RÉPUBLIQUE ALGÉRIENNE DÉMOCRATIQUE ET POPULAIRE

MINISTÈRE DE L'ENSEIGNEMENT SUPÉRIEUR
ET DE LA RECHERCHE SCIENTIFIQUE

ÉCOLE NATIONALE POLYTECHNIQUE

DÉPARTEMENT DE GÉNIE ÉLECTRIQUE
Option : Automatique

PROJET DE FIN D'ÉTUDES

SUJET :

RECONNAISSANCE DE CARACTÈRES PAR RÉSEAUX DE NEURONES

Proposé par :
M. M. C. SOUAMI

Étudié par :
M. KAMED SOFIANE

Dirigé par :
M. M. C. SOUAMI

PROMOTION

JUIN 1996

E.N.P. 10, Avenue Hacen Badi El-Harrach – ALGER

RÉPUBLIQUE ALGÉRIENNE DÉMOCRATIQUE ET POPULAIRE

MINISTÈRE DE L'ENSEIGNEMENT SUPÉRIEUR
ET DE LA RECHERCHE SCIENTIFIQUE

ÉCOLE NATIONALE POLYTECHNIQUE

DÉPARTEMENT DE GÉNIE ÉLECTRIQUE
Option : Automatique

PROJET DE FIN D'ÉTUDES

SUJET :

RECONNAISSANCE DE CARACTÈRES PAR RÉSEAUX DE NEURONES

Proposé par :
M. M. C. SOUAMI

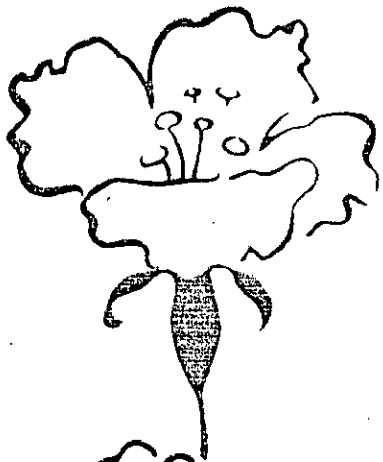
Étudié par :
M. KAMED SOFIANE

Dirigé par :
M. M. C. SOUAMI

PROMOTION

JUIN 1996

E.N.P. 10, Avenue Hacen Badi El-Harrach – ALGER



DEDICACES

Je dedie ce modeste travail à mon pays, mes parents, ma famille, mes amis(es) et à tous ceux qui m'ont aidé de près au de loin.

Remerciements

Je remercie mon promoteur

Je remercie tous les enseignants de l'École Nationale Polytechnique,

Je remercie tous mes amis(es)

... et mon Pentium.

KAMED SOFIANE

Introduction.

Les ordinateurs séquentiels ont imposé une conception de mémoire très particulière qui consiste à stocker les informations à des endroits bien déterminés et localisés : un 'mot binaire' dans une 'case'. Les mémoires de type biologique sont de nature fondamentalement différente. En effet, elles ne réalisent pas un simple stockage des informations qui leur sont proposées par l'environnement mais effectuent en plus divers traitements qui leur permettent de remplir des fonctions complexes, parmi lesquelles la fonction d'**association** entre informations.

Des fonctions mimant celles des mémoires biologiques peuvent être réalisées de plusieurs manières, que l'on peut ranger dans deux catégories ; une approche « algorithmique » et une approche « biologique » qui est les réseaux de neurones.

Depuis 1982, l'étude des réseaux de neurones connaît un développement extrêmement rapide dans des directions très variées. Une telle évolution aurait de quoi surprendre un observateur superficiel qui se souviendrait de l'échec du Perceptron à la fin des années 1960. En effet, l'idée de réaliser des structures de calcul cellulaires, plus ou moins inspirées du système nerveux central n'est pas nouvelle, mais l'évolution des modèles de réseaux et des algorithmes d'apprentissage pendant ces dernières décennies, a donné aux réseaux de neurones une grande richesse de comportement et une capacité à résoudre des problèmes complexes.

L'étude que nous présentons ici est motivée par un besoin de technologie nouvelle dans la reconnaissance des formes et spécialement dans la reconnaissance de caractères.

CHAPITRE I

Reconnaissance des formes et de caractères

1. Reconnaissance des formes. [FAU 1990]

Le fait est désormais accepté que l'ordinateur n'est pas une simple machine à calculer géante, mais un outil universel capable de réaliser n'importe quelle tâche à condition d'avoir été correctement programmé. Son usage s'est répandu dans des activités traditionnellement qualifiées «d'intelligentes»: prévisions boursières, contrôle de salles de machines, jeux de stratégie, lecture de codes postaux manuscrits, reconnaissance de la parole, ...,etc. Pour pouvoir réaliser de telles tâches avec des performances comparables à celles de l'homme, les machines doivent être programmées pour simuler son comportement dans deux domaines complémentaires et difficiles à séparer nettement : celui du raisonnement et celui de la perception des stimuli extérieurs. D'une part, le raisonnement peut être inconscient, complexe, intégré des contradictions et précédé d'un apprentissage complexe et mal connu ; d'autre part, la perception fait appel à des allers et retours incessants entre le domaine sensoriel pur et l'activité cérébrale la plus abstraite. Ces deux remarques montrent bien que l'abord de la simulation de tâches intelligentes, par ordinateur, nécessitera une programmation extrêmement complexe ; et ce n'est pas étonnant si l'on songe aux millions d'années d'évolution biologique qui ont façonné les organes des sens et le cerveau humain.

Dans ce cadre général, la reconnaissance des formes s'intéresse au traitement par ordinateur de données naturelles : cette discipline vise donc à se placer dans le champ de la perception automatique ; l'identification d'une signature sur un chèque en est un exemple, puisque les données doivent passer par des « organes des sens » (caméras, scanners, etc.) et présentent une variabilité intrinsèque à leur nature même.

La reconnaissance des formes vise donc à équiper les ordinateurs d'organes des sens ; cela ne signifie pas simplement la mise en place de systèmes physiques de capteurs, mais aussi et surtout :

- L'extraction d'informations pertinentes.
- La catégorisation des phénomènes perçus (y compris l'apprentissage des catégories).

- L'utilisation des connaissances des experts humains dans les domaines concernés.

Exemple: La reconnaissance d'un numéro de téléphone prononcé à la voix doit intégrer :

- La capture et la numérisation du son.
- L'analyse du signal de parole ainsi obtenu.
- Sa comparaison instant par instants avec tous les prototypes de sons de la langue.
- L'analyse de cette suite de décisions locales pour reconnaître la série des chiffres prononcés.

Il s'agit donc de faire abstraction des bruits, des différences d'élocution; des accents...etc. Pour retrouver le message commun dont les formes sonores, qui en sont les manifestations perçues, sont d'une variété infinie.

1.1. Extraction d'informations pertinentes.

Résoudre un problème de reconnaissance des formes , cela veut dire tout d'abord capter les informations du monde extérieur, à l'aide d'instruments de mesure assez précis pour ne rien perdre de la subtilité du phénomène. A titre d'exemple, la parole humaine devra être saisie à une fréquence d'échantillonnage de l'ordre de 8000 Hz, compte tenu du fait que très peu d'informations significatives s'y trouvent au-dessus de 4000 Hz. . Si l'on s'en tient aux outils qui sont comparables aux organes des sens humains, on va donc chercher à réaliser une automatisation de certains phénomènes de la perception. Mais on voit que la reconnaissance des formes pourra élargir son champ d'action à la catégorisation d'événements perçus par l'homme non directement, mais à travers des instruments de mesure (signaux infrarouges, signaux ultrasons,...,etc.). Les organes des sens dont dispose la reconnaissance de formes sont donc très variés, et ces techniques devront pouvoir; s'appliquer à un vaste ensemble de phénomènes.

La difficulté qui surgit d'emblée est que la quantité d'information directement issue des instruments de mesure est extraordinairement volumineuse. Une seconde de parole, en conservant l'exemple précédent, occupera un espace mémoire de l'ordre de 10^5 bits, si l'on garde une précision de 12 bits à chaque échantillon pour avoir une bonne dynamique. Or, ce signal est a priori composé d'une dizaine de phonèmes (sons élémentaires), pris dans un ensemble que les phonéticiens estiment à une quarantaine. En pure théorie de l'information, ce signal ne représente, dans l'optique de la reconnaissance de ces phonèmes, que 50 à 60 bits. La redondance, est donc extrême. En d'autres termes, l'information pertinente pour le problème doit être extraite d'une grande quantité d'information non pertinente.

Il n'en reste pas moins qu'il faut transformer les données issues des capteurs pour se placer dans un espace de représentation où l'on peut faire des calculs, ce qui signifie en

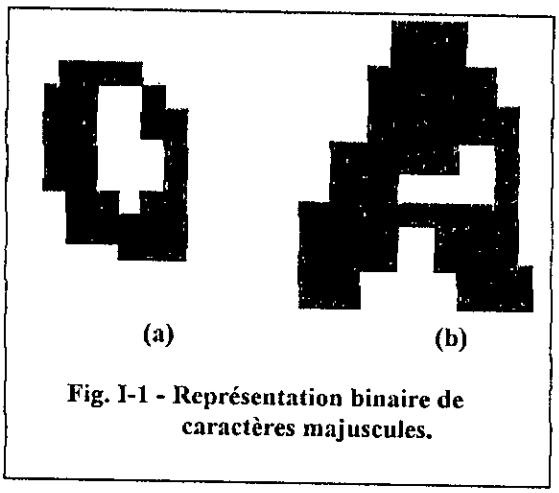


Fig. I-1 - Représentation binaire de caractères majuscules.

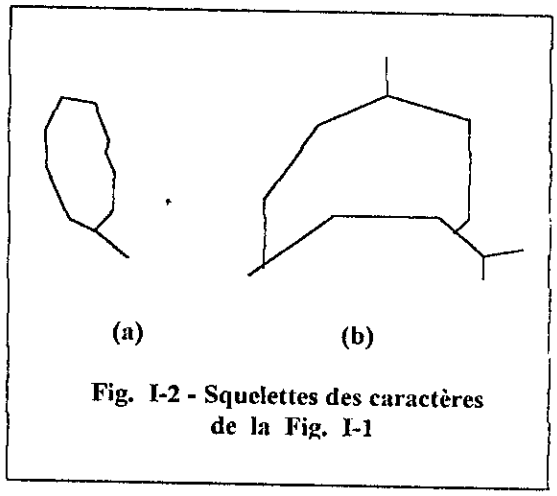


Fig. I-2 - Squelettes des caractères de la Fig. I-1

pratique deux choses:

1. Que la représentation y soit économique (en terme de place-mémoire).
2. Que l'espace possède des propriétés mathématiques assez fortes pour que l'on puisse y effectuer des opérations, faire de l'apprentissage, prendre des décisions.

Voyons l'exemple donné par la Fig. I-1 pour illustrer le premier point. Chaque image de ce type définit un caractère d'une façon extrêmement grossière (le premier -Fig. I-1 a- est-il un O ou un Q?). Et pourtant il existe environ 2^{100} images différentes de cette définition (naturellement, une grande majorité ne représentera pas un caractère).Cette représentation est d'évidence impossible à utiliser directement.

Le second point implique que l'on choisisse un espace de représentation sur lequel des algorithmes puissent travailler en temps raisonnable.

Reprenons l'exemple du signal de parole. On sait que l'oreille est sensible, d'une façon non linéaire, au spectre des sons, c'est-à-dire à la répartition de leur énergie selon les fréquences. Le calcul d'un spectre projettera la forme de départ (signal temporel) dans un espace de représentation muni de propriétés propices à la reconnaissance ; une distance entre spectre correspondra grossièrement à la distance entre les sons perçus à l'audition. Deux sons de spectres très proches (au sens d'une certaine métrique dans l'espace de représentation spectral) seront également proches du point de vue de la perception. C'est cette propriété de l'espace spectral qui le rend en l'occurrence pratique : le calcul d'une distance y est un algorithme simple, et significatif sur le plan perceptif.

Le choix de l'espace de représentation est donc un compromis à effectuer entre, d'une part, les possibilités mathématiques et algorithmiques qu'il présente et, d'autre part, la fidélité de la description des formes originales qu'il assure.

Prenons un autre exemple de la représentation d'une forme interne à l'ordinateur. Une économie de place considérable est faite si l'on assimile un caractère manuscrit à son squelette (ligne représentant en quelque sorte le tracé de ce caractère par un crayon d'épaisseur nulle).

La Fig. I-2 reprend les exemples de la Fig. I-1, sous forme de squelettes. Elle est une projection du caractère dans un espace de représentation a priori favorable. Ici surgit la question de la métrique: l'œil humain est capable de comparer deux squelettes, de décider s'ils sont proches ou non. Comment formaliser cette procédure naturelle sous la forme d'un algorithme de temps de calcul raisonnable ? Quels types de modèles mathématiques faut-il utiliser pour représenter cette information ?

On voit dans ces exemples apparaître une distinction importante, entre deux familles d'espaces de représentation, qui induiront deux types de méthodologies en reconnaissance des formes.

La première famille est celle des espaces métriques au sens habituel ; on effectue d mesures sur une forme, qui est donc représentée par un vecteur de l'espace $\mathbf{R}^d$. La décision d'appartenance à une classe se fait en utilisant les propriétés des distributions des distances dans cet espace.

La seconde famille de méthodes est celle à laquelle l'on va faire appel dans un problème comme celui de la Fig. I-2, la reconnaissance d'un caractère se fera, non par sa projection dans un espace vectoriel, mais en cherchant à traduire sa structure dans un espace de représentation adapté. Les outils dont on disposera ne viendront plus des statistiques, mais de domaines comme la théorie des langages. Notons cependant que le schéma général de la classification, quel que soit l'espace de représentation, reste semblable à lui-même : il est celui de la Fig. I-3. Comme on l'a vu, les capteurs

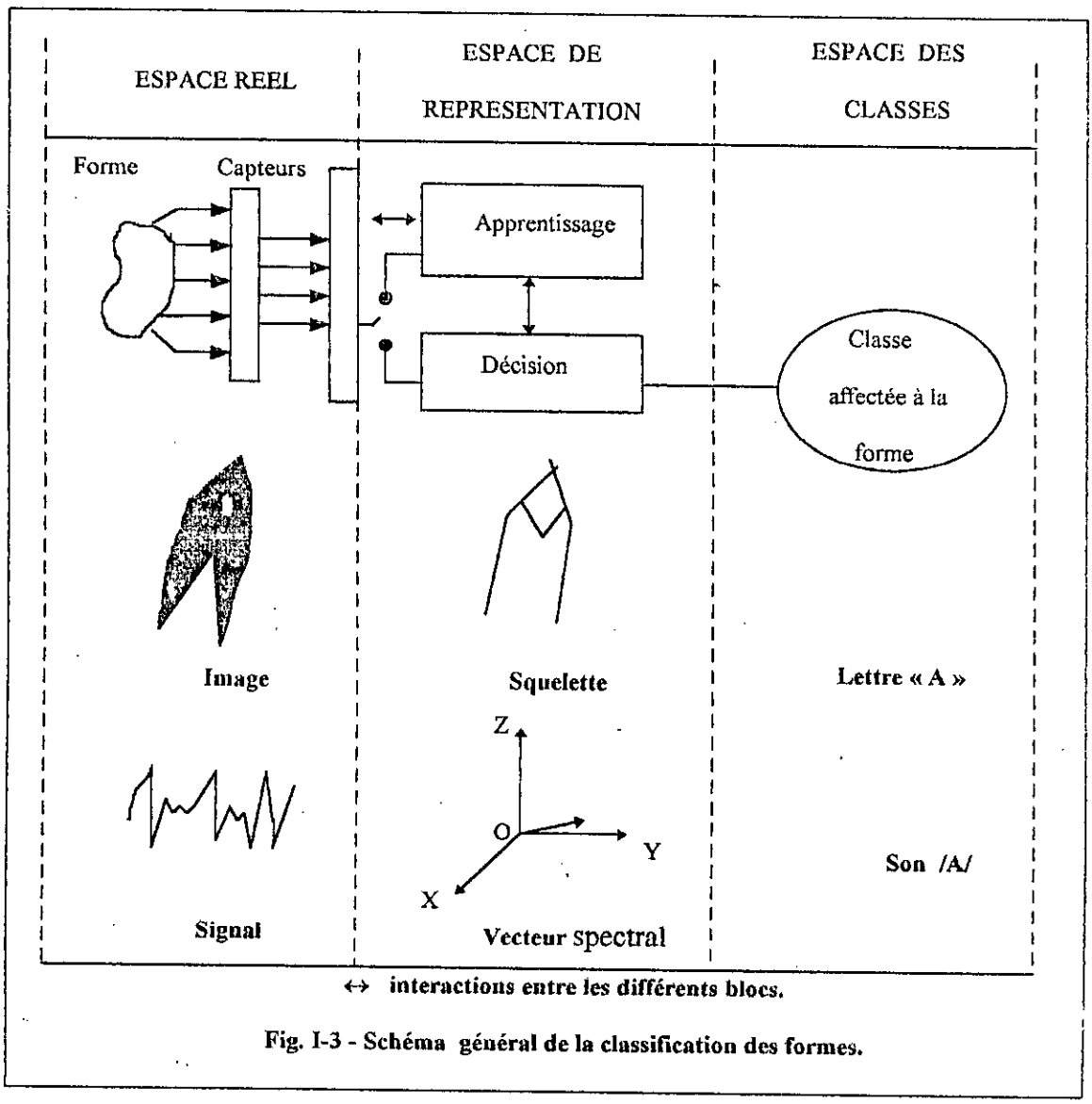


Fig. I-3 - Schéma général de la classification des formes.

saisissent une information, qui est transformée pour mettre chaque forme dans l'espace de représentation. Une phase d'apprentissage permet de trouver quelle sera la formule à appliquer pour trouver la classe d'une forme : un certain nombre de formes dont on connaît la classe d'appartenance est utilisé dans ce but. La reconnaissance elle-même projette finalement la forme inconnue dans l'espace des classes : elle est reconnue.

Dans le premier cas, on parle de **méthodes de décision statistiques** et, dans le second, de **méthodes de décision syntaxiques** (par référence à la théorie des langages et des grammaires formelles).

1.2. Techniques statistiques de classification.

1.2.1. Décision bayésienne.

On suppose être dans un espace métrique Ω de dimension ***d.*** Imaginons connaître le nombre ***m*** de classes possibles (par exemple le nombre de sons de la langue ou le nombre de lettres de l'alphabet) : soit ω_1 , ω_2 ,..., ω_m les ***m*** classes possibles. Elles sont munies d'une probabilité a priori $\mathbf{P_i}$, i = 1,...,***m*** (calculée à partir des fréquences d'occurrence de ces classes).

Les formes à reconnaître sont des vecteurs aléatoires **x** distribués selon les lois de probabilité $\mathbf{P}\ (\ \mathbf{x} / \omega_i\ )$, densité conditionnelle à l'appartenance à une classe donnée. On notera ϖ une règle de décision, c'est-à-dire une fonction qui applique l'espace des représentations dans l'espace des classes. Par conséquent $\varpi(\ \mathbf{x}\ ) = \omega_i$ signifie que la classe d'indice **i** a été attribuée au vecteur représentant la forme **x**. Supposons enfin connaître une fonction $\lambda\ (\ \omega_i\ ,\ \omega_j\ )$ qui nous indique le coût de la décision erronée suivante : « La forme **x** a été affectée à la classe ω_i , alors qu'elle aurait dû être affectée à la classe ω_j » .

Le rôle de la théorie de la décision statistique est de fournir la fonction ϖ qui minimise le coût moyen par décision prise.

Pour cela, les calculs montrent qu'il faut choisir la règle de décision bayésienne ϖ^*, qui est tel que:

$$\varpi^*(\ \mathbf{x}\ ) = \omega_i \qquad \text{si} \qquad \ell^i(\mathbf{x}) \leq \ell^j(\ \mathbf{x}\ ) \quad \text{pour } j = 1,\ldots,m$$

où $\ell^j(\mathbf{x}) = \sum_{k=1}^{m} \lambda(\omega_j, \omega_k)\ \mathbf{P}(\omega_k / \mathbf{x})$ est le risque de prendre la décision $\varpi(\mathbf{x}) = \omega_j$

Pour chaque classe d'indice k , $\mathbf{P}(\omega_k / \mathbf{x})$ et $\mathbf{P}(\mathbf{x} / \omega_k)$ sont reliés par la règle de Bayes, qui s'écrit :

$$\mathbf{P}(\omega_k / \mathbf{x}) = \frac{\mathbf{P}(\mathbf{x} / \omega_k)\ \mathbf{P}_k}{\mathbf{P}(\mathbf{x})}$$

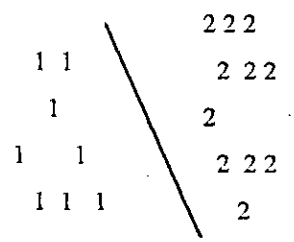
Cette règle de décision traduit bien l'intuition simple qu'il faut à chaque décision sélectionner la classe qui fait courir le moins de risque d'erreur.

L'interprétation géométrique de cette décision bayésienne aboutit à une partition de l'espace en ***m*** régions. Le vecteur $\mathbf{x}$, représente la forme inconnue, se trouve dans une de ces régions et appartient donc à la classe correspondante.

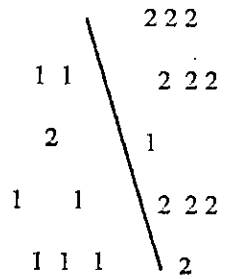
Si l'on possède la connaissance suffisante sur le problème pour remplir toutes ces hypothèses, cette décision est évidemment optimale. Toute la difficulté est d 'estimer les probabilités inconnues $\mathbf{P}(\mathbf{x} / \omega_k)$. Dans la plupart des cas, on supposera les distributions en question gaussiennes. Cela permet à la fois de développer les calculs jusqu'à arriver à une fonction de choix ϖ explicitement calculable, et en même temps d'estimer les paramètres de distribution (vecteur, moyenne et matrice de covariance) à partir d'un échantillon d'apprentissage. Celui-ci est constitué d'un ensemble $\mathbf{R}^d$ dont on connaît la classe ω_i (c'est-à-dire de formes que l'on a reconnu ou que l'on connaît).

Séparation par hyperplans.

On a vu dans le paragraphe précédent qu'une décision bayésienne était équivalente à partitionner l'espace de représentation en sous-espace attribués aux classes. Une autre façon de faire est naturellement de définir directement de tels sous-espace par leurs frontières. On dira dans le cas général que l'on cherche des surfaces discriminantes entre les classes

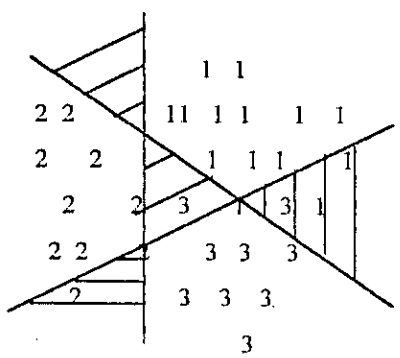


(a) deux classes gaussiennes
séparable par un hyperplan

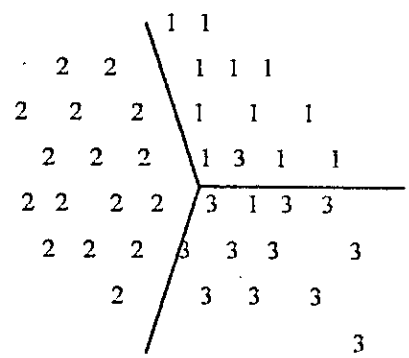


(b) deux classes gaussiennes
non séparable par hyperplan.

Fig. I-4 - Exemples de répartition de classes.



(a) Séparation du premier type:
les zones d'indécision sont hachurée



b) sépartion du deuxieme type:
pas de zone d'indicision mais quleqlues erreurs

Fig. I-5 -Séparation de trois classes par trois hyperplans.

Pour des raisons de simplicité de calcul, et sans aucun soutien de la théorie, on déterminera dans la plupart des cas des hyperplans pour séparer les classes. La Fig. I-4 montre, pour $m = 2$ et $d = 2$, quelques cas de répartition de classes en liaison avec leur séparation par des hyperplans (ici, des droites). L'algorithme du *Perceptron* résout en particulier ce problème de séparation (voir chapitre II).

Le cas multiclasses peut se ramener au cas à deux classes mais de deux façons distinctes : Soit en séparant chaque classe de toutes les autres par un hyperplan, soit en séparant les classes deux à deux.

Le premier cas mène à de larges zones d'indécision ; le second demande plus de calculs pour effectuer la décision de reconnaissance. La Fig. I-5 montre ces deux possibilités sur trois classes à deux dimensions.

Pour éviter les défauts inhérent au choix d'un hyperplan (comme vu sur la Fig. I-4), on élargit la famille des séparatrices soit en cherchant des frontières linéaires par morceaux (des lignes brisées, pour $d = 2$), soit en calculant les paramètres de surfaces, par exemple quadriques, qui séparant les classes. Les calculs sont en général beaucoup plus longs tant à l'apprentissage des frontières qu'au stade de la décision.

Une autre politique est de chercher malgré tout des hyperplans même quand les classes ne sont pas parfaitement séparables ; par exemple, ceux qui vont induire le minimum de mauvaises classifications dans l'ensemble d'apprentissage.

Décision par plus proches voisins.

Cet méthode consiste à prendre la décision d'appartenance en fonction de l'ensemble d'apprentissage sans faire d'hypothèse a priori sur la forme des classes ou sur celle de leurs séparatrice.

L'idée est simple : la forme à reconnaître est un point dans l'espace R^d. On cherche autour de lui les K plus proches voisins de l'ensemble d'apprentissage ; ils sont donc affectés d'un numéro de classe, la forme à reconnaître sera catégorisée dans la classe majoritaire parmi les K numéros déterminés.

La version la plus simple consiste ,pour K =1 , à choisir pour une forme la classe de la forme d'apprentissage la plus proche ,au sens de la métrique définie dans l'espace de représentation. La décision est alors simplement celle du plus proche voisin.

Cette technique de décision est donc à la fois cohérente avec l'intuition . Non paramétrique (pas d'hypothèse sur les classes), simple à mettre en oeuvre et, de plus on démontre qu'elle est statistiquement efficace. Son seul et gros défaut est d'avoir un temps de calcul proportionnel au nombre de formes d 'apprentissage : si celles-ci sont des centaines ou des milliers, le temps de décision devient prohibitif. C'est pour cela qu'une importante littérature s'est développée sur la recherche rapide du plus proche voisin ou des K plus proches voisins. Les algorithmes correspondants sont en général d'autant moins efficaces que la dimension de l'espace augmente .

1.2.4. Réseaux de neurones ou classification connexionistes.

Récemment sont apparues des méthodes de classification fondées sur des principes différents. Elles relèvent du champ général des « réseaux de neurones formels », qui intéressent à la fois les neuro-biologistes, les physiciens et les informaticiens. En ce qui concerne la connaissance des formes, on ne s'intéresse qu'à un sous-ensemble de ce champ d'études, celui où se trouvent des outils qui permettent une décision de type statistique dans $\mathbf{R}^d$. Nous verrons plus en détail cette technique dans les prochains chapitres.

1.3. Techniques syntaxiques de décision.

Nous allons tenter de décrire et de reconnaître les formes avec une approche tout à fait différente.

Chaque forme est exprimée maintenant comme un agencement structuré de formes élémentaires ou primitives.

Ce qui est déterminant, c'est moins la présence d'une sous-forme primitive que la façon dont elle se compose avec d'autre pour structurer la forme globale.

L'exemple de la Fig. I-6 aidera à préciser ce concept : les formes élémentaires sont

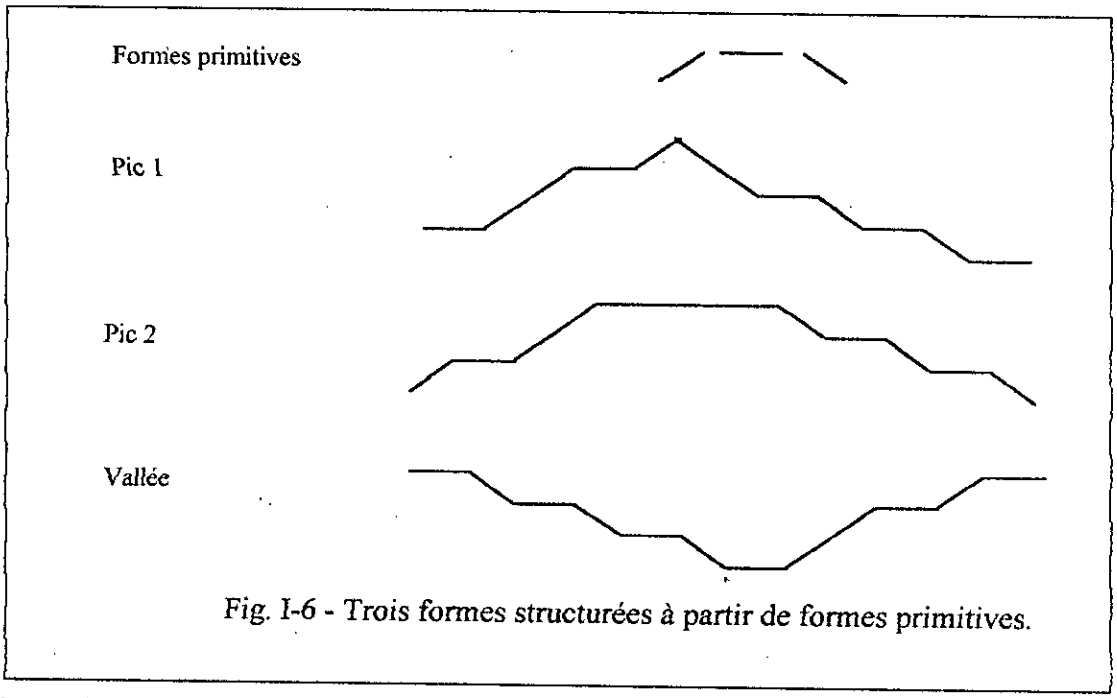


Fig. I-6 - Trois formes structurées à partir de formes primitives.

les petits segments de trois type possibles : montants, horizontaux, descendants. On peut former avec eux, par exemple trois assemblage différent, en les ajoutant les uns à la suite des autres : deux *pics* et une *vallée* .

Il est clair que si l'on veut dans cette représentation distinguer les *pics* des *vallées*, il va falloir utiliser des outils différents de ceux employés au paragraphe précèdent : ce n'est pas le nombre relatif des segments montants, horizontaux ou descendant qui va importer, mais lès propriétés de leur enchaînement. Quels sont les outils permettant de décrire la forme général *pic* sous cette représentation ? C'est à ce genre de questions que les méthodes syntaxique tentent de répondre.

1.3.1. Structures de chaînes .

Comme l'exemple précèdent l'a montré, une description simple de la structure d'une forme peut être faite en la codant comme une suite ordonnée de formes élémentaires appartenant à un ensemble finis. En terme de théorie des langages, un tel ensemble de symboles :

$$[\ \mathbf{X} = \{ a, b, \ldots, n \}\ ]$$

est appelé un alphabet, et une telle suite notée simplement :

$$[\ \mathbf{x} = a_1 a_2 \ldots a_p\ ] \qquad \text{avec } a_i \in \mathbf{X}$$

est appelée une phrase (on dit aussi chaîne) ; leur ensemble est noté $\mathbf{X}^*$ cette description est, bien entendu adaptée aux formes munies d'un ordre total naturel (signaux fonction du temps, par exemple) ; elle se prête aussi à la description de la frontière d'une image ou à toute séquence d'événement dont l'ordre est la structure fondamentale.

Dans l'optique de la reconnaissance des formes, un ensemble d'apprentissage est donc un ensemble de phrases munies d'étiquettes indiquant la classe d'appartenance. Si l'on sait comparer deux phrases c'est-à-dire définir une métrique sur l'espace $\mathbf{X}^*$, rien n'empêche d'utiliser des techniques de décisions par plus proches voisins pour aider à l'apprentissage.

Des algorithmes de comparaison de chaînes ont été inventés dans cette optique; on impose en général des contraintes au propriétés de cette distance, de sorte que les calcul puissent s'effectuer en un temps raisonnable, grâce aux principes de la programmation dynamique, des circuits intégrés spécialisés sont déjà réalisés pour effectuer par ce type de technique de la reconnaissance de plusieurs milliers de mots parlés.

1.3.2. Automates et grammaires.

Les automates et les grammaires sont des outils mathématiques dont le rôle est d'engendrer des ensembles de phrases munies d'une structure commune. Le principe de leur application à la reconnaissance syntaxique est donc simple : on tente de modéliser une classe de formes par une grammaire ; une forme inconnue, représentée

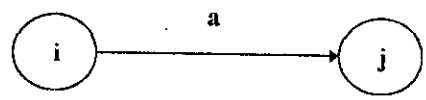
par une phrase, est testée pour déterminer si la grammaire peut l'engendrer auquel cas, elle est susceptible d'appartenir à la classe correspondante. En d'autres termes, sa syntaxe correspond à la grammaire de la classe.

Reprenons tout d'abord l'exemple des pics (Fig. I-6), et construisons un automate dont le but est de détecter la forme pic dans un signal. Décrit de manière informelle, un automate (ou, pour être plus exacte un automate fini) est représenté par un graphe, dont on appelle les nœuds des états et dont les flèches entre nœuds portent des symboles de l'alphabet.

L'élément de base d'un automate est donc la figure suivante. Cela se traduit par : on peut aller de l'état n° i à l'état n° j par le symbole **a**.

Dans notre exemple, l'alphabet est composé des trois symboles: ↗ , → ,↘ .

Définissons simplement un pic par les formules suivante :



a) un pic est composé d'une partie ascendante, d'une partie horizontale (éventuellement de longueur nulle) et d'une partie, descendante;

b) une partie ascendante est composée de symboles ↗ , parmi lesquels on peut trouver un symbole → , mais jamais deux à la suite;

c) une partie horizontale est composée d'une suite de symboles → .

Une partie descendante a la même définition qu'une partie ascendante, en remplaçant le symbole ↗ par le symbole ↘ .

Moyennant cette définition, on peut alors représenter l'ensemble des pics par l'automate de la Fig. I-7. Son interprétation est la suivante:

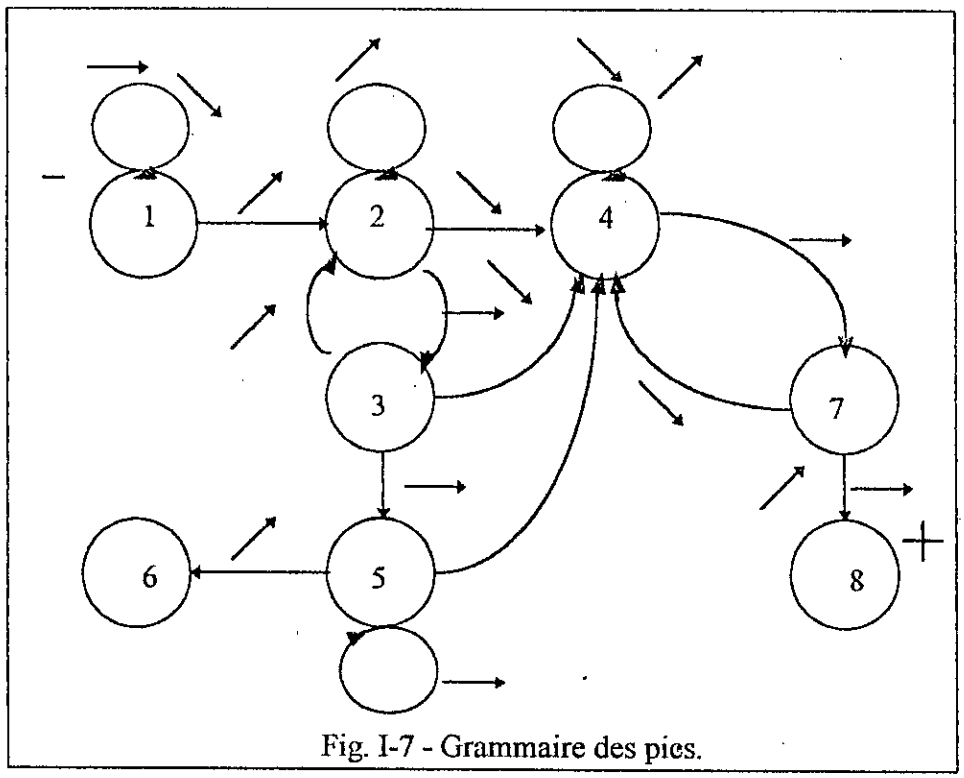


Fig. I-7 - Grammaire des pics.

a) L'état 1 est l'état initial (noté par le signe -) ; on y reste tant que la partie ascendante n'a pas démarré par la lecture du symbole ↗ ;

b) L'état 2 est celui qui indique que l'on est dans la partie ascendante; on le quitte par le symbole → (éventuellement pour revenir immédiatement par ↗) et par le symbole ↘ , qui envoie dans l'état symétrique 4, indiquant la partie descendante;

c) L'état 3 indique que, durant la montée, on a lu un symbole horizontal; on se décide sur le symbole suivant pour soit revenir à l'état 2 par une montée. Soit entamée la descente par le symbole ↘ (menant vers l'état 4), soit aller en 5;

d) L'état 5 est l'état indiquant la partie horizontale, l'état 6 est une indication d'erreur;

e) L'état 7 joue le même rôle que l'état 3, pour la partie descendante;

f) L'état 8 indique la fin du pic ; c'est l'état de réussite, indiqué par le signe +.

On voit donc qu'une phrase composée sur l'alphabet { ↗ , → , ↘ }, et menant de l'état 1 à l'état 8 représente un pic, au sens de la définition donnée précédemment. Cet automate reconnaît donc la forme pic. Toute autre forme sera rejetée.

Les automates finis sont les modèles grammaticaux les plus simples dans la hiérarchie classique de la théorie des langages. Les propriétés structurelles des familles de formes qu'ils peuvent représenter sont limitées. Leur avantage est la simplicité et la souplesse d'utilisation, même pour un grand nombre d'états.

D'autres types d'automates et de grammaires sont utilisés par les spécialistes de la reconnaissance des formes. Contrairement aux cas simplifié des automates finis, ces modèles peuvent représenter des structures complexes.

2. Reconnaissance de caractères. [REM 1991]

Depuis quelques décennies, de multiples recherches et développements ont été consacrés à la reconnaissance de textes écrits, dactylographiés ou imprimés. Le premier lecteur optique de caractères date de 1955. En 1956, la Farrington Company introduit la lecture optique dans l'industrie pétrolière avec Scandex. Ce procédé apparaît alors comme une simplification par rapport aux techniques de saisie de l'époque, qui étaient la carte ou le ruban perforés. Surtout, la lecture optique devait permettre d'entrer les données dix fois plus vite qu'un opérateur sur clavier, ils traitaient de très gros volumes de documents par lots.

Lorsque furent introduits les systèmes de saisie sur supports magnétiques et que parurent, plus tard, les premiers résultats de la reconnaissance vocale, les efforts de recherche sur la lecture optique furent abandonnés par la plupart des grands groupes .

Seuls des constructeurs spécialisés restaient en lice. Au début des années soixante. Des firmes travaillant pour la presse, tels CompuScan, ECRM, etc., avaient eu l'idée de réaliser des machines permettant de composer automatiquement les journaux à partir des textes dactylographiés fournis par les journalistes. C'est ainsi qu'un éditeur de presse américain, Perry Publications , appliqua la technique de reconnaissance optique de caractères dès 1965. En 1974. L'Américain Raymond Kurrweil a eu l'idée de concevoir un lecteur optique qui permettrai aux aveugles de lire.

La machine qu'il réalise associe la reconnaissance optique à la technique de synthèse vocale pour la restitution. C'est le point de départ de plusieurs générations de lecteurs optiques, appliqués non seulement pour les non-voyants, mais également en association avec des systèmes de traduction automatique et pour la saisie en imprimerie.

Originellement réservée aux grands systèmes, la lecture optique de caractères connaît depuis quelques années un nouvel essor, dû à la disponibilité de processeurs plus puissants et de mémoires plus spacieuses. Alors que les premiers systèmes étaient réalisés en logique câblée, donc sur des matériels dédiés, la lecture optique est désormais accessible aux micro ordinateurs, dotés de cartes de numérisation et de logiciels. Dès lors, son prix s'est fait plus abordable. Par ailleurs, après plusieurs

générations de logiciels. Les systèmes actuels sont parvenus à une maturité suffisante, tant du point de vue de la rapidité et de la convivialité que de la fiabilité.

Toutefois, quel que soit leur degré de sophistication, aucun des logiciels de lecture de caractères actuellement disponibles n'est encore capable d'atteindre la perfection:

- aucun système ne sait reconnaître efficacement n'importe quel type de document dactylographié ou imprimé;

- les meilleurs systèmes ont des taux de reconnaissance de l'ordre de 99 %. Si L'on entend par taux d'erreur le pourcentage de caractères mal reconnus, celui-ci doit être, en effet, minime. Un taux de 1%, par exemple, est inadmissible: cela représenterait quelque 80 erreurs sur une page imprimée, comme celle-ci !

2.1. Principe de l'OCR.

Un texte écrit peut être considéré soit comme un document image, représenté par un ensemble de pixels, soit comme un ensemble de caractères ASCII (American Srandard Code for Information Interchange) ou autres, ou document texte. Cette seconde forme prend environ 80 fois moins de place en machine que la première. Elle peut être reprise par des logiciels pour subir des traitements informatiques (traitement de texte, publication assistée par ordinateur, ou PAO, tableur, base de données, traduction automatique, indexation, etc.). D'où l'intérêt de pouvoir passer de la première forme à la seconde. C'est ce que fait la reconnaissance optique de caractères, plus fréquemment désignée par le sigle OCR (Optical Characters Recognition), qui transforme le document image (papier numérisé représentant un texte dactylographié, imprimé ou, plus rarement, manuscrit) en mode texte.

Pour cela, un système d'OCR doit remplir deux fonctions:

- la saisie (c'est à elle qu'est lié le qualificatif d'optique puisqu'elle se fait par éclairement du texte à lire);

- l'analyse des caractères et la reconnaissance proprement dite, fonctions assurées par un logiciel spécifique.

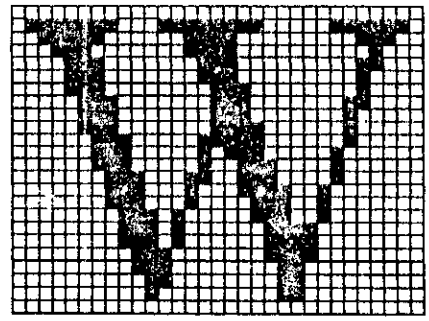


Fig. I-8

la première tâche correspond à celle de la rétine humaine qui perçoit des signaux. Le texte est balayé par une tête de lecture mobile comprenant un faisceau lumineux. et le capteur optique, caméra ou numériseur (scanner). Alors que, dans un photocopieur, une lumière projetée sur une feuille de papier imprimée est réfléchie de manière plus ou moins intense selon qu'elle éclaire les parties blanches ou encrées de la page, ici ce phénomène est transformé en signaux électriques, mais ceux-ci au lieu de servir à impressionner une feuille blanche par passage sur un tambour encre, sont mémorisés sous forme binaire (un point noir est représenté par 1 et un blanc par 0 par exemple). Ainsi, à un caractère lu correspond une matrice de points telle que celle représentée à la Fig. I-8 .

L'image numérisée est transférée dans une mémoire qui permet le stockage d'un ou plusieurs caractères, voire une ou plusieurs lignes.

C'est le principe de la lecture optique, dont les lecteurs de codes à barres sont une application largement répandue. Le but de l'OCR est la conversion de l'image d'un texte (dactylographié, typographie ou, plus rarement, manuscrit), numérisée par un scanner, en un fichier exploitable par un logiciel de traitement de texte, un tableur, ou tout autre traitement informatique.

L'utilisation d'un système d'OCR impose un certain nombre de contraintes sur le matériel. Outre l'ordinateur (aujourd'hui, un PC ou Macintosh sont généralement suffisants), les éléments nécessaires sont:

- Un scanner, et souvent une carte d'interface scanner,
- Parfois un co-processeur spécialisé,
- Une bonne puissance de calcul et de la mémoire vive (au moins 1 Mo sur PC).

2.2. Technique de reconnaissance.

Il existe plusieurs catégories de logiciels d'OCR, depuis ceux qui se limitent à la reconnaissance de caractères jusqu'aux systèmes capable d'analyser des pages entières, des paragraphes, colonnes, tableaux etc. Avant de procéder à la reconnaissance proprement dite. Le traitement des informations saisies passe par deux étapes préliminaires :

- La recherche des données à lire,
- La segmentation des caractères.

La recherche des données peut se faire sur L'ensemble du document numérisé ou dans des plages définies par leur position, au moment de la saisie.

Les caractères sont isolés, à l'intérieur des plages ainsi définies, en tenant compte ou non des espaces inter-caractères et de leurs caractéristiques dimensionnelles moyennes. C'est ainsi que certains systèmes ne sont capables de reconnaître que des caractères à chasse fixe, alors que d'autres, plus généraux, s'appliquent aux chasses variables. En ce qui concerne la reconnaissance de ces caractères, on distingue principalement deux techniques : la méthode analytique et la méthode omnifonte.

2.2.1. Reconnaissance analytique.

La méthode analytique de reconnaissance, ou méthode matricielle, est à la fois la plus simple et la plus ancienne. Chaque signe « vu » par le logiciel est comparé à l'un des modèles préalablement mémorisés, qui constituent les différents signes d'un jeu de caractères. A partir de cette comparaison matricielle, consistant à superposer la matrice du caractère à reconnaître avec le modèle correspondant, on obtient un ensemble de coefficients de corrélation indiquant la concordance ou la discordance entre le caractère à reconnaître et chacun des caractères de référence auxquels il est confronté. Ce caractère recherché est soit celui dont le coefficient de corrélation est le plus élevé, soit celui dont ce coefficient dépasse un certain seuil. Cette méthode a l'avantage de

demander peu d'instructions, donc d'être facilement implémentée sur microprocesseur mais, fondée sur la logique binaire elle manque de souplesse:

- L'espacement fixe entre les caractères est nécessaire;
- Un signe est rejeté dès qu'il ne correspond pas exactement au modèle enregistré.

Les limitations d'un tel système apparaissent immédiatement: un coefficient de 100 % (superposition parfaite entre le caractère vu et l'un des caractères mémorisés) est rarement atteint. On fixe donc généralement le seuil entre 80 et 90%. Dans ce cas, plusieurs caractères peuvent répondre aux critères de reconnaissance et il y a donc ambiguïté, si l'on ne peut départager les différents caractères pertinents, ou erreur, si le caractère retenu n'est pas le bon.

Par ailleurs, comme il n'est pas possible de mémoriser une quasi-infinité de formes différentes, correspondant à toutes les fontes possibles, un tel système est donc limité à certaines fontes.

Pour faciliter la reconnaissance des caractères par les premiers systèmes d'OCR, la forme de ceux-ci fut normalisée. C'est pourquoi l'American National Standards Institute (ANSI) développa en 1960 la norme OCR A, selon laquelle chaque caractère, à espacement fixe, a une forme bien distincte de tous les autres ; en particulier, le 0, le O et le Q, le 2 et le Z sont nettement différenciés. Cinq ans plus tard, l'Association européenne des constructeurs d'ordinateurs (ECMA) proposait la norme OCR B, dont les caractères ont des formes moins étranges et plus esthétiques. Depuis lors, de nouveaux systèmes sont apparus qui s'accommodent des caractères habituels d'imprimerie ou de dactylographie.

Une variante de la méthode de corrélation matricielle est la méthode vectorielle. Chaque caractère y est représenté par un ensemble de vecteurs que le système d'OCR compare avec lés modèles déjà connus par le système. Cette méthode s'avère plus puissante que la méthode matricielle, à la fois par sa vitesse et sa capacité d'adaptation: cependant, sa principale difficulté réside dans la détection de vecteurs à partir d'ensembles de points.

Comme les logiciels de lecture optique fondés sur la méthode analytique ne peuvent pré-mémoriser un grand nombre de fontes différentes (à la fois pour des raisons de capacités de mémoire et de temps de traitement), ils sont souvent dotés d'une fonction d'apprentissage qui permet de les enrichir de nouveaux caractères pour une application donnée.

La fiabilité de ces logiciels dépend de nombreux facteurs, comme le type de scanner, sa résolution, la qualité du papier. Cette méthode impose tout un ensemble de contraintes concernant les propriétés optiques du papier (réflectance, pureté, opacité), le scanner (type, résolution, réglages de luminosité et de contraste), l'encre et les caractéristiques d'impression des caractères (couleur noire obligatoire, régularité et largeur des traits, etc.), ainsi que les espacements minimal et maximal des caractères et la hauteur des interlignes .

2.2.2. *Reconnaissance omnifonte.*

Alors que la méthode précédente nécessite un enregistrement préalable de tous les caractères, définis dans plusieurs polices, cet enregistrement se trouve inutile dans le cadre d'une méthode de type morphologique, fondée sur la reconnaissance de formes. Les méthodes de ce type cherchent à établir des propriétés synthétiques représentatives des formes de caractères ; elles sont donc fondées sur la mise au point d'algorithmes de reconnaissance les plus généraux possible. Ces techniques sont généralement précédées d'un traitement d'image comme la squelettisation qui réduit la forme du caractère à quelques segments symboliques.

Dans l'idéal, il s'agit de reconnaître universellement toutes les polices de caractères, d'où le qualificatif d'«omnifontes » attribué à ces méthodes. Le premier de ces systèmes omnifontes est le Kurzweil Data Entry Machine, ou KDEM.

On distingue les méthodes de type morphologique, ou par caractéristiques locales, et les méthodes de type statistique, ou par caractéristiques globales. Elles sont, les unes et les autres, indépendantes de la taille des caractères. Les premières sont efficaces pour tous les chiffres, qu'ils soient manuscrits, imprimés, issus de machines à écrire ou

d'imprimantes. La reconnaissance statistique est plutôt appliquée à certaines fontes spécifiques .

2.2.3. *Apport des réseaux neuronaux.*

Les réseaux de neurones, bien adaptés aux problèmes de classification et d'apprentissage, conviennent assez bien à la reconnaissance de caractères omnifontes et manuscrits. L'application de ces systèmes est envisagée pour le tri automatique de codes postaux manuscrits, les taux de reconnaissance sont comparables, voire supérieurs à ceux des méthodes habituelles de lecture optique.

2.3. Caractéristiques de systèmes d'OCR.

Beaucoup de logiciels de lecture optique tournent actuellement sur micro-ordinateur (PC, PS, ou Macintosh), éventuellement muni d'une carte spécialisée. La plupart des systèmes actuels sont omnifontes.

Ils se distinguent par:

- La nécessité ou la possibilité de l'apprentissage,
- L'existence et l'importance du dictionnaire intégré (utile pour l'auto-apprentissage),
- La vitesse de lecture, de quelques caractères par seconde à plusieurs centaines,
- La conservation de la mise en page,
- La prévisualisation du document,
- La possibilité de conserver l'enrichissement typographique du texte,
- L'interfaçage avec d'autres logiciels (traitements de texte, tableurs, bases de données, etc.),

2.3.1. *Contraintes sur le papier et l'encre.*

Le lecteur optique détecte le caractère par différence de contraste avec le papier, Pour obtenir le meilleur contraste, le papier et l'encre du document doivent respecter certaines conditions. En ce qui concerne le papier:

- Il doit réfléchir le maximum de quantité de lumière (facteur de réflectance);
- L'opacité doit être forte et régulière;
- La brillance doit être faible;
- Le papier doit être homogène et exempt d'impuretés (moins de 10^{-5} par mètre carré), en particulier ne comporter ni filigrane ni motif coloré;

Pour permettre un entraînement régulier, éviter les bourrages ou les cas de double alimentation (deux documents traités ensemble), le papier ne doit être ni trop lisse ni trop rugueux, respecter des dimensions précises ; il doit encore posséder d'autres qualités physiques comme la résistance à la déchirure, la rigidité, la porosité à l'air, etc.

En ce qui concerne l'encre :

- Pour les caractères à lire; elle doit présenter un très bon contraste.
- La densité d'encrage doit être bonne, les rubans encreurs doivent être d'excellente qualité;

Enfin, quelques recommandations générales permettent d'éviter certaines erreurs de lecture:

- Il est préférable d'éviter les caractères à empattement, lesquels sont souvent à l'origine de lettres collées;
- Si l'interligne est assez important (1 ½ par exemple), il n'y a pas risque de passage d'une ligne à l'autre, notamment si la ligne de texte à lire n'est pas tout à fait parallèle au scanner;
- Les marges et espaces intercolonnes doivent être suffisamment larges;
- Il faut éviter les surcharges au crayon sur le document à lire;
- Il faut éviter le recto-verso, notamment dans le cas de papier à grammage faible; la transparence peut perturber la lecture;

Mais l'utilisateur n'est pas toujours maître de la qualité du document original; un essai s'impose, qui indiquera le taux d'erreur et donc la faisabilité.

2.3.2. *Apprentissage.*

Parmi les systèmes omnifontes, on distingue les systèmes sans apprentissage (Kurzweil par exemple), et les systèmes avec apprentissage, qui constituent la majeure partie du marché de l'OCR.

Les premiers n'admettent en général que des documents de très bonne qualité, sans ligatures ou lettres collées. Les systèmes à apprentissage sont plus ouverts et généralement très performants, à condition de pouvoir leur consacrer de 20 à 40 min. pour l'apprentissage. Cela ne se justifie donc que pour des documents assez volumineux, de plusieurs pages, ou fréquents. En revanche, pour lire des document d'origines variées et de petite taille, il vaut mieux choisir un système sans apprentissage.

L'apprentissage consiste pour l'utilisateur à assister le système dé qu'il aborde un nouveau type de document. A chaque signe nouveau rencontré, celui-ci est signalé à l'utilisateur, lequel effectue la reconnaissance à la place de la machine. L'utilisateur constitue ainsi une bibliothèque de polices de caractères, qui peut s'étendre aux alphabet étrangers (russe, grec, caractères spécifiques aux langues scandinaves, espagnol ou tchèque...). L'apprentissage se fait au cours d'une phase où l'utilisateur est sollicité chaque fois que le logiciel rencontre un caractère qu'il n'a pas encore mémorisé : généralement, l'image du caractère inconnu du système s'affiche en mode bitmap (Fig. I-8) sur une fenêtre de l'écran, et l'utilisateur doit frapper au clavier le caractère correspondant pour que celui-ci soit mémorisé par le système. Dans un contexte donné (documents de même type et de qualité typographique constante), il faut généralement compter une ou deux pages de lecture « assistée » par l'utilisateur, avant que la lecture puisse se faire automatiquement, encore que cette automaticité ne puisse jamais être total dans la mesure où il est impossible de garantir que tous les caractères ont été utilisés au début du texte. De plus, même après apprentissage, lorsque le seuil de reconnaissance n'est pas atteint, le système affiche l'image du caractère à lire et, en regard, le caractère supposé reconnu, et l'utilisateur doit valider la reconnaissance. Les systèmes à apprentissage sont, de ce fait, assez contraignants.

Leur avantage est la capacité à gérer un alphabet complet, avec minuscules et majuscules accentués, cédille et autres signes diacritiques. Les limitations observées par certains systèmes, sur micro-ordinateur notamment, tiennent à la taille de la mémoire. Certains systèmes disposent d'une fonction d'apprentissage spéciale qui permet de lever des ambiguïtés, par exemple, distinguer la barre oblique (/) de la lettre ' l ' minuscule italique. Ils peuvent aussi être dotés d'une fonction d'auto-apprentissage, notamment les systèmes à réseaux de neurones.

2.3.3. Dictionnaires et reconnaissance de mots.

La tendance actuelle consiste à doter les logiciels de lecture optique de dictionnaires multilingues qui servent à lever certaines incertitudes. Le caractère peut ainsi être identifié dans le contexte du mot dans lequel il se trouve. Ces dictionnaires permettent, dans certains cas, de se dispenser de la présence d'un opérateur et d'automatiser complètement la lecture. C'est le cas des systèmes ReadStar (Inovatic) et AccuText (Xerox Imaging System). Il serait plus juste de parler dans ce cas de reconnaissance de mots plutôt que de reconnaissance de caractères. Le dictionnaire permet aussi de lever les ambiguïtés entre caractères très semblables (lettre 'O' et chiffre zéro, lettres 'l', 'I' et chiffre '1', lettre 'S' et chiffre '5',..., etc.) ou de pallier une mauvaise qualité d'impression du document (similitudes entre 'e' et 'c', 'i' et 'l', 'D' et 'O', 'b' et '6', 'r n' et 'm',..., etc.).

Le dictionnaire permet aussi de reconstituer les mots césures en fin de ligne : restituer le trait d'union lorsqu'il fait partie d'un mot composé ou le supprimer lorsqu'il est une simple marque de césure. L'usage du dictionnaire permet accessoirement de corriger les fautes d'orthographe.

Ce procédé de lecture globale, permet d'atteindre des vitesses de traitement bien plus élevées que celles des procédés classiques de lecture optique qui, pour la plupart, n'autorisent la visualisation que d'un seul caractère à la fois, sans connaissance de son environnement. Par exemple, il est souvent difficile de trancher entre un '1' et un '7' manuscrits, lorsque ce dernier est écrit sans barre transversale. La visualisation d'un nombre dans son ensemble ('3 417' par exemple) permet de décider qu'il s'agit de '1'

pour l'avant-dernier chiffre et de '7' pour le dernier, quelle que soit la manière dont ils sont tracés. La décision peut donc se faire automatiquement sans risque d'erreur, et sans être obligé de se référer au document original.

Ces systèmes OCR possédant en aval une partie de contrôle orthographique permettant de compléter un mot, lorsqu'un ou plusieurs caractères de ce mot ne sont pas reconnus, ou de déceler des erreurs (reconnaissance erronée) ne fonctionnent donc pas dans toutes les langues :

- Ils peuvent ou non tenir compte d'accents, apostrophes et autres cédilles dans une langue, tildes, trémas, o barrés,..., etc., dans d'autres langues;
- Ils détectent des erreurs liées à une langue particulière : longueur maximale de mots, associations impropres de certaines lettres, irrégularités statistiques, accentuations incorrectes,..., etc.;
- Ils comprennent des dictionnaires dans une langue donnée et ne vérifient donc pas l'orthographe des mots d'une langue étrangère.

2.3.4. Lecteurs de document et lecteurs de pages.

Les systèmes d'OCR appartiennent à deux grandes familles :

Les lecteurs de documents, conçus pour lire des bordereaux ou des imprimés en informatique de gestion (par exemple, la carte de crédit bancaire);

Les lecteurs de pages, capables de reconnaître des textes non structurés, produits par des machines à écrire, ou des textes d'imprimerie produits par composition et impression.

L'appellation «lecteur optique de pages » désigne un ensemble informatique dont l'organe d'entrée privilégié est un dispositif de captation d'images, et qui inclut un système de reconnaissance de texte fondé sur une reconnaissance de caractères. Dans ce cas, le logiciel d'OCR est complété par d'autres sous-programmes qui permettent d'effectuer, par exemple, la localisation de l'information, le calcul automatique du seuil de luminosité, le pré-traitement, la segmentation des caractères, l'analyse de page...

L'analyse de page comprend la détection des différents éléments composant la page considérée : graphiques, tableaux, titres, intertitres, éléments courants, logos,..., etc., et prend en compte la disposition du texte en colonnes. Ces systèmes sont capables de repérer les colonnes et les paragraphes, de décomposer le texte en blocs, de restituer les enrichissements typographiques (caractères gras, italiques, soulignement...) et les différents corps, d'éliminer les graphiques (pour les traiter séparément dans une autre étape), de faire l'analyse contextuelle portant sur un mot et non plus seulement sur les caractères (recours aux dictionnaires, systèmes experts,..., etc.). Certains systèmes ajoutent à ces fonctionnalités la saisie d'images et la lecture sélective d'une partie de document à partir d'une fenêtre définie par l'utilisateur au moment de la saisie (multi-fenêtrage).

La lecture est faite colonne après colonne, en éliminant les autres éléments de la page. La restitution du texte est faite par paragraphes. Le lecteur doit d'abord connaître les limites de son domaine d'exploration (marges horizontales et verticales). Selon les cas, le lecteur requiert la fourniture de ces informations au début du traitement, ou bien les détermine automatiquement au cours de l'exploration.

Le repérage de la ligne courante se fait sur une notion de « ligne de base» droite imaginaire sur laquelle sont alignés tous les caractères. Les systèmes de lecture optique ont une tolérance plus ou moins grande vis-à-vis de l'alignement sur cette ligne.

La ligne étant identifiée, le système optique recherche les caractères un par un en utilisant le principe de la fenêtre pour sélectionner le caractère. Cette sélection peut être délicate, notamment dans le cas de textes imprimés à chasse variable. En cas d'anomalie (caractères éclatés ou confondus), il faut parfois faire appel à des algorithmes de séparation ou de recollement des caractères. Une fois séparés, les caractères sont soumis un à un à une logique de reconnaissance de caractères.

Dans une page imprimée moyenne, le logiciel d'OCR détecte en moyenne 130 modèles, selon la multiplicité des polices qui y sont utilisées.

Tout caractère non reconnu est signalé comme rejet, et le système propose éventuellement à l'utilisateur un caractère comme meilleur choix. La correction des rejets peut se faire soit immédiatement, en cours de lecture, soit de manière différée.

Certains systèmes, comme Omnipage (Caere), possèdent des extensions pour la reconnaissance de page :

- Un dictionnaire modulable, OmniSpell (480 000 mots);
- Un outil graphique, OmniTrace, qui permet la conversion en format PostScript;
- Un utilitaire de comparaison de documents, Omniproof, mettant en évidence les différences sur deux documents d'apparence identique; il souligne et fait ressortir les corrections et modifications qui y ont été apportées.

2.3.5. Sorties.

Les sorties se font sur support physique (bande magnétique, disque optique numérique) ou bien par télétransmission, selon différents types de protocoles synchrones ou asynchrones. Les lecteurs optiques orientés vers la Bureautique offrent des interfaces de format compatible avec les principaux logiciels : ASCII, traitement de texte (Word par exemple), tableurs,..., etc. Le texte capté par le lecteur est directement transmis à l'unité de contrôle de la disquette; il peut ensuite être rappelé comme un fichier informatique ordinaire.

2.4. Applications.

L'OCR a des applications dans différents domaines :

- Bureautique, pour la gestion électronique de documents;
- PAO, afin de reprendre, dans les logiciels de mise en page, des textes saisis à la machine à écrire ou autres;
- Archivage électronique de documents imprimés, alimentation de bases de données,..., etc.

Dans toutes ces applications, l'OCR a de multiples avantages.

Elle délivre l'utilisateur de la tâche ingrate consistant à ressaisir manuellement des documents imprimés lorsque ceux-ci doivent être retraités par ordinateur. L'OCR évite ainsi des erreurs dues à diverses manipulations lorsqu'il y a ressaisie.

C'est un mode de saisie particulièrement rapide : alors que la saisie au clavier d'une page A4 moyennement pleine (2 200 caractères) requiert environ 20 min. de travail (saisie, relecture, correction), la même saisie par OCR se fait en 30s , plus 2 à 4 min. de correction, soit moins de 5 min. (sans compter la phase d'apprentissage, qui peut être relativement longue, mais vite amortie sur des documents volumineux).

La lecture optique offre une plus grande fiabilité que la saisie manuelle puisque, en principe, toutes les incertitudes de lecture sont signalées.

Lors de la lecture optique, un code ASCII est associé à chaque modèle de caractère, ce qui représente un octet en mémoire. En revanche, l'image numérisée d'un caractère occupe plusieurs dizaines d'octet. Le gain de place de stockage en mémoire est donc considérable, lorsqu'on passe d'une image produite par scanner (1 Mo par page A4 numérisée à 300 points par inch) à un fichier texte obtenu par OCR (quelques Kilo-octets par page).

2.4.1. Imprimerie.

Les professions de l'imprimerie ont tout naturellement des volumes importants de textes à traiter. Depuis qu'il existe des logiciels de PAO, les imprimeurs ont pris l'habitude de reprendre des textes provenant de fichiers informatiques obtenus par traitement de texte en particulier.

Lorsqu'ils disposent d'une version imprimée ou dactylographiée, et non d'un enregistrement de ces données, l'OCR permet de ressaisir le texte, qui peut ensuite être soumis à divers traitements de présentation, mise en page,..., etc. Combinée a des logiciels dits « de compression hybride », la lecture optique permet de distinguer, sur un même document, les textes des images, logos et autres schémas, en vue du stockage sur disque optique numérique, de la composition et de la restitution par imprimante à laser.

Dans l'imprimerie, des systèmes d'OCR peuvent être utilisés pour vérifier la qualité et la conformité des chiffres et lettres imprimés sur des étiquettes, billets de loterie, billets de banque, chèques,..., etc. Il est ainsi possible de détecter des caractères absents

ou erronés, taches rendant un caractère illisible, encrage insuffisant, surabondant, ou non homogène....

2.4.2. Bureautique.

La lecture optique constitue l'un des modes d'entrée de données informatiques les plus importants, à l'exception du clavier. Elle est utilisée pour la récupération de fichiers d'adresses, de tableaux de chiffres, l'insertion de citations ou de références dans une publication, l'alimentation de bases de données, la constitution automatique de bases documentaires, l'archivage électronique de documents imprimés, l'association à des systèmes de traduction automatique ou de conversion en Braille,..., etc.

L'OCR permet de contourner les problèmes de compatibilité entre systèmes informatiques hétérogènes ou d'extension du système au cours du temps. Elle peut également remplacer le clavier d'ordinateur dans le cas de matériels miniaturisés. Dans ce cas, les caractères sont entrés à la volée, via une petite tablette à numériser, sur laquelle l'utilisateur écrit manuellement.

2.4.3. Administration.

Les débouchés de la reconnaissance optique de caractères dans l'administration sont nombreux, notamment dans les cas de saisie numérique : tri postal, bordereaux de Sécurité Sociale, formulaires d'assurance, chèques bancaires, déclarations d'impôts, étiquettes et autres QCM (questionnaires au choix multiples)....

L'une des premières applications a vu le jour dans l'administration des PTT dans différents domaines d'activité:

- Saisie des adresses postales (80 % du courrier est tapé à la machine, et donc susceptible d'être traité automatiquement);
- Lecture de pages de données administratives.

Pour la lecture automatique des adresses postales, la Poste a fait développer le système Triex (Cap Sesa), capable de reconnaître les adresses postales écrites en caractères imprimés, quelle que soit la police de caractères utilisée. Le résultat est exploité par un système à base de connaissances, capable d'interpréter le contenu des

adresses malgré les variantes de formulation et en dépit des informations manquantes ou erronées. Triex permet ainsi de passer du tri manuel au tri automatique, sans introduire d'erreurs d'indexation.

La reconnaissance de signatures peut être appliquée à l'amélioration de la sécurité dans différents domaines (paiement électronique, accès à des fichiers informatiques, accès à des locaux protégés,..., etc.). Au Service d'Étude des Postes et Télécommunications (SEPT) ont été développés deux types différents de reconnaissance de signature:

- La reconnaissance dynamique de signature (RDS) pour le contrôle accès;
- La reconnaissance statique de signature (RSS) qui permet de vérifier a posteriori des signatures apposées sur des documents.

2.4.4. Industrie.

Les applications de la lecture optique de caractères en milieu industriel sont variées : identification automatique des produits industriels par lecture d'un numéro de référence, contrôle de qualité ou de conformité, suivi de production. Gestion de stock, tri de colis, adressage, routage...etc. La première société européenne de reconnaissance de caractères en milieu industriel, tant en ce qui concerne l'effort de recherche et de développement que la base installée, est la firme française Al Vision Systèmes. Celle-ci a mis en place plus de cent systèmes d'OCR (Stirca: Système de traitement d'images et de reconnaissance de caractères) chez une trentaine d'industriels, essentiellement dans le domaine de la construction automobile (General Motors, Volkswagen, BMW), mais aussi en électronique, électricité, métallurgie, nucléaire,... etc.

Dans l'industrie automobile, il s'agit, par exemple, de lire des étiquettes d'identification de voitures, de contrôler la qualité des références de bougies. En électronique, la lecture optique permet de lire des chiffres sur des plaques de circuits intégrés ou de mettre au point des moniteurs vidéo. Dans le secteur nucléaire, on l'utilise pour identifier des tubes de plutonium...etc.

L'identification automatique des produits industriels s'effectue par la lecture de leurs numéros à distance, parfois éloignée, ce que ne permettent pas les codes à barres, qui n'offrent d'ailleurs pas la même fiabilité en durée de vie du matériel ni en performance de reconnaissance dans un milieu industriel.

2.4.5. Commerce et emballage.

Dans le commerce comme dans l'industrie, la lecture optique de caractères constitue une autre possibilité intéressante par rapport aux lecteurs de codes à barres. L'OCR permet de lire des étiquettes, vérifier la conformité et la lisibilité des codes produits, numéros de lots, date de préemption, etc., pour éviter notamment toute erreur d'étiquetage sur un produit pharmaceutique ou alimentaire.

Dans le secteur de l'emballage, il s'agit de lire des chiffres inscrits sur des colis afin de les identifier, d'assurer le contrôle et le suivi des emballages des produits sous toutes leurs formes : étiquettes, documents joints, caisses, palettes,...,etc. Cette application intéresse les producteurs, transporteurs, distributeurs, professionnels de la vente par correspondance,..., etc. Dans ce secteur, le système de lecture optique est généralement associé à un système d'impression et de marquage .

CHAPITRE II

Réseaux de neurones

1. Fondement biologique. [PAN 1987] , [DAV 1990]

1.1. Le cerveau.

Ce n'est qu'au XVIII[e] siècle que la théorie sur le rôle du cerveau comme organe de commande centrale de l'organisme a été reconnue en Europe, affirmée par La Mettrie et Cabanis ' le cerveau secrète la pensée comme le foie la bile'. Après des siècles d'occultation durant la période médiévale. Démocrite, puis Platon, furent les premiers à l'expliciter. Il aura cependant fallu attendre Hippocrate pour que commencent les premières observations cliniques, puis Hérophile, trois siècles avant Jésus-Christ, pour Les premières dissections. Les bases de la Physiologie cérébrale remontent réellement à Galien (deux siècles après Jésus-Christ), qui démontre à l'aide d'expérimentations sur des animaux que le cerveau est bien l'organe de commande centrale du corps.

A partir du XIX[e] siècle se poursuivront les recherches qui aboutiront aux théories actuelles.

1.1.1. Les méthodes d'études.

On peut faire trois remarques sur les différentes méthodes utilisées pour étudier le cerveau.

- La méthode, qui a donné naissance à la neuropsychologie, est basée sur l'étude des relations entre faits d'anatomie et faits du comportement. C'est au XIX[e] siècle qu'elle prend toute son importance au travers des expériences de Broca, qui grâce à ses expériences, montrera que les fonctions cérébrales (motrices, sensibles..) sont localisées d'une façon précise dans le cerveau.
- Cette localisation fournit aux méthodes analytiques un champ d'étude idéal. La méthode analytique a cependant porté ses fruits en particulier dans l'étude des systèmes visuels. Elle reste aujourd'hui la base de toutes les études scientifiques.
- Enfin, l'évolution la plus récente porte sur la recherche des bases physico-chimiques des fonctions cérébrales. Le niveau d'explication du fonctionnement du cerveau descend de plus en plus vers le moléculaire. L'étude du cerveau est donc passée

d'une caractérisation des lieux, des fonctions cérébrales par ablation, à une étude des relations entre faits du comportement et faits de nature électrique, puis chimique.

1.1.2. L'évolution du cerveau.

La réussite des travaux de Broca, est à l'origine de nos cartes du cerveau utilisées pour décrire son évolution chez les différentes espèces animales.

Le tout premier cerveau apparut sur la terre chez un poisson. Ce cerveau représente un stade primitif de l'évolution chez les vertèbres. Il comprenait trois compartiments :

* Une partie antérieure consacrée à l'odorat,
* Une partie médiane consacrée à la vision,
* Une partie postérieure consacrée à l'équilibre.

Ce cerveau était bien incapable de coordination entre ses différentes parties. Chacune correspondait à un type de comportement parfaitement déterminé en réponse à certains stimulus.

L'évolution des espèces à partir de ce poisson passe par de nombreuses étapes, avant d'aboutir à l'homo sapiens. Il est remarquable de noter que cette évolution a toujours été dans le sens d'un accroissement du rapport entre le poids du cerveau et le poids du corps total, pour les représentants de chaque espèce.

1.2. Les éléments de base.

1.2.1. Le neurone.

Les cellules nerveuses, appelées neurones, sont les éléments de base du système nerveux central. Celui-ci en posséderait environ cent milliards. Les neurones possèdent de nombreux points communs dans leur organisation générale et leur système biochimique avec les autres cellules. Ils présentent cependant des caractéristiques qui leur sont propres et se retrouvent au niveau des cinq fonctions spécialisées qu'ils assurent:

a) Recevoir des signaux en provenance de neurones voisins,

b) Intégrer ces signaux,

c) Engendrer un influx nerveux,

d) Le conduire,

e) Le transmettre à un autre neurone capable de le recevoir.

1.2.2. Structure des neurones.

Un neurone est constitué de trois parties :

- Le corps cellulaire,
- Les dendrites,
- L'axone.

Le corps cellulaire.

Il 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. Sa forme est pyramidale ou sphérique dans la plupart des cas. Elle dépend souvent de sa position dans le cerveau, ainsi les neurones du néo-cortex ont principalement une forme pyramidale. Ce corps cellulaire fait quelques microns de diamètre.

Les dendrites.

Chaque neurone possède une 'chevelure' de dendrites. Celles-ci sont de fines extensions tubulaires, de quelques dixièmes de microns de diamètre et d'une longueur de quelques dizaines de microns. Elles se ramifient, ce qui les amène à former une espèce d'arborescence autour du corps cellulaire. Elles sont les récepteurs principaux du neurone pour capter les signaux qui lui parviennent.

L'axone.

L'axone, qui est à proprement parler la fibre nerveuse, sert de moyen de transport pour les signaux émis par le neurone. Il se distingue des dendrites par sa forme et par les propriétés de sa membrane externe. En effet, il est généralement plus long (sa

longueur varie d'un millimètre à plus d'un mètre) que les dendrites, et se ramifie à son extrémité, là où il communique avec d'autres neurones, alors que les ramifications des dendrites se produisent plutôt près du corps cellulaire.

Pour former le système nerveux, les neurones sont connectés les uns aux autres suivant des répartitions spatiales complexes. Les connexions entre deux neurones se font en des endroits appelés synapses où ils sont séparés par un petit espace synaptique de l'ordre d'un centième de microns.

1.2.3. Le fonctionnement des neurones.

Les fonctions spécifiques réalisées par un neurone dépendent essentiellement des propriétés de sa membrane externe.

La membrane extérieure

La membrane externe d'un neurone remplit cinq fonctions principales :

- Elle sert à propager des impulsions électriques tout au long de l'axone et des dendrites,
- Elle libère des médiateurs à l'extrémité de l'axone,
- Elle réagit à ces médiateurs au niveau des dendrites,
- Elle réagit au niveau du corps cellulaire aux impulsions électriques que lui transmettent les dendrites pour générer ou non une nouvelle impulsion,
- Enfin, elle permet au neurone de reconnaître les autres neurones afin qu'il puisse se situer au cours de la formation du cerveau, et trouver les cellules auxquelles il doit être connecté.

1.2.4. Le corps cellulaire.

D'une façon simple, on peut dire que le soma du neurone traite les courants électriques qui lui proviennent de ses dendrites, et qu'il transmet le courant électrique résultant de ce traitement aux neurones auxquels il est connecte par l'intermédiaire de son axone.

Le schéma classique présenté par les biologistes est celui d'un soma effectuant une sommation des influx nerveux transmis par ses dendrites. Si la sommation dépasse un seuil, le neurone répond par un influx nerveux ou potentiel d'action qui se propage le long de son axone. Si la sommation est inférieure à ce seuil, le neurone reste inactif.

L'influx nerveux, qui se propage entre différents neurones, est au niveau de ces neurones, un phénomène électrique.

1.2.5. Les synapses.

Le rôle des synapses est fondamental pour permettre aux cellules nerveuses de communiquer entre elles.

Les signaux qui se propagent dans les neurones sont de nature électrique. Cependant, Il n'existe pas de liaisons directes entre deux cellules nerveuses. Celles-ci sont séparées par un espace appelé fente synaptique que l'influx électrique ne peut traverser. Le relais s'effectue à ce niveau par l'intermédiaire d'un médiateur chimique.

Les synapses se rencontrent plus particulièrement entre les axones et les dendrites.

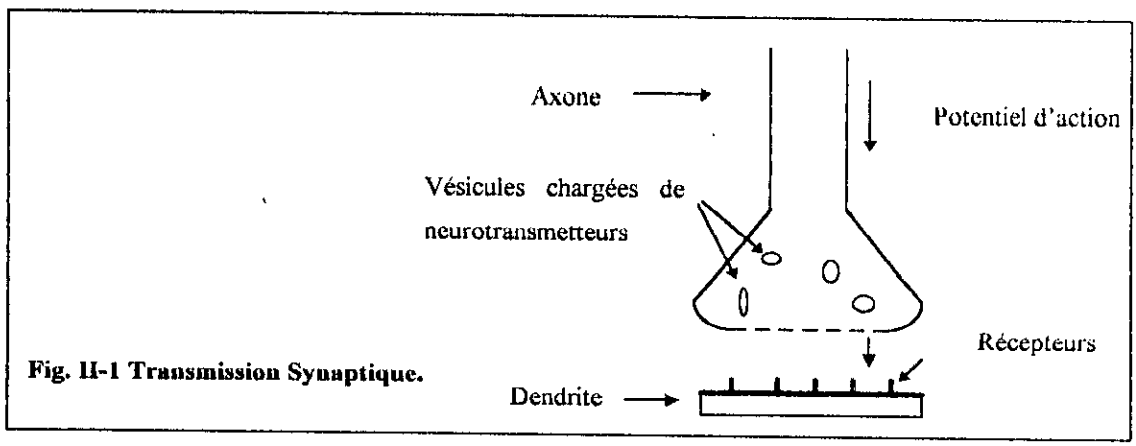


Fig. II-1 Transmission Synaptique.

Le fonctionnement général d'une synapse est le suivant (voir Fig. II-1) :

L'arrivée du potentiel d'action à l'une des extrémités de l'arborisation terminale, appelée 'bouton terminal', déclenche la libération d'une substance chimique appelée neurotransmetteur. Cette substance diffuse dans l'espace synaptique et vient se fixer sur des récepteurs spécifiques appelés neuro-recepteurs, situés sur la terminaison du

neurone cible. Cette fixation provoque l'ouverture de canaux ioniques, ce qui peut donner naissance a un nouveau signal électrique.

Avant l'arrivée d'une seconde impulsion, la fente synaptique est nettoyée soit par recapture du médiateur par le premier neurone, soit par destruction du médiateur par des enzymes.

2. Modélisation. [WEI 1993] , [DAV 1990]

2.1. Modélisation générale.

D'une façon générale, on peut définir un neurone formel par les cinq éléments suivants (voir Fig. II-2) :

◊ La nature de ses entrées;

◊ La fonction d'entrée totale qui définit le prétraitement effectué sur les entrées;

◊ La fonction d'activation (ou d'état) 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 état d'activation;

◊ La nature de la sortie du neurone.

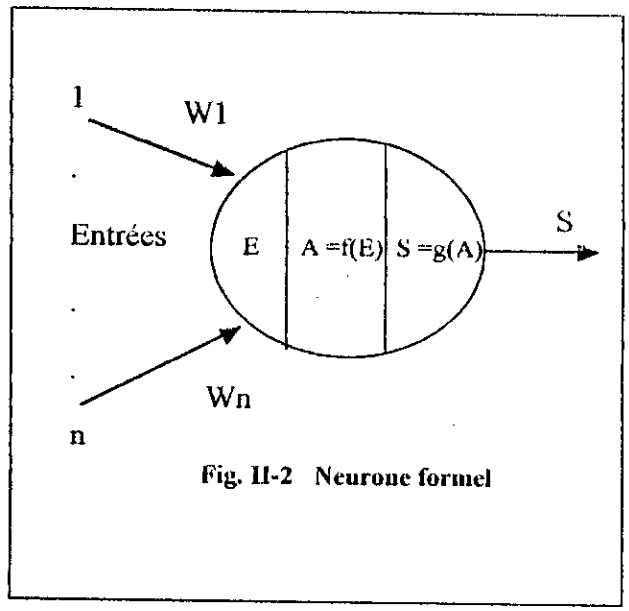


Fig. II-2 Neurone formel

Nous adopterons par la suite les notations suivantes :

- (e_i) i=1...n seront les entrées;
- *h* sera la fonction d'entrée totale;
- *f* sera la fonction d'activation;
- *g* la fonction de sortie.

D'autre part, nous utiliserons aussi :

- $E = h(e_1, \ldots, e_n)$ comme entrée totale;

- S = g (A) comme sortie.

2.1.1. La nature des entrées et de la sortie.

Elles peuvent être :

- Binaires (-1,+1) ou (0,1).
- Réelles.

2.1.2. La fonction d'entrée totale.

h peut être:

- Booléenne;
- Linéaire: $h(e_1,\ldots,e_n) = \sum_{i=1\ldots n} W_i e_i$;
- Affine : $h(e_1,\ldots,e_n) = \sum_{i=1\ldots n} W_i e_i - W_a a$;

Ces deux derniers cas sont les plus fréquents. Dans le deuxième on peut interpréter *a* comme l'utilisation d'un neurone qui fournirait toujours l'entrée 1 au neurone étudié.

- Polynomiale de degré supérieur à deux.

2.1.3. La fonction d'activation.

f peut être :

- Une fonction binaire à seuil tel que la fonction de Heaviside ou la fonction signe ;
- Une fonction linéaire à seuil ou multi-seuils;
- Une fonction hyperbolique : $f(x) = [\ 1 + \exp(-\alpha\ x\ )\ ]^{-1}$;
- Une fonction hyperbolique : $f(x) = th(\alpha\ x)$;

2.1.4. La fonction de sortie.

En général, cette fonction *g* est considérée comme la fonction identité :

$$S = f(E) = A$$

2.2. Le modèle de MacCulloch et Pitts.

La première modélisation d'un neurone date de 1943. Elle a été présentée par MacCulloch et Pitts.

S'inspirant de leurs travaux sur les neurones biologiques, ils ont proposé le modèle suivant:

« Un neurone formel fait une somme pondérée des potentiels d'actions qui lui parviennent (chacun de ces potentiels est une valeur numérique qui représente l'état du neurone qui l'a émis), puis s'active suivant la valeur de cette sommation pondérée. Si cette somme dépasse un certain seuil, le neurone est activé et transmet une réponse (sous forme de potentiel d'action) dont la valeur est celle de son activation. Si le neurone n'est pas activé, il ne transmet rien. »

Ce neurone formel est un automate booléen, c'est-à-dire que ses entrées et sa sortie sont booléennes.

2.3. La structure des connexions.

Les structures qui peuvent être utilisées sont très variées. Si l'on se réfère aux études biologiques du cerveau, on constate que le nombre de connexions est énorme. Par exemple, des chercheurs ont montre que le cortex était divisé en différentes couches. A l'intérieur d'une même couche les interactions entre neurones sont très grandes, mais les neurones d'une couche sont aussi reliés aux neurones d'autres couches, le tout formant un système d'une complexité gigantesque.

D'une manière générale, l'architecture des réseaux de neurones formels peut aller d'une connectivité totale (tous les neurones sont reliés les uns aux autres), à une connectivité locale où tes neurones ne sont reliés qu'a leurs plus proches voisins. Il est courant d'utiliser des réseaux à structure régulière pour faciliter leurs utilisations.

3. Taxonomie et terminologie des réseaux usuels.

[WEI 1993] , [FRE 1992]

3.1. Réseaux statique et dynamique.

On peut classer les réseaux en deux grandes catégories, selon la dépendance de l'évolution de ceux-ci en fonction explicite du temps.

Dans le cas des réseaux statiques, le temps n'est pas un paramètre significatif. En d'autres termes, la modification de l'entrée n'entraînera qu'une modification stable de la sortie, mais n'entraîne pas de retour d'information vers cette entrée.

Les **réseaux dynamiques**, comme leur nom l'indique, contiennent des rebouclage partiels ou totaux entre neurones, et ont donc une évolution dépendante du temps. Il faut bien distinguer la dépendance théorique, pour laquelle l'état du réseau à un certain instant dépend de son état à l'instant ou aux instants précédents, et du temps nécessaire à obtenir une réponse, dans le cas d'une réalisation matérielle ou d'une simulation sur ordinateur, que le réseau soit statique ou dynamique. On peut ajouter que, formellement, le fonctionnement d'un réseau quelconque dépend de son histoire passée, par le biais de l'apprentissage. Mais cette dépendance est simplement causale, et non pas fonctionnelle, et n'a pas à être prise en compte dans l'étude détaillée de leurs mécanismes. Le Perceptron multicouche ordinaire ou la carte auto-organisatrice sont des réseaux statiques. Par contre, le réseau de Hopfield ou le Perceptron avec rebouclage sont des réseaux dynamiques.

3.2. Apprentissages supervisés, non supervisés.

Les procédures d'apprentissage peuvent se subdiviser, elles aussi, en deux grandes catégories : apprentissage supervisé ou apprentissage non supervisé.

- L'**apprentissage supervisé** implique l'existence d'un « professeur » qui a pour rôle d'évaluer le succès (ou l'échec) du réseau quand il lui est présenté un stimulus connu (on dit que le stimulus est un exemple appartenant à la base d'apprentissage). Cette supervision consiste à renvoyer au réseau une information lui permettant de faire évoluer ses connexions (parfois aussi sa propre architecture) afin de faire diminuer

son taux d'échec. L'information peut être explicite, sous la forme d'une mesure de l'erreur commise, exemple par exemple, ou globalement sur l'ensemble des exemples de la base. C'est ce qui se passe par exemple dans le cas du Perceptron. Mais elle peut être implicite (apprentissage par renforcement), sous forme d'une simple appréciation (« bon » ou « mauvais », punition ou récompense), sans mesure d'erreur, et même être globale, sur l'ensemble des tâches que le réseau doit exécuter. Dans ce dernier cas, il est facile de comprendre que l'apprentissage est plus difficile, la difficulté majeure consistant pour le réseau à identifier les étapes du processus qui sont responsables de l'échec ou du succès (*credit assignement problem*).

- L'**apprentissage non supervisé** implique la fourniture à un réseau autonome d'une quantité suffisante d'exemples contenant des corrélations (c'est-à-dire de la redondance), telles que celui-ci en dégage les régularités automatiquement. Ces réseaux sont souvent appelés « auto-organisateurs », ou encore « à apprentissage compétitif ». Bien entendu, l'architecture du réseau, préalablement définie par son utilisateur, est une forme de supervision définie.

4. Réseaux classifieurs statique supervisés à propagation.

[DAV 1990] , [FRE 1992] , [WEI 1993]

4.1. Le Perceptron.

F. Rosenblatt a publié en 1958 un article sur un modèle de stockage de l'information et de l'organisation dans le cerveau, intitulé « Perceptron ». En simplifiant il a tenté de montrer qu'une machine simulant la vision (une « rétine », à la réponse binaire, connectée à une couche constituée de neurones formels, elle-même connectée à une couche analogue dite « d'association ») était capable de distinguer entre plusieurs stimulus, et même capable de grouper des stimulus ayant entre eux des relations de proximité, en les séparant d'autres groupes de stimulus. En ce sens, il rejoignait les travaux sur la classification de données, mais dans le cadre du connexionnisme et de l'auto-organisatio. L'intérêt de ce concept tient dans ce que les connexions peuvent être ajustées par un processus itératif dans lequel on cherche à minimiser l'erreur de

classification en faisant évoluer progressivement leurs valeurs (appelées aussi « poids synaptiques » ou simplement « poids »).

4.2. Apprentissage du Perceptron.

Supposons un réseau monocouche à un neurone classifieur, connecté à une couche perceptive de N cellules par des poids W_i. L'activation V du neurone s'écrit :

$$V = \sum_{i=0}^{N} W_i x_i + W_0 = \sum_{i=1}^{N} W_i x_i \qquad (x_0 = 1)$$

Où les x_i représentent les composantes du vecteur stimulus.

La sortie y de neurone vaut y = f (V), étant la fonction de transfert. Dans le cas le plus simple, c'est la fonction signe, qui peut prendre une de deux valeurs +1 ou -1 (elle n'est donc pas dérivable). Le terme constant dans l'expression précédente peut être entré sous le signe somme, il correspond à la contribution d'une entrée fictive x_0 de valeur constante +1, connectée par un poids W_0 qui est précisément ce terme constant. Par la suite, pour éviter toute lourdeur d'écriture, nous supposerons que cette contribution existe toujours, mais sans le mentionner explicitement.

Nous souhaitons apprendre au neurone à séparer deux classes, pour lesquelles on possède des exemples. On initialise les poids W_i à des valeurs aléatoires, puis on présente successivement les exemples. Si la réponse est correcte, on ne change pas la valeur des poids. Sinon, on applique une correction qui correspond à la règle de Hebb :

$$\Delta W_i^{(t+1)} = \eta \; x_i \; W_i^{(t)}$$

Où on note par t et t+1 deux étapes successives de l'apprentissage le facteur η étant appelé coefficient d'apprentissage.

La Fig. II-3 illustre cette procédure dans le cas facile à représenter d'un espace d'entrée à deux dimensions : elle revient à déplacer la séparatrice dans l'espace des

exemples jusqu'à ce qu'elle joue effectivement son rôle, c'est-à-dire que tous les exemples soient bien classés. En d'autres termes, cette règle d'apprentissage *dénombre* les exemples mal classés, et s'efforce d'en faire tomber le nombre à zéro. Une fois la position définitive de la séparatrice trouvée (cette position n'est pas nécessairement unique), on classera un nouvel exemple (un stimulus inconnu) selon sa position par rapport à celle-ci.

On peut montrer que s'il existe une solution au problème, la règle permet de la trouver en un nombre fini de pas. Mais cette condition d'existence, que l'on appelle la séparation linéaire pose d'importants problèmes.

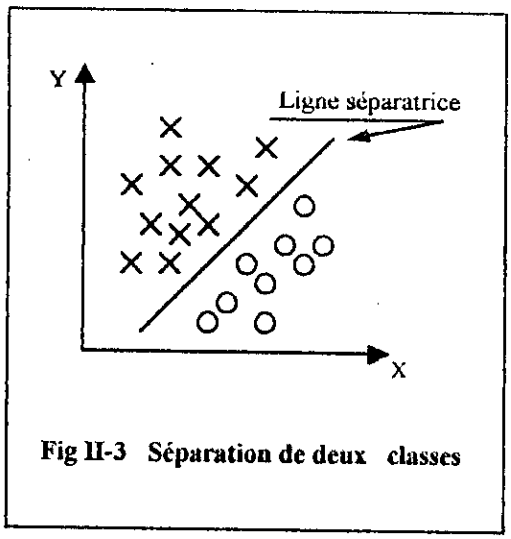


Fig II-3 Séparation de deux classes

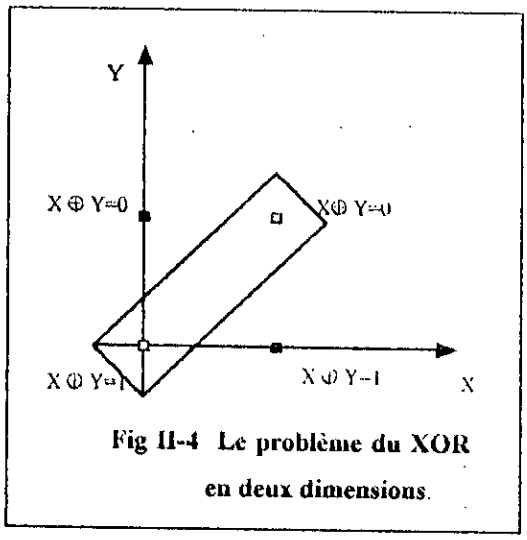


Fig II-4 Le problème du XOR en deux dimensions.

4.3. Séparabilité linéaire.

On peut remarquer que le Perceptron ne peut pas fonctionner correctement dans le cas, où les points représentatifs des deux classes sont groupés dans des nuages qui s'interpénètrent, même légèrement. Dans ce cas, aucune droite séparatrice (aucun hyperplan) ne permettra de bien classer tous les exemples, c'est-à-dire de satisfaire la règle d'apprentissage, ce qui empêchera les coefficients de jamais se stabiliser.

Le problème ainsi représenté est dit **non linéairement séparable**. Or la grande majorité des problèmes de classification des espaces d'entrée de forte dimensionnalité (qui sont les problèmes intéressants du point de vue pratique), et pour lesquels on

possède un nombre significatif d'exemples, son non linéairement séparable, ce qui ruine apparemment la méthode que nous venons d'exposer sommairement. On verra que ce problème, peut être résolu par des structures multicouches ; Notons au passage que la forte dimensionnalité n'est pas une condition de la non-séparabititée linéaire : la fonction booléenne OU exclusif, à deux entrées, n'est pas linéairement séparable (Fig. II-4).

L'incapacité démontrée du Perceptron à résoudre des problèmes intéressants entraîna, en son temps, un ralentissement notable de la recherche sur le connexionnisme. Mais certains travaux démontrèrent, dans le courant des années 70, que l'on pouvait quand même dépasser ces limitations en intercalant, entre la couche de cellules d'entrées et la couche de neurones de sorties, une ou plusieurs couches intermédiaires (appelées aussi couches cachées), l'information circulant d'une couche à la suivante. Dans ce cas on est certain de pouvoir approximer à la sortie n'importe quelle fonction dans l'espace de d'entrée.

4.4. Perceptron multicouche.

L'algorithme du Perceptron monocouche offre une garantie de convergence dès que le problème considéré à une solution. Malheureusement, on l'a vu, la classes des problèmes résolus est celle des problèmes lineairement séparables.

Pour illustrer ce dernier point, il est intéressant de considérer de nouveau l'exemple du XOR. On connaît la représentation graphique de ce problème en deux dimensions, et on sait qu'il n'admet pas de séparation linéaire.

Toutefois, si nous considérons le problème suivant:

0 0 0 → 0

0 1 0 → 1

1 0 0 → 1

1 1 1 → 0

Qui est une extension en trois dimensions du problème du XOR (la troisième dimension étant obtenue comme le ET des deux premières), on constate qu'il admet une séparation linéaire, comme le montre la Fig. II-5.

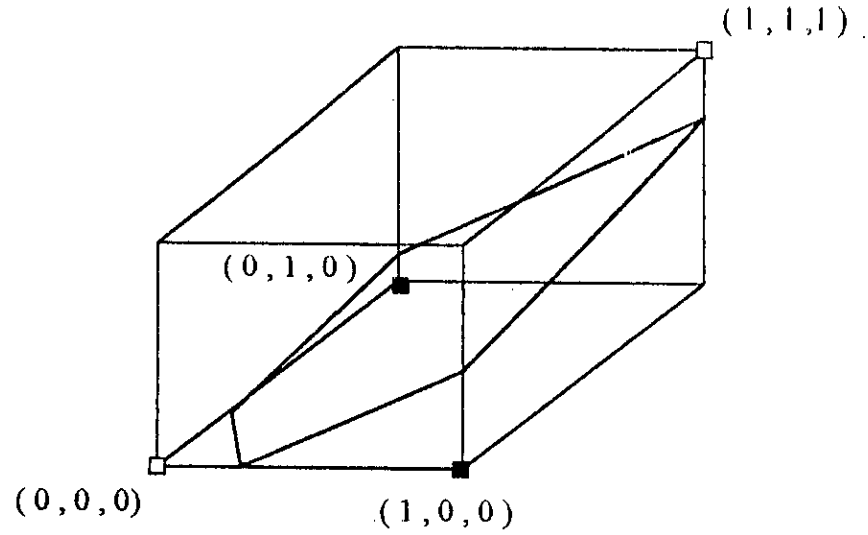


Fig. II-5 La solution en 3 dimensions du problème du XOR

On en déduit que le XOR est réalisable avec un réseau de neurones linéaires à seuil à 3 couches, dont une configuration pourrait être :

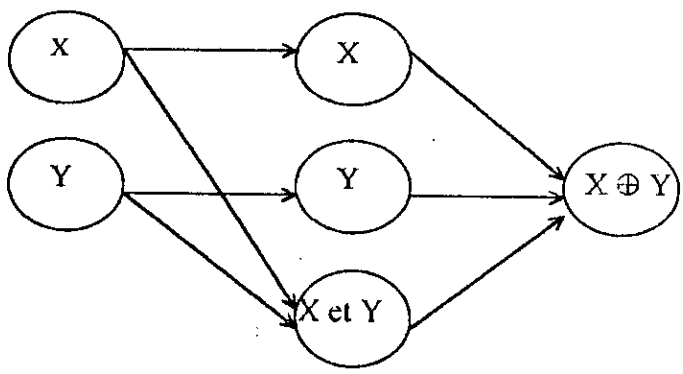


Fig. II-6 Configuration d'un réseau à 3 couches pour la résolution du problème du XOR.

Ces remarques montrent l'intérêt qu'il y aurait à généraliser l'algorithme du Perceptron à un réseau à plus d'une couche ; on serait alors en mesure de résoudre tous les problèmes de séparation, linaire ou pas (**il faut toutefois remarquer que le problème de dimensionnement du réseau n'en serait pas résolu pour autant**).

Le principe des méthodes d'adaptation de poids est basé sur un apprentissage supervisé : en d'autres termes, on connaît à chaque fois la sortie désirée. Un « professeur » qui connaît parfaitement la classification attendue guide le réseau en lui rappelant à chaque étape le bon résultat.

Mais si l'on introduit une couche supplémentaire de neurones entre la couche d'entrée et la couche de sortie, comment évaluer l'impact de la modification d'un poids de la première couche sur la réponse finale ?

Ou encore que doit indiquer le « professeur » comme sortie attendue aux neurones de la couche intermédiaire?

4.5. Le « credit Assignment problem ».

Ce problème est un cas particulier du « Credit Assigement problem » qui est général à tous les systèmes à apprentissage, qu'ils soient naturels ou artificiels.

Dans le cas réseaux de neurones à plusieurs couches de connexions modifiables, ce problème s'exprime de la façon suivante ; comment répercuter sur chacune des connexions le signal d'erreur qui n'a été mesurée que sur la couche de sortie, après avoir traversé plusieurs étapes non linéaires?

Ce problème a été rapidement identifié et a donné lieu à de nombreuses solutions partielles, dont aucune n'était vraiment satisfaisante jusqu'à la mise au point dans les années 1980 de l'algorithme de rétropropagation du gradient, qui sera traité dans le prochain chapitre.

CHAPITRE III

L'algorithme de rétropropagation du gradient

1. Présentation.

Cet algorithme que l'on désigne couramment par « back-propagation » a été mis au point simultanément par deux équipes indépendantes en France (Fogelman-Soulié, Gallinari, Le Cun) et aux États-Unis (Rumelhart, Hinton, Williams).

L'idée simple qui est à la base de cet algorithme, et qui permet de lever la difficulté du « Credit Assignement Problem » est l'utilisation d'une fonction dérivable (fonction sigmoïde) en remplacement de la fonction de seuil utilisée dans les neurones linéaire à seuil.

Dans cet algorithme, de même que l'on est capable de propager un signal provenant des cellules d'entrées vers la couche de sortie, on peut suivant le chemin inverse, rétropropager l'erreur commise en sortie vers les couches internes.

1.1. Le modèle du neurone.

Le neurone utilisé est fondamentalement de même nature que le neurone linéaire à seuil du Perceptron, il applique une fonction à la somme pondérée de ses entrées.

La question de la fonction de transfert mérite un peu d'attention. On peut facilement imaginer que les corrections de poids doivent être d'autant plus fortes, que l'activation du neurone est plus proche de zéro, ce qui signifie que l'exemple traité se trouve très près de l'hyperplan séparateur. Par contre, s'il est suffisamment éloigné de cet hyperplan, on peut accepter une correction faible, voire nulle. Une fonction de transfert qui répond bien à ces critères est la **tangente hyperbolique,** dont la dérivée est maximale pour un argument nul, et tend vers zéro pour un argument grand en valeur absolue (nous verrons plus loin, qu'effectivement, la correction des poids est proportionnelle à la dérivée de la fonction d'activation). De plus, sa dérivée est facile à calculer, ce qui n'est pas sans importance pour la mise en œuvre des algorithmes d'apprentissage sur ordinateurs. Si l'on pose : $f(v) = \text{th}(\alpha v)$

Où α mesure la dérivée de la fonction à l'origine, on a : $f'(v) = \alpha(1-f^2)$.

Dans ce cas, la sortie du neurone varie continûment entre -1 et +1. On utilise aussi, couramment la **fonction sigmoïde**, qui est une homothétie de la tangente hyperbolique, et varie entre 0 et +1 :

$$f(v) = (1 + e^{-v})^{-1} \quad \text{et} \quad f'(v) = \alpha f (1-f).$$

On remarque que, dans les deux cas, le coefficient α se trouve en facteur et que, par conséquent, c'est, en première approximation, le produit $\alpha\eta$ qui apparaît en facteur dans l'expression de la correction de poids.

1.2. Le modèle du réseau.

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, qu'on peut interpréter comme contentant des représentations internes du problème.

Chaque neurone est connecté à l'ensemble des neurones de la couche suivante, par des connexions dont les poids sont des nombres réels quelconques.

1.3. L'apprentissage.

On dispose d'un ensemble d 'exemples qui sont des couples (entrées, sorties désirées). A chaque étape, un exemple est présenté en entrée du réseau. Une sortie réelle est 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, ou encore relaxation du réseau.

Ensuite l'erreur (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 exemple. 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, considérée comme une fonction des poids par une approximation d'une descente du gradient.

Toute la difficulté pour effectuer cette décente dans un réseau multicouche était de pouvoir calculer la dérivée de l'erreur quadratique par rapport à un poids donné. L'utilisation de neurones à fonction d'activation dérivable permet de résoudre ce problème simplement.

2. Formalisation.

Dans ce paragraphe, nous allons présenter une description du formalisme mathématique, du réseau à rétropropagation. La Figure III-1 nous servira, de référence à cette description tout au long du paragraphe.

Pour un exemple à apprendre donné, on note x_i le vecteur des entrées et y_i les vecteurs des sorties. Si le réseau comporte **N** neurones en entrée et **M** en sortie, on a :

$$x_i = (x_{i1}, x_{i2}, \ldots, x_{iN}) \quad \text{et} \quad y_i = (y_{i1}, y_{i2}, \ldots, y_{iM})$$

L'ensemble des **P** paires de vecteurs d'apprentissage sera, (x_1, y_1), (x_2, y_2),..., (x_P, y_P). Nous supposerons que les vecteurs d'apprentissages sont **bien choisis** et que **leur nombre est suffisant**.(Nous verrons plus en détail ces deux notions dans le paragraphe « *considérations pratiques* »).

Présentant le vecteur $x_p = (x_{p1}, x_{p2}, \ldots, x_{pN})$, à la couche d'entrée, les neurones de cette couche, distribuerons ces valeurs à la couche cachée, l'entrée totale du $j^{ième}$ neurone caché est :

$$E^h_{pj} = \sum_{i=1}^{N} \omega^h_{ji} x_{pi} + \theta^h_j \qquad \text{Eq III.1}$$

Où ω^h_{ji} est le poids de la connexion, qui part du $i^{ème}$ neurone d'entrée.

θ^h_j Est le biais (les neurones biais sont optionnels, mais leur utilisation ajoute une dimension supplémentaire à l'espace de codage du réseau).

Dans l'équation Eq. III.1, la lettre 'h' en exposant est utilisée, pour référer à la couche cachée.

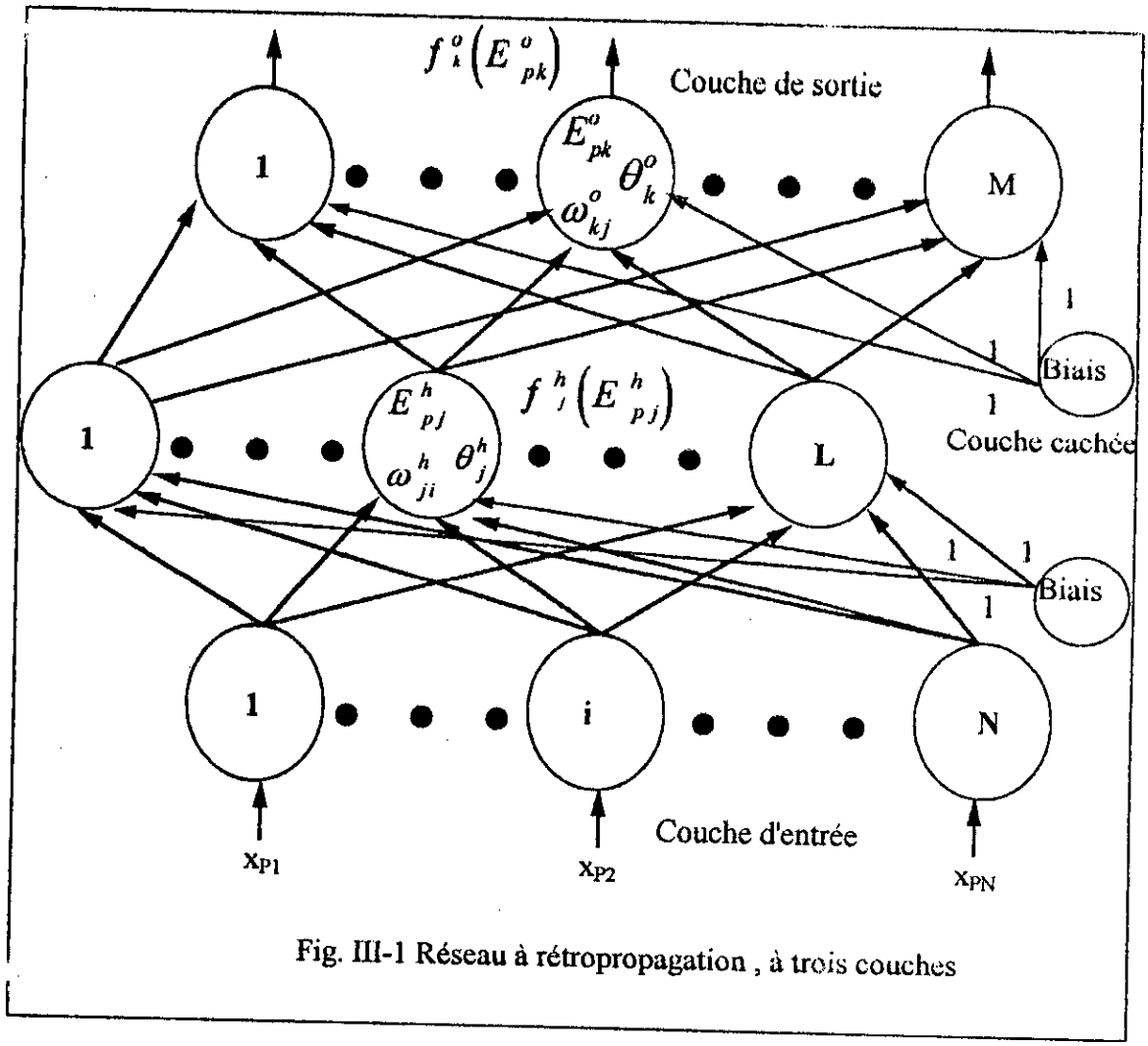


Fig. III-1 Réseau à rétropropagation , à trois couches

La sortie du neurone j de la couche cachée, est donnée par l'équation :

$$S^h_{pj} = f^h_j\left(E^h_{pj}\right) \qquad \text{Eq. III.2}$$

Avec $f(x) = (1 + \exp(-x))^{-1}$, la fonction sigmoïde, ou $f(x) = \text{th}(\alpha x)$ la fonction tangente hyperbolique.

Les équations pour les neurones de sorties, seront :

$$E^o_{pk} = \sum_{j=1}^{L} \omega^o_{kj} S^h_{pj} + \theta^o_k \qquad \text{Eq. III.3}$$

$$S^o_{pk} = f^o_k\left(E^o_{pk}\right) \qquad \text{Eq. III.4}$$

La lettre 'o' en exposant est utilisée, pour référer à la couche de sortie.

Les grandes lignes de l'entraînement du réseau sont les suivantes :

1. Appliquer le vecteur d'exemple à l'entrée du réseau et calculer les valeurs correspondantes à sa sortie.
2. Comparer la sortie réelle du réseau à celle désirée et déterminer une mesure de l'erreur.
3. Déterminer dans quelle direction (+ et -), le changement des poids devra être effectué, pour réduire l'erreur.
4. Déterminer la valeur avec laquelle chaque poids devra être corrigé.
5. Appliquer les corrections aux poids.
6. Répéter les étapes de 1 à 5, avec tous les vecteurs d'apprentissage jusqu'à ce que l'erreur pour tous ces vecteurs soit acceptable.

2.1. Corrections des poids de la couche de sortie.

Définition l'erreur d'un neurone de sortie $\delta_{pk} = (y_{pk} - S^{o}_{pk})$ avec l'indice 'p' qui correspond au $\mathbf{P}^{\text{iéme}}$ vecteur d'apprentissage, l'indice 'k' correspond au $k^{\text{iéme}}$ neurone de sortie .Dans notre cas y_{pk} est la sortie désirée, et S^{o}_{pk} la sortie réelle du neurone k de la couche de sortie.

L'erreur minimisée avec cet algorithme est la somme des carrés des erreurs de tous les neurones de sorties.

$$Z_p = \frac{1}{2}\sum_{k=1}^{M}\delta_{pk}^2 \qquad \text{Eq. III.6}$$

Pour déterminer dans quelle direction la correction des poids doit être effectué, nous calculons la valeur, changée de signe, du gradient $\mathbf{Z}_p$, $\nabla\mathbf{Z}_p$ par rapport au poids ω^{o}_{kj}, c'est alors qu'on peut modifier les valeurs des poids afin de minimiser l'erreur totale.

On peut écrire l'Eq. III.6 sous cette forme :

$$Z_p = \frac{1}{2}\sum_{k=1}^{M}\left(y_{pk} - S^o_{pk} \right)^2 \qquad \text{Eq. III.7}$$

Et

$$\frac{\partial Z_p}{\partial \omega^o_{kj}} = -\left(y_{pk} - S^o_{pk}\right)\frac{\partial f^o_k}{\partial\left(E^o_{pk}\right)}\frac{\partial\left(E^o_{pk}\right)}{\partial \omega^o_{kj}} \qquad \text{Eq. III.8}$$

Où nous avons remplacé S^o_{pk} par ça valeur dans l'Eq. III.4

D'autre part nous avons :

$$\frac{\partial\left(E^o_{pk}\right)}{\partial \omega^o_{kj}} = \left(\frac{\partial\left(\sum_{j=i}^{L} \omega^o_{kj}\, S^o_{pk} + \theta^o_k\right)}{\partial \omega^o_{kj}}\right) = S^o_{pk} \qquad \text{Eq. III.9}$$

En combinant les deux équations Eq. III.8 et Eq. III.9 , nous avons pour la valeur négative du gradient :

$$-\frac{\partial Z_p}{\partial \omega^o_{kj}} = -\left(y_{pk} - S^o_{pk}\right) f^{o'}_k\left(E^o_{pk}\right) S^h_{pj} \qquad \text{Eq. III.10}$$

Les poids de la couche de sortie seront corrigés comme suit :

$$\omega^o_{kj}(t+1) = \omega^o_{kj}(t) + \Delta_p\, \omega^o_{kj}(t) \qquad \text{Eq. III.11}$$

Avec :

$$\Delta_p\, \omega^o_{kj} = \eta\left(y_{pk} - S^o_{pk}\right) f^{\omega'}_k\left(E^o_{pk}\right) S^h_{pj} \qquad \text{Eq. III.12}$$

η est appelé le coefficient d'apprentissage ,il est positif et inférieur à 1.

Dans le cas où $f^o_i\left(E^o_{pk}\right)$ est la fonction sigmoïde :

$$f^o_k\left(E^o_{pk}\right) = \left(1 + \exp\left(-E^o_{pk}\right)\right)^{-1}$$

d'où : $f_k^{o'} = f_k^o\left(1 - f_k^o\right) = S_{pk}^o\left(1 - S_{pk}^o\right)$ Eq. III.13

l'Eq. III.11 devient :

$$\omega_{kj}^o(t+1) = \omega_{kj}^o(t) + \eta\left(y_{pk} - S_{pk}^o\right) S_{pj}^o\left(1 - S_{pj}^o\right) S_{pj}^h \quad \text{Eq. III.14}$$

Nous voulons condenser l'Eq. III.14 en définissant :

$$\delta_{pk}^o = \left(y_{pk} - S_{pk}^o\right) f_k^{o'}\left(E_{pk}^o\right)$$

$$= \delta_{pk} f_k^{o'}\left(E_{pk}^o\right) \quad \text{Eq. III.15}$$

L'Eq. III.11 devient :

$$\omega_{kj}^o(t+1) = \omega_{kj}^o(t) + \eta\,\delta_{pk}^o E_{pj}^h \quad \text{Eq. III.16}$$

2.2. Corrections des poids de la couche cachée.

Nous allons reprendre les mêmes étapes que ceux pour la couche de sortie, mais un problème surgit quand on essaie de déterminer une mesure de l'erreur de sortie des neurones de la couche cachée, pour cela reprenons l'Eq. III.7 :

$$Z_p = \frac{1}{2}\sum_k\left(y_{pk} - S_{pk}^o\right)^2$$

$$Z_p = \frac{1}{2}\sum_k\left(y_{pk} - f_k^o\left(E_{pk}^o\right)\right)^2$$

$$Z_p = \frac{1}{2}\sum_k\left(y_{pk} - f_k^o\left(\sum_k \omega_{kj}^o S_{pj}^h + \theta_k^o\right)\right)^2 \quad \text{Eq. III.17}$$

Nous savons que S_{pj}^h dépend des poids de la couche cachée, à travers l'Eq. III.1, utilisons cette équation pour calculer le gradient de Z_p par rapport aux poids de la couche cachée :

$$\frac{\partial Z_p}{\partial \omega_{ji}^h} = \frac{1}{2}\sum_k \frac{\partial}{\partial \omega_{ji}^h}\left(y_{pk} - S_{pk}^o\right)^2$$

$$\frac{\partial Z_p}{\partial \omega_{ji}^h} = -\sum_k \left(y_{pk} - S_{pk}^o\right)\frac{\partial S_{pk}^o}{\partial\left(E_{pk}^o\right)}\frac{\partial\left(E_{pk}^o\right)}{\partial S_{pj}^h}\frac{\partial S_{pj}^h}{\partial\left(E_{pj}^h\right)}\frac{\partial\left(E_{pj}^h\right)}{\partial \omega_{ji}^h} \qquad \text{Eq. III.18}$$

Chaque terme de l'Eq. III.18 peut être calculé explicitement à partir des équations précédentes d'où:

$$\frac{\partial Z_p}{\partial \omega_{ji}^h} = -\sum_k \left(y_{pk} - S_{pk}^o\right) f_k^{o'}\left(E_{pk}^o\right)\omega_{kj}^o f_j^{h'}\left(E_{pj}^h\right) x_{pi} \qquad \text{Eq. III.19}$$

La correction des poids de la couche cachée sera proportionnelle, à la valeur négative, exprimer par l'Eq. III.19 :

$$\Delta_p \omega_{ji}^o = \eta f_j^{h'}\left(E_{pj}^h\right) x_{pi}\sum_k \left(y_{pk} - S_{pk}^o\right) f_k^{o'}\left(E_{pk}^o\right)\omega_{kj}^o \qquad \text{Eq. III.20}$$

Où η le coefficient d'apprentissage.

on peut utiliser la définition de S_{pk}^o donnée dans la section précédente pour écrire :

$$\Delta_p \omega_{ji}^o = \eta f_j^{h'}\left(E_{pj}^h\right) x_{pi}\sum_k \delta_{pk}^o \omega_{kj}^o \qquad \text{Eq. III.21}$$

Noter que chaque poids corriger, de la couche cachée, dépend de **tous les termes d'erreur,** δ_{pk}^o, de la couche de sortie.

C'est le fondement de la notion de rétropropagation, car les erreurs connues sur la couche de sortie, sont rétropropagés vers la couche cachée, pour déterminer les changements appropriés des poids de cette couche.

En définissant le terme d'erreur de la couche cachée :

$$\delta_{pj}^h = f_j^{h'}\left(E_{pj}^h\right)\sum_k \delta_{pk}^o \omega_{kj}^o \qquad \text{Eq. III.22}$$

Équation de correction des poids devient similaire à celle de la couche cachée :

$$\omega_{ji}^{h}(t+1) = \omega_{ji}^{h}(t) + \eta\, \delta_{pj}^{h} x_{pi} \qquad \text{Eq. III.23}$$

2.3. Résumé de l'algorithme de rétropropagation.

Dans cette section ,nous regroupons toutes les équations de l'algorithme de rétropropagation, elles sont présentées dans l'ordre dans lequel elles doivent être utilisées pendant l'apprentissage :

1) Appliquer le vecteur d'entrée $x_p = (x_{p1}, x_{p2}, \ldots, x_{pN})$, à la couche d'entrée.

2) Calculer les entrées totales de chaque neurone de la couche cachée :

$$E_{pj}^{h} = \sum_{i=1}^{N} \omega_{ji}^{h} x_{pi} + \theta_{j}^{h}$$

3) Calculer les sorites de la couche cachée :

$$S_{pj}^{h} = f_{j}^{h}\left(E_{pj}^{h}\right)$$

4) Passer à la couche de sortie ,calculer les entrées totales de chaque neurone:

$$E_{pk}^{o} = \sum_{j=1}^{L} \omega_{kj}^{o} S_{pj}^{h} + \theta_{k}^{o}$$

5) Calculer les sorties :

$$S_{pk}^{o} = f_{k}^{o}\left(E_{pk}^{o}\right)$$

6) Calculer les termes d'erreurs de la couche de sortie :

$$\delta_{pk}^{o} = \left(y_{pk} - S_{pk}^{o}\right) f_{k}^{o'}\left(E_{pk}^{o}\right)$$

7) Calculer les termes d'erreurs des neurones de la couche cachée :

$$\delta_{pj}^{h} = f_{j}^{h'}\left(E_{pj}^{h}\right) \sum_{k} \delta_{pk}^{o} \omega_{kj}^{o}$$

(noter que les termes d'erreurs de la couche cachée sont calculés avant la correction des poids de la couche de sortie)

8) Corriger les poids de la couche de sortie :

$$\omega_{kj}^{o}(t+1) = \omega_{kj}^{o}(t) + \eta\, \delta_{pk}^{o} E_{pj}^{h}$$

9) Corriger les poids de la couche cachée :

$$\omega_{ji}^{h}(t+1) = \omega_{ji}^{h}(t) + \eta\, \delta_{pj}^{h} x_{pi}$$

10) Calculer le terme d'erreur : $Z_p = \frac{1}{2}\sum_{k=1}^{M} \delta_{pk}^{2}$; car cette valeur, traduit le taux d'apprentissage du réseau, quand elle est *acceptable* (petite) pour tous les vecteurs l'apprentissage, ce dernier peut être arrêté.

3. Considérations pratiques.

Investir dans le travail parfois considérable que constitue l'apprentissage ne se justifie que si l'on souhaite utiliser ensuite le réseau dans une véritable tâche de reconnaissance. Pour cela, il faut être en mesure d'estimer d'abord le degré de confiance que l'on peut placer dans le réseau. C'est pourquoi on scinde la base de données contenant les exemples connus en deux sous-ensembles : la base d'apprentissage proprement dite, à l'aide de laquelle on effectue l'apprentissage, et la base de test sur laquelle on essaie l'aptitude du réseau à reconnaître des exemples non appris. Cette dernière opération doit donc permettre d'estimer la capacité de **généralisation**, qui est le critère déterminant. On comprend facilement que le taux d'erreur mesuré en généralisation est nécessairement plus grand que l'erreur résiduelle de l'apprentissage.

◊ La première condition fondamentale à respecter est que tous les exemples utilisés soient équitablement représentatifs des classes à reconnaître. De plus, on a pu démontrer que l'erreur de généralisation était directement liée au rapport entre le nombre d'exemples et le nombre total de connexions dans le réseau. Une règle de base à respecter est qu'**il faut que le rapport du nombre d'exemples d'apprentissage au nombre de connexions soit supérieur à un** : plus ce rapport est élevé, plus basse peut être l'erreur de généralisation. Ce qui signifie qu'une quantité insuffisante d'exemples devrait décourager d'entreprendre l'apprentissage. C'est ce que nous voulions dire par, vecteurs d'apprentissages **bien choisis** et de **nombre suffisant.**

◊ Une deuxième condition à respecter, liée à la précédente, est de **trouver la taille appropriée de la couche cachée** (ou les couches cachées). On s'aperçoit que, si le

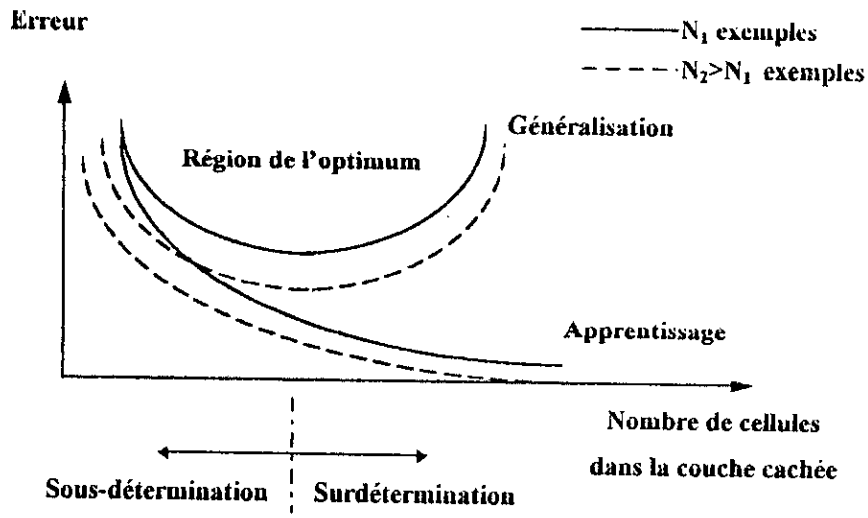


Fig. III-2 Évolution de l'erreur d'apprentissage et de généralisation en fonction de la taille de la couche cachée et de la taille de la base d'exemples.

nombre de neurones de cette couche est insuffisant, on n'arrive pas à atteindre une erreur d'apprentissage suffisamment faible. En revanche, cette erreur peut être rendue de plus en plus petite si la taille augmente (au prix d'ailleurs d'un apprentissage de plus en plus lourd). Mais, dans le même temps, l'erreur de généralisation diminue, passe par un minimum, puis recommence à croître. La Figure III-2 montre cet effet qualitativement.

Revenons sur le rôle de la couche cachée : elle réalise une projection des exemples d'entrée dans un espace de représentation interne, où ceux-ci deviennent linéairement séparables, puisque la couche de sortie, qui est un Perceptron, est capable d'en faire la classification.

Cet espace interne a une dimension qui est le nombre de neurones de la couche cachée. Si ce nombre est trop faible, le réseau n'a pas assez de degrés de liberté pour représenter séparément les caractéristiques classifiantes, et l'apprentissage reste

mauvais. A l'opposé, si ce nombre est trop grand, le réseau trouve une projection interne trop détaillée des exemples, et en réalise en quelque sorte l'apprentissage « par cœur » ; dans ce cas, les exemples de base de test sont mal reconnus, donc la généralisation mauvaise. Enfin, il ne faut pas oublier que, quand on augmente progressivement la taille de la couche cachée, on augmente assez rapidement le nombre de connexions, donc on risque de tomber dans le cas d'insuffisance du nombre d'exemples mentionné plus haut.

Le problème de minimisation que la rétropropagation tente de résoudre n'est pas simple. En effet, la surface de la fonction d'erreur présente des caractéristiques peu satisfaisantes pour effectuer une descente de gradient : minima locaux, qui empêchent la convergence de l'algorithme, plateaux où les pentes sont très faibles, etc...

D'autres parts l'algorithme de rétropropagation est très gros consommateur de temps de calcul sur des problèmes de taille, c'est pour ça qu'on choisit le coefficient d'apprentissage η aussi grand que possible pour la rapidité et pour éviter la capture dans un minimum local, mais il faut surveiller l'évolution de l'erreur pour détecter les oscillations qui résultent d'une valeur trop forte. Rien n'empêche d'ailleurs de faire évoluer ce paramètre en cours d'apprentissage, fort d'abord puis de plus en plus faible.

Un autre moyen efficace consiste à utiliser une technique dite « d'inertie » (en anglais '*momentum*'), dans laquelle on tient compte de la correction du poids à l'étape précédente l'équation III-23 devient :

$$\omega_{kj}^{o}(t+1) = \omega_{kj}^{o}(t) + \eta\,\delta_{pk}^{o} E_{pj}^{h} + \gamma\,\Delta_p \omega_{kj}^{o}(t-1) \qquad \text{Eq. III.24}$$

Dans NeuralWorks tous ces paramètres sont regroupés
Nous utiliserons ces deux derniers moyens dans notre application.

4. Application.

4.1 Description du réseau.

Dans ce qui suit, nous allons essayer de créer un réseau de neurones, basé sur l'algorithme de rétropropagation du gradient. Ce réseau aura pour objet de reconnaître les dix chiffres arabes (de '0' à '9'). Nous utiliserons pour la conception, l'apprentissage et le test de ce réseau, un logiciel spécialisé ; il s'agit du « **NeuralWorks Professional II/PLUS** » édité par la société américaine « **NeuralWare,Inc.** ».

Nombre de couches	3
Nombre de cellules de la couche d'entrée	12x8 (rétine d'une largeur de 8 pixels et une hauteur de 12 pixels)
Nombre de cellules de la couche cachée	45
Nombre de cellules de la couche de sortie	10 (un neurone pour chaque chiffre)
Modèle du neurone (le même pour tous les neurones) - Entrée totale : - Fonction d'activation : - Fonction de sotie :	 - Somme des entrées pondérer par les poids des connexions - Tangente hyperbolique - Fonction identité

Tableau III-1

Un neurone de biais est connecté à tous les neurones de la couche cachée ainsi qu'à ceux de la couche de sortie.

4.2. Base d'apprentissage et base de test.

La Fig. III-3 illustre les exemples d'apprentissage, on s'est inspiré des polices de caractères du WINDOWS pour leur génération. La Fig. III-4 illustre les exemples de tests.

4.3. Apprentissage et résultats.

L'algorithme utilisé lors de l'apprentissage est le même que celui exposé à la section précédente. Cependant, pour accélérer l'apprentissage, nous avons introduit un **momentum** γ , ainsi qu'un coefficient d'apprentissage η dépendant du temps, comme suggérer au (§ 3).

Nous avons fixé un seuil d'erreur (somme quadratique des erreurs sur chaque cellule de sortie) à **0.05**, l'apprentissage est arrêté dés que cette valeur est atteinte pour tous les vecteurs d'apprentissage.

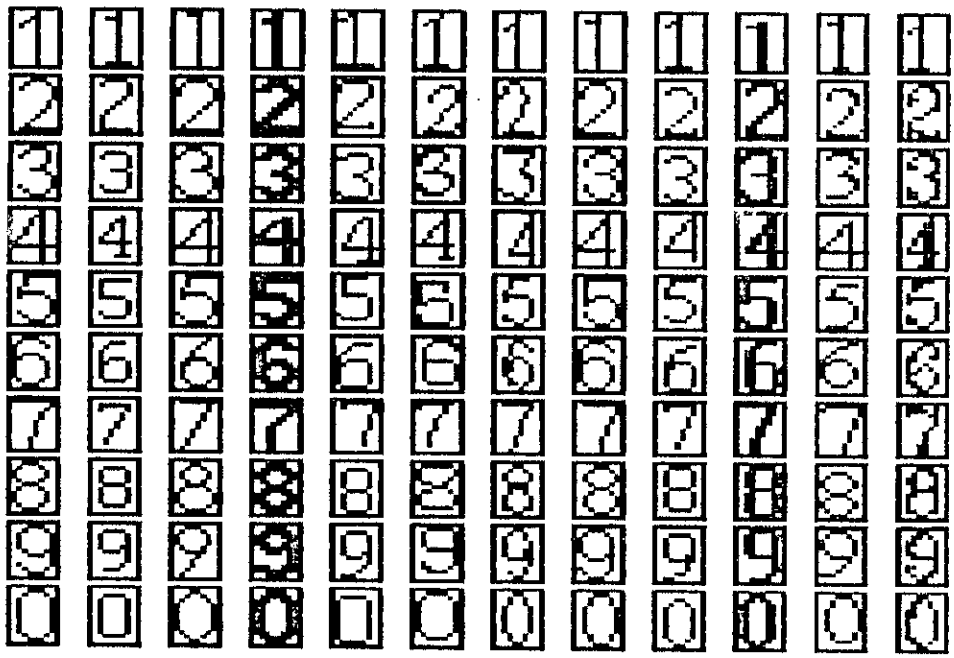


Figure III-3 Base d'apprentissage

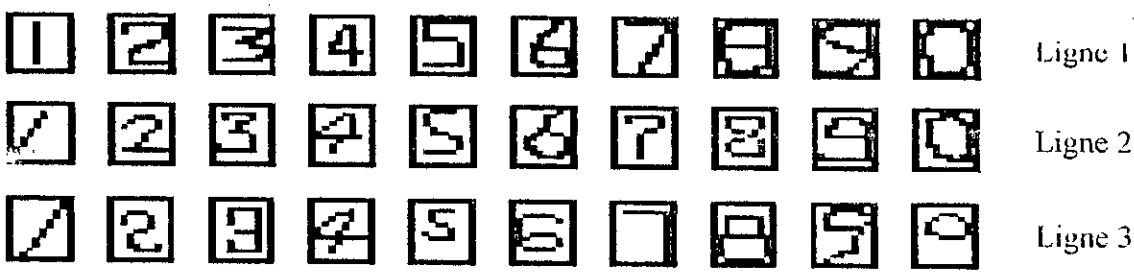


Fig III-4 Base de test

Les tableau III.5 et III.6 récapitulent les valeurs de η et γ pour la couche cachée ainsi que la couche de sortie. Les nombres inscrits dans la première colonne indiquent les valeurs de η et γ pour les 5000 premiers passages des exemples d'apprentissage, ceux de la deuxième colonne les valeurs de η et γ pour les 5000 passages suivant, et ainsi de suite.

Passages	1-5000	5000-10000	10000-70000
η	**0.2000**	**0.3000**	**0.3375**
γ	**0.4000**	**0.3000**	**0.0500**

Tableau III-5 Valeurs de η et γ pour la couche cachée.

Passages	1-5000	5000-10000	10000-70000
η	**0.2000**	**0.3000**	**0.0188**
γ	**0.2000**	**0.3000**	**0.0500**

Tableau III-6 Valeurs de η et γ pour la couche de sortie.

L'apprentissage a été effectué en 120 minutes, au total 38520 passages, nous avons présenté les exemples de tests de la Fig. III-4 au réseau, la réponse du réseau a été très rapide.

Les résultats des tests sont récapitulés dans les tableaux III-2 , III-3 et III-4 ,qui correspondent respectivement aux ligne 1, 2 et 3 de la Fig. III-4. Chaque chiffre est associé à un neurone ; le neurone N°1 correspond au chiffre '1', le neurone N°2 au chiffre '2', et ainsi de suite. Le neurone N°10 correspond au chiffre '0'.

Le réseau ne reconnait pas correctement le chiffre '6' dans les trois lignes puisqu' il le détecte comme étant un '4', les chiffres '1' à '5' de la ligne N°2 ainsi que tous les chiffres de la ligne N°3 (mis à par le chiffre '4'), qui sont légèrement déformés ou inclinés, ne sont pas correctement reconnues non plus.

	Neurone 1	Neurone 2	Neurone 3	Neurone 4	Neurone 5	Neurone 6	Neurone 7	Neurone 8	Neurone 9	Neurone 10
chiffre '1'	**0.964531**	-0.982704	-0.997206	-0.905021	-0.992626	-0.996366	-0.984749	-0.999999	-0.999638	-0.988033
chiffre '2'	-0.962793	**0.416446**	-0.979046	-0.999250	-0.998592	-0.999955	-0.818917	-0.999955	-0.703843	-0.986142
chiffre '3'	-0.995465	-0.998843	**0.199760**	-0.875628	-0.997135	-0.997897	-0.981553	-0.997694	-0.998504	-0.999698
chiffre '4'	-0.994079	-0.999117	-0.999975	**0.650450**	-0.715916	-0.999949	0.292011	-0.992947	-0.218852	-0.999851
chiffre '5'	-0.995826	-0.998979	-0.979727	-0.999236	**0.882682**	-0.999700	-0.966616	-0.973979	-0.943593	-0.993685
chiffre '6'	-0.992848	-0.999944	-0.933912	0.345254	-0.999879	**-0.871759**	-0.982284	-0.999906	-0.998489	-0.998916
chiffre '7'	-0.990578	-0.968893	-0.979665	-0.985569	-0.998943	-0.999982	**0.985827**	-0.999280	-0.988339	-0.999972
chiffre '8'	-0.993354	0.599743	-0.433430	-0.997110	-0.753861	-0.934815	-0.888983	**0.975788**	-0.981167	0.957689
chiffre '9'	-0.988085	-0.893969	-0.999847	-0.999785	-0.920239	-0.999966	-0.883537	-0.987005	**0.993376**	-0.993781
chiffre '0'	-0.990509	-0.845889	-0.996744	-0.997820	-0.999039	-0.985170	-0.998868	-0.814743	-0.996383	**0.959940**

Tableau III-2

	Neurone 1	Neurone 2	Neurone 3	Neurone 4	Neurone 5	Neurone 6	Neurone 7	Neurone 8	Neurone 9	Neurone 10
chiffre '1'	**0.048982**	-0.932481	-0.999994	-0.601355	-0.999214	-0.750242	-0.975095	-0.995472	-0.999949	-0.999006
chiffre '2'	-0.853572	**-0.998998**	-0.984579	-0.785573	-0.999989	-0.997084	-0.854158	-0.999940	-0.986326	-0.992361
chiffre '3'	-0.820173	-0.998824	**-0.879814**	-0.705291	-0.996646	-0.999993	-0.760078	-0.999094	-0.996167	-0.996745
chiffre '4'	-0.926734	-0.997993	-0.986023	**-0.869473**	-1.000000	-0.987095	-0.900129	-0.998643	0.794107	-0.999727
chiffre '5'	-0.997698	-0.999463	-0.981735	-0.996052	**-0.058023**	-0.986212	-0.988692	-0.994853	-0.841843	-0.999651
chiffre '6'	-0.981743	-0.999972	-0.999855	0.591882	-0.999491	**0.018808**	-0.997918	-0.985019	-0.999752	-0.958817
chiffre '7'	-0.944208	-0.968111	-0.929080	-0.997385	-0.985547	-0.999990	**0.812268**	-0.996611	-0.998101	-0.999750
chiffre '8'	-0.999945	-0.993743	-0.958296	-0.978762	-0.982597	-0.986404	-0.996929	**0.692033**	-0.990871	-0.999723
chiffre '9'	-0.997103	-0.972865	-0.999034	-0.999443	-0.928537	-0.998535	-0.994935	-0.998089	**0.409199**	-0.968177
chiffre '0'	-0.996747	-0.999333	-0.998460	-0.995519	-0.999341	-0.947120	-0.999729	-0.999161	-0.305353	**0.837534**

Tableau III-3

	Neurone 1	Neurone 2	Neurone 3	Neurone 4	Neurone 5	Neurone 6	Neurone 7	Neurone 8	Neurone 9	Neurone 10
chiffre '1'	**-0.990326**	-0.892793	-0.987340	-0.965027	-0.999175	-0.999985	0.885323	-0.998798	-0.994667	-0.999930
chiffre '2'	0.708355	**-0.895112**	-0.999429	-0.669350	-0.999054	-0.599007	-0.990270	-0.999997	-0.999975	-0.998886
chiffre '3'	-0.999701	-0.979474	**0.107414**	-0.998333	-0.999559	-0.999728	-0.997421	-0.999353	0.153859	-0.990534
chiffre '4'	-0.836833	-0.968880	-0.998841	**0.928640**	-0.999986	-0.188994	-0.994539	-0.999723	-0.999983	-0.999226
chiffre '5'	0.664295	-0.859140	-0.989414	-0.974016	**-0.966071**	-0.999608	-0.807083	-0.999983	-0.999423	-0.999867
chiffre '6'	-0.999532	-0.994652	-0.977346	-0.510289	-0.999890	**-0.937789**	-0.999635	-0.665630	-0.998995	-0.975527
chiffre '7'	-0.991446	0.554657	-0.986230	-0.999124	-0.996659	-0.999930	**-0.912226**	-0.999955	-0.831185	-0.977535
chiffre '8'	-0.905958	-0.998391	-0.999643	-0.997651	-0.997359	-0.508003	-0.999989	**-0.998720**	-0.717616	0.024214
chiffre '9'	-0.974787	-0.995224	-0.999945	-0.993211	-0.962633	-0.999987	0.754415	-0.999252	**0.090531**	-0.991599
chiffre '0'	-0.864662	-0.999719	-0.999915	-0.999239	-0.988644	-0.998350	-0.996846	-0.999685	0.984542	**-0.709933**

Tableau III-4

4.4. Commentaires.

- Le réseau à rétropropagation que nous avons conçu ne répond pas aux exigences attendues, un moyen de corriger cela, serait de reprendre l'apprentissage avec une base d'apprentissage plus large. Comme nous l'avons dit au (§3), il faut que le nombre d'exemples d'apprentissage soit, au moins, égale au nombre de connexion qui est 4780 ((96+1) x 45 + (45+1) x10), or les exemples de notre base d'apprentissage ne sont que de 120 exemples.

- Une autre manière de corriger ce problème (complémentaire à la première), serait d'augmenter le nombre de neurones de la couche cachée, mais dans ce cas il faut augmenter encore plus le nombre d'exemples d'apprentissage.

Dans les deux solutions proposées, l'élargissement de la base d'apprentissage est nécessaire, mais comment allons nous choisir 4780 (ou plus) exemples d'apprentissages, en maintenant une **représentation équitable des classes à reconnaître,** sans redondance d'une représentation d'un exemple particulier ? Comment intégrer les déformations, décalages et différentes tailles des caractères à reconnaître ? Surtout qu'en combinant ces trois dernières caractéristiques des caractères on arrive à une infinité de cas possibles.

Remarque: Nous aurions pus choisir une règle d'apprentissage plus rapide et qui ne pose pas de problème de minima locaux, mais nous considérons que le choix des dimensions du réseau et de la base d'apprentissage sont prépondérant au choix de la règle d'apprentissage.

Si le réseau n'arrive pas à trouver une représentation interne correcte (par manque de neurones dans la couche cachée), ou s'il trouve une représentation trop détaillée (par surplus de neurones dans la couche cachée), la règle d'apprentissage n'y changera rien. D'autres parts une base d'apprentissage mal choisit conduira à une mauvaise généralisation.

5. Conclusion.

Bien que l'algorithme de rétropropagation du gradient donne de bons résultats dans de nombreuses applications, on constate que rien ne nous permet de choisir correctement les dimensions du réseau et les exemples de la base d'apprentissage afin d'arriver à une généralisation acceptable.

Dans le chapitre suivant, nous allons présenter un réseau spécialisé dans la reconnaissance de caractères et qui s'inspire du plus parfait des systèmes dans ce domaine, c'est **Le système visuel des mammifères.**

CHAPITRE IV

Le Neocognitron

1. Introduction [PAN 1987] , [DAV 1990]

Les modèles de réseaux, comme ceux à rétropropagation, tendent à avoir une application générale. On peut utiliser un seul réseau dans des applications différentes, en changeant les dimensions du réseau, ses paramètres et la base d'apprentissage. Cependant , les développeurs du Neocognitron ont entrepris de concevoir une architecture spéciale pour une application spécifique ; la reconnaissance de caractères.

L'un des premiers chercheurs à développer ce réseau et le professeur Kunihiko Fukushima, du département de Biologie de l'Université d'Osaka au Japon, ces travaux étaient inspirés du système visuel de mammifères.

Nous n'allons pas aborder en profondeur l'anatomie et la physiologie du cortex visuel, cependant, une description bref des caractéristiques de cette partie du cerveau, nous permettra de comprendre les concepts de base du Neocognitron.

2. Système visuel des mammifères.

2.1. L'œil.

Le système visuel est le système sensoriel le mieux connu. Il est constitué de capteurs qui sont les yeux, du nerf optique qui assure la liaison entre les capteurs et le cerveau , et du cortex visuel qui assure (chez les vertébrés supérieurs) une partie importante du traitement de l'information visuelle.

2.2. La structure de l'oeil

Les rayons lumineux qui arrivent sur l'oeil passent tout d'abord par le cristallin qui joue le rôle d'une lentille. Cette lentille focalise une image du monde extérieur sur le fond de l'oeil qui est constitué de la rétine .

L'image projetée sur la rétine est précise et nette elle active alors le « film photographique » de la rétine. Celui-ci est constitué de cellules photosensibles: les cellules cônes et les cellules bâtonnets.

- Les cellules bâtonnets se retrouvent pratiquement sur toute la rétine. Elles contiennent un pigment qui se décompose lorsqu'il est soumis à la lumière , puis se resynthétise de façon lente (ce qui explique les phénomènes d'éblouissement)
- Les cellules cônes se trouvent essentiellement dans le centre de la rétine et sont sensibles aux couleurs.

L'image du monde que « voit » la rétine de l'oeil est donc constituée d'une série de points ou taches noires, lumineuses ou colorées.

Les signaux engendrés par ces cellules connaissent alors un traitement préliminaire par des cellules situées au fond de l'oeil avant d'être envoyés par l'intermédiaire du nerf optique au cerveau.

2.3. Le cortex visuel.

Les signaux électriques transmis par le nerf optique passent dans un centre de relais situé au centre de cerveaux ,avant d'aboutir au cortex visuel situé à l'arrière du cerveau.

Une découverte récente que l'on doit à Huble et Wiesel, montre que le cortex visuel est organisé selon un système de colonnes de cellules nerveuses, étendues de la superficie à la profondeur. Ces colonnes seraient l'unité fonctionnelle de base du cortex. Elles semblent coder les données premières de l'expérience visuelle .

Ceci amène à se faire une première idée d'organisation hiérarchique au niveau biologique du traitement des signaux électriques transmis par le nerf optique.

2.4. Le système nerveux visuel des mammifères

Nous allons étudier les effets de stimuli constitués par des lignes sur le système visuel du chat pour essayer de comprendre d'une manière plus générale comment le cerveau décode les messages sensoriels qui lui parviennent.

Ces études furent menées à l'aide de micro-électrodes qui permettent de mesurer l'activité des cellules nerveuses .

Elles ont permis de distinguer trois groupes de cellules :

◊ Les cellules ganglionnaires situées juste derrière la rétine

◊ Les cellules dites « simples » du cortex visuel

◊ Les cellules dites « complexes » qui appartiennent aussi au cortex visuel

Chacun de ces groupes joue un rôle spécifique dans le traitement des signaux visuels perçus:

◊ Le groupe des cellules ganglionnaires assure une division de l'image perçue en un ensemble de petites aires circulaires (qu'on appelle aussi aire de perception) auxquelles est affectée une cellule ganglionnaire bien précise. Le niveau de luminosité de chacun de ces aires est représenté par le niveau d'activité de la cellule ganglionnaire qui lui est affectée .

◊ Le groupe des cellules simples, assure le même type de division de l'image mais des aires plutôt rectangulaires et plus grandes. Celles-ci contiennent plusieurs des aires circulaires affectées aux cellules ganglionnaires. Le niveau d'activité des cellules simples est conditionné par la présence de lignes brillantes ou foncées dans les aires qui leur sont affectées. De plus la force de cette activité dépend de l'angle d'inclinaison des lignes repérées ,leur activité étant minimale pour des lignes qui font un angle droit avec celles qui correspondent à leur activité maximale On est donc parvenu à la conclusion que la sensibilité d'une cellule simple à une orientation particulière dépend des connexions entre cette cellule et les cellules ganglionnaires.

◊ Le groupe des cellules complexes assure des fonctions plus larges. Par exemple , on a remarqué que certaines de ces cellules étaient sensibles au fait qu'il y ait ou non une ligne droite en un endroit quelconque de l'espace visuel du chat. Ceci impliquerait que ces cellules soient reliées à toutes les cellules simples qui repérant une ligne droite en un endroit précis de l'image et qu'elles appliquent la fonction logique « OU » aux signaux émis par ces cellules.

2.5. Conclusion

Si l'on récapitule les données précédentes, on peut penser que les cellules réceptrices de la rétine répondent à un stimulus de type ligne droite en excitant certaines cellules ganglionnaires dont les aires réceptives sont voisines du stimulus.

Ces cellules ganglionnaires stimulent alors des cellules simples spécialisées dans la reconnaissance de lignes droites dans certaines aires de l'espace visuel, la spécification de ces cellules se fait par leurs connexions avec les cellules ganglionnaires.

Enfin, un groupe de cellules simples peut être relié à une cellule complexe qui sera activée à partir du moment où une de ces cellules simples le sera. On peut conclure que ces cellules opèrent d'une façon hiérarchique ; et qu'elles extraient des signaux électriques qui sont transmis une l'information de plus en plus abstraite , afin de représenter finalement l'image vue sous la forme d'un ensemble de traits et de contours.

3. Architecture et fonctionnement du Neocognitron. [FUK 1991]

Un modèle de réseau de neurones pour la reconnaissance de caractères a été proposé par K. Fukushima . Il est capable de reconnaître des caractères déformés et de différentes tailles. Le Neocognitron est un réseau hiérarchique, composé de plusieurs couches de neurones, les connexions sont directes entre les neurones de couches voisines ; quelques unes sont variables et peuvent être modifiées lors de l'apprentissage. Ce réseau a un grand pouvoir de généralisation, la présentation de quelques exemples de caractères déformés, est suffisant. Il n'est pas nécessaire de lui présenter toutes les versions déformées des caractères qui peuvent apparaître lors de son utilisation. Après apprentissage il peut reconnaître des caractères déformés, de tailles différentes ou décalés.

Contrairement à d'autres réseaux, il n'a pas besoin d'un pretraitement comme la normalisation de position, de taille ou de déformation du caractère à reconnaître.

3.1. Architecture du Neocognitron.

Le Neocognitron, a évolué à partir d'une ancienne version appelée le ***cognitron***. Il existe plusieurs versions du Neocognitron lui même, celui que nous allons décrire a 9

couches de neurones, incluant la couche d'entrée ou rétine, ce réseau a été conçu pour reconnaître les 10 chiffres arabes ainsi que les 25 lettres majuscules de l'alphabet (le O est assimilé au zéro), cette reconnaissance est effectuée indépendamment de la position, la taille et la déformation du caractère à reconnaître.

La propagation du signal s'effectue de la rétine vers la couche de sortie, il n'y a pas de bouclages des connexions.

Les neurones du Neocognitron sont organisés en module, chaque module est constitué de deux couches, une couche de cellules simples dites **cellules-S**, suivi d'une couche de cellules complexes dites **cellules-C**, chaque couche à son tour, (qu'elle soit simple ou complexe) est divisée en un certain nombre de **plans**, chaque plan est un tableau carré de neurones. Tous les plans d'une couche donnée ont le même nombre de neurones, cependant ce nombre peut changer d'une couche à l'autre au sein du même module.

Il y'a d'autres neurones appelés **cellules-V**, ils sont arrangés en tableaux et chaque couche de cellules-S en possède un, de même dimension que l'un de ces propres plans.

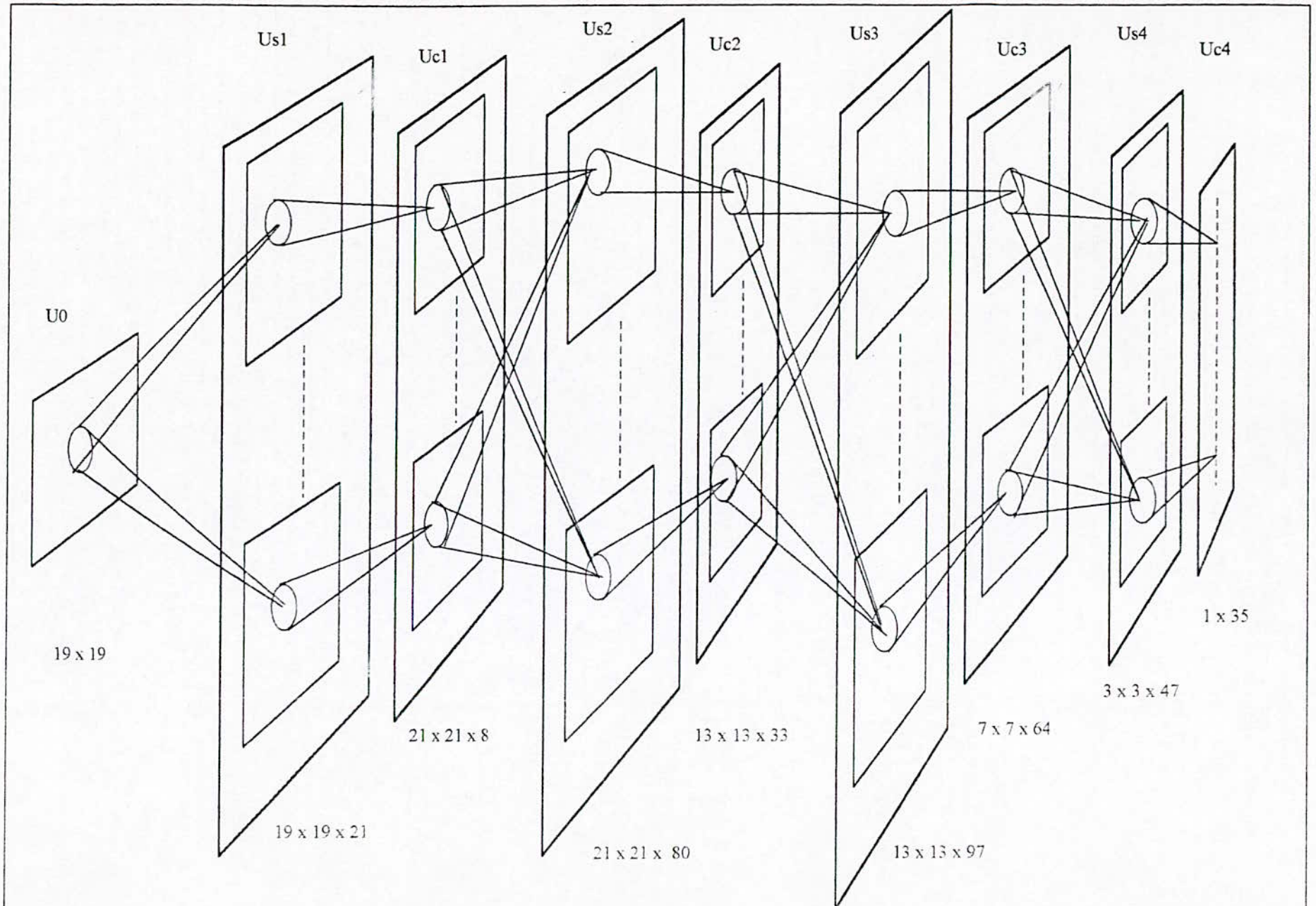
On construit le réseau complet en cascadant, une couche d'entrée, un certain nombre de modules (dans notre cas il y'a 4 modules) , la figure IV-1 illustre la hiérarchie du réseau ainsi que le nombre de plans par couches.

3.2. Fonctionnement du Neocognitron.

La stratégie de connexion est différente de celle des réseaux complètement connectés entre couche (comme c'est le cas du réseau à rétropropagation), la figure IV-3 schématise ces connexions.

Chaque couche de cellules-S, agit comme un système qui extrait des caractéristiques locales à partir des réponses de la couche précédente. Sur la première couche de cellules-S U_{s1} , les neurones d'un plan donné sont sensibles à la présence, sur la rétine, d'un segment de droite à une certaine orientation. Les neurones d'un même plan-S sont sensibles à une seule forme, mais dans différents endroits de la rétine.

Plus nous avançons dans le réseau, et plus les caractéristiques auxquels les neurones des couches-S sont sensibles, deviennent compliquées.



Figur e IV.1 : structure hierarchique du neocognitron

Les neurones des couches-C intègrent les réponses d'un groupe de plans-S , du même module, ainsi les cellules-C sont moins sensibles à la position exacte des caractéristiques qu'elles identifient. C'est cette stratégie qui donne au Neocognitron, son aptitude à identifier des caractères indépendamment de leur position sur la rétine.

Une fois la couche finale atteinte, le champ réceptive de chaque neurone (en anglais *receptive-field*), embrasse toute la rétine.

3.2.1. Fonctionnement des cellules-S

Pour comprendre le fonctionnement des cellules-S, nous allons concentrer notre attention sur un seul plan de la couche U_{sl} .

La couche U_0 est un tableau de 19 par 19 pixels, chaque plan de U_{sl} est un tableau de même dimension que la rétine, ces neurones scannent entièrement la rétine, pour détecter un stimulus particulier à différents endroits , ce stimulus change d'un plan à l'autre.

Chaque cellule-S est connectée à un ensemble de pixels (un carré de 3x3 pour U_{sl}) de la rétine, qu'on appelle 'ensemble de propagation des connexions'(en anglais *spatial spread*), ces pixels sont centrés sur le pixel qui occupe la même position relative que la cellule-S où convergent ces connexions.

Un seul plan de cellules-V (U_{vl}) est associé à U_{sl}. Le plan de cellules-V contient le même nombre de cellules que les plans de U_{sl} , en plus U_{vl} a le même ensemble de propagation des connexions que les cellules-S de U_{sl} (3x3).

Une cellule-S est connectée à la cellule-V qui occupe la même position relative qu'elle, ces connexions sont inhibitrices. La Fig. IV-2 illustre ces connexions. La notation $u_{Sl}(\boldsymbol{n}, k)$, est utilisée pour indiquer la sortie d'une cellule-S de la couche U_{sl} , où $\boldsymbol{n}$ est la paire de coordonnées du centre de l'aire de perception dans la couche U_0, l'indice l indique le numéro du module.

Pour les cellules-S , l'indice k et le numéro du plan-S , avec $1 \leq k \leq K_{Sl}$, pour les cellules-C , il indique le numéro du plan-C, avec $1 \leq k \leq K_{Cl}$. Les valeurs de K_{Sl} et K_{Cl} ainsi que les dimensions du réseau sont dans les figures Fig. IV-1 et Fig. IV-2. La sortie d'une cellule-S est donnée par l'équation Eq. VI-1 :

$$u_{Sl}(n,k) = r_l(k)\,\varphi\left[\frac{1+\sum_{i=1}^{K_{Cl-1}}\sum_{v\in A_l} a_l(v,i,k)\,u_{Cl-1}(n+v,k)}{1+r_l(k)\left\{1+r_l(k)\right\}^{-1} b_l(k)\,u_{Vl}(n)} - 1\right] \qquad \text{Eq. VI-1}$$

Où : $\varphi[x] = x$ si $x \geq 0$, $\varphi[x] = 0$ si $x<0$

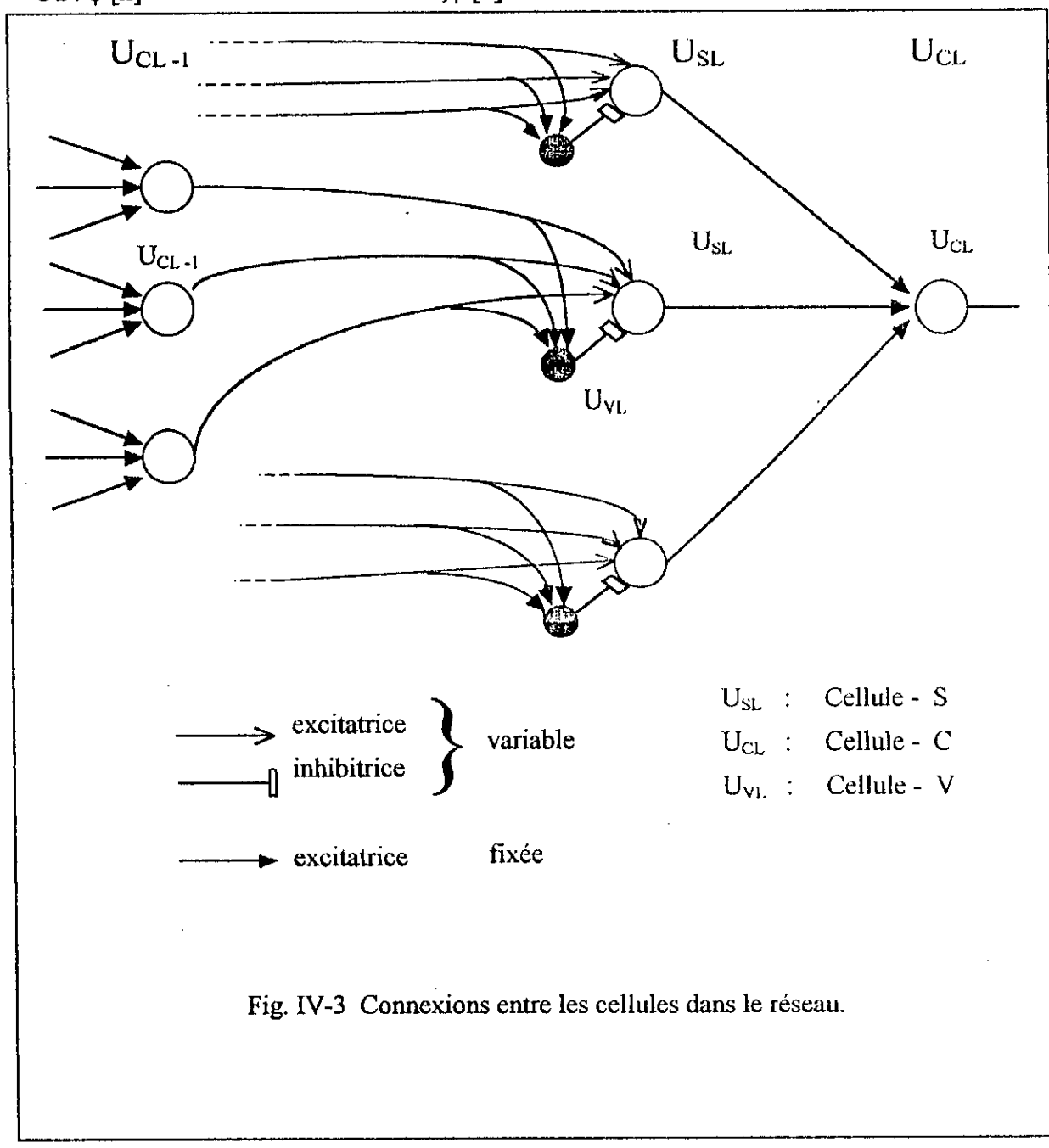
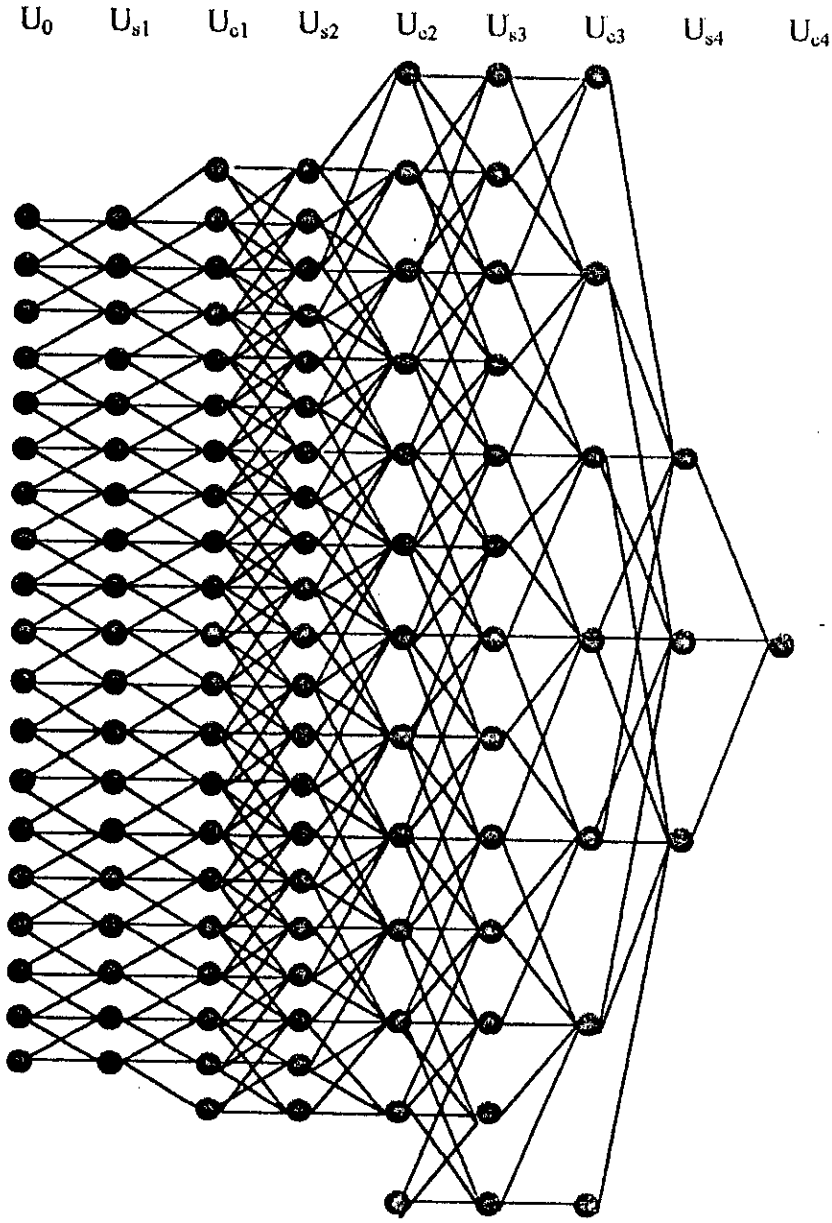


Fig. IV-3 Connexions entre les cellules dans le réseau.



1	12	8	80	33	97	64	47	35
x	x	x	x	x	x	x	x	x
19	19	21	21	13	13	7	3	1
x	x	x	x	x	x	x	x	x
19	19	21	21	13	13	7	3	1

Fig. IV-2 : connexions du Necognitron , ici un seul plan est visible

Dans le cas ou $l = 1$ dans l'équation Eq. VI-1 , $u_{Cl-1}(n,i)$ correspond à $u_0(n)$, qui est l'état du pixel de coordonnées n (pixel allumé u_0=1, pixel éteint u_0=0), et nous avons dans ce cas K_{cl-1}=1.

La quantité $a_l(v, i, k)$ (≥0) est le poids de la connexion entre la cellule-C de la couche précédente et la cellule-S dont on calcule la sortie.

A_l est la portée des connexions de la cellule-S, ainsi v parcourt toutes les cellules-C auxquelles la cellule-S est connectée. La taille de A_l dans notre système peut être lu sur la Fig. IV-2 : (3 x 3) pour A_1 ,(5 x 5) pour A_2 , A_3 et A_4 .

$b_l(k)$ (≥0) est le poids de la connexion avec la cellule-V de la même couche que la cellule-S.

Puisque toutes les cellules-S d'un même plan ont le même ensemble de poids, les termes $a_l(v, i, k)$ et $b_l(k)$ ne contiennent pas l'argument n.

Le paramètre $r_l(k)$ détermine l'efficacité de l'entrée inhibitrice et la sélectivité dans l'extraction des caractéristiques de l'exemple présenté au réseau.

La cellule-V qui envoie le signal inhibiteur à la cellule-S, a pour sortie :

$$u_{vl}(n) = \left[\sum_{i=1}^{K_{Cl-1}} \sum_{v \in A_l} c_l(v) \left\{ u_{Cl-1}(n+v, i) \right\}^2 \right]^{1/2} \qquad \text{Eq. VI-2}$$

Où $c_l(v)$ (≥0) représente les poids des connexions, excitatrices fixes, que la cellule-V a avec les cellules-C de la couche précédente (de tous les plans-C et qui appartiennent a l'ensemble de propagation des connexions de la cellule-S) ces poids sont une fonction monotone et décroissante dont l'argument est $|v|$ (module du vecteur v), dans notre système :

$$c_l(v) = \gamma_l^{|v|}$$

Avec $\gamma_1 = \gamma_2 = \gamma_3 = 0.9$ et $\gamma_3 = 0.8$.

3.2.2. *Fonctionnement des cellules-C*

Les cellules-C reçoivent leurs connexions, d'un ou plusieurs plans-S de la couche précédente, la sortie d'une cellule-C est donnée par l'équation suivante :

$$u_{Cl}(n,k) = \psi\left[\sum_{i=1}^{K_{Sl}} j(i,k)_l \sum_{v \in A_l} d_l(v)\, u_{Sl}(n+v, i)\right] \qquad \text{Eq. VI-3}$$

Où, $\psi[\,]$ est une fonction de saturation spécifique aux cellules-C , elle est définie par :

$$\psi[x] = \frac{\varphi[x]}{1+\varphi[x]}$$

$d_l(v)$ (≥ 0), est le poids des connexions fixes, c'est une fonction monotone et décroisante dont l'argument est $|v|$ (module du vecteur v), dans notre système :

$$d_l(v) = \delta_{l0}\ \delta_l^{\,|v|}$$

Avec $\delta_{10} = \delta_{20} = 4.0$, $\delta_{30} = 2.5$, $\delta_{40} = 2.5$;

$\delta_1 = 0.9$, $\delta_2 = 0.8$, $\delta_3 = 0.7$, $\delta_4 = 1.0$;

D_l est la portée des connexions de la cellule-C, ainsi v parcourt toutes les cellules-S auxquelles la cellule-C est connectée. La taille de D_l dans notre système peut être lu sur la Fig. IV-2 : (3 x 3) pour D_1 , (7 x 7) pour D_2 , (5 x 5) pour D_3 et (3 x 3) pour D_4 .

Les sorties de plusieurs plans-S, sont souvent regroupées, et convergent vers un plan-C . Ce regroupement des sorties est exprimé par j_l (i, k) , cette valeur est égale à 1 si le $k^{ème}$ plan-C reçoit des connexions du $i^{ème}$ plan-S, sinon nulle.

3.3. Entraînement du Neocognitron.

Le Noecognitron peut être entraîné avec un apprentissage supervisé ou non supervisé, dans notre cas l'apprentissage est supervisé.

Les poids des connexions des cellules-S sont renforcés pendant l'apprentissage, leurs valeurs initiales sont nulles, l'apprentissage est effectué pas à pas, c'est-à-dire que

l'apprentissage d'un module est effectué après avoir entraîné tout les modules précédents.

Tous les modules sont entraînés avec le même processus. Considérons un module particulier, dans une couche de cellules-S , un 'professeur' choisit un plan à entraîner, ensuite il présente un exemple d'apprentissage à la couche d'entrée U_0 et indique quelle cellule-S sera l'objet de l'apprentissage, cette cellule est dit 'cellule gagnante' (en anglais *seed cell*), la cellule gagnante est repérée par la position du centre de son aire de perception.

Le renforcement des poids des connexions de la cellule gagnante est proportionnel à l'intensité de la réponse des cellules auxquelles la cellule gagnante est connectée (voir Eq. IV-4 et Eq. IV-5) , les réponses en question sont obtenues (ou calculer) en présentant l'exemple d'apprentissage à la couche d'entrée et en propageant le signal jusqu'à la couche qui précède la couche à entraîner (l'entraînement des couches antérieures étant déjà effectué).

$$\Delta a_l(v,i,\hat{k}) = q_l\, c_l(v)\, u_{Cl-1}(\hat{n}+v,i) \qquad \text{Eq. IV-4}$$

$$\Delta b_l(\hat{k}) = q_l\, u_{vl}(\hat{n}) \qquad \text{Eq. IV-5}$$

Avec $\hat{n}$ la position de la cellule gagnante et $\hat{k}$ le numéro d'ordre du module.

Après l'entraînement de la cellule gagnante, on recopie les poids ainsi obtenus à toutes les autres cellules du même plan.

3.3.1. Stratégie d'extraction des caractéristiques locales.

La reconnaissance de caractères avec le Neocognitron consiste en l'extraction, dans les premiers modules, des caractéristiques locales tels que, les segments de droites dans différentes orientations, intersection de segments de droites, angles droits, la courbature d'une ligne et les fins de ligne. Parmi ces caractéristiques les segments de droites ont un intérêt particulier.

Supposons qu'une couche intermédiaire, par exemple la couche U_{s2} , contient des plans-S spécialisés dans l'extraction de segments de droites, d'autre dans l'extraction

d'angles droits, ...etc. Considérant un exemple particulier le caractère 'L' qu'on présente à la couche d'entrée, plusieurs cellules-S seront activées dans le plan-S qui extrait les lignes horizontales, ainsi que dans le plan-S qui extrait les lignes verticales, cependant, seulement une cellule-S (ou quelques unes) serons activées dans le plan-S qui extrait les angles droits, de même pour les plans-S qui extraient les fins de lignes. On remarque que l'influence des cellules-S qui extraient les segments de lignes sera

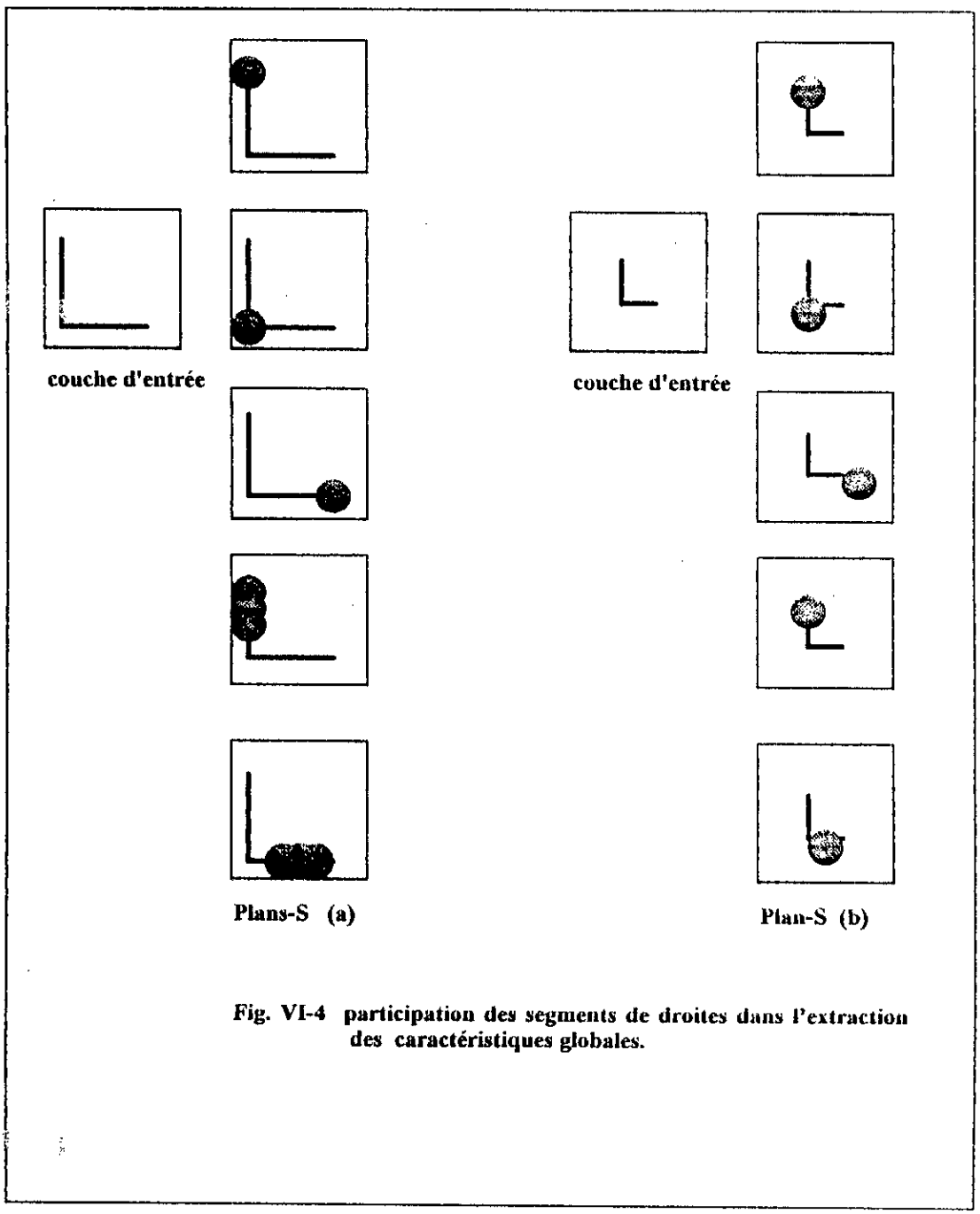


Fig. VI-4 participation des segments de droites dans l'extraction des caractéristiques globales.

d'autant plus forte que les segments de lignes en question sont longs (voir Fig. VI-4).

Cette tendance, rend la reconnaissance des caractères, indépendamment de leurs tailles, difficile.

Cependant il suffit de détecter la présence des deux bouts de la ligne, et aucune autre caractéristique entre eux. C'est pour cela que les couches U_{S2} et U_{S3} sont entraînées à extraire d'autres caractéristiques que les segments de lignes, l'extraction des segments de lignes est faite par la couche U_{S1}.

3.3.2. Exemples d'apprentissage.

3.3.2.1. Exemples d'apprentissage pour la couche U_{s1}.

La couche U_{s1} est entraînée pour extraire, les segments de droites dans différentes orientations. La Fig. IV-5, montre les 12 exemples d'apprentissage utilisés pour cette couche. Chaque exemple est présenté au réseau une seule fois. La cellule au centre du plan-S est toujours sélectionnée comme étant la cellule gagnante. Comme on peut le voir sur la Fig. IV-2, chaque cellule de cette couche a une aire de perception de

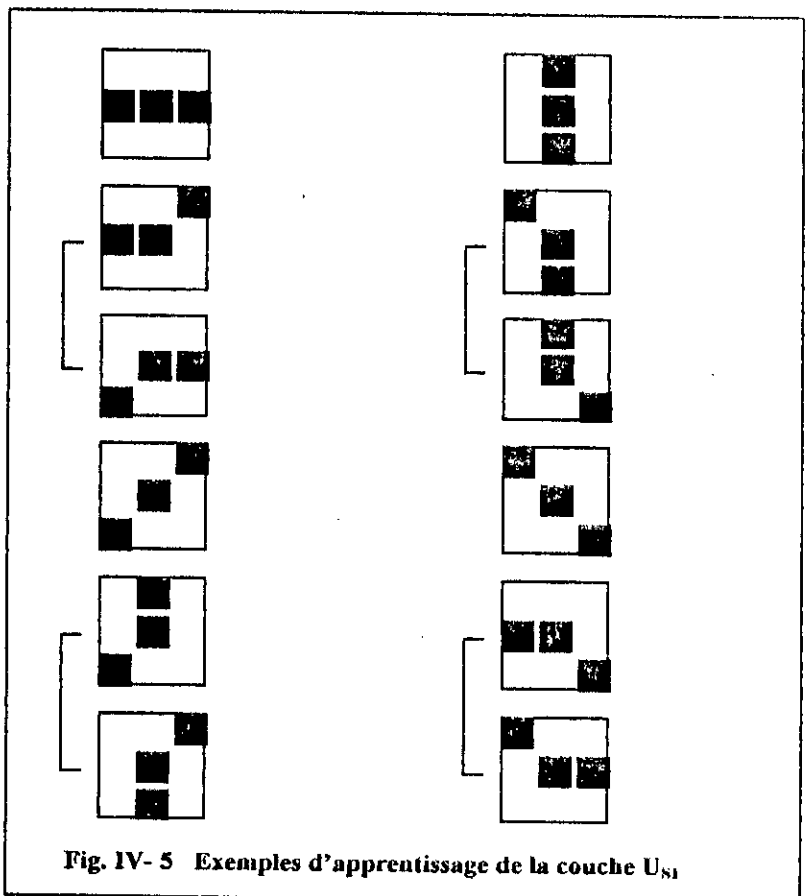


Fig. IV- 5 Exemples d'apprentissage de la couche U_{S1}

dimension (3 x 3), ainsi la matrice centrale (montrée dans la Fig. IV-5) de dimension (3 x 3) sera la seule à contribuer effectivement pendant l'apprentissage.

Puisque la dimension du champ de perception n'est que de (3 x 3), il est difficile d'extraire toutes les parties d'une ligne de pente, 1: 2 par exemple , avec un seul plan, c'est pour cela que deux plans sont utilisés pour extraire de telles lignes, les sorties des deux plans sont regroupées et convergent vers une seule cellule-C. Dans la Fig. IV-5, les crochets indiquent les plans-S qui sont regroupés ensembles.

La sélectivité des cellules-S dans l'extraction des caractéristiques est contrôlée par l'efficacité des entrées inhibitrices. Cette efficacité est déterminée par le paramètre $r_1(k)$, dans l'équation EQ. VI-1. Une grande valeur de $r_1(k)$, correspond à une petite tolérance aux bruits et déformations.

Nous avons choisi $r_1(k) = 1.7$,

3.3.2.2. Exemples d'apprentissage pour la couche U_{s2}.

La Fig. IV-6, montre les 80 exemples d'apprentissage utilisés pour l'entraînement de la couche U_{S2} . La matrice centrale de dimension (9 x 9) est montrée, caractères le champ de perception des cellules-S de cette couche est de (9 x 9). Comme pour la couche précédente, la cellule au centre du plan-S est toujours sélectionnée comme étant la cellule gagnante,

Comme on peut le voir sur la Fig. IV-6, ces exemples d'apprentissage sont des portions de caractères déformées, c'est-à-dire ceux susceptibles d'apparaître au cours de l'utilisation du réseau.

Quelques plans de cette couche sont entraînés avec plusieurs exemples, pour augmenter la capacité des cellules-S à extraire des caractéristiques déformées.

Dans la reconnaissance de caractères, plusieurs formes différentes doivent être traitées comme une seule caractéristique, si cette différence est trop importante, on confie l'extraction des caractéristiques à plusieurs plans-S, les réponses des cellules-S ainsi obtenues convergent vers le même plan-C.

Les crochets dans la Fig. IV-6 indiquent les plans-S qui sont regroupés ensembles.

Le paramètre $r_2(k)$ est choisi égale à 4.0 .

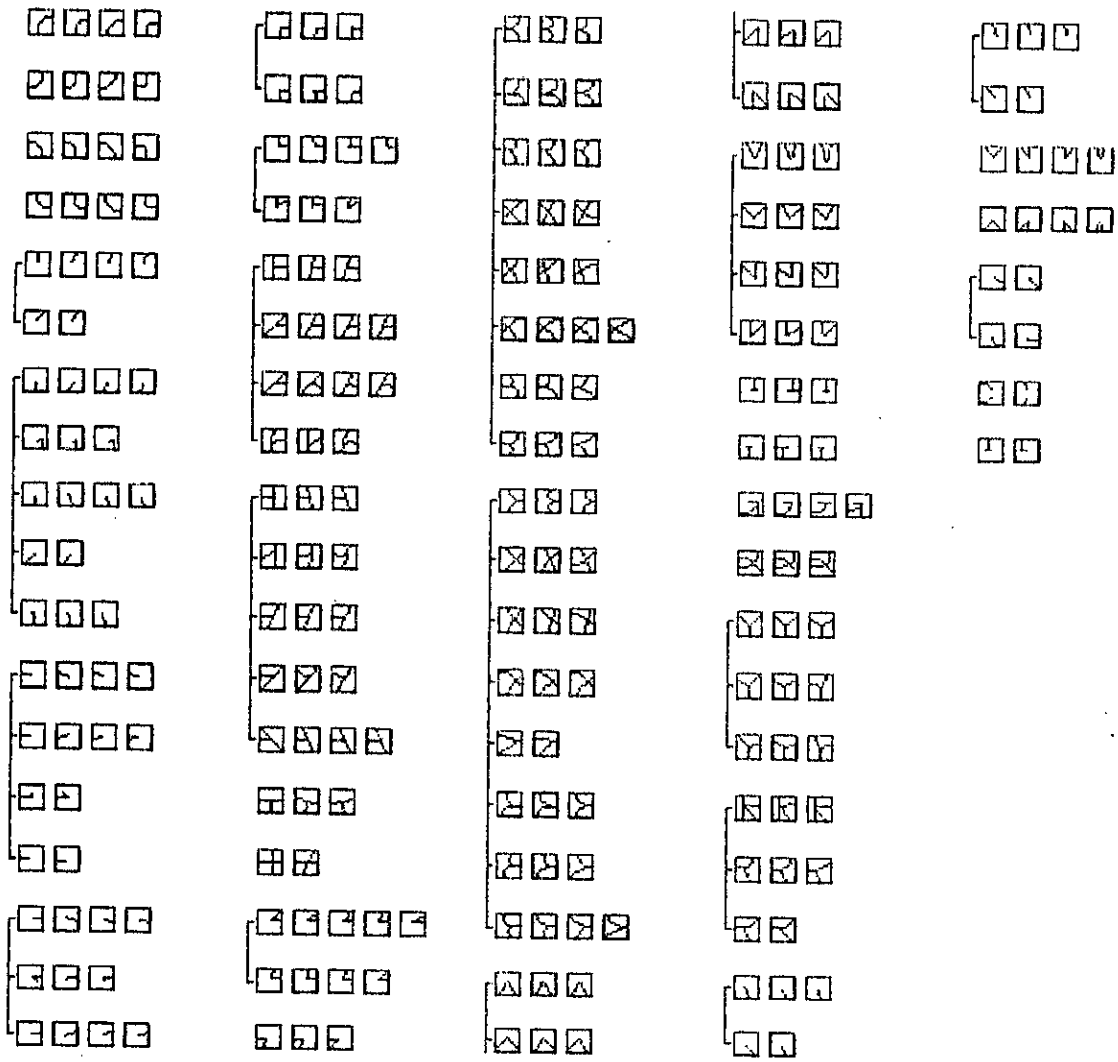


Fig. IV-6 Exemples d'apprentissage de la de U_{s2}

3.3.2.3. Exemples d'apprentissage pour la couche U_{S3}.

La couche U_{S3} extrait les caractéristiques globales, en combinant les caractéristiques locales extraites dans la couche précédente. La Fig. IV-7 montre les 97 exemples d'apprentissage utilisés dans cette couche.

L'aire de perception des cellules-S de cette couche est plus grande que la rétine, on ne peut plus choisir le centre des plans-S comme cellule gagnante. La position de la cellule gagnante est marquée d'une croix.

Le paramètre $r_3(k)$ est choisi égal à 1.5 .

3.3.2.4. Exemples d'apprentissage pour la couche U_{S4}.

La Fig. IV-8 montre les 47 exemples utilisés pour l'entraînement des 47 plans de la couche U_{S4} .

Comme pour les couches U_{S1} et U_{S2} , la cellule centrale de chaque plan-S est la cellule gagnante , le paramètre $r_3(k)$ est choisi égal à 1 .

Quelques caractères ont différents styles d'écriture, qui sont difficiles à détecter par un seul plan. C'est pourquoi plusieurs plans-S détectent le même caractère. Les réponses des cellules-S ainsi obtenues, sont regroupées et convergent vers le même plan-C.

3.4. Réponse du réseau.

Ce réseau est simulé sur un PC équipé d'un processeur PENTIUM à 75 Mhz. Le programme est écrit en langage C (Borland C++ version 4), le listing du programme et donné à l'Annexe A.

Nous avons utilisé la même technique que dans le chapitre III (§ 4), pour introduire les exemples d'apprentissage, ainsi la rétine est enregistrée dans un fichier texte que notre programme chargera. Un seul caractère à la fois, est présenté au réseau.

Les temps d'apprentissage pour les différentes couches sont présentés dans le tableau suivant :

Première couche	moins d'une seconde
Deuxième couche	≅ 120 min.
Troisième couche	≅ 60 min.
Quatrième couche	≅ 30 min.
Total	≅ 310 min.

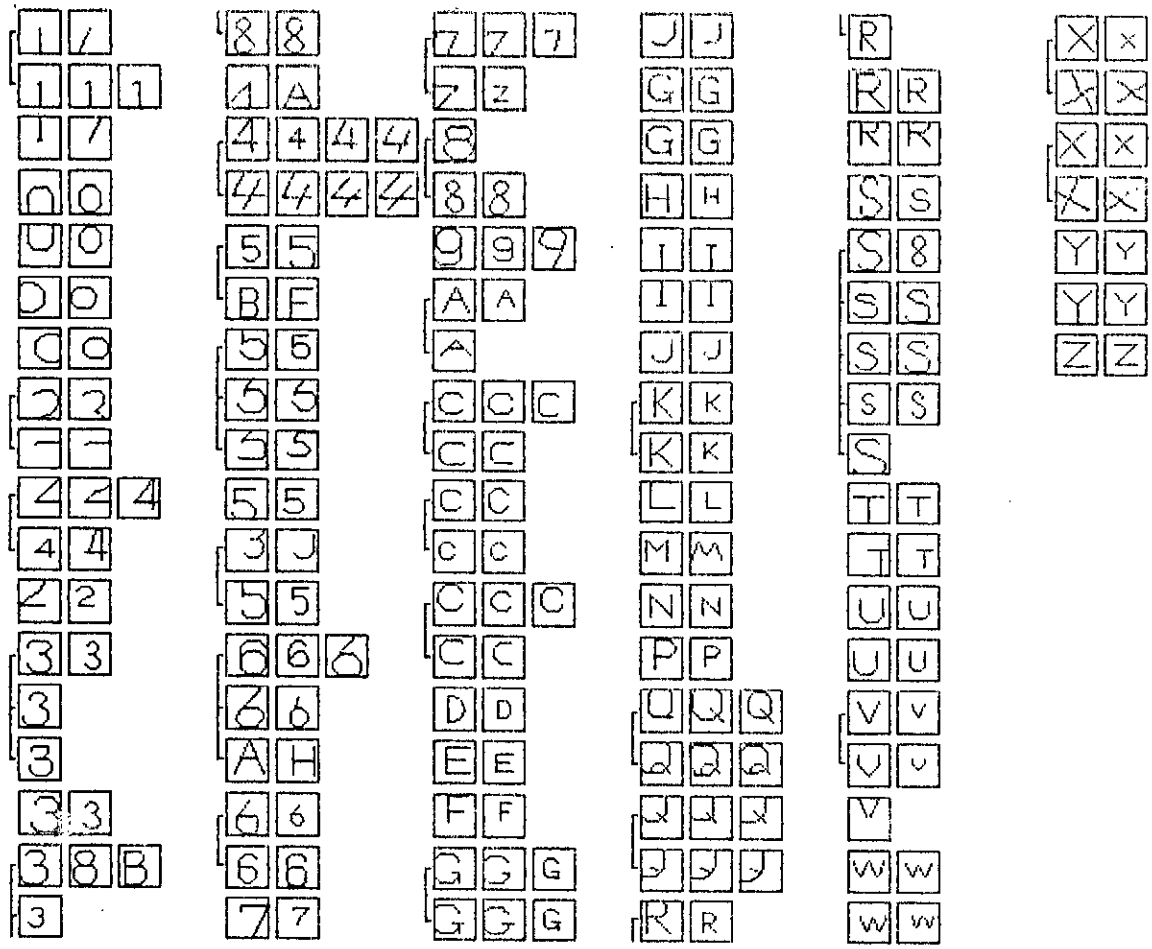


Fig. IV-7 Exemples d'apprentissage de la couche U s3

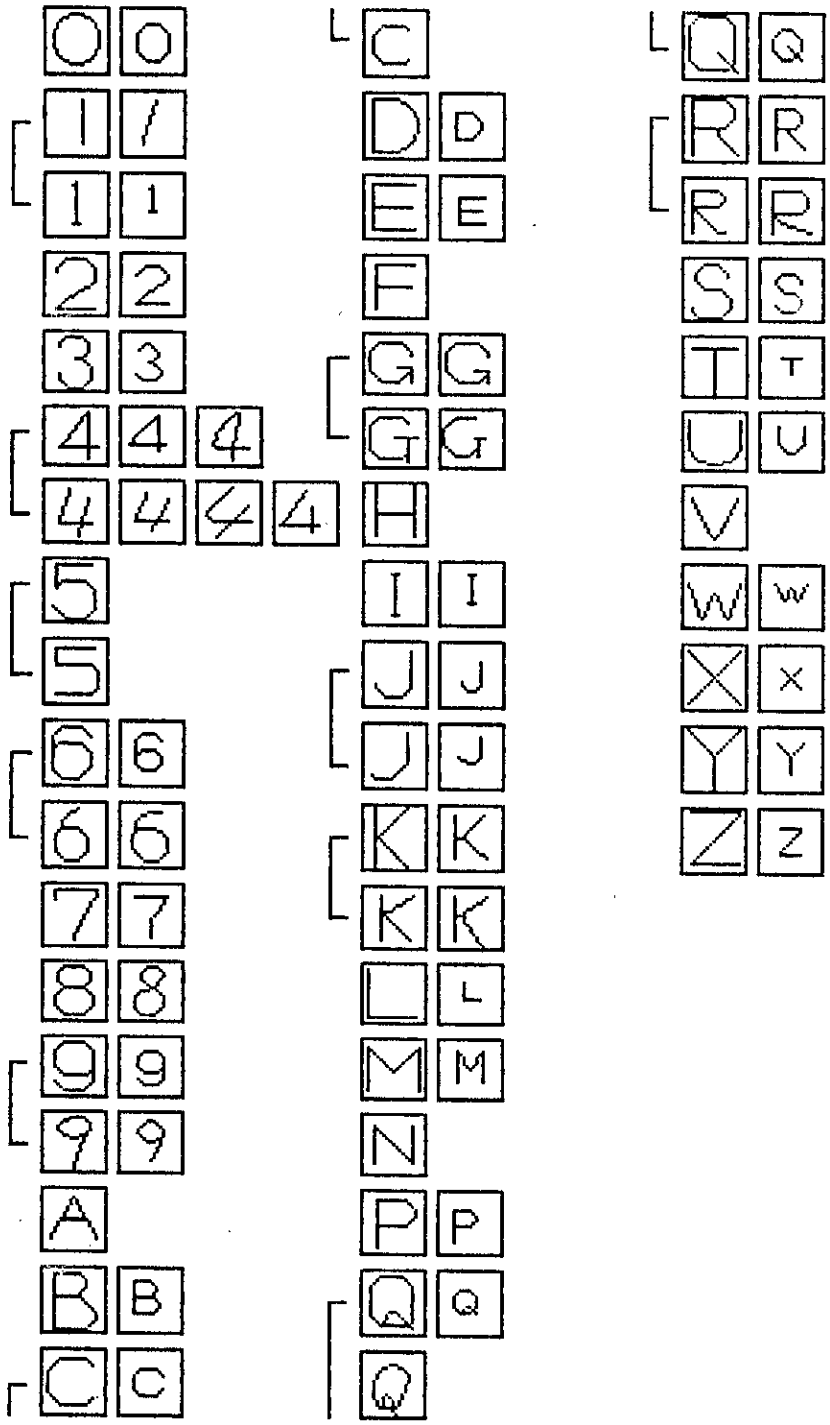


Fig. IV-8 exemples d'apprentissage de la couche U

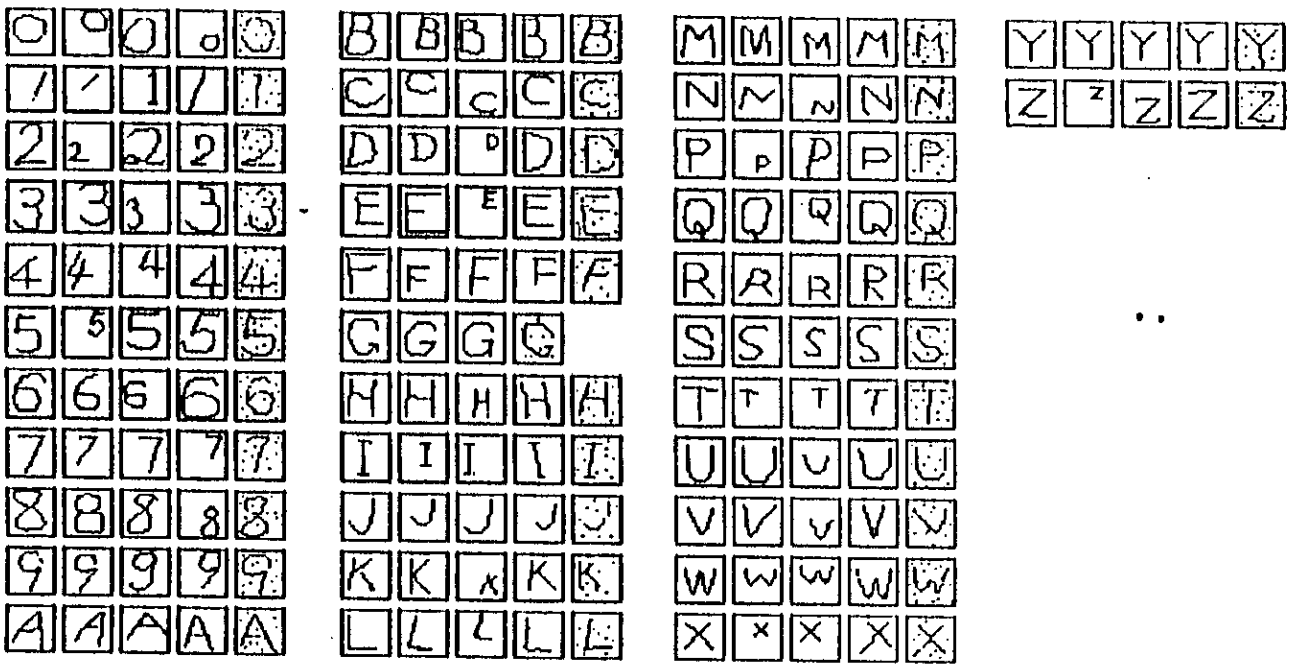


Fig. IV-9 Base de test du Neocognitron

Le temps de réponse du réseau est très long, il est de 30 secondes (c'est dû au grand nombre de neurones , qui est de 70045 neurones).

On peut utiliser un microprocesseur spécialisé, pour accélérer la reconnaissance, comme c'est le cas dans [LEC 1989].

La Fig. IV- 9 montre les caractères que le Neocognitron a reconnus avec succès, on remarque qu'il est insensible au bruit.

3.5. Conclusion.

Le Neocognitron est sans doute le réseau le mieux adapté pour la reconnaissance de caractères, Il est capable de reconnaître des caractères déformés et de différentes tailles, le temps de calcul ne doit pas diminuer de l'intérêt de ce réseau, car il existe des technologies adaptées à ça structure ; les neuro-ordinateurs, circuits intégrés spécialisés, circuits à impulsions,...,etc.

Annexe A.

1. Listing du programme simulant le Neocognitron.

```
#include <stdio.h>
#include <alloc.h>
#include <stdlib.h>
#include <conio.h>
#include <math.h>
#include <time.h>
#define CALCWS3 "c:\\langages\\tc\\dat\\p3.dat"
#define CALCWS4 "c:\\langages\\tc\\dat\\p4.dat"
#define WS1FILE "c:\\langages\\tc\\dat\\poids1.dat"
#define WS2FILE "c:\\langages\\tc\\dat\\poids2.dat"
#define WS3FILE "c:\\langages\\tc\\dat\\poids3.dat"
#define WS4FILE "c:\\langages\\tc\\dat\\poids4.dat"
#define EXEFILE "c:\\exe\\exemple.nna"
#define ERR {printf("\a erreur dans allocation u0  \n");getch();exit(1);}
#define US1 19
#define US2 21
#define UC2 13
#define US3 13
#define UC3 7
#define US4 3
#define NEX  5
#define LAR  9

void alou(void);
void alou3(void);
void alou4(void);

void charge_s1poids(void);
void charge_s2poids(void);
void charge_s3poids(void);
void charge_s4poids(void);
```

```
void charge_retinc(void);

float caluv(int x0,int y0,int kdeb,int kfin ,int couhe,int pas,int lig);
float calus(int x1,int y1,int k1,int kdeb,int kfin,int couche,int pas,int lig);
float caluc(int x2,int y2,int lmin,int lmax,int couche,int pas2,int lig);


typedef struct      {float ws1[3][3]; float b;} rec;
typedef struct      {float ws2[8*5][5]; float b2;} rec2;
typedef struct      {float ws3[33*5][5]; float b3;} rec3;
typedef struct      {float ws4[64*5][5]; float b4;} rec4;
typedef struct      {char calcws3[33][97]; }cws3;
typedef struct       {char calcws4[64][47];}cws4;

rec poids;
rec2 poids2;
rec3 poids3;
rec4 poids4;
cws3  sof;
cws4 sof4;

char  (huge *u0) [US1];
float (huge *us1)  [US1];
float (huge *uv1)  [US1];
float (huge *wv1)  [3];
float (huge *tabws1)  [3];
float (huge *tabb1)  [1];
float (huge *uc1)  [US2];
float (huge *wc1)  [3];

float (huge *uv2)  [US2];
float (huge *wv2)  [5];
float (huge *tabws2)  [80*5];
float (huge *tabb2)  [1];
float (huge *us2) [US2];
float (huge *wc2) [7];

float (huge *wv3) [5];
```

```
float (huge *uv3) [US3];
float (huge *tabws3) [97*5];
float (huge *tabb3)  [1];
float (huge *us3) [US3];
float (huge *wc3) [5];

float (huge *wv4) [5];
float (huge *uv4) [3];
float (huge *tabws4) [47*5];
float (huge *tabb4)  [1];
float (huge *us4) [3];



float temp[13*35][13];


char corres1[8][2] = { {0,0},{1,2},{3,3},{4,5},{6,6},{7,8},{9,9},{10,11} };
char corres2[33][2]= { {0 ,0 },{1 ,1 },{2 ,2 },{3 ,3 },{4,5},{610},{11,14},
                {15,17},{18,19},{20,21},{22,25},   {26,30},{31,31} ,
                {32,32},{33,34},{35,35},{36,43},{44,51},{52,55},
                {56,59},{60,60},{61,61},{62,62},{63,63}, {64,66},
        {67,69},{70,71},{72,73},{74,74},{75,75},{76,77},
        {78,78},{79,79}, };
char corres3[64][2]={{0 ,1 },{2 ,2 },{3 ,3 },{4 ,4 },{5 ,5 },{6 ,6 },{7 ,8 },{9 ,10},{11,11},{12,14},
                        {15,15},{16,18}, 19,19},{20,21},{22,23},{24,26},{27,27},{28,29},{30,32},
                        {33,34},{35,35},{36,37},{38,39},{40,40},   {41,42},{43,44},{45,46},{47,48},
                        {49,49},{50,50},{51,51},{52,53},{54,54},{55,55},{56,56},{57,57},     {58,58},
                        {59,59},{60,60},{61,62},{63,63},{64,64},{65,65},{66,66},{67,68},{69,70},
                        {71,72},{73,73}, {74,74},{75,75},{76,80},{81,81},{82,82},{83,83},{84,84},
                        {85,86},{87,87},{88,88},{89,89},{90,91},   {92,93},{94,94},{95,95},{96,96}} ;


char corres4[47][2]={{0 ,0 },{1 ,2 },{3 ,3 },{4 ,4 },{5 ,6 },{7 ,8 },{9
10},{11,11},{12,12},{13,14},{15,15},
        {16,16},{17,18},{19,19},{20,20},{21,21},{22,23},{24,24},{25,25},{26,27},{28,29},
                {30,30},{31,31},{32,32},{33,33},{34,36},{37,38},{39,39},{40,40},{41,41},{42,42},
                {43,43},{44,44},{45,45},{46,46}} ;


char in[81][6];
float r1=1.7,r2=4.0,r3=1.5,r4=1.0,r11,r22,r33,r44;
char chi;
void main(void)
{
char i,j,k;
clock_t start1,end1, start3,end3, start4,end4;
r11=(r1/(r1+1));r22=(r2/(r2+1));r33=(r3/(r3+1));r44=(r4/(r4+1));
```

```
textmode(C4350);
clrscr();
start1=clock();
alou();
charge_retine();
charge_s1poids();
for (i=1;i<81;i++)
      { for(j=1;j<6;j++)
        {if (in[i][j]==0) in[i][j]=9;          }
      }
//*****************************UV1************************
printf("\n couche 1 en cour \n");
for (i=0;i<US1;i++)
      { for (j=0;j<US1;j++)
      {uv1[i][j] = caluv(i,j,0,1,1,1,US1);}
      }
farfree((void *) wv1);
//*****************************US1************************
for (k=0;k<12;k++)
 {
      for  (i=0;i<US1;i++)
        {
              for (j=0;j<US1;j++)
      { us1[((k*US1)+i)][j] = calus(i,j,k,0,1,1,1,US1); }
        }
 }
farfree((void *) u0);
farfree((void *) uv1);
farfree((void *) tabws1);
farfree((void *) tabb1);
//*****************************UC1************************
for (k=0;k<8;k++)
 {for (i=0;i<US2;i++)
      {for (j=0;j<US2;j++)
      {uc1[((k*US2)+i)][j]= caluc(i-1,j-1,corres1[k][0],corres1[k][1],1,1,US1);}
        }
 }
for (k=0;k<8;k++)
```

```
{
printf("plan %d , une touche pour cintinuer , <q> pour annuler \n",k+1);
      chi=getch();
      if (chi=='q') break;
     for (i=0;i<21;i++)
      {for (j=0;j<21;j++)
     {printf("%1.0f ",10*uc1[((k*21)+i)][j]);  }
        printf("\n");
      }
}
farfree((void *) us1);
farfree((void *) wc1);
printf("couche 1 terminée \n");
//*****************************UV2*************************
printf("\ncouche 2 en cour\n");
for (i=0;i<US2;i++)
{   for (j=0;j<US2;j++)
      { uv2[i][j] = (caluv(i,j,0,8,2,2,US2)); }
}
farfree((void *)wv2);
//*****************************US2*************************
charge_s2poids();
for (k=0;k<80;k++)
 {for (i=0;i<US2;i++)
     { for (j=0;j<US2;j++)
       { {if ((k==39)|| (k==40)|| (k==41)|| (k==45)|| (k==46)|| (k==47)|| (k==48))
          r2=3.8;
          else
           r2=4.0;}
         us2[(k*US2)+i][j] = calus(i,j,k,0,8,2,2,US2);
       }
      }
  }
farfree((void *)uv2);
farfree((void *)uc1);
farfree((void *)tabws2);
farfree((void *)tabb2);
//*****************************UC2*************************
```

```
for (k=0;k<33;k++)
{for (i=0;i<UC2;i++)
   { for (j=0;j<UC2;j++)
      {temp[(k*UC2)+i][j] = caluc(i+4,j+4,corres2[k][0],corres2[k][1],2,3,US2);  }
   }
}
for (k=0;k<4;k++)
{ printf("plan %d , une touche pour continuer , <q> pour annuler \n",k+1);
 chi=getch();
 if (chi=='q') break;
for (i=0;i<UC2;i++)
   { for (j=0;j<UC2;j++)
     {printf("%1.0f ",10*temp[(k*UC2)+i][j]);            }
      printf("\n");
     }
}
farfree((void *)wc2);
farfree((void *)us2);

printf("couche 2  terminee \n");
end1=clock();
printf ("temps écoulé pour couhe1 + couche2 : %f \n ",(end1-start1)/CLK_TCK);
//*****************************************uv3****************
start3=clock();
printf("\ncouche 3 en cour\n");
alou3();
for (i=0;i<US3;i++)
 {  for (j=0;j<US3;j++)
      { uv3[i][j] = caluv(i,j,0,33,3,2,UC2); }
 }
//************************************us3************************
charge_s3poids();
for (k=0;k<97;k++)
  {for (i=0;i<US3;i++)
      { for (j=0;j<US3;j++)
          {us3[(k*US3)+i][j] = calus(i,j,k,0,33,3,2,UC2);}
      }
  }
```

```
//*************************************uc3************************
for (k=0;k<64;k++)
{
for (i=0;i<7;i++)
 { for (j=0;j<7;j++)
      {temp[(k*7)+i][j] = caluc(i+3,j+3,corres3[k][0],corres3[k][1],3,2,US3);}
 }
}
for (k=0;k<64;k++)
{ printf("plan %d , une touche pour continuer , <q> pour annuler \n",k+1);
 chi=getch();
 if (chi=='q') break;
 for (i=0;i<UC3;i++)
   { for (j=0;j<UC3;j++)
       {printf("%1.0f ",10*temp[(k*UC3)+i][j]);}
     printf("\n");
 }
}
farfree((void *)wv3);
farfree((void *)uv3);
farfree((void *)tabws3);
farfree((void *)tabb3);
farfree((void *)us3);
farfree((void *)wc3);
printf("couche 3  termineé \n");
end3=clock();
printf ("temps écoulé pour couche3 : %f \n ",(end3-start3)/CLK_TCK);
//*****************************UV4
start4=clock();
alou4();
printf("\ncouche 4 en cour\n");
for (i=0;i<3;i++)
  { for (j=0;j<3;j++)
      { uv4[i][j] = caluv(i+2,j+2,0,64,4,2,7); }
  }
//*****************************US4*********************************
charge_s4poids();
for (k=0;k<47;k++)
```

```
 {for (i=0;i<US4;i++)
     { for (j=0;j<US4;j++)
         {us4[(k*US4)+i][j] = calus(i+2,j+2,k,0,64,4,2,UC3);}
     }
 }
//*****************************************UC4*****************
for (k=0;k<35;k++)
 {temp[k][0] = caluc(1,1,corres4[k][0],corres4[k][1],4,1,3); }
for (k=0;k<35;k++)
{ printf("reponse  du nerone  %d ",k);
 printf("     %1.5f\n",  temp[k][0]);
}



farfree((void *)wv4);
farfree((void *)uv4);
farfree((void *)tabws4);
farfree((void *)tabb4);
farfree((void *)us4);
printf("\ncouche 4  terminee \n");
end4=clock();
printf ("temps écoulé pour couche4 : %f \n ",(end4-start4)/CLK_TCK);
printf ("temps écoulé total : %f \n ",((end4-start4)+(end3-start3)+(end1-start1))/CLK_TCK);
printf("\n fin\n");
chi=getch();
textmode(LASTMODE);
}//*****************************fin de main************************



//*********************CALCUL UV*****************************************
float caluv (int x0, int y0,int kdeb,int kfin,int couche,int d0,int lig)
{
float x=0;
int i0,j0,k0;
```

```
for (k0=kdeb;k0<kfin;k0++)
  {for (i0=-d0;i0<d0+1;i0++)
    {for (j0=-d0;j0<d0+1;j0++)
      { if( ((x0+i0)>=0) && ((y0+j0)>=0) && ((x0+i0)<lig)  && ((y0+j0)<lig) )
            switch(couche)
            {
            case  1:
            if ((u0[x0+i0][y0+j0])==0) break;
            x+=((wv1[i0+d0][j0+d0])*(u0[x0+i0][y0+j0]));
            break;
            case  2:
            if ((uc1[(k0*US2)+x0+i0][y0+j0])==0) break;

      x+=((wv2[i0+d0][j0+d0])*(uc1[(k0*US2)+x0+i0][y0+j0])*(uc1[(k0*US2)+x0+i0][y0+j0]) );
            break;
            case 3:
            if ((temp[(k0*US3)+x0+i0][y0+j0])==0) break;

      x+=((wv3[i0+d0][j0+d0])*(temp[(k0*US3)+x0+i0][y0+j0])*(temp[(k0*US3)+x0+i0][y0+j0])
);
            break;
            case 4:
            if ((temp[(k0*7)+x0+i0][y0+j0])==0) break;

      x+=((wv4[i0+d0][j0+d0])*(temp[(k0*7)+x0+i0][y0+j0])*(temp[(k0*7)+x0+i0][y0+j0]) );
            break;
        }
      }
    }
  }
      return(sqrt(x));
  }
  //*******************CALCUL DE US*****************************************
  float calus(int x1,int y1,int k1,int kdeb,int kfin,int couche,int d0,int lig)
  {
  int i0,j0,k0;
  float x=0.0;
  for (k0=kdeb;k0<kfin;k0++)
    { if ( (couche==2) && (((k0+1)==in[k1+1][1]) ||((k0+1)==in[k1+1][2]) ||((k0+1)==in[k1+1][3])
                                                ||((k0+1)==in[k1+1][4]) ||((k0+1)==in[k1+1][5]))
)
```

```
        continue;
        if ( (couche==3) && (sof.calcws3[k0][k1]==1)  )  continue;
        if ( (couche==4) && (sof4.calcws4[k0][k1]==1)  )  continue;
        {for (i0=-d0;i0<d0+1;i0++)
         {  for (j0=-d0;j0<d0+1;j0++)
                { if( ((x1+i0)>=0) && ((y1+j0)>=0) && ((x1+i0)<lig)  && ((y1+j0)<lig) )
                switch(couche)
                 {
                 case 1:
                 if( (tabws1[k1*3+i0+d0][j0+d0]==0.0)  || (u0[x1+i0][y1+j0]==0) )break;
                 x+=tabws1[k1*3+i0+d0][j0+d0]*u0[x1+i0][y1+j0];
                 break;
                 case 2:
                 if( (tabws2[k0*5+i0+d0][k1*5+j0+d0]==0.0) || (uc1[k0*US2+x1+i0][y1+j0]==0.0) )
                 break;
                 x+=tabws2[k0*5+i0+d0][k1*5+j0+d0]*uc1[k0*US2+x1+i0][y1+j0] ;
                 break;
                 case 3:
                 if( (tabws3[k0*5+i0+d0][k1*5+j0+d0]==0.0) || (temp[k0*US3+x1+i0][y1+j0]==0.0)
)                break;
                 x+=tabws3[k0*5+i0+d0][k1*5+j0+d0]*temp[k0*US3+x1+i0][y1+j0];
                 break;
                 case 4:
                 if( (tabws4[k0*5+i0+d0][k1*5+j0+d0]==0.0) || (temp[k0*7+x1+i0][y1+j0]==0.0) )
break;
                 x+=tabws4[k0*5+i0+d0][k1*5+j0+d0]*temp[k0*7+x1+i0][y1+j0];
                 break;
                 }
               }
            }
        }
    }
     switch(couche)
     {
     case 1:
     if((uv1[x1][y1])==0.0) break;
     x=r1*( ( (x+1)/ ( 1+r11*tabb1[k1][0]*uv1[x1][y1]) )-1   );
     break;
     case 2:
     if((uv2[x1][y1])==0.0) break;
```

```
x =r2*( ( (x +1)/ ( 1+r22*tabb2[k1][0]*uv2[x1][y1]) ) -1  );
break;
case 3:
if((uv3[x1][y1])==0.0) break;
x =r3*( ( (x +1)/ ( 1+r33*tabb3[k1][0]*uv3[x1][y1]) ) -1  );
break;
case 4:
if((uv4[x1-2][y1-2])==0) break;
x =r4*( ( (x +1)/ ( 1+r44*tabb4[k1][0]*uv4[x1-2][y1-2]) ) -1  );
break;
}
if (x<0.0)   x=0.0;
return(x);
}
//********************CALCUL DE UC**************************************
float caluc(int x2,int y2,int lmin,int lmax,int couche,int d0,int lig)
{
int i1,j1,ind0;
float x=0.0;
for (ind0=lmin;ind0<lmax+1;ind0++)
 {for (i1=-d0;i1<d0+1;i1++)
    {for (j1=-d0;j1<d0+1;j1++)
     { if( ((x2+i1)>=0) && ((y2+j1)>=0) && ((x2+i1)<lig)  && ((y2+j1)<lig) )
       switch(couche)
       {
       case 1:
       if (us1[ind0*US1+x2+i1][y2+j1]==0.0) break;
       x+=wc1[i1+d0][j1+d0]*us1[ind0*US1+x2+i1][y2+j1];
       break;
       case 2:
       if (us2[ind0*US2+x2+i1][y2+j1]==0.0) break;
       x+=wc2[i1+d0][j1+d0]*us2[ind0*US2+x2+i1][y2+j1];
       break;
       case 3:
       if (us3[ind0*US3+x2+i1][y2+j1]==0.0) break;
       x+=wc3[i1+d0][j1+d0]*us3[ind0*US3+x2+i1][y2+j1];
       break;
       case 4:
```

```
        if ((us4[(ind0*US4)+(x2+i1)][(y2+j1)])==0) break;
        x+=us4[ind0*3+x2+i1][y2+j1];
        break;
        }
       }
      }
  }
  if (x<0.0)  x=0.0;
  x=x/(x+1);
  return(x);
}
//**********************FONCTION ALOU*************************************
void alou(void)
{
if(((char *)u0=(char *)farcalloc(sizeof(*u0),US1))==NULL)          ERR;
if(((float *)us1=(float *)farcalloc(sizeof(*us1),US1*12))==NULL)    ERR;
if(((float *)uv1=(float *)farcalloc(sizeof(*uv1),US1))==NULL)       ERR;
if(((float *)wv1=(float *)farcalloc(sizeof(*wv1),3))==NULL)         ERR;
if(((float *)uc1=(float *)farcalloc(sizeof(*uc1),US2*8))==NULL)     ERR;
if(((float *)wc1=(float *)farcalloc(sizeof(*wc1),3))==NULL)         ERR;
if(((float *)uv2=(float *)farcalloc(sizeof(*uv2),US2))==NULL)       ERR;
if(((float *)wv2=(float *)farcalloc(sizeof(*wv2),5))==NULL)         ERR;
if(((float *)tabws1=(float *)farcalloc(sizeof(*tabws1),12*3))==NULL)  ERR;
if(((float *)tabb1=(float *)farcalloc(sizeof(*tabb1),12))==NULL)    ERR;
if(((float *)tabws2=(float *)farcalloc(sizeof(*tabws2),8*5))==NULL)  ERR;
if(((float *)tabb2=(float *)farcalloc(sizeof(*tabb2),80))==NULL)    ERR;
if(((float *)us2=(float *)farcalloc(sizeof(*us2),80*US2))==NULL)    ERR;
if(((float *)wc2=(float *)farcalloc(sizeof(*wc2),7))==NULL)         ERR;

wv1[1][1] = 1.0;
wv1[0][1] =wv1[1][0] =wv1[1][2] =wv1[2][1] =0.9;
wv1[0][0] =wv1[0][2] =wv1[2][0] =wv1[2][2] = 0.8615;

wc1[1][1] = 4.0;
wc1[0][1] =wc1[1][0] =wc1[1][2] =wc1[2][1] =3.6;
wc1[0][0] =wc1[0][2] =wc1[2][0] =wc1[2][2] = 3.446;
```

```
wv2[2][2] = 1.0;
wv2[1][2] =wv2[2][1] =wv2[2][3] =wv2[3][2] =0.9 ;
wv2[1][1] =wv2[1][3] =wv2[3][1] =wv2[3][3] =0.8616 ;
wv2[0][2] =wv2[2][0] =wv2[2][4] =wv2[4][2] =0.81 ;
wv2[0][0] =wv2[0][4] =wv2[4][0] =wv2[4][4] =0.7423 ;
wv2[0][1] =wv2[0][3] =wv2[1][0] =wv2[1][4] = 0.7901 ;
wv2[3][0] =wv2[3][4] =wv2[4][1] =wv2[4][3] = 0.7901 ;

wc2[0][0] =wc2[0][6] =wc2[6][0] =wc2[6][6] =1.552;
wc2[0][1] =wc2[0][5] =wc2[1][0] =wc2[1][6] =1.7891;
wc2[5][0] =wc2[5][6] =wc2[6][1] =wc2[6][5] =1.7891;
wc2[0][2] =wc2[0][4] =wc2[2][0] =wc2[2][6] =1.9752;
wc2[4][0] =wc2[4][6] =wc2[6][2] =wc2[6][4] =1.9752;
wc2[1][2] =wc2[1][4] =wc2[2][1] =wc2[2][5] =2.429;
wc2[4][1] =wc2[4][5] =wc2[5][2] =wc2[5][4] =2.429;
wc2[1][1] =wc2[1][5] =wc2[5][1] =wc2[5][5] =2.1279;
wc2[1][3] =wc2[3][1] =wc2[3][5] =wc2[5][3] =2.56;
wc2[0][3] =wc2[3][0] =wc2[3][6] =wc2[6][3] =2.048;
wc2[2][2] =wc2[2][4] =wc2[4][2] =wc2[4][4] =2.9175;
wc2[2][3] =wc2[3][2] =wc2[3][4] =wc2[4][3] =3.2;
wc2[3][3] =4.0;

in[5][1]=in[6][1]=in[7][1]=in[9][1]=in[10][1]=in[57][1]=in[71][1]=in[72][1]=in[73][1]=in[74][1]=
in[79] [1]=1;
in[9][2]=in[71][2]=in[72][2]=in[73][2]=in[74][2]=2;
in[3][1]=in[4][1]=in[9][3]=in[11][1]=in[12][1]=in[15][1] =in[16][1] =in[27][1] =in[31][1] =in[71][3]
in[72][3]=in[73][3]=in[74][3]=in[77][1]=in[78][1]=3;
in[12][2]=in[15][2]=in[16][2]=4;
in[12][3]=in[13][1]=in[15][3]=in[16][3]=in[18][1]=in[49][1]=5;
in[13][2]=in[15][4]=in[18][2]=6;
in[1][1]=in[2][1]=in[5][2]=in[6][2]=in[7][2]=in[8][1]=in[10][2]=in[13][3]=in[15][5]=in[18][3]= 7;
in[22][1]=in[23][1]=in[24][1]=in[29][1]=in[33][1]=in[34][1 ]=in[35][1 ]=in[36][1]=in[63][1]=7;
in[5][3]=in[6][3]=in[7][3]=in[10][3]=8;
}

//*****************************alou3*********************************
void alou3(void)
```

```
wv2[2][2] = 1.0;
wv2[1][2] =wv2[2][1] =wv2[2][3] =wv2[3][2] =0.9 ;
wv2[1][1] =wv2[1][3] =wv2[3][1] =wv2[3][3] =0.8616 ;
wv2[0][2] =wv2[2][0] =wv2[2][4] =wv2[4][2] =0.81 ;
wv2[0][0] =wv2[0][4] =wv2[4][0] =wv2[4][4] =0.7423 ;
wv2[0][1] =wv2[0][3] =wv2[1][0] =wv2[1][4] = 0.7901 ;
wv2[3][0] =wv2[3][4] =wv2[4][1] =wv2[4][3] = 0.7901 ;

wc2[0][0] =wc2[0][6] =wc2[6][0] =wc2[6][6] =1.552;
wc2[0][1] =wc2[0][5] =wc2[1][0] =wc2[1][6] =1.7891;
wc2[5][0] =wc2[5][6] =wc2[6][1] =wc2[6][5] =1.7891;
wc2[0][2] =wc2[0][4] =wc2[2][0] =wc2[2][6] =1.9752;
wc2[4][0] =wc2[4][6] =wc2[6][2] =wc2[6][4] =1.9752;
wc2[1][2] =wc2[1][4] =wc2[2][1] =wc2[2][5] =2.429;
wc2[4][1] =wc2[4][5] =wc2[5][2] =wc2[5][4] =2.429;
wc2[1][1] =wc2[1][5] =wc2[5][1] =wc2[5][5] =2.1279;
wc2[1][3] =wc2[3][1] =wc2[3][5] =wc2[5][3] =2.56;
wc2[0][3] =wc2[3][0] =wc2[3][6] =wc2[6][3] =2.048;
wc2[2][2] =wc2[2][4] =wc2[4][2] =wc2[4][4] =2.9175;
wc2[2][3] =wc2[3][2] =wc2[3][4] =wc2[4][3] =3.2;
wc2[3][3] =4.0;

in[5][1]=in[6][1]=in[7][1]=in[9][1]=in[10][1]=in[57][1]=in[71][1]=in[72][1]=in[73][1]=in[74][1]=
in[79] [1]=1;
in[9][2]=in[71][2]=in[72][2]=in[73][2]=in[74][2]=2;
in[3][1]=in[4][1]=in[9][3]=in[11][1]=in[12][1]=in[15][1]=in[16][1]=in[27][1]=in[31][1]=in[71][3]=
in[72][3]=in[73][3]=in[74][3]=in[77][1]=in[78][1]=3;
in[12][2]=in[15][2]=in[16][2]=4;
in[12][3]=in[13][1]=in[15][3]=in[16][3]=in[18][1]=in[49][1]=5;
in[13][2]=in[15][4]=in[18][2]=6;
in[1][1]=in[2][1]=in[5][2]=in[6][2]=in[7][2]=in[8][1]=in[10][2]=in[13][3]=in[15][5]=in[18][3]= 7;
in[22][1]=in[23][1]=in[24][1]=in[29][1]=in[33][1]=in[34][1 ]=in[35][1 ]=in[36][1]=in[63][1]=7;
in[5][3]=in[6][3]=in[7][3]=in[10][3]=8;
}

//*****************************alou3*********************************
void alou3(void)
```

```
{
if(((float *)wv3=(float *)farcalloc(sizeof(*wv3),5))==NULL)          ERR;
if(((float *)uv3=(float *)farcalloc(sizeof(*uv3),US3))==NULL)        ERR;
if(((float *)tabws3=(float *)farcalloc(sizeof(*tabws3),33*5))==NULL)  ERR;
if(((float *)tabb3=(float *)farcalloc(sizeof(*tabb3),97))==NULL)     ERR;
if(((float *)us3=(float *)farcalloc(sizeof(*us3),US3*97))==NULL)     ERR;
if(((float *)wc3=(float *)farcalloc(sizeof(*wc3),5))==NULL)          ERR;

wv3[2][2] = 1.0;
wv3[1][2] =wv3[2][1] =wv3[2][3] =wv3[3][2] =0.9 ;
wv3[1][1] =wv3[1][3] =wv3[3][1] =wv3[3][3] =0.8616 ;
wv3[0][2] =wv3[2][0] =wv3[2][4] =wv3[4][2] =0.81 ;
wv3[0][0] =wv3[0][4] =wv3[4][0] =wv3[4][4] =0.7423 ;
wv3[0][1] =wv3[0][3] =wv3[1][0] =wv3[1][4] = 0.7901 ;
wv3[3][0] =wv3[3][4] =wv3[4][1] =wv3[4][3] = 0.7901 ;

wc3[2][2] = 2.5;
wc3[1][2] =wc3[2][1] =wc3[2][3] =wc3[3][2] =1.75 ;
wc3[1][1] =wc3[1][3] =wc3[3][1] =wc3[3][3] =1.5096 ;
wc3[0][2] =wc3[2][0] =wc3[2][4] =wc3[4][2] =1.225 ;
wc3[0][0] =wc3[0][4] =wc3[4][0] =wc3[4][4] =0.9116 ;
wc3[0][1] =wc3[0][3] =wc3[1][0] =wc3[1][4] = 1.1261 ;
wc3[3][0] =wc3[3][4] =wc3[4][1] =wc3[4][3] = 1.1261 ;
}
//*********************FONCTION ALOU4*****************************************
void alou4(void)
{
if(((float *)uv4=(float *)farcalloc(sizeof(*uv4),3))==NULL)          ERR;
if(((float *)wv4=(float *)farcalloc(sizeof(*wv4),5))==NULL)          ERR;
if(((float *)tabws4=(float *)farcalloc(sizeof(*tabws4),64*5))==NULL)         ERR;
if(((float *)tabb4=(float *)farcalloc(sizeof(*tabb4),47))==NULL)         ERR;
if(((float *)us4=(float *)farcalloc(sizeof(*us4),3*47))==NULL)           ERR;
wv4[2][2] = 1.0;
wv4[1][2] =wv4[2][1] =wv4[2][3] =wv4[3][2] =0.8 ;
wv4[1][1] =wv4[1][3] =wv4[3][1] =wv4[3][3] =0.7294 ;
wv4[0][2] =wv4[2][0] =wv4[2][4] =wv4[4][2] =0.81 ;
wv4[0][0] =wv4[0][4] =wv4[4][0] =wv4[4][4] =0.64 ;
wv4[0][1] =wv4[0][3] =wv4[1][0] =wv4[1][4] = 0.60716 ;
```

```
wv4[3][0] =wv4[3][4] =wv4[4][1] =wv4[4][3] = 0.60716 ;
}
//*******************FONCTION CHARGE WS1**********************************
void charge_s1poids(void)
{
int i,j,k, l = sizeof(poids);
FILE *fpt;
k=0;

if  ((fpt = fopen(WS1FILE,"r")) == NULL )
      printf("fichier %s non trouv,",WS1FILE);
      else
       {
                while (fread(&poids,l,1,fpt))
                        {
                        for (i=0;i<3;i++)
                                {for (j=0;j<3;j++)
                                   {  tabws1[(k*3)+i][j]=poids.ws1[i][j]; }
                                }
                        tabb1[k][0]=poids.b;
                         k++;
                         }
      }
fclose(fpt);
}
//************************FONCTION CHARGE RETINE*****************************
void charge_retine(void)
{
 int i=0,j=0;
 FILE *fpl;
 char *c =" " ;
 fpl=fopen(EXEFILE,"r");
 do
 {*c=getc(fpl);
  if ((*c != EOF) && ((*c == '1' ) || (*c == '0')))
   {
      {
              {if (j==US1)
```

```
                    { i++;
                     j=0;
                    }
                u0[i][j] = atoi(c);
                j++;
                }
        }
  }
} while( (*c != EOF) );
fclose(fpl);
 for (i=0;i<US1;i++)
   {for (j=0;j<US1;j++)
      {
       if (u0[i][j] == 1)
        chi = '\xDB ';
         else
         chi = '\xB0';
         printf("%c",chi);
       }
    printf("\n");
   }
}
//************************charge_s2poids*****************************
void charge_s2poids(void)
{FILE *fpt1;
int i,j,k,g,l=sizeof(poids2);
k=0;
if  ((fpt1 = fopen(WS2FILE,"rb")) == NULL )
printf("fichier %s non trouv,",WS2FILE);
else
{  for (g=0;g<80;g++)
      { fread(&poids2,l,1,fpt1);
        tabb2[g][0]=poids2.b2 ;
        for (k=0;k<8;k++)
          {
              {for (i=0;i<5;i++)
                    {for (j=0;j<5;j++)
                          {
```

```
                                tabws2[(k*5)+i][(g*5)+j]=poids2.ws2[(k*5)+i][j];
                                }
                        }
                }
        }


}
fclose(fpt1);
}
}
//*************************charge_s3poids*****************************
void charge_s3poids(void)
{
FILE *fpt3,*fpt33;
int i,j,k,g,l=sizeof(poids3);
char touche3=' ';
k=0;
if  ((fpt3 = fopen(WS3FILE,"rb")) == NULL )
printf("fichier %s non trouve,",WS3FILE);
else
    {for (g=0;g<97;g++)
            {
            fread(&poids3,l,1,fpt3);
            tabb3[g][0]=poids3.b3 ;
            for (k=0;k<33;k++)
                    {for (i=0;i<5;i++)
                            {for (j=0;j<5;j++)
                                    {tabws3[(k*5)+i][(g*5)+j]=poids3.ws3[(k*5)+i][j]; }
                            }

                    }
            }
    }
fclose(fpt3);
if  ((fpt33 = fopen(CALCWS3,"rb")) == NULL )
printf("fichier %s non trouvé\n",CALCWS3);
else
```

```
        {fread(&sof,sizeof(sof),1,fpt33);}
fclose(fpt33);
 }
//************************charge_s4poids*****************************
void charge_s4poids(void)
{
FILE *tpt4,*fpt44;
int i,j,k,g,l=sizeof(poids4);
char touche3=' ';
k=0;
if  ((tpt4 = fopen(WS4FILE,"rb")) == NULL )
printf("fichier %s non trouve,",WS4FILE);
else
      {for (g=0;g<47;g++)
              {fread(&poids4,l,1,tpt4);
               tabb4[g][0]=poids4.b4 ;
                for (k=0;k<64;k++)
                      {for (i=0;i<5;i++)
                              {for (j=0;j<5;j++)
                                      {tabws4[(k*5)+i][(g*5)+j]=poids4.ws4[(k*5)+i][j]; }
                               }

                      }
       }
}
fclose(tpt4);
if  ((fpt44 = fopen(CALCWS4,"rb")) == NULL )
printf("fichier %s non trouve\n",CALCWS4);
else
 {
  fread(&sof4,sizeof(sof4),1,fpt44);
 }
fclose(fpt44);
 }
```

Références bibliographiques.

[BAB 1989] BABA (N.). « A new approch for finding the global minimum of error function of neural networks. ». Neural Networks, Vol. 2,pp. 367-373, 1989

[DAV 1992] DAVALO (E.), NAïM (P.). « Des réseaux de neuronnes. ». EYROLLES, 1990.

[FAU 1990] FAURE. (C.), MICLET. (L.). « Reconnaissance des formes. ». Technique de L'Ingénieur, Traité informatique, H 1920, pp. 1-10, 1990.

[FRE 1992] FREEMAN (A.), Skapura (M.). « Neural networks, algorithms, applications and programming techniques .». Addison-Wesley Publishing Company, 1992.

[FUK 1991] FUKUSHIMA (K.), NOBUKI (W.). « Handwritten alphanumeric character recognition by the Neocognitron.». IEEE Tran. On Neural Networks, Vol.2, N° 3,pp.355-365, May 1991.

[KHO 1988] KHOTANZARD (A.), LU (J.). « Distortion invariant character recognition by a multi-layer Perceptron and Back-Propagation learning. ». Electrical Enginearing Departement Southern Methodist University Dallas, Texas, 1988.

[KOS 1992] KOSKO. (B.). « Neural networks and fuzzy systems. ». Prectice-Hall, Inc., 1992.

[LEC 1989] LE CUN (Y.), JAKLEY (L.). « Handwritten digit recognitron : Application of neural network chips and automatic learning. ». IEEE Communications Magazine,pp.41-46,1989.

[PAN 1987] PANINI. (P.). « Le livre de l'homme. ». Edition P.G Rouge et Or, 1987.

[PER 1986] PERONNAZ (L.). « Etude de réseaux de neuronnes formels : Conception, propriétés et applications. ». Thèse de Doctorat d'Etat ès Sciences physiques, 1986.

[REM 1991] REMY (C.). « Lecture optique de caractères . ». Technique de L'Ingénieur, Traité informatique, H 1405, pp. 1-10, 1991.

[SOU 1990] SOUCHIER (C.). « Analyse d'image ». Technique de L'Ingénieur, Traité Analyse chimique et Caractérisation, P 855, pp. 1-18, 1990.

[WEI 1993] WEINFELD (M.). « Résaux de neuronnes. ». Technique de L'Ingénieur, Traité informatique, H 1 990, pp. 1-16, 1993.