j];end
    else
    begin
    mat[i,j]:=random(15)+3;
    end;
    begin
    if j=dim+1 then begin
    j:=1;i:=i+1;
    end;
    begin
    if i=dim+1 then begin
    i:=i-1;j:=dim;end;
    end;
    end;
    end;
    end;{ case readkey }
until done;
  i:=1;
  j:=1;
  mat[i,j]:=random(15)+3;
  vecteur_referance;
  i:=1;
  j:=1;
  mat[i,j]:=random(15)+3;
  generalise;

end;
{*************************************************************************}
          {     V I D E R    L A     M E M O I R E   }
{*************************************************************************}

procedure vide_memoire;
begin
     m:=0;
     for k:=1 to dim*dim do
     for l:=1 to dim*dim do
     begin
     a[k,l]:=0;g[k,l]:=0;
     end;
      window(50,20,79,23);
      textbackground(white);
      textcolor(black);
      encadre(1,2,17,4);
      textbackground(black);
```

```
      textcolor(white);
      gotoxy(2,3);
      writeln('Je l''ai vidé ');
      delay(2500);
      textbackground(black);
      textcolor(white);
      clrscr;
end;

{*************************************************************************}
      { INTRODUCTION DU BRUIT }

 {*************************************************************************}
procedure affbruit;
begin
     window(25,5,60,8);
     textbackground(white);
     textcolor(black);
     encadre(1,2,31,4);
     textbackground(black);
     textcolor(white);
     gotoxy(2,3);
     write('donnez le bruit en db: ');
     readln(db);
     max:=(64/(1+exp(db/20)));
     max:=round(max);
     textbackground(black);
     textcolor(white);
     clrscr;
end;

{*************************************************************************}
 { ***************** APPRENTISSAGE ***********************}

 {*************************************************************************}
procedure apprentissage;
begin
n:=dim*dim;
{ window(25,4,40,20);
 textbackground(white);
 textcolor(black); }
   for k:=1 to n do
     begin
        for l:=1 to n do
          begin
              a[k,l]:=a[k,l]+(mtf1[k]*mtf1[l]);
            {  writeln(a[k,l]:2:2);}
          end;{readln;}

    end;
     window(50,20,70,22);
     textbackground(white);
     textcolor(black);
     write('Je l''ai appris');
end;
procedure fenetre;
begin
  window(50,20,79,23);
     textbackground(white);
     textcolor(black);
     encadre(1,2,17,4);
     textbackground(black);
     textcolor(white);
     gotoxy(2,3);
     writeln('Je l''ai appris ');
     delay(2500);
```

```
        textbackground(black);
        textcolor(white);
        clrscr;
   end;

  {*************************************************************************}
   {*C L A S S I F I C A T I O N   A P R E S   A P P R E N T I S S A G E*}
   {*************************************************************************}

  procedure grossberg;
  begin
      vecteur;
         begin
              for k:=1 to n do
              begin
                  x[k]:=-1;
              end;
          end;
      window(25,4,40,20);
      textbackground(white);
      textcolor(black);
      m:=m+1;
      x[m]:=1;
  for k:=1 to n do
       begin
            for l:=1 to n do
            begin
            g[k,l]:=g[k,l]+mtf[k]*x[l];
            end;
       end;

       window(50,20,79,23);
       textbackground(white);
       textcolor(black);
       encadre(1,2,17,4);
       textbackground(black);
       textcolor(white);
       gotoxy(2,3);
       writeln('Je l''ai appris ');
       delay(2500);
       textbackground(black);
       textcolor(white);
       clrscr;
end;
procedure polar;
begin
   for l:=1 to dim*dim do
      begin
           if mtf[l]=-1 then mtf[l]:=0;
      end;
end;

{*************************************************************************}
  { ***************** R E C O N N A I S S A N C E ********************}


procedure reconnaissance;
begin
vecteur;
nbri:=0;
repeat
nbri:=nbri+1;
      for k:=1 to dim*dim do
         begin
            s:=0;
               for l:=1 to dim*dim do
```

```
                    begin
                        s:=s+a[k,l]*mtf1[l];
                    end;
                      {   s:=(exp(s)-exp(-s))/(exp(s)+exp(-s));}
                      if s>0 then s:=+1 else s:=-1 ;

                   h[k]:=s;
            end;
 begin
   for k:=1 to dim*dim do
         mtf1[k]:=h[k];
 end;

 begin
    window(45,4,45+dim+1,4+dim+1);
    textbackground(white);
    textcolor(black);
    k:=1;
     for i:=1 to dim do
         begin
             for j:=1 to dim do
                begin
                    z[i,j]:=mtf1[k];
                 if z[i,j]=+1  then write(#178)
                  else write(' ');
                    k:=k+1;
                end;writeln;
        end;
 end;

 until nbri>=nbrimax;
 window(48,19,79,23);
 textbackground(white);
 textcolor(black);
 encadre(1,1,19,4);
 textbackground(black);
 textcolor(white);
 gotoxy(2,2);
 begin
    s:=0;
    for i:=1 to dim do
    for j:=1 to dim do
    s:=s+z[i,j];
    if s=-dim*dim then  write('je suis saturé')

                else
 begin
 write('patientez');
 gotoxy(2,3);
 write('       un peu');
 end;
 delay(1200);
 textbackground(black);
 textcolor(white);
   clrscr;
 end;
 end;
 procedure energie;
 begin
 begin
     for l:=1 to dim*dim do
        si[l]:=0;
 end;
 begin
    for l:=1 to dim*dim do
        begin
```

```
            for k:=1 to dim*dim do
              begin
               si[l]:=si[l]+a[k,l]*mtf1[k];
              end;
         end;
end;
begin
      enr:=0;
     for l:=1 to dim*dim do
       begin
         for k:=1 to dim*dim do
          begin
           enr:=enr+mtf1[l]*si[k];
          end;
       end;
end;
       enr:=-(enr)/2;
       window(50,20,79,23);
       textbackground(white);
       textcolor(black);
       encadre(1,2,17,4);
       textbackground(black);
       textcolor(white);
       gotoxy(2,3);
       writeln('enr=',enr:6:4);
       delay(2500);
       textbackground(black);
       textcolor(white);
       clrscr;
end;
procedure idvecteur;
begin
     for i:=1 to dim*dim do
     begin
      mtf1[i]:=mtf[i];
     end;
end;
procedure bruit;


   begin
     for l:=1 to 64 do
      begin
         v[l]:=0;
      end;
     j:=0;
     repeat
     j:=j+1;
     fi:=j;
     i:=random(64)+1;
     for l:=1 to 64 do
      begin
         if v[l]=i then begin
          j:=j-1;
      end;
      end;
     if fi=j then   begin  v[j]:=i;
                        if mtf1[i]=1 then mtf1[i]:=-1
                             else mtf1[i]:=1;
                   end;
     until j=max;
end;

{*********************************************************************}
{********************  C L A S S I F I C A T I O N************}
{*********************************************************************}
```

```
  procedure identification_grossberg;
  begin
    window(50,20,76,22);
    textbackground(white);
    textcolor(black);
      for l:=1 to n do
        begin
            s:=0;
               for k:=1 to n do
                begin
                     s:=s+mtf1[k]*g[k,l];
                end;
            if s>0 then s:=+1 else s:=-1;
            h[l]:=s;
       end;
   begin
       for k:=1 to n do
       begin
          mtf1[k]:=h[k];
       end;
   end;
 window(48,19,79,23);
 textbackground(white);
 textcolor(black);
 encadre(1,1,21,4);
 textbackground(black);
 textcolor(white);
 gotoxy(2,2);

 begin
     s:=0;
        for k:=1 to n do
         begin
           s:=s+mtf1[k];
         end;
     if s<>(-dim*dim+2)  then begin  write(' non ');
              gotoxy(2,3);write('je n''arrive pas'); end;
               for k:=1 to n do
                 begin
                 if (mtf1[k]=1) and (s=(-dim*dim+2)) then
                 writeln ('l''objet est le n° ',k)


                 end;
 delay(1200);
textbackground(black);
textcolor(white);
   clrscr;
end;
end;
{*********************************************************************}

 {********** P R O G R A M M E   P R I N C I P A L**************}
 {*****************************************************************}
begin   {program principal}
initialize1;
iimage;
initialize;
repeat
 kh:=readkey;
case kh of

    #59:begin
     {  curseur(true);}
       saisie;
```

```
        vecteur;
        idvecteur;
        apprentissage ;
        grossberg;
      {  curseur(false); }


         end;

     #60 :begin
           {   curseur(true);}
              saisie;
              reconnaissance;
              identification_grossberg;
{             curseur(false); }
              window(45,4,60,20);
              textbackground(1);
              textcolor(0);
              clrscr;
         end;
    #61 :begin
              vide_memoire;
         end;
    #62 :begin
           {    curseur(true);}
              affbruit;
              saisie;
              vecteur;
              w:=0;
              repeat
              w:=w+1;
              idvecteur;
              bruit;
              mat1;
              affiche2;
              reconnaissance;
    {         identification_grossberg;  }
              until w=20;
{             curseur(false); }
              window(25,4,40,20);
              textbackground(1);
              textcolor(0);
              clrscr;
              window(45,4,60,20);
              textbackground(1);
              textcolor(0);
              clrscr;
           end;
           {  #61 :begin
              affbruit;
              saisie;
              vecteur;
              idvecteur;
              apprentissage;
              w:=0;
              repeat
              w:=w+1;
              idvecteur;
              bruit;
              mat1;
              affiche2;
              readln;}
            {  reconnaissance;}
             {  apprentissage; }
            {  identification_grossberg;}
             {  until w=4; }
```

```
{            curseur(false); }
           {  window(25,4,40,20);
              textbackground(black);
              textcolor(white);
              clrscr;
              window(45,4,60,20);
              textbackground(black);
              textcolor(white);
              clrscr;
          end;}

  end;
until kh='F';
end.
```

```
    {*******************************************************}
    {***************R E S E A U   M . L . P ***************}
    {*******************************************************}
uses crt;
{$i cadrant.prc}
 const
 dim=19;
 alf=0.25;
 pci=0.75;
 cap=10;
 cl=10;
 he=6;
 fe=7;
var
   origmode,lastcol,lastrow,n,k,l,nbri,p,q,u,nbr,cr,fii,max,dd,wi,wss,xxx,sss,sss1 :integer;
   s,h,g,d,e,o,a,r,a1,a2,ero,ET,vit,db,nnn:real;
   mat:array[1..dim,1..dim] of shortint;
   sau:array[1..dim,1..dim] of shortint;
   z  :array[1..dim,1..dim] of shortint;
   lt :array[1..26,1..dim*dim] of shortint;
   de :array[1..cl,1..cl] of shortint;
   mo :array[0..1,0..1] of real;
   vi :array[1..dim*dim] of integer;
   mu :array[0..3,0..3] of real;
   br :array[1..dim,1..dim] of shortint;
   mtf1:array[1..dim*dim] of shortint;
   mtf:array[1..dim*dim] of shortint;
   w1 :array[1..fe,1..he] of real;
   v1 :array[1..he,1..cl] of real;
   nu :array[0..3,0..3] of real;
   fe1 :array[1..26,1..7] of real;
{   fe2 :array[1..47] of real;}
   fi :array[1..7] of real;
   t  :array[1..he] of real;
   b  :array[1..dim,1..dim+1] of shortint;
   w  :array[1..7,1..he] of real;
   v  :array[1..he,1..cl] of real;
   y  :array[1..cl] of real;
   delta1 :array[1..cl] of real;
   delta2 :array[1..he] of real;
   Err1   :array[1..cap,1..he,1..cl] of real;
   Err2   :array[1..cap,1..fe,1..he] of real;
   Energ  :array[1..cap] of real;
   er1 :array[1..cl] of real;
   er2 :array[1..he] of real;
   vec:array[1..dim,1..dim] of shortint;
   namef:string[20];
   i,j,m:integer;
   f:text;

   done:boolean;
   ch,kh  :char;
{*****************************************************************}
{$i initiali.prc}

{*****************************************************************}
{********* C O D A G E   P A R   L E S   M O M E N T S **********}
{*********          I N V A R I A N T S        ****************}
{*****************************************************************}

function pui(g:real;t:real):real;
begin
```

```
    if (g=0) and (t=0) then pui:=0;
    if (g=0) and (t<>0) then pui:=0;
    if (t=0) and (g<>0) then pui:=1;
    if (t<>0) and (g<>0) then pui:=exp(t*ln(Abs(g)));
    u:=round(t);
    if (g<0) and (odd(u)=true) then pui:=-exp(t*ln(abs(g)));
end;
procedure moment;
begin
    window(24,4,39,19);
    textbackground(black);
    textcolor(white);
    for p:=0 to 1 do
    for q:=0 to 1 do
      begin
        s:=0;
            for i:=1 to dim do
             begin
                 for j:=1 to dim do
                  begin
                   s:=s+(pui(i,p)*pui(j,q))*sau[i,j];
                  end;
             end;mo[p,q]:=s;
      end;
end;
procedure MIT;
begin
    window(24,4,50,30);
    textbackground(black);
    textcolor(white);
    for p:=0 to 3 do
    for q:=0 to 3 do
    begin
        s:=0;
        h:=0;
        g:=0;
          for i:=1 to dim do
             begin
                 for j:=1 to dim do
                  begin
                   h:=(i-(mo[1,0]/mo[0,0]));
                   g:=(j-(mo[0,1]/mo[0,0]));
                  s:=s+pui(h,p)*pui(g,q)*sau[i,j];
                  end;
             end;mu[p,q]:=s;
    end;

end;
procedure MIS;
begin
    window(24,4,50,30);
    textbackground(black);
    textcolor(white);
    for p:=0 to 3 do
    for q:=0 to 3 do
    begin
         h:=((p+q)/2)+1;
         s:=(mu[p,q]/pui(mu[0,0],h));
         nu[p,q]:=s;
    end;
end;
procedure MRTS;
begin

   fi[1]:=nu[2,0]+nu[0,2];
   fi[2]:=sqr(nu[2,0]-nu[0,2])+4*sqr(nu[1,1]);
```

```
    fi[3]:=sqr(nu[3,0]-3*nu[1,2])+sqr(3*nu[2,1]-nu[0,3]);
    fi[4]:=sqr(nu[3,0]+nu[1,2])+sqr(nu[2,1]+nu[0,3]);
       d:=(nu[3,0]-3*nu[1,2])*(nu[3,0]+nu[1,2]);
       e:=sqr(nu[3,0]+nu[1,2])-3*sqr(nu[2,1]+nu[0,3]);
       o:=(3*nu[2,1]-nu[0,3])*(nu[2,1]+nu[0,3])*(3*sqr(nu[3,0]+nu[1,2])-sqr(nu[2,1]+nu[0,3]));
    fi[5]:=((d*e)+o);
       a:=sqr(nu[3,0]+nu[1,2])-sqr(nu[2,1]+nu[0,3]);
       r:=4*nu[1,1]*(nu[3,0]+nu[1,2])*(nu[0,3]+nu[2,1]);
    fi[6]:=r+((nu[2,0]-nu[0,2])*a);
      a1:=(3*nu[2,1]-nu[3,0])*(nu[3,0]+nu[1,2]);
      a2:=3*sqr(nu[3,0]+nu[1,2])-sqr(nu[2,1]+nu[0,3]);
    fi[7]:=(a1*e)+a2*((3*nu[1,2]-nu[3,0])*(nu[2,1]+nu[0,3]));



end;
procedure crtfi1;
begin
    s:=0;
 for j:=1 to fe do
      begin

        if fi[j]=0 then fi[j]:=0.00000001;
           s:=s+sqr(ln(abs(fi[j])));
      end; s:=sqrt(s);
      begin
          for j:=1 to fe do
          fe1[cr,j]:=ln(abs(fi[j]))/s;
      end;
end;
{******** C O D A G E   P A R   L E S  M O M E N T S ********}
{***********         D E   Z E R N I K E         *********** }
{*********************************************************************}
procedure centgrv;
begin
    window(24,4,50,22);
     textbackground(black);
     textcolor(white);
  xg:=mo[1,0]/mo[0,0];
  yg:=mo[0,1]/mo[0,0];
{  writeln('xg=',xg); }
{  writeln('yg=',yg);readln;}
  end;
procedure rayon;
begin
     window(24,4,50,22);
     textbackground(black);
     textcolor(white);
     clrscr;
     for i:=1 to dim do
     for j:=1 to dim do
     begin
     if sau[i,j]=1 then begin
     r1:=0;
     r2:=0;
     r1:=r1+sqrt(sqr(xg-j)+sqr(yg-i));
     writeln('rayon est',r1);

      end;
     end;   readln;
end;
function R(n,l,i,j:integer):real;
var
   z1,z2 :real;
begin
     z3:=0;
```

```
     xx1:=0;
     xx2:=0;
     xx1:=round(int((n-abs(l))/2));
     xx2:=round(int((n+abs(l))/2));
     for s1:=0 to xx1 do
     begin
     r1:=sqrt(sqr(xg-j)+sqr(yg-i));
     if r1<0.000001 then r1:=0;
     z1:=(pui(-1,s1)*fonct(n-s1));
     z2:=(fonct(s1)*fonct(xx2-s1)*fonct(xx1-s1));
     z3:=z3+(z1/z2)*(pui(r1,n-2*s1));
     end;
     R:=z3;
end;
function angle(i,j:integer):real;

begin
    x:=0;
    y:=0;
    teta:=0;
       begin
       x:=(j-xg);
       y:=(yg-i);
       if x=0 then  x:=0.000001;
       if (x<0) and (y>=0) then teta:=teta+3.14;
       if (y<0) and (x<0) then teta:=teta+3.14;
       if (y<0) and (x>0) then teta:=teta+6.28;
       teta:=teta+arctan((yg-i)/(x));
       end;
       angle:=teta;
end;
procedure zerco1;
begin
    A1[n,l]:=A1[n,l]+sau[i,j]*R(n,l,i,j)*cos(l*angle(i,j));
    A1[n,l]:=((n+1)*A1[n,l])/3.14;
end;
procedure zerco2;
begin
     A2[n,l]:=A2[n,l]+sau[i,j]*R(n,l,i,j)*sin(l*angle(i,j));
     A2[n,l]:=((n+1)*A2[n,l])/3.14;
end;
procedure zerco;
begin
    for n:=2 to ord do
    for l:=0 to ord do
       if odd(n-l)=false  then
       if  l<=n then
         begin
              for i:=1 to dim do
              for j:=1 to dim do
              if sau[i,j]=1 then begin
                                    zerco1;
                                    zerco2;
                                 end;

           Az[n,l]:=Az[n,l]+sqrt(sqr(A1[n,l])+sqr(A2[n,l]));
           writeln('Az[',n,',',l,']',Az[n,l]);
         end;
end;
procedure normal;
begin
for n:=2 to ord do
for l:=0 to ord do
 begin
 if odd(n-l)=false then
 if l<=n then
```

```
 azg:=azg+sqr(Az[n,l]);
 end;
 azg:=sqrt(azg);
for n:=2 to ord do
for l:=0 to ord do
 begin
  if odd(n-l)=false then
  if l<=n then begin
  Az[n,l]:=(Az[n,l]/azg);
  writeln('Az[',n,',',l,']',Az[n,l]);
              end;
 end;
end;
procedure initbr;
begin
for i:=1 to dim do
for j:=1 to dim do
   begin
   sau[i,j]:=br[i,j];
   br[i,j]:=0;
   end;
end;
procedure crtfi;
begin
    j:=0;
    for n:=2 to ord do
    for l:=0 to ord do
     begin
       if odd(n-l)=false then
       if l<=n then begin
                     j:=j+1;
                     if Az[n,l]=0 then Az[n,l]:=0.00000001;
                       fe1[cr,j]:=Az[n,l];
                  end;
end;
end;
 {************************************************************************}
 {**************    A P P R E N T I S S A G E   **********************}
 {************************************************************************}

procedure aleatoir;
begin
    for i:=1 to fe do
  begin
    for j:=1 to he do
    begin
        w[i,j]:=(random)-0.5;
        w1[i,j]:=w[i,j];
       {  writeln(w[i,j],'   ',i,'   ',j);readln;}
    end;
  end;
    for i:=1 to he do
  begin
    for j:=1 to cl do
     begin
       v[i,j]:=(random)-0.5;
       v1[i,j]:=v[i,j];
     end;
  end;

end;
procedure app1;
 begin
       for j:=1 to he do { hidden layer }
           begin
             s:=0;
```

```
              for i:=1 to fe do  {features}
                begin
                       s:=s+(fe1[k,i]*w[i,j]);
                end;
                if s>80 then s:=80 else s:=s;
                if s<-80 then s:=-80 else s:=s ;
                    s:=(1/(1+exp(-s)));
               { y[j]:=s;}   t[j]:=s;
         end;
end;
procedure app2;
    begin
            for j:=1 to cl do
            begin
               s:=0;
               for i:=1 to he do
                 begin
                    s:=s+t[i]*v[i,j]
                 end;
                 if s>80 then s:=80 else s:=s;
                 if s<-80 then s:=-80 else s:=s;
                 s:=(1/(1+exp(-s)));
                 ys[j]:=s;
            end;

        end;
procedure app3;
   begin
       for j:=1 to cl do
       begin
          for i:=1 to he do
            begin
               delta1[j]:=(de[k,j]-ys[j])*(1-ys[j])*ys[j];
               Err1[k,i,j]:=(delta1[j]*t[i]);
            end;

       end;

    end;
procedure app4;
  begin
     for i:=1 to he do
        begin
          s:=0;
          for j:=1 to cl do
            begin
               s:=s+delta1[j]*v[i,j];
            end;
               delta2[i]:=t[i]*(1-t[i])*s;
               begin
                   for m:=1 to fe do
                       Err2[k,m,i]:= (delta2[i]*fe1[k,m]);
               end;
       end;
end;
procedure app5;
begin
    for j:=1 to cl do
      begin
          s:=0;
            for i:=1 to he do
             begin
                for k:=1 to cap do
                     begin
                     s:=s+Err1[k,i,j];
                     end;
```

```
                    vit:=pci*(v[i,j]-v1[i,j]);
                    v[i,j]:=v[i,j]+(alf*s)+vit;
                    v1[i,j]:=(vit/pci)+v1[i,j];
              end;

      end;
end;
procedure app6;
begin
    for j:=1 to he do
      begin
        s:=0;
          for i:=1 to fe do
           begin
              for k:=1 to cap do
               begin
                  s:=s+Err2[k,i,j];
               end;
              vit:=pci*(w[i,j]-w1[i,j]);
              w[i,j]:=w[i,j]+(alf*s)+vit;
              w1[i,j]:=(vit/pci)+w1[i,j];
           end;
     end;
end;

procedure apprentissage;
begin
    n:=0;
    begin
       for k:=1 to cap do
           Energ[k]:=0;
       begin
          for k:=1 to cap do
          for i:=1 to cl do
          for m:=1 to fe do
           Err2[k,m,i]:=0;
       end;
       begin
          for k:=1 to cap do
          for j:=1 to cl do
          for i:=1 to he do
          Err1[k,i,j]:=0;
       end;
    end;
repeat
begin

 for k:=1 to cap do
  begin
       app1;
       app2;
           begin
              s:=0;
                for j:=1 to cl do
                    begin
                       s:=s+sqr(de[k,j]-ys[j]);
                    end;
               Energ[k]:=s;
           end;
          app3;
          app4;
 end;

   begin
        s:=0;
        for k:=1 to cap do
```

```
            begin
               s:=s+Energ[k];
            end;
          ET:=s;

    end;
   if ET>0.1 then begin
                   app5;
                   app6;
              end;
end;
n:=n+1;
encadre(10,10,60,14);
gotoxy(11,11);
writeln('le nombre d''iteration est:',n);
gotoxy(11,13);
writeln('l''energie totale:',ET);

until ET<0.1;
      begin
      encadre(20,21,60,24);
      gotoxy(21,22);
      writeln('J''ai terminne''l''apprentissage ');
      delay(2500);
      end;

end;
{*****************************************************************************}
{************  R E C O N N A I S S A N C E  ***************************}
{*****************************************************************************}

procedure reconnaissance;
begin

         for j:=1 to he do { hidden layer }
         begin
              s:=0;
              for i:=1 to fe do  {features}
                begin
                      {s:=s+((ln(abs(fi[i]))*w[i,j])); }
                       s:=s+(fe1[cr,i]*w[i,j]);
                end;
                if s>80 then s:=80 else s:=s;
                if s<-80 then s:=-80 else s:=s ;
                    s:=1/(1+exp(-s));
               { y[j]:=s;}   t[j]:=s;
        end;

       begin
           for j:=1 to cl do
           begin
              s:=0;
              for i:=1 to he do
                begin
                   s:=s+t[i]*v[i,j]
                end;
                if s>80 then s:=80 else s:=s;
                if s<-80 then s:=-80 else s:=s;
                s:=1/(1+exp(-s));
                ys[j]:=s;
           end;
      end;
     encadre(24,10,50,10+(cl*2));
    begin
        for j:=1 to cl do
        begin
```

```
          gotoxy(25,10+j);
          writeln('ys[',j,']=',ys[j]);
          end;
          readln;
      end;

       delay(2500);
       textbackground(black);
       textcolor(white);
       clrscr;
end;
procedure sauvefile1;
begin
 namef:='zer1.dat';
 assign(f,namef);rewrite(f);
 begin
      for i:=1 to cap do
       begin
         for j:=1 to dim*dim do
          writeln(f,lt[i,j]);
       end;writeln;
 end;
 close(f);
 end;
procedure sauvfile2;
begin
 namef:='zer2.dat';
 assign(f,namef);rewrite(f);
 begin
      for k:=1 to cap do
      for i:=1 to fe do
      begin
      writeln(f,fe1[k,i]);
      end;
 end;
 close(f);
 end;
procedure sauvfilew;
begin
 namef:='zerw2.dat';
 assign(f,namef);rewrite(f);
 begin
      for k:=1 to fe do
      for i:=1 to he do
      begin
      writeln(f,w[k,i]);
      end;
 end;
 close(f);
 end;
procedure sauvfilev;
begin
 namef:='zerv2.dat';
 assign(f,namef);rewrite(f);
 begin
      for k:=1 to he do
      for i:=1 to cl do
      begin
      writeln(f,v[k,i]);
      end;
 end;
 close(f);
 end;
procedure lirefilev;
begin
  namef:='zerv2.dat';
```

```
  assign(f,namef);
  reset(f);
   begin
     for k:=1 to he do
     for j:=1 to cl   do
     readln(f,v[k,j]);
   end;
  close(f);
end;
procedure lirefilew;
begin
  namef:='zerw2.dat';
  assign(f,namef);
  reset(f);
   begin
     for k:=1 to fe do
     for j:=1 to he   do
     readln(f,w[k,j]);
   end;
  close(f);
end;
procedure caraclt;  { formation de la matrice globale des images}
 begin
      i:=cr;
      k:=1; m:=1;
 for j:=1 to dim*dim do
    begin
        lt[i,j]:=sau[k,m];
        m:=m+1;
   if m>dim then begin k:=k+1; m:=1; end;
   end;
end;
procedure reprend;  {reprise d'une image à partir de la matrice globale}
begin
     i:=cr;
     j:=1;
  for k:=1 to dim do
  for m:=1 to dim do
     begin
       sau[k,m]:=lt[i,j];
       j:=j+1;
     end;
end;
procedure initmoment;  {initialisation  des moments de zernike}
begin
  for i:=2 to ord do
  for j:=0 to ord do
       begin
       Az1[i,j]:=0;
       Az2[i,j]:=0;
       Az[i,j]:=0;
      end;
end;
procedure initializlt;   { initialisation de la matrice image globale}
begin
    for i:=1 to 24 do
    for j:=1 to dim*dim do
      lt[i,j]:=0;
end;
procedure initializde;  {initialisation du vecteur désiré}
begin
     for i:=1 to cl do
     for j:=1 to cl do
     begin
         if i=j then de[i,j]:=1 else de[i,j]:=0;
     end;
```

```
end;
procedure initializfe;    {initialisation du vecteur code d'entrée}
begin
     for k:=1 to cap do
     for j:=1 to fe do
      fe1[k,j]:=0;
end;
procedure lirefile;  {lecture du fichier code images}
begin
  namef:='zer2.dat';
  assign(f,namef);
  reset(f);
   begin
     for k:=1 to cap do
     for j:=1 to fe   do
     readln(f,fe1[k,j]);
   end;
  close(f);
end;
procedure lirefiletest;  {lecture du fichier code des images test}
begin
  namef:='zertest.dat';
  assign(f,namef);
  reset(f);
   begin
     for k:=1 to cap do
     for j:=1 to fe   do
     readln(f,fe1[k,j]);
   end;
  close(f);
end;
{*****************************************************************}
{*******************PARTIE    BRUIT*****************************}
{*****************************************************************}
procedure vecteur1; { transformation de la matrice image en vecteur}
begin
k:=1;
begin
    for i:=1 to dim do
     begin
        for j:=1 to dim do
           begin
            mtf[k]:=sau[i,j];
            k:=k+1;
           end;
     end;
end;
end;
procedure lire_fil2;
begin
     namef:='zer1.dat';
     assign(f,namef);
     reset(f);
       begin
         for k:=1 to cap do
         for j:=1 to dim*dim do
         readln(f,lt[k,j]);
       end;
      close(f);
end;
procedure mat1;{transformation du vecteur bruit en matrice bruit}
begin
   k:=0;
      for i:=1 to dim do
       begin
        for j:=1 to din do
```

```
        begin
         k:=k+1;
         br[i,j]:=mtf1[k];
        end;
      end;
end;
procedure idvecteur;
begin
    for i:=1 to dim*dim do
     begin
       mtf1[i]:=round(mtf[i]);
    end;
end;
procedure bruit; { le vecteur mtf1 est affecté du bruit}
begin
  for i:=1 to dim*dim do
    begin
     vi[l]:=0;
    end;
    j:=0;
  repeat
   j:=j+1;
   fii:=j;
   i:=random(dim*dim)+1;
   for l:=1 to dim*dim do
    begin
      if vi[l]=i then begin j:=j-1;
    end;
    end;
    if fii=j then begin vi[j]:=i;
                        if mtf1[i]=1 then mtf1[i]:=0
                        else mtf1[i]:=1;
                end;

   until j=max;
end;
procedure affbruit;
begin
    encadre(40,15,77,20);
    textbackground(black);
    textcolor(white);
    gotoxy(42,16);
    write('donner le bruit en db: ');
    readln(db);
    max:=round((dim*dim)/(1+exp(db/20*2.3)));
    max:=round(max);
    gotoxy(42,18);
    write('le nombre de pixel changer est :',max);
    window(1,1,80,25);
    clrscr;
end;
procedure affiche2; {affichage sur ecran de l'image avec le bruit}
 begin
     for i:=1 to dim do
        begin
           for j:=1 to dim do
               begin
               if br[i,j]=1 then begin gotoxy(1+j,1+i);write(#178);end
               else begin gotoxy(1+j,1+i);write(' ');end;
                end;
         end;writeln;
         encadre(1,1,dim,dim);
 end;
procedure vecteur_bruit;
begin
 k:=1;
```

```
   begin
      for i:=1 to dim do
       begin
         for j:=1 to dim do
           begin
            mtf1[k]:=br[i,j];
           end;
       end;
   end;
end;
procedure matbruit;  {identification de la matrice br à sau}
begin
 for i:=1 to dim do
 for j:=1 to dim do
     sau[i,j]:=br[i,j];
end;
procedure calcmoment;
begin
initmoment;
moment;
centgrv;
MIT;
zerco;
{readln;}    { à supprimer *}
normzerc;
crtfi;
end;
procedure initsau1; {sau prend la valeur de sau1 et initialisation de sau1}
begin
  for i:=1 to dim do
  for j:=1 to dim do
    begin
     sau[i,j]:=sau1[i,j];
     sau1[i,j]:=0;
    end;
end;
procedure inisau1;
begin
 for i:=1 to dim do
 for j:=1 to dim do
    sau1[i,j]:=0;
end;
procedure centralmoment;
begin
 inisau1;
 xg1:=round(dim/2+0.5);
 yg1:=round(dim/2+0.5);
 repeat
 moment;
 centgrv;
    begin
       for i:=1 to dim do
       for j:=1 to dim do
         if sau[i,j]=1 then begin  k:=j;
                                   k:=k+round(xg1-xg);
                                   sau1[i,k]:=1;
          {  if sau[i,j]=1 then begin gotoxy(40+j,9+i);write(#178) end}
             {  else gotoxy(40+j,9+i);write(' ');}
                 { writeln;end;}
                   { window(1,1,80,25);}
                   { encadre(40,9,dim+41,dim+10); }
                         end;
                         initsau1;
     end;
until abs(xg1-xg)<1  ;
repeat
```

```
moment;
centgrv;
   begin
     for i:=1 to dim do
     for j:=1 to dim do
       if sau[i,j]=1 then begin     k:=i;
                                    k:=k+round(yg1-yg);
                                    sau1[k,j]:=1;
           {if sau[i,j]=1 then begin  gotoxy(40+j,9+i);write(#178) end}
            {else gotoxy(40+j,9+i);write(' ');}
              {writeln;}
                  {window(1,1,80,25);}
                  {encadre(40,9,dim+41,dim+10);}
                        end;
                        initsau1;
   end;
until abs(yg1-yg)<1  ;
 end;
procedure esthetique;
begin
 initiali;
 encadre(18,8,60,24);
 textbackground(black);
 textcolor(white);
end;



begin             {programme principale}
clrscr;
esthetique;
repeat
  kh:=readkey;
case kh of

      #59:begin              {sauvegarde  des caractères}
         initializfe;
         initializlt;
         repeat
         window(1,1,80,25);
         clrscr;
         encadre(2,2,52,5);
         gotoxy(3,3);
         write('donnez la classe du carractère à sauvegarder'':');
         readln(cr);
         clrscr;
         window(1,1,80,30);
         encadre (9,9,10+dim,10+dim);
         saisie;
       { initmoment;}
         caraclt;
         sauvefile1;   {r}
         centralmoment;
         calcmoment;
         sauvfile2;    {r}
        until cr=10;
        window(1,1,80,25);
        clrscr;
        encadre(20,11,50,16);
        gotoxy(22,13);
        write('la sauvegarde est finie');
        readln;
             end;

     #60:begin   {apprentissage}
           clrscr;
```

```
            initializde;
            lirefile;
            aleatoir;
            apprentissage;
            sauvfilew;
            sauvfilev;
        end;

    #61:begin                          {reconnaissance}
             lirefile;
           {  lirefiletest;}
             lirefilev;
             lirefilew;
             window(1,1,80,25);
             clrscr;
             encadre(2,2,52,5);
             gotoxy(3,3);
             write('donnez la classe à reconnaitre:');
             readln(cr);
             reconnaissance;
        end;
    #62:begin
             lire_fil2;
             lirefilev;
             lirefilew;
             window(1,1,80,25);
             clrscr;
             encadre(2,2,52,5);
             gotoxy(3,3);
             write('donner la classe du caractere à reconnaitre: ');
             readln(cr);
             dd:=0;
             reprend;
             affbruit;
             vecteur1;
             wi:=0;
             repeat
             wi:=wi+1;
             encadre(40,10,77,24);
             gotoxy(42,13);
             write('test n° ',wi);
             gotoxy(42,16);
             write('signal sur bruit égal :',round(db));
             gotoxy(42,19);
             write('distance de Hamming :',max);
             gotoxy(42,22);
             write('le caractere est le',cr,'ieme appris');
             idvecteur;
             bruit;
             mat1;
             window(1,1,80,25);
            affiche2;
             matbruit;
             centralmoment;
             calcmoment;
             window(1,1,80,25);
             clrscr;
             reconnaissance;
             until wi=1;
             clrscr;
        end;
    end; esthetique;


until kh='f';
```

```
end.

{*************************************************************************}

{************************** fin ************************************}

{*************************************************************************}
```

BIBLIOGRAPHIE

[1] Bruce Alberts, Denis Bray, Julian Lewis, Martin Raff, Keith Roberts, James D. Watson." Biologie moléculaire de la cellule." Ed. Nov 1987, pp 1146 .

[2] A.Amin, G.Masini." Deux méthodes de reconnaissance de mots pour l'écriture arabe manuscrit; Actes Congrés AFECT,Grenoble Tome 2 pp.837-848.

[3] H. Almuallim, S.Yamaguchi." A méthod of recognition of arabic cursive handwriting; Proc.IEEE, vol.9, no.5, 1987, pp 715-22.

[4] A.G. Barto, R S. Sutton and C W. Anderson," Neuronlike adaptative elements that can solve difficult learning controlproblems, IEEE Transactions on systems, Man, and Cybernetics, vol. smc-13,no. 5, sept/oct 1983.

[5] H.D.Block." The perceptron: A model for the brain". Fonctionning I Reviews of modern physics, V.34,no 1, jan 62, pp 123-135.

[6] Eric Davalo, Patric Naim." Des réseaux de neurones".Eyrolles ed mai 1989,p 205,211,215.

[7] JEAN Delacour, Neurobiologie de l'apprentissage, Masson.

[8] Kunihiko Fukushima, Sei Miyak, and Takayuki Ito, Neocognitron: a neural network model for a mechanism of visual pattern recognition. IEEE Transactions on systems, Man, and Cybernetics SMC-13: pp 826-834.

[9] D.O.Hebb "the organisation of the behaviour". Wiley New York 1949.

[10] Hecht-Nielsen, R.(1988). Neurocomputing: picking the human brain IEEEspectrum,VOL 25,no 3, pp 36- 41.

[11] C. Arl.G. Hempel " élément d'épistémologie". Armand Colin Ed 1972. pp 15-19.

[12] J.J Hopfield " Neural networks and physical systems with emergent collective computational abilities." Proc of the National Academy of science. USA,V 19, April 1982, pp 2554-2558 .

[13] CHristian Jutten et Jeanny Heraut." Une solution neuro-mimétique au problème de séparationde sources.Soumis à la revue Traitement du signal dans cours N° 1 Neuro-mimes 88, Réseaux Neuro-mimétiques: principes. applications et réalisation .matérielles.

[14] A.Khotanzad And Y.H.Hong, " Rotation invariant pattern recognition using Zernike moments", in Proc.9th ICPR (Rome) Nov. 14-17 , 1988, pp326-328

[15] A.Khotanzad and Y.H.Hong ." Zernike moment based retation between invariant features for pattern recognition." in Proc. SPIE conf.Intelligent robots and computer vision (Cambridje,MA Nov 6-11, 1980.

[16] A.Khotanzad and J.H . Lu." Classification of invariant image représentation using a Neural Network" , IEEE Transactions on acoustics, speech, and signal processing. Vol 38. no 6, June 1990. pp 1028- 1038 .

[17] Philippe Lagrange. Les industriels découvrent l'intérêt des réseaux de neurones. Industries et techniques, Janv 1991 pp 12.

[18] Yann Le Cun."Une procedure d'apprentissage pour réseau à seuil symétrique.Proc. Cognitiva 85, Paris 1985, pp 599-604.

[19] Yann Le Cun. " Modèles connexionstes de l'apprentissage" Thèse de Doctorat Paris 6, 1987,223 p.

[20] W.S . Mc.Culloch, W.Pitts. " A logical calculus of the idea immanent in nervous activity". Bultin of Mathématicals Biophysics V.5, 1943, pp 115-133.

[21] Carver Mead. " Analog VLSI and Neural systems." Addison Wesly Publishing Company Inc. 1989,371p

[22] M.L .Minsky, S.A. Papert." Perceptro: An introduction to computational geometry." MA.MILT Press 1969.

[23] Nestor Inc. Introduction au système de développement NDS. Nestor Developpement System. Europixel Montpelier FRANCE.

[24] F. Rosenblatt " Principles of neurodynamics:Perceptrons and the théorie of the brain mécanisms".Spartan Books, Washington DC 1961? 616 p.

[25] David E.Rumelhart, G.Hinton and R.J.Williams. " Learning représentation by back propagation errors". Nature, V 323, 9 oct 1986, pp 533-536.

[26] T J .Sejnowski and C.R. Rosenberg, NETTALK: a parallel network that learn to read aloud, the Johns Hopkins University Electrical Engneering and computer Science Technical Reprt,JHU/EECS-86/01, 32 pp.

[27] S.Talaat El- Sheikh, Ramez M.Guindi: Computer recognition of arabic cursive scripts; Pattern recognition, vol.21,no.4,1988, pp.293-302.

[28] Michael Reed Teague." Image analysis via general theory of moments. J.Opt. Soc.AM. Vol.70,no 8 , aug 80. pp 920-930.

[29] C.H.Teh and R.T. Chin, " On image analysis by the methods of moments ." IEEE trans. On Patterns Analysis and machine intel. VOL.10, no 4, July 88. pp 496-513.

[30] David S.Touretzky and Dean A.Pomerleau." What's Hidden in the hidden layers?" Byte, august 1989, pp 227-233.

[31] G.Widrow,M.E.HOFF. " Adaptative switching circuits. Institute of Radio Engineer, Western Electronic Show and Convention Record, Part 4, pp 96-104.

[32] F.Zernike, Physica, Vol.1 , pp689, 1934.

[33] J.J Rousselle," Présentation des réseaux Neuro-mimétique", C.N.A.M . TOURS (France), pp 58-68 ,1991.

[34] M.K.Hu."Visual pattern recognition by moments invariants."
IRE. Trans. Inform. Theory, vol.IT-8, pp 179-187, Feb.1962.