MASTER EN SCIENCES INFORMATIQUES

Développement et évaluation d'un système de reconnaissance de cellules basé sur l'approche neuronale

van der Kaa, Vincent

Award date:
1995

Awarding institution:
Universite de Namur

FACULTES
UNIVERSITAIRES
N.D. DE LA PAIX
NAMUR

INSTITUT D'INFORMATIQUE

Développement et évaluation d'un système de reconnaissance de cellules basé sur l'approche neuronale

Vincent van der Kaa

Promoteur : Professeur M. Noirhomme
Co-Promoteur : Professeur J. Remacle

Mémoire présenté en vue de
l'obtention du titre de :
Licencié en Informatique

Année Académique 1994 - 1995

Remerciements

Je tiens à exprimer mes plus vifs remerciements à madame M. Noirhomme et monsieur J. Remacle, promoteur et co-promoteur de ce mémoire, pour l'aide et le soutien qu'ils m'ont apportés tout au long de ce travail.

Un tout grand merci à madame A. De Baenst pour sa disponibilité et les nombreux conseils qu'elle m'a dispensés.

Ma gratitude s'adresse également aux membres du laboratoire de Biochimie et plus particulièrement à mesdames M. Raes, A. Houbion, à mademoiselle M. Botman et à messieurs O. Toussaint et M. Dieu; l'intérêt qu'ils ont porté à ce travail, leurs services et leurs remarques m'ont été très précieux.

A messieurs J-P. Rasson, G. Huys et à mademoiselle F. Montaigne, appartenant à la faculté de Mathématique, pour leur assistance et leurs conseils.

A messieurs K. Bertels et L. Neuberg, appartenant à la faculté des Sciences Economiques, pour avoir partagé leur expérience en matière de réseaux de neurones.

A messieurs R. Mairesse et R.V. Cotet pour leur disponibilité.

Enfin, à tous ceux qui de près ou de loin, ont contribué à la réalisation de ce mémoire, veuillez trouver ici l'expression de mes remerciements les plus chaleureux.

Résumé

Les biologistes de l'unité de biochimie des F.U.N.D.P. de Namur développent et évaluent des théories sur le vieillissement cellulaire, sur base de cultures de fibroblastes. A cette fin, il est nécessaire de classifier ces cellules.

Dans le cadre de ce mémoire, nous nous proposons de développer et d'étudier la faisabilité d'un système automatique de reconnaissance de ces cellules. Pour ce système, nous avons opté pour la technique des réseaux de neurones et plus particulièrement pour l'algorithme de 'rétropropagation'. Cet algorithme est déjà utilisé dans divers domaines dont la reconnaissance de caractères.

Après une description du contexte biologique, une énumération des phases d'un système de reconnaissance d'images et une introduction théorique aux réseaux de neurones, nous décrirons le système de reconnaissance de cellules que nous avons développé. Ensuite, nous évaluerons la capacité de ce système à reconnaître les cellules. Nous constaterons que les résultats sont encourageants, nous réaliserons une comparaison avec une méthode statistique (l'analyse discriminante) et nous proposerons quelques recommandations pour l'implémentation du système et la poursuite de cette recherche.

Abstract

The biologists in the Department of Biochemistry of the University of Namur develop and study theories about cellular aging, based on human fibroblasts in culture. It's therefore a necessity to classify the cells.

Within the framework of our dissertation, we intend to develop and to study the feasibility of an automatic system to recognize cells. For this system, we decided upon the neural networks technique and particularly the 'back-propagation' algorithm. This algorithm is already used in various applications such as optical characters' recognition.

After having described the biologic situation, listed the phases of an image recognition system and made a theoric introduction to neural networks, we will describe the cells recognition system that we developed. Afterwards, we will test the capacity of the system to recognize cells. After analysis, results turn out to be encouraging. We will thereafter draw a comparison with a statistical method (discriminant analysis) and make some recommandations as regards the implementation of the system.

Table des matières

Introduction

Depuis l'apparition des hiéroglyphes et surtout depuis l'invention du premier alphabet (Ugarit, Syrie, il y 4 mille ans), l'être humain n'a cessé de développer son habileté à gérer des symboles et des abstractions - les mots en sont un exemple. Et pourtant, nous ne les utilisons pas tout le temps. L'information sensorielle, qui ne requiert pas une activité mentale élevée pour l'apprécier, reste généralement plus efficace que les mots, les colonnes de chiffres ou les formules.
Pour illustrer cette idée nous pouvons prendre deux exemples: n'est-il pas «plus aisé» de lire une bande dessinée par rapport à un texte? Et, n'est-on pas capable de mémoriser des centaines de visages, et de les remettre en situation alors qu'une multiplication à plus de six chiffres ou la mémorisation de quelques numéros de téléphone est considérée généralement comme un calvaire.

L'information de nos sens et plus particulièrement la vision reste donc une source importante de renseignements sur notre environnement. Or, l'informatique a, jusqu'il y a peu, sous-estimé cet aspect; le système d'exploitation MS-DOS et son interface homme-machine en est une illustration. Pour représenter cette information visuelle, nous utiliserons dorénavant le terme « image » qui sera défini ultérieurement. Mais pour l'instant, nous dirons qu'une image est une représentation digitale d'une partie du monde réel.
Ce terme fait référence aux nouveaux systèmes d'exploitation: Windows 95, X Window, ... On parle de graphiques, d'outils de présentation mais aussi de traitement d'images. Parmi toutes les possibilités proposées par les techniques de traitement d'images, le domaine de la reconnaissance de celles-ci semble le plus délicat et est de plus en plus étudié. Deux raisons à cela!

- Au niveau fondamental: un ordinateur pourrait-il égaler l'énorme capacité de perception et surtout d'interprétation visuelle de l'homme telles que la reconnaissance de milliers de visages et la capacité de lire des écritures différentes, pour ne citer que ces deux exemples.
- Au niveau appliqué: la reconnaissance de caractères (Optical Caracter Recognition) manuscrits ou typés, la détection de mauvaises conformations de produits sortant d'une unité de fabrication, la reconnaissance d'avions pour la défense aérienne [Reid, 94], ... sont des domaines pour lesquels l'informatique est requis dans le souci d'une automatisation et donc d'une augmentation de l'efficacité.

Autre domaine où l'image a son importance: la biologie cellulaire, pour laquelle la reconnaissance de cellules constitue une étape clé, notamment en validation de théories.

Dans le cadre de ce mémoire, notre propos sera de définir le cadre d'un système expert de reconnaissance de cellules.
Dans un premier temps, sera décrit le problème biologique et les objectifs de notre recherche. Ensuite, sera proposée une architecture théorique d'un système classique

d'analyse et de reconnaissance d'images comprenant quatre phases: acquisition d'images, prétraitement, analyse et reconnaissance d'images. Pour cette dernière phase, différentes méthodes sont actuellement disponibles et nous décrirons plus particulièrement l'approche 'réseaux de neurones' sur laquelle s'est posé notre choix.
Les traitements automatiques conçus et utilisés seront alors décrits en détails et nous rédigerons un bref compte-rendu sur l'architecture de ce système.
Quant aux résultats, ils présenteront l'ensemble des tests réalisés, dans le but d'évaluer les trois principales sources de 'bruits' à savoir:

- la structure du réseau de neurones
- les techniques de prétraitement
- le choix de l'entraînement

La méthode des réseaux de neurones sera, alors, comparée à une méthode statistique classique: l'analyse discriminante.
Enfin, nous statuerons sur la faisabilité d'un tel système, les inconvénients et avantages de la méthode et énoncerons quelques perspectives.

1. Le contexte biologique

1.1 Définition de la cellule

La cellule est l'unité fondamentale, morphologique et fonctionnelle de tout organisme vivant. Les cellules sont composées d'une membrane cytoplasmique limitant le cytosol au sein duquel se trouve les organites (noyau, vacuoles, mitochondries, reticulum endoplasmique, lysosomes, appareil de golgi).
Chacun des organites possède une fonction particulière. Parmi ceux-ci, le noyau et les mitochondries ont des rôles essentiels pour les cellules. Le premier, qui contient l'ADN (Acide désoxyribonucléique) cellulaire ou génome, est le centre de contrôle et les secondes constituent la source principale de production d'énergie sous forme d'ATP (Adénosine triphosphate).

1.2 Le vieillissement cellulaire

Afin d'expliquer le vieillissement de la cellule, différentes théories ont été proposées. D'une part, la théorie qui considère le vieillissement comme un processus programmé par le génome et d'autre part, la théorie stochastique qui insiste sur l'importance des événements environnementaux (stress) qui conduisent à des altérations irréversibles et donc au vieillissement de la cellule. La théorie thermodynamique du vieillissement, développée au laboratoire de biochimie cellulaire des facultés universitaires de Namur, propose une approche globale du vieillissement cellulaire et dans cette perspective, fait transparaître les aspects génétiques et stochastiques du processus de vieillissement.
Cette théorie explique que la cellule, au cours de sa vie, passe par différents états et que chacun de ceux-ci est caractérisé par une phase stationnaire, avec un certain niveau d'erreurs. A la suite d'une diminution de réactions de la cellule, le niveau d'erreurs, lié à des stress de toute nature, augmente. Cette accumulation fait passer graduellement la cellule d'un état à un autre pour arriver finalement à un seuil critique où la cellule dégénère et meurt.
Dans cette théorie, on porte surtout l'attention sur deux aspects: d'une part, l'évolution du niveau des erreurs et d'autre part, la production d'énergie dans les cellules et son utilisation pour les multiples fonctions cellulaires. Les systèmes de défense, de protection et de réparation doivent être considérés dans ce processus car ils préviennent l'accumulation des erreurs et évitent une mort précoce de la cellule.

1.3 Le modèle expérimental du vieillissement sur les fibroblastes

Pour valider cette théorie et étudier l'influence du stress sur le vieillissement, ce modèle a été testé sur des populations de cellules cultivées in vitro, les fibroblastes [Toussaint, 92].
Les fibroblastes sont des cellules du tissu conjonctif, support structural et métabolique des autres organes et tissus du corps humain (sang, muscles,...)

Différentes sources de la littérature ont montré que les fibroblastes passent progressivement d'un type de morphologie à un autre et que ce passage se fait toujours dans un ordre bien précis. Les trois premiers types MF I, II, III sont appelés mitotiques (cellules en division), et les quatre derniers types sont appelés postmitotiques PMF IV, V, VI et VII. Ces caractéristiques se retrouvent de façon similaire au cours du vieillissement normal ou après un vieillissement accéléré sous l'effet de stress et sont décrites par des critères morphologiques et biochimiques.

1.4 Descriptions morphologiques et biochimiques des fibroblastes

1.4.1 Caractéristiques morphologiques

Grâce aux techniques microscopiques classiques (microscope optique), on peut observer précisément la forme des fibroblastes (figure 1.1).
Les morphotypes MF I sont des petites cellules en forme de fuseau et très réfringentes en microscopie de phase, les morphotypes MF II sont des petites cellules épithéloïdes grossièrement rectangulaires et les morphotypes MF III sont des cellules plus grandes, à l'aspect plus triangulaire. Viennent ensuite les cellules postmitotiques viables (PMF IV à VI). Les morphotypes IV sont des grandes cellules en forme de fuseau dont les contours sont crénelés, les morphotypes V sont des grandes cellules épithéloïdes avec un rapport noyau/cytoplasme plus élevé, critères qui s'accentuent pour les morphotypes VI. Enfin, les morphotypes VII sont des cellules en dégénérescence. Si l'on observe l'évolution des morphotypes MF I, MF II et MF III en fonction du nombre de générations, on constate qu'aux premières générations les cellules de type MF I sont majoritaires mais que progressivement, elles sont remplacées par les morphotypes MF II puis par les cellules MF III qui deviennent majoritaires en fin de culture. Les cellules PMF IV, V, VI et VII ne sont présentes que dans les dernières générations au moment où les cellules ne se divisent plus ou très peu, et leurs proportions augmentent après le dernier passage en culture, lorsque la culture est formée de cellules postmitotiques uniquement.

1.4.2 Caractéristiques biochimiques

La cellule produit une grande variété de protéines. Les gènes contenus dans le noyau sont les modèles permettant de fabriquer les protéines. Celles-ci jouent parfois un rôle structural mais leur principale fonction est de catalyser des réactions chimiques (la protéine est alors appelée enzyme). Certaines protéines jouent un rôle dans le vieillissement cellulaire; elles agissent comme un système de défense.
L'analyse de l'expression protéique des cellules, suivant un processus de différenciation, peut se faire à l'aide des électrophorèses en gel 2D. Cette technique permet de séparer, d'identifier et de quantifier un mélange complexe de protéines grâce à la combinaison, dans un premier temps, de l'électrofocalisation et, deuxièmement, d'une électrophorèse sur de grands gels plats, en conditions dénaturantes. La séparation est basée sur des propriétés dépendantes des protéines, à savoir, d'une part, la charge, mise en évidence par le point isoélectrique lors de l'électrophorèse dans la première dimension, et d'autre part le poids moléculaire qui détermine la mobilité des

complexes protéines-SDS dans les gels polyacrylamide, au cours de l'électrophorèse, dans une seconde dimension.

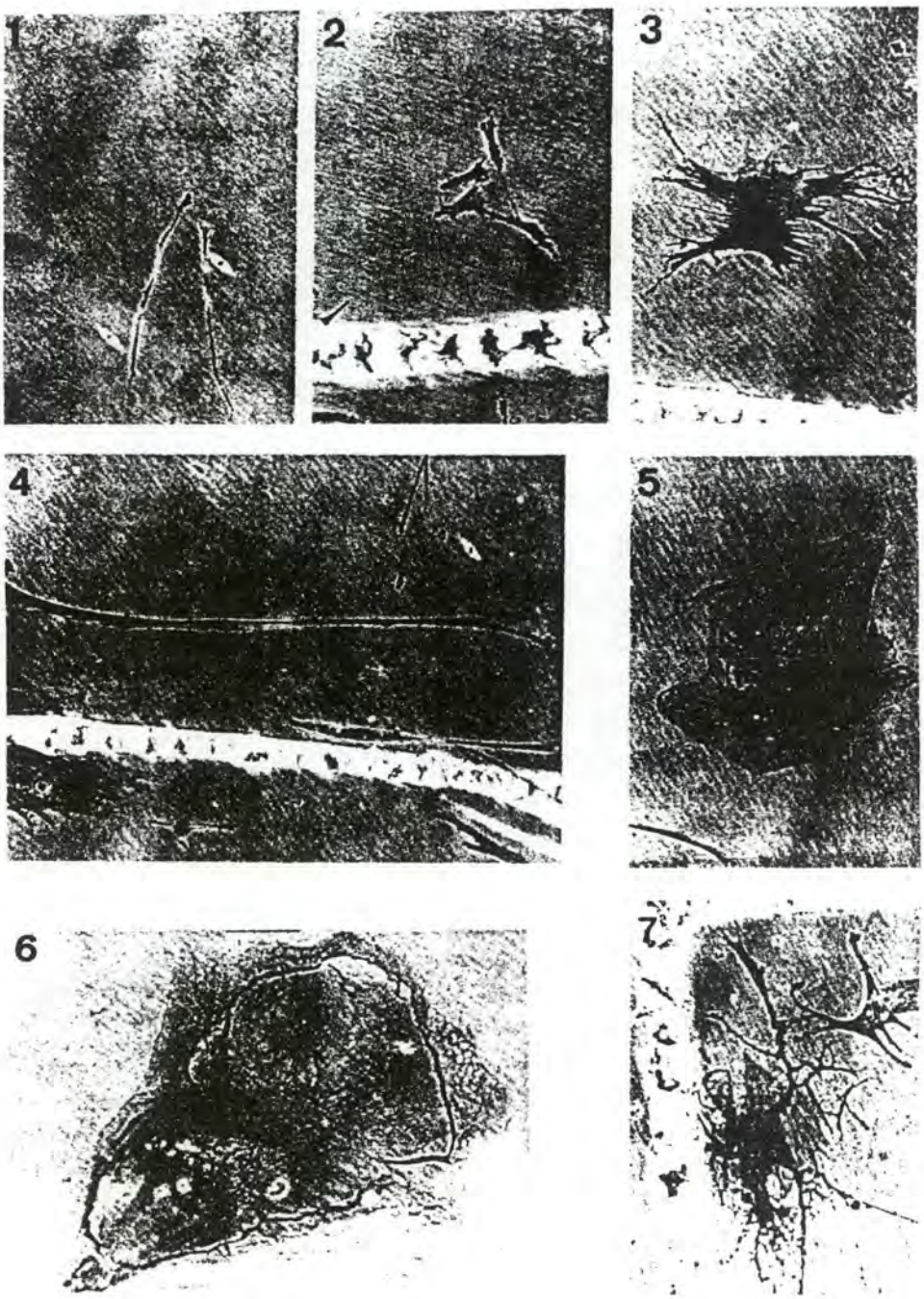


Figure 1.1 Photographie des 7 morphotypes de fibroblaste. (1) MF I, (2) MF II, (3) MF III, (4) PMF IV, (5) PMF V, (6) PMF VI et (7) PMF VII.

Une prévision du modèle théorique est que, lorsque la cellule passe d'un état stationnaire à un autre, elle va s'adapter à un nouvel état en réajustant et en réorganisant son génome, c'est-à-dire en modifiant l'expression de ses gènes. Lors d'un passage vers des cellules de morphotype âgé, par une transition naturelle ou par l'effet de stress, on peut, donc, s'attendre à ce qu'il y ait toute une série de protéines qui soient induites alors que d'autres verraient leur synthèse déprimée ou réprimée.

Chacun des sept morphotypes cellulaires présente des polypeptides spécifiques. Quatorze marqueurs protéiques spécifiques de ces morphotypes ont été trouvés au niveau de la fraction cytoplasmique et nucléaire, vingt-quatre autres faisant partie des protéines membranaires et des protéines sécrétées. Les PMF VII, comprenant les fibroblastes en dégénérescence, montrent des changements de protéines très importants. La nature de ces marqueurs polypeptidiques n'est pas encore parfaitement connue. Grâce à des anticorps monoclonaux dirigés contre ces marqueurs, on espère purifier les polypeptides spécifiques des différents morphotypes.

1.5 But du mémoire

Nous proposons une étude de faisabilité d'un système automatique de reconnaissance des morphotypes de fibroblastes. Ce système permettrait de compter le nombre de représentants de chacun des sept morphotypes à partir d'images procurées par le système SUN-VIEW du laboratoire de biochimie des facultés universitaires Notre-Dame de la Paix.

Nous utiliserons les caractéristiques morphologiques développées dans le paragraphe 1.4. Dorénavant, nous emploierons le terme 'classe' en lieu et place de morphotype ou type cellulaire. De plus, nous circonscrirons notre problème à la reconnaissance de quatre classes afin de faciliter les tests, limiter la complexité du problème mais aussi parce que les classes II, III, IV et V sont les plus fréquemment rencontrées dans les cultures étudiées en biochimie.

Puisque nous souhaitons réaliser un système expert, nous devons travailler en collaboration avec des experts. Madame A. Houbion et Monsieur O. Toussaint partageront leur expérience en matière de reconnaissance de fibroblastes. A cette fin, un interview sera réalisé, afin de cerner les arguments utilisés par les experts pour reconnaître les cellules. De plus, leur collaboration nous sera précieuse pour l'acquisition des images représentant des fibroblastes de classes II, III, IV et V.

De cette rencontre avec les experts, se profilent deux résultats préliminaires:

1. L'existence d'une divergence, de près de 10%, entre les classifications des deux experts

Il est donc illusoire de chercher à dépasser ce résultat! Cependant, les cellules, sources de divergences, ne seront pas utilisées dans les tests préliminaires.

2. Une description informelle des quatre classes de fibroblastes

- classe II: ces cellules sont petites, fréquemment en division et ont un rapport longueur sur largeur de 4 pour 1. Elles se rassemblent souvent en agglomérats et peuvent être pourvues de filaments cytoplasmiques.

- classe III: il en existe beaucoup de formes différentes. Leur taille est assez grande avec une largeur approximativement égale à la longueur. On note la présence de nombreux filaments cytoplasmiques.

- classe IV: les cellules de cette classe sont grandes, longiformes mais avec une épaisseur cytoplasmique.

- classe V: ces cellules sont grandes; leur cytoplasme est strié.

Remarque: Le filament cytoplasmique est un artefact morphologique provoqué par le mouvement des cellules dans leur milieu.

2. La reconnaissance de patterns

2.1 Du pattern à l'image

Le monde réel est constitué d'un ensemble d'objets tels une lampe, Jean Dupont, une fourchette, ... chacun de ceux-ci a un nom qui l'identifie et appartient, au moins, à une classe. Par exemple, des trois objets uniques François, Jean et Titi; les deux premiers adhèrent à la classe homme et le dernier, à la classe canari.
Un pattern est un modèle simplifié d'un objet ou d'une structure, autrement dit, il s'agit d'un ensemble de mesures ou caractéristiques pouvant être représentées par une notation vectorielle ou matricielle [Schalkoff, 89]. Une caractéristique est une mesure ayant une signification particulière mais pouvant aussi être calculée à partir d'autres mesures. Le pattern d'un visage peut être une image de celui-ci, mais il peut aussi être la couleur des yeux, la surface, ... Dans cet exemple, l'ensemble des pixels de l'image constitue un vecteur à d dimensions appelé le vecteur de mesures ou espace de mesures, alors que la couleur des yeux ou la surface du visage est une caractéristique.
Une théorie adéquate permet de classifier les objets en classes. Si celle-ci est valide et que l'on dispose d'un nombre suffisant de mesures (un « bon pattern ») et d'un modèle mathématique correct, un système informatique pourra facilement être programmé pour classifier ces objets. La difficulté vient du fait que les théories sont soit trop complexes, soit trop partielles pour avoir un modèle mathématique complet. Imaginons qu'il soit complet, les mesures dont il doit disposer ne seraient peut être pas disponibles dans leur totalité.

Souvent, nous cherchons un pattern qui est invariant aux changements, soit le pattern idéal. Ces changements sont dus à des causes variées, entre autre le bruit. Dans beaucoup de situations, un ensemble de patterns, appartenant à une même classe, varie fortement dans cette classe.
Par exemple, des caractères manuscrits peuvent être reconnus malgré la large variété d'écriture, ce qui nécessite une analyse approfondie du caractère. Dans ce cas, une tentative de recherche de correspondance s'avère plus difficile à moins de choisir un certain nombre de mesures invariables.

Dans notre étude, le pattern ne sera pas un vecteur de caractéristiques mais un vecteur de mesures: l'image.

Mais comment définir le concept « image »? [Rimmer, 93]
Avant tout, il convient de savoir que notre définition ne prendra pas en compte: les graphiques d'ordinateurs tels que les courbes graphiques, les diagrammes en barres, ... ceux-ci permettant de percevoir les choses de manière visuelle plutôt qu'analytique. La façon dont ils sont construits (programme qui dessine selon des fonctions mathématiques) sort de notre description de l'image.
Le concept « image » sera illustré dans ce présent travail par le bitmap, image représentée sous forme digitale et stockée dans un tableau de pixels pour être affichée sur un écran, imprimée avec une imprimante graphique ou traitée par un programme quelconque. Lorsque nous parlerons d'une image, il s'agira donc toujours d'une image bitmap (soit digitalisée) et non pas de l'image mentale d'une personne. Plutôt que de

manipuler des lignes, des cercles, des objets virtuels formant un dessin, l'image bitmap est une analogie digitale de la façon dont nous voyons les choses mais elle est aussi un vecteur de mesures où chaque mesure correspond à un pixel de l'image. La valeur de cette image représente la codification de l'intensité en niveaux de gris de ce pixel.

Qu'entend-t'on par pixel (Picture Element)?
Sur une image, c'est la plus petite surface homogène; c'est donc, en quelque sorte l'équivalent d'un point. On utilise habituellement ce terme pour définir les caractéristiques de l'affichage en haute résolution. Dans le cas d'une image monochrome, le pixel s'identifie à un point. Dans le cas d'un moniteur couleur, il faut trois points de couleurs différentes (rouge, vert, bleu) pour constituer un pixel.

La plupart des programmes stockent les images avec des formats particuliers de fichiers, très différents et nombreux en informatique. Pour n'en citer que les plus courants: le format MacPaint, le format IMG de Digital Research, le format PCX de PaintBrush, le format GIF de Compuserve et enfin le format TIFF. Une image bitmap d'un format donné doit pouvoir être affichée, imprimée et traitée, par le truchement d'une application. Une image couleur de 1024 X 768 pixels peut rapidement prendre 1 MégaByte en mémoire. Dés lors, on comprend aisément l'intérêt qu'il y a d'avoir un format réduisant la taille mémoire de l'image, gérant les caractéristiques de celle-ci (couleurs et dimension) et utilisable par l'application souhaitée.

2.2 Un cadre théorique pour la reconnaissance d'images

La classification consiste à assigner des données en entrée à une classe préspécifiée, grâce à l'extraction de caractéristiques ou de mesures significatives et grâce au traitement de ces mesures. La reconnaissance est définie simplement comme une capacité à classifier. Comme notre pattern initial est une image, nous proposons de décrire les étapes classiques d'un système de reconnaissance de celle-ci. Ce cadre théorique nous permettra de construire un système de reconnaissance de fibroblastes.
Dans un exemple de reconnaissance d'avions [Reid, 94], les auteurs utilisent les contours des avions représentés sur les images afin de les reconnaître (figure 2.1).

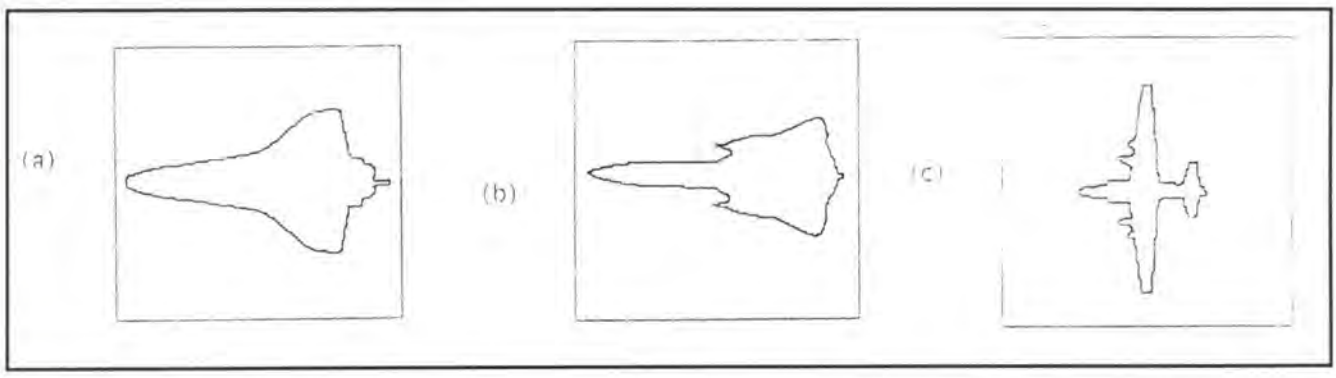


Figure 2.1 Images binaires représentant les contours de (a) une navette spatiale, (b) un SR-71 et (c) un bombardier U-2.

Dans ces conditions, le problème de reconnaissance d'images se restreint à un problème de reconnaissance de formes. Un tel système est classiquement composé de quatre phases [Belaïd, 92]: codage, prétraitement, analyse, reconnaissance.

Comme point de départ, il y a le monde physique ou monde réel c'est-à-dire un espace analogique de dimension infinie appelé espace de formes (F).

2.2.1 Phase I: Codage

Le codage est une opération de conversion numérique du monde physique continu vers un monde numérique discret, appelé espace de représentation (R). La dimension r de cet espace est volontairement grande de manière à pouvoir disposer d'un maximum d'informations sur la forme. Cet espace de représentation R ainsi que ses r dimensions peuvent aussi être représenté par un vecteur de mesures à d dimensions. Le résultat du codage est donc un pattern. Cette transformation permettra de traiter les données par ordinateur.
Pour cela, nous avons besoin d'un capteur et d'un récepteur:

- Le capteur assure la transformation physique d'un objet du monde réel en un signal adapté au système de traitement. Le capteur est, par exemple, un dispositif optique bidimensionnel (le couple microscope-caméra),

- Le récepteur convertit le signal fourni par le capteur en données numériques.
A cette fin, il réalise deux opérations: l'échantillonnage et la quantification. L'échantillonnage permet la représentation de l'amplitude d'un signal à un instant donné. Exemple: 60.000 échantillons par seconde sont nécessaires pour représenter un signal continu tel que le son.
La quantification consiste à donner une valeur numérique à un signal analogique.

2.2.2 Phase II: Prétraitement

Le prétraitement permet de sélectionner, dans l'espace de représentation, l'information nécessaire à l'application; ceci génère l'apparition d'un nouvel espace de représentation R' à r' dimensions.

Cinq types de prétraitements existent [Belaïd, 92]:

- La suppression de bruit vise à supprimer les informations résiduelles venant perturber les données propres à la forme. Le bruit peut être provoqué par les capteurs, les conditions de prise de vue (éclairage, mise au point, calibrage) et l'environnement de la scène: le fond, la composition de la forme, la nature de la matière, les autres objets, ...
La suppression du bruit met en oeuvre des techniques de correction de la non-linéarité du capteur, de lissage (un pixel étant représenté par la valeur moyenne de ses pixels voisins) et de seuillage. Cette dernière fonction consiste à examiner les pixels et à ne retenir que ceux dont la valeur est comprise entre 2 seuils donnés. Pour mémoire, on distingue seuillage global et seuillage local.

- La correction d'erreurs telles que les distorsions géométriques et la variabilité de certains objets.
L'erreur vient du matériel de saisie ou de l'objet lui-même. En effet, le matériel de saisie peut provoquer des distorsions géométriques qui vont modifier la forme de l'objet. Cependant, dans la plupart des applications, cette erreur n'est pas sensible pour l'utilisateur (vu la haute technicité du matériel de prise de vue). Comme il s'agit de donner, au système informatique, une information semblable à celle traitée par un être humain, ce type d'erreur ne doit pas être absolument corrigé.
Une autre source d'erreur vient de la variabilité intrinsèque (ne dépendant pas du système de reconnaissance d'images) de certains objets. La variabilité des boulons sortant d'une chaîne de fabrication est faible, par contre la variabilité des caractères manuscrits est élevée. Or, dans les deux cas, un écart sera perçu comme erreur. Il est inopportun d'utiliser le terme « erreur », et il est indispensable de le remplacer par le terme « variabilité ». De même, cette variabilité ne concerne plus le prétraitement mais doit être traitée dans une phase ultérieure.
La variabilité des objets a, cependant, des limites: ainsi dans des cas extrêmes d'objets pathologiques, on peut accepter le terme « erreur ». Exemple: une personne dont l'écriture n'est pas impeccable pourrait dessiner un 'a' en lieu et place d'un 'o', ce cas pathologique rendrait la réponse du système de reconnaissance fausse. Un tel cas devrait être corrigé par une phase de prétraitement. On imagine, à peine, la complexité de la tâche sans une aide humaine! Ce type d'erreur est le signe d'une théorie lacunaire mais est-il possible d'avoir une théorie parfaite surtout avec des objets aussi variables que l'écriture.
Trois pistes sont envisageables pour résoudre ces difficultés dues aux « erreurs »:
- davantage d'informations sur l'objet, sa variabilité et son environnement (on donnerait en entrée du système un mot entier au lieu d'un caractère)
- une possibilité de répondre de manière probabiliste
- une extension de la théorie

- L'homogénéisation des données a pour but de se débarrasser de l'information redondante, superflue ou inutile pour l'application à réaliser. La fonction de transfert qui relie la grandeur numérique délivrée par le capteur à la grandeur physique mesurée, peut être rectifiée par une table de correction. Il est, ainsi, possible de réduire les effets de non-uniformité spatiale, de sensibilité de capteur et d'éclairement. Le but est d'augmenter le contraste et d'améliorer la netteté des frontières des objets représentés, ce qui est hautement nécessaire pour un système de reconnaissance de formes.
Par exemple, après une étape de seuillage, la binarisation consiste à créer une image binaire où le noir représente toutes les régions de l'objet et le blanc représente le fond.

- La normalisation des données permet de s'affranchir de différentes distorsions de formes. Ainsi, trois transformations géométriques sont rencontrées (figure 2.2).

1. La dilatation (échelle)
2. Le déplacement (translation)
3. La rotation

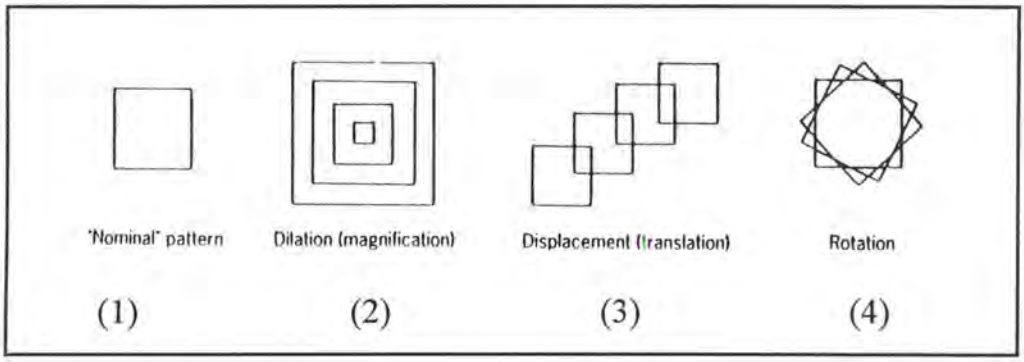


Figure 2.2 Les trois transformations géométriques (1) pattern initial, (2) dilatation, (3) translation et (4) rotation

La normalisation n'est pas toujours triviale et nécessite parfois une réflexion approfondie, dépassant le stade du prétraitement; la distorsion doit alors être traitée à un plus haut niveau, à une autre phase du système de reconnaissance.

- La réduction des données intervient lorsque les données produites par le capteur sont trop importantes, sans être toutes utiles à la reconnaissance. Par exemple, il n'est pas pratique d'utiliser directement toutes les intensités en pixels d'une image comme vecteur de mesures, car une image de 512 x 512 pixels est représentée par un vecteur de 262144 x 1 mesures.

Ces prétraitements visent à améliorer la qualité de l'image sans toutefois rechercher l'image parfaite. Le but est de reconnaître l'image et donc de poursuivre le prétraitement afin de faciliter cette reconnaissance.

De plus, nous allons introduire une autre différenciation dans ces prétraitements:

- Indépendants de la méthode de reconnaissance: le seuillage, le réduction du bruit, ...
- Dépendants de la méthode de reconnaissance: par exemple, ceux développés dans le cadre de ce mémoire

2.2.3 Phase III: Analyse

Il s'agit du calcul d'un certain nombre de caractéristiques ou paramètres permettant de circonscrire les données en une information pertinente. L'espace de paramètre P obtenu est très inférieur à R'. Ces paramètres géométriques, topologiques ou statistiques constituent les seules données représentant la forme. Ces caractéristiques sont par exemple: la surface, le périmètre, le nombre de cavités, la compacité, l'allongement, la dissymétrie, l'aplatissement, ...

Cette analyse est indispensable dans un système classique de reconnaissance d'images. En effet, le vecteur de mesures est inadéquat pour représenter des relations

entre les composants du pattern. Un pattern-image est difficilement interprétable en tant que tel contrairement au pattern constitué de caractéristiques.

Cependant certaines méthodes, permettent le support aux relations de données et ne nécessitent pas a priori d'analyse. Ces méthodes sont dites connexionistes, ce sont les réseaux de neurones. La suppression de la phase d'analyse est évident: le gain de temps et de réflexion. De plus, fondamentalement, il est peu probable que le système nerveux de l'homme calcule la distance entre les yeux d'une personne afin d'identifier son visage.
Les méthodes connexionistes sont aussi capables de traiter des données analytiques, mais un des intérêts de la méthode s'estompe.

2.2.4 Phase IV: Reconnaissance

A partir d'un pattern, en l'occurrence une description de la forme (analytique ou non), la phase de reconnaissance, proprement dite, tente d'attribuer cette forme à un modèle de référence (prototype) ou plus généralement à une classe. La difficulté vient du fait qu'il est souvent illusoire de choisir le prototype d'une classe parmi un ensemble d'objets appartenant à celle-ci. Si le système doit reconnaître des chiffres manuscrits, quelle écriture choisir comme prototype?

On utilise le concept « Région de décision » [Schalkoff, 94]:
L'espace de mesures (d dimensions) est classifié en une partition, chacune de ses parts constitue une région de décision dont le label est la classe. Dans le but d'utiliser ces régions de décisions pour une possible et unique affectation à une classe, celles-ci doivent couvrir l'espace R' ou P de dimension d et être disjointes (pas de superposition). Les intersections des parts, deux à deux, sont donc vides. Il existe une exception à cette dernière contrainte: les systèmes flous. Ceux-ci permettent au système de proposer plusieurs classes possibles au lieu d'une seule.
Le bord de chacune des régions de décisions est la frontière de décision. Le challenge consiste à déterminer ces régions de décision. Les séparations entre régions peuvent être linéaires, quadratiques ou relatives (figure 2.3).

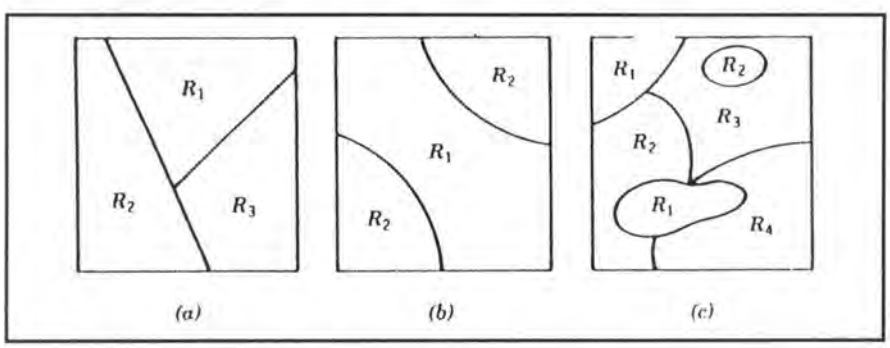


Figure 2.3 Exemples de régions de décisions pour un espace de mesures à deux dimensions: (a) frontières linéaires, (b) frontières hyperboliques et (c) frontières relatives.

Déterminer cette frontière revient à chercher une fonction discriminante de dimension R' ou P selon le type de vecteur (le pattern) utilisé. Il faut donner au système les moyens de modeler cette fonction grâce à des exemples et/ou des lois. L'apprentissage par l'exemple est le plus utilisé et sera développé dans le chapitre 3.

Pour ce qui est du mécanisme d'entraînement du système, la phase de reconnaissance regroupe deux tâches, quelle que soit l'approche utilisée:

- L'apprentissage:	Réorganisation / Renforcement d'une région existante ou Création d'une nouvelle région représentant la forme entrée
- La décision:	Avis sur l'appartenance ou non de la forme aux modèles de l'apprentissage

La tâche de décision pourra être soit une tâche de test, si l'utilisateur connaît la classe de l'objet présenté au système de reconnaissance, soit une tâche de production, dans laquelle ce système classifie des objets dont la classe est inconnue.

Les applications en reconnaissance de patterns et plus particulièrement d'images sont nombreuses. Elles se différencient par les bases statistiques ou structurelles des patterns. Il s'agira de choisir le bon outil pour le job souhaité. On dispose principalement de trois approches pour la reconnaissance de patterns:

- L'approche statistique: l'ensemble des mesures est extrait des données d'entrée (l'image) et est utilisé pour assigner, à chaque vecteur de mesures, une des c classes existantes. On peut citer l'analyse discriminante dont le but est de trouver une fonction linéaire de d mesures qui rendra le mieux compte de la séparation des observations en k groupes.

- L'approche syntaxique consiste à mettre en relation la structure des patterns avec la syntaxe d'un langage défini formellement. Cette technique se limite au pattern de type caractéristique (surface, taille, ...). Cette méthode est à rejeter pour la reconnaissance d'une image 'brute'.

- L'approche réseaux de neurones fournit un moyen pour réaliser l'apprentissage de cette fonction, en souplesse.

Enfin, la compréhension de la scène est une éventuelle cinquième étape permettant une action en fonction d'une interprétation de l'image. Un système de détection d'avion, après avoir classifié celui-ci, permettrait de donner des renseignements sur cet avion et proposerait, éventuellement, sa destruction. [Reid, 1994].

2.2.5 Le 'Bruit' dans un système de reconnaissance

Nous avons étudié que le prétraitement est une source possible de 'bruits'. Mais ce concept peut s'étendre à l'entièreté du système. Ce que nous appellerons dorénavant des 'bruits', seront des circonstances non idéales pour la classification des objets à partir d'informations ou patterns.
On distinguera cinq sources de bruits:

- erreurs de mesures: signal (phase I)
- erreurs de prétraitement (phase II)
- erreurs d'extraction de caractéristiques ou d'analyse (phase III)
- erreur dans le choix des patterns d'entraînement (phase IV)
- méthode de reconnaissance inadéquate (phase IV)

3. L'approche réseau de neurones: une méthode de reconnaissance d'images

3.1 L'intelligence artificielle et les systèmes experts

Depuis quelques années, l'intelligence artificielle est un point de focalisation des médias, du public et des scientifiques. En effet, on s'interroge sur :

- son existence en tant que discipline scientifique,
- ses buts,
- son champ d'investigation,
- sa situation,
- ses limites,
- ses applications actuelles et futures,
- son impact social, économique, éthique ou psychologique.

Qu'est-ce que concrètement l'intelligence artificielle ?
Sous ce terme peu précis, on regroupe, d'une part, l'ensemble des technologies ayant pour but l'analyse et la compréhension du fonctionnement du cerveau humain (aspect scientifique) et d'autre part, la création de machines (automates, robots) ou de programmes réagissant suivant une démarche proche de l'intelligence humaine (aspect technique). Dans le cadre de ce travail, notre intérêt se portera essentiellement sur l'aspect technique.

Au même titre que la psychologie, l'intelligence artificielle vérifie les modèles et théories sur les connaissances et les raisonnements, à la différence qu'elle n'expérimente pas sur des sujets humains mais qu'elle programme des calculateurs. Dès lors, elle se distingue des autres sciences cognitives par son outil central d'investigation: l'ordinateur.

Dans la méthodologie conventionnelle, il y a une certaine incohérence entre le parallélisme du système nerveux et la nature sérielle des voies de l'intelligence artificielle. C'est justement ce parallélisme, cette capacité de prendre plusieurs contraintes simultanément qui définit le cerveau humain [Aleksander, 89]. Or, le réseau de neurones, dont le parallélisme est la principale caractéristique constitue, sans aucun doute, un paradigme prometteur de l'intelligence artificielle.
Ses domaines d'application sont nombreux: pilotage des procédés industriels, reconnaissance de véhicules aux péages autoroutiers, prévision des consommations d'eau et d'électricité d'un secteur géographique, ou encore prédiction des défauts, microporosités dans les alliages d'aluminium sans citer la reconnaissance d'avions et l'OCR [Hérault, 93].
Ces dernières années, les applications industrielles des travaux de recherche ont été nombreuses: lecture automatisée des codes postaux écrits à la main, réservations des billets d'avion, dictée automatique ... [Abu-Mostafa, 95].

S'intéresser à l'intelligence artificielle implique nécessairement la prise en compte des systèmes experts, logiciels exploitant des informations qui leurs ont été transmises pour simuler le comportement d'un expert humain. Ces systèmes abordent des tâches très difficiles pour l'homme dans divers domaines d'applications: médecine, chimie, géologie, droit, robotique, ...

3.2 Définition d'un réseau de neurones

Réseaux de neurones, connexionisme, « Parallel Distributed Processing », sous ces vocables se cache un seul concept !
Ces noms font référence à des systèmes informatiques, qui à l'inverse des techniques informatiques conventionnelles, ont une structure qui, à un certain niveau, reflète ce qui est connu de la structure du cerveau, soit que le système est équipé de plusieurs processeurs travaillant en parallèle, soit que le système est équipé d'un seul processeur mais qu'il simule un fonctionnement en parallèle. Cette dernière solution, la plus souple et la moins coûteuse, est la plus souvent utilisée mais elle a comme inconvénient majeur sa lenteur.

Une fois pour toutes, au dessus de cette querelle de noms, nous utiliserons le terme de 'réseau de neurones' (RN). Ce terme est peut-être abusif mais il reflète bien l'architecture et l'historique des réseaux. D'autant que « RN » est la notation la plus fréquemment rencontrée. Nous refuserons de polémiquer sur le bien-fondé de ce choix, les ambitions prométhéennes des débuts de l'intelligence artificielle n'ayant plus cours.
Il importe de savoir que les réseaux de neurones ne peuvent résoudre tous les problèmes, il s'agit plutôt d'une technique de pointe qui donne des résultats positifs pour certaines applications et qui n'est aucunement une réplique du cerveau humain.

Selon B. Kröse et P. van der Smagt, de l'université d'Amsterdam, un réseau de neurones consiste en un ensemble de neurones (processeurs simples ou unités) qui communique entre eux par l'envoi de messages via un large nombre de connexions pondérées [Kröse, 93]. Ses propriétés fonctionnelles sont les suivantes:

une adaptabilité une généralisation ou organisation de données un traitement parallèle des données

Une autre définition attire l'attention sur la mémoire: « un réseau de neurones est une organisation interne des éléments de mémoire selon le modèle du cerveau humain » [Hérault, 94]. Les éléments ne sont plus reliés de manière séquentielle mais chacun est relié à tous les autres. Ceci ouvre la porte à l'apprentissage par l'exemple, au raisonnement par généralisation, aux processus d'auto-adaptation, ... Cette définition de mémoire structurée à l'aide d'exemples me semble la plus appropriée.

Belaïd parle des réseaux de neurones en ces termes: « Un parallélisme à grain très fin et à forte connectivité » [Belaïd, 92]. C'est un discriminateur non linéaire complexe qui permet de délimiter avec précision les régions dans l'espace de mesures. La complexité de la discrimination fonctionnelle réside dans le fait que le réseau doit s'adapter à des surfaces de séparation complexe (ex: espace à 244192 dimensions) et que la méthode d'apprentissage permet de 'trouver' automatiquement cette fonction à partir d'exemples. Le réseau de neurones peut se passer d'analyse et de spécification explicite de la fonction effectuée. Le fort pouvoir discriminant de celui-ci peut permettre d'éliminer l'étape d'analyse, voir de prétraitement.

3.3 Bref historique

[Aleksander, 1990]

3.3.1 Les années de définitions: 1943-1969

Deux articles sont véritablement à la base de la théorie des neurones. En 1943, Mc Culloch et Pitts développèrent un modèle d'une cellule fondamentale du cerveau: le neurone [McCulloch, 43]. En 1949, D.O. Hebb, un neurophysiologiste, proposait l'idée de mémoire dynamique, ce qui signifie qu'un groupe de neurones peut réagir à différents patterns, par rappels successifs d'une expérience précédente. Plus formellement, la règle d'apprentissage de Hebb stipule que lorsque deux neurones A et B interconnectés sont simultanément activés, il faut augmenter la force de connexion qui les relie.

La première application du réseau de neurones face à un problème du monde réel, *adaline*, est un élément linéaire adaptatif proposé par G. Widrow [Widrow, 59]. Ce réseau était utilisé pour supprimer les échos sur une ligne téléphonique.

Un autre événement clé est l'invention d'un élément de calcul appelé *perceptron* [Rosenblatt, 57]. Ce système permettait de reconnaître les images.

3.3.2 Les années de transitions: 1969-1982

La critique de Minsky et Pappert donne un coup fatal aux réseaux de neurones en démontrant l'incapacité du système à traiter des problèmes non linéairement séparables [Minsky, 69]. Les prétentions des méthodes connexionistes vont être revues à la baisse.

3.3.3 La renaissance du réseau de neurones: 1982-....

Depuis une dizaine d'années, beaucoup de nouveaux paradigmes neuronaux sont apparus et ont augmenté le champ d'application de ces systèmes. On peut citer le réseau de Hopfield [Hopfield, 82], le réseau *counter-propagation* [Hecht, 88], etc.

3.4 Un cadre pour la représentation distribuée

3.4.1 Le neurone

La figure 3.1 montre un neurone formel appelé simplement *neurone*, *cellule* ou *unité*.

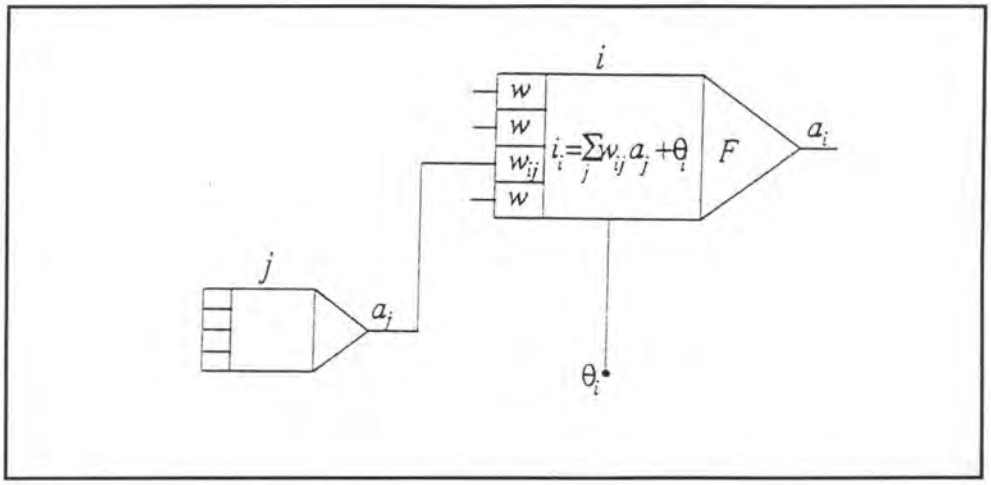


Figure 3.1 Le neurone formel

Notations

- Unité de calcul = neurone = cellule = unité	neurone i (1⇒n)
- Poids de la connexion entre les neurones i et j	w_{ij}
- Règle de propagation = entrée effective du neurone	i_i
- Fonction d'activation (détermine le nouveau niveau d'activation)	F_i
- Etat d'activation pour chaque neurone = sortie	a_i
- Une entrée externe (bias)	θ_i

Chaque neurone ou unité (processeur simple) réalise une tâche simple, à savoir recevoir des entrées de neurones voisins ou de sources externes et les utiliser pour le calcul d'un signal de sortie propagé vers d'autres unités [Kröse, 94].
Le système construit par l'ensemble des neurones est intrinsèquement parallèle: chaque unité réalise ses calculs au même moment.

Cette tâche est réalisée en deux temps:

1. La règle de propagation permet de calculer l'entrée effective du neurone $i_i(t)$ en sommant chacune des entrées $a_i(t)$ (constituées par les sorties des neurones de la couche précédente) pondérée par le poids $w_{ij}(t)$ de la connexion entre le neurone d'entrée et le neurone courant, éventuellement augmentée par le terme du bias $\theta_i(t)$.

$$i_i(t) = \sum_j w_{ij}(t) \, . \, a_i(t) + \theta_i(t)$$ *Règle de propagation*

2. La règle d'activation permet de connaître l'état de sortie du neurone qui pourra être propagé vers d'autres neurones. Cet état d'activation est directement fonction de l'entrée effective du neurone.

$$a_i(t+1) = F_i(i_i(t))$$ *Règle d'activation*

Différents types de fonctions d'activation F_i existent. Ainsi, la fonction la plus utilisée est la sigmoïde qui répond à la formule suivante:

$$F_i(i_i) = 1 / (1 + e^{-\lambda i_i})$$

Le λ, dont la valeur varie de 0 à l'infini, représente la pente de la sigmoïde; la valeur de λ pour la fonction sigmoïde classique est 1.
Quelques fonctions d'activations, dont la sigmoïde, sont représentés sur la figure 3.2.

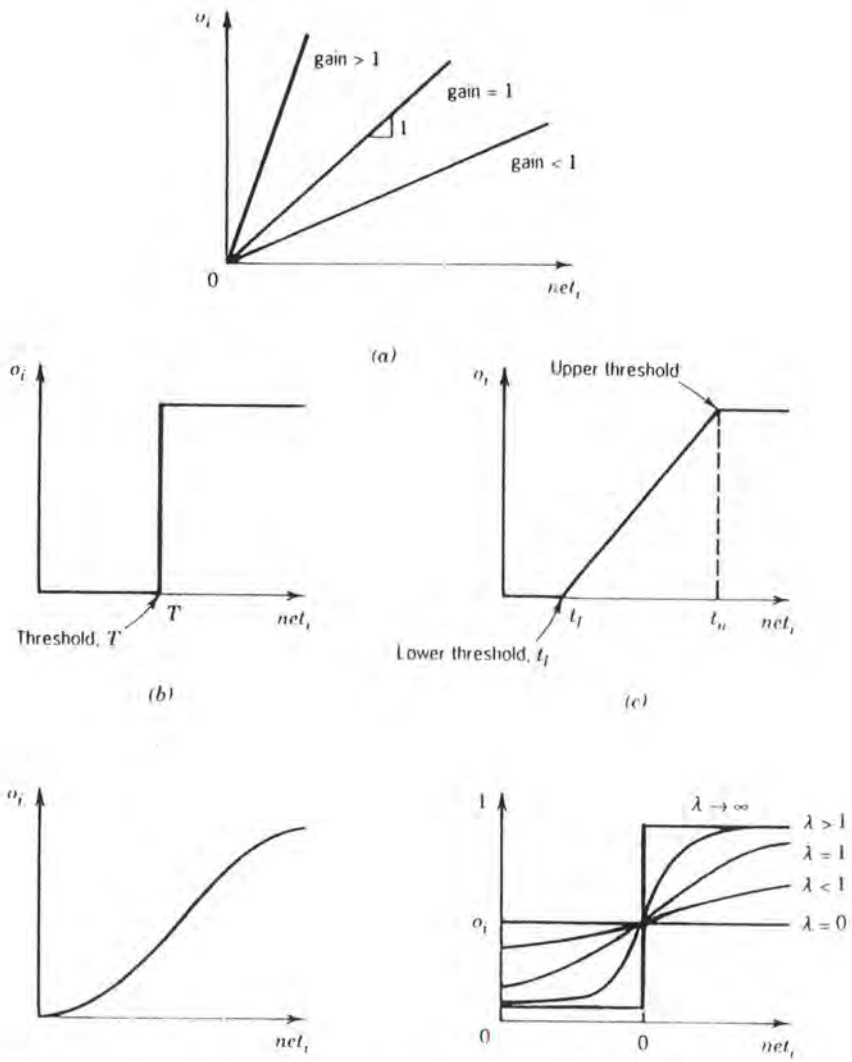


Figure 3.2 Différents exemples de courbes définies par les fonctions Fi caractéristiques des réseaux de neurones: (a) linéaire, (b) seuil, (c) linéaire à seuil, (d) sigmoïde caractéristique et (e) fonctions sigmoïdes pour les différentes valeurs de λ.

Une deuxième tâche incombe au réseau de neurones: l'ajustement des pondérations.

Chaque neurone est classé selon sa répartition dans la topologie du réseau. Les unités d'entrée reçoivent les données de l'extérieur. Les unités cachées ont des signaux d'entrée et de sortie interne au système lui-même. Les unités de sortie, quant à elles, envoient les données hors du système. Ainsi, par exemple, un système composé d'un seul neurone recevant des informations en entrée et fournissant une réponse en sortie, est à la fois un neurone d'entrée et de sortie [Dayhoff, 90].

La description de la structure d'une unité et de son fonctionnement est générale pour tous les réseaux. Dès lors, les réseaux de neurones vont se différencier principalement par:

- leur façon de modifier le poids de chaque connexion,
- leur structure macroscopique ou architecture.

3.4.2 La structure du réseau

La structure du réseau représente comment les connexions se distribuent entre les unités. Deux types de réseaux s'opposent sur ce principe:

- les réseaux 'feed-forward' dans lesquels le flux de données est strictement de l'entrée vers la sortie. Ainsi, un neurone appartenant à une couche d'entrée ne recevra jamais, en entrée, la sortie d'un autre neurone et un neurone de la couche de sortie n'aura jamais sa sortie propagée vers un autre neurone (figure 3.3.a).

- les réseaux récurrents contenant des connexions 'feed-back', ce qui signifie qu'un neurone peut avoir en entrée un signal provenant directement ou indirectement de sa propre sortie (figure 3.3.b).

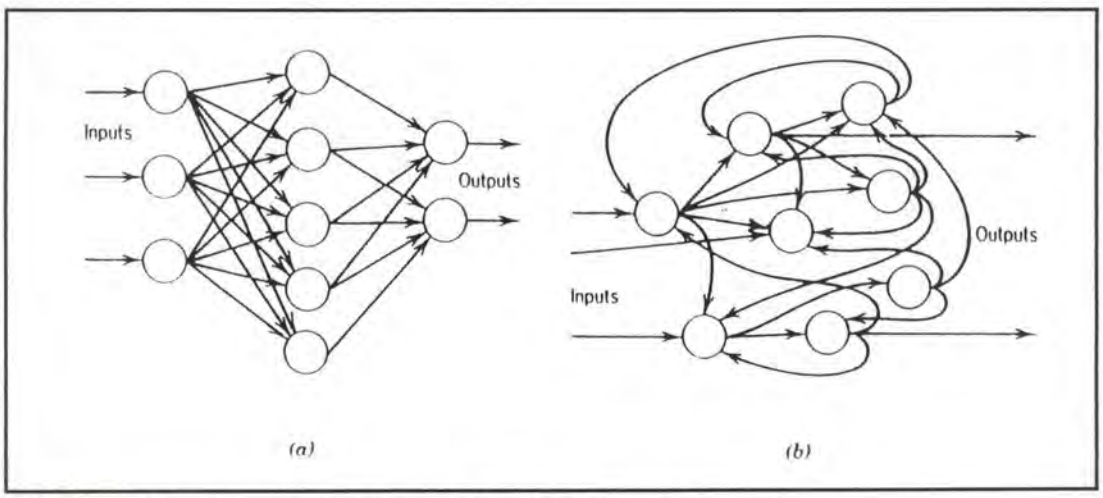


Figure 3.3 (a) réseau 'feed-forward' et (b) réseau récurrent

Si nous appliquons la règle d'activation pour un pattern d'entrée du réseau, nous remarquons qu'un réseau 'feed-forward' est stable en un passage, l'activation d'un

neurone n'étant modifiée qu'une seule fois. Par contre, un réseau récurrent verra son activation changer au cours du temps et évoluer vers un état stable dans lequel elle ne se modifiera plus. Toutefois, cette activation n'atteindra, dans certains cas, jamais sa stabilité!

Un réseau de type 'feed-forward' est composé, classiquement, d'une couche d'entrée, de couches cachées et d'une couche de sortie. Par définition, il n'y a pas de connexion inter-couche et de connexion d'une couche supérieure vers un neurone de couche inférieure.

La structure du réseau récurrent est fonction de l'imagination de chercheurs: en anneau, interconnectée, ...
Quelques structures classiques de réseaux seront abordées dans la section 3.5 sur la taxonomie.

3.4.3 L'entraînement des neurones

Notations

- Pattern (d données en entrée)	x^p
- 'Output' désiré pour le pattern x^p (m données fournies par l'instructeur)	d^p
- 'Output' proposé par le réseau pour le pattern x^p (m données en sortie)	o^p

Il nous reste à étudier comment les poids des connexions, la mémoire du réseau, vont pouvoir être modifiés; comment le réseau va s'organiser pour créer une fonction qui permettra de représenter les classes.
Un réseau de neurones doit être configuré de telle sorte que l'application d'un ensemble de valeurs d'entrées produise un ensemble désiré de sorties, de manière directe ou par un processus de relaxation. Ainsi, on peut, soit choisir les poids, explicitement, à l'aide d'une connaissance à priori, soit entraîner le réseau de neurones à changer les poids avec des patterns d'apprentissage.
Un algorithme d'apprentissage est un programme dont le but est de gérer la modification des poids tout au long de l'apprentissage.

Deux catégories d'apprentissages se distinguent:

1. L'apprentissage supervisé qui suppose une classification à priori. Le réseau est entraîné avec des couples [pattern, 'output' désiré], donnés par un professeur externe.

Un cas particulier est l'apprentissage auto-supervisé dans lequel le système qui contient le réseau a la possibilité de vérifier automatiquement la sortie. Par exemple, un robot qui est entraîné à suivre un point lumineux peut générer un signal d'erreur calculé d'après la distance séparant l'impact du point lumineux sur la rétine stimulée et le centre de cette rétine [DARPA, 88].

2. L'apprentissage non supervisé suppose que le système va créer sa propre représentation du stimuli d'entrée et créer ses propres classes. Il n'y a pas de professeur externe. Le processus d'apprentissage extrait les propriétés statistiques de l'ensemble d'entraînement et groupe les patterns similaires en classes. En appliquant, lors de la phase de rappel, un pattern d'une classe donnée à l'entrée du réseau, ce dernier produira un vecteur d''output' spécifique à la classe.

Bien que différents algorithmes d'apprentissages aient été développés pour des applications diverses, les plus populaires et les plus puissants sont supervisés [Anderson, 87].

Les ajustements des poids des connexions se font grâce à des règles de modification contenues dans l'algorithme d'apprentissage. Ces règles dépendent du type de réseau étudié. Celles-ci seront étudiées en détail dans le cas particulier du réseau à rétropropagation, le 'back-propagation' (BPN).

3.5 Taxonomie des réseaux de neurones

Notre but n'est pas de présenter une taxonomie exhaustive mais de s'y retrouver dans cette forêt que constitue les nombreux types de réseaux de neurones:

1. réseaux multicouches	: multicouches, supervisés
2. réseaux à auto-organisation	: multicouches, non supervisés
3. réseaux récurrents ('feed-back')	: pas de couche, supervisés

3.5.1 Les réseaux multicouches

Les réseaux multicouches sont des réseaux de neurones qui réalisent un apprentissage supervisé et ont des connexions de type 'feed-forward'. Ils sont organisés en couches de neurones.
Les réseaux multicouches sont entraînés à rechercher une fonction de 'recherche de régions' F: $R_d \Rightarrow R_m$ par présentation d'exemples (d: dimension du pattern et m: dimension de l'output) de type (x^p, o^p) avec $o^p = F(x^p)$ de cette recherche.

Au départ, *l'adaline* (adaptative linear element) [Widrow, 1960] ne contenait qu'un neurone et la fonction d'activation était binaire. L'apprentissage se faisait par une minimisation de l'erreur moyenne (ou différence entre la sortie désirée et la sortie obtenue).

Le *perceptron*, évolution *d'adaline*, proposé par Rosenblatt, est un réseau constitué d'unités binaires [Rosenblatt, 1959]. Initialement prévu pour fonctionner avec une couche de neurones, il a suscité beaucoup d'optimisme. Cependant, Minsky et Pappert ont montré qu'un perceptron, à une couche, ne pouvait pas représenter une simple fonction 'ou exclusif' (figure 3.4).

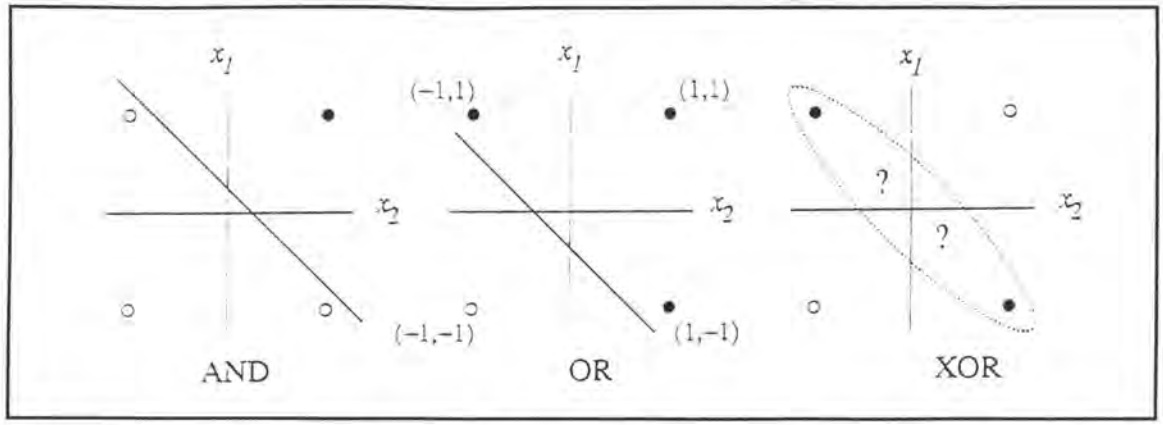


Figure 3.4 Représentations géométriques de l'espace d'entrée des fonctions ET, OU et OU Exclusif. Cette dernière fonction n'est pas représentable de façon linéaire.

Ce système est incapable de déterminer, dans l'espace des données, une région de décision suffisamment précise [Minsky, 69]. Dès lors, la non-linéarité a été introduite dans les réseaux de neurones. Il s'agit d'utiliser plusieurs couches ainsi qu'une fonction d'activation non linéaire, sigmoïde par exemple. C'est le cas du 'back-propagation' [Rhumelhart, 85].

3.5.2 Les réseaux à auto-organisation

Les réseaux de neurones à apprentissage supervisé sont capables de traiter beaucoup de problèmes différents. Cependant, des problèmes existent lorsque l' 'ensemble' d'entraînement, consistant en un pattern et un 'output' désiré, n'est pas à la disposition de l'utilisateur.
La seule information disponible est fournie par un ensemble de patterns x^p. Dès lors, l'information nécessaire à la structuration du réseau (modification des poids) doit être trouvée dans l'ensemble d'entraînement disponible redondant x^p.

La réduction de dimensions est une illustration d'un tel problème. Dans cette application, les données d'entrée doivent être groupées dans un nouvel espace de dimension plus faible que l'espace initial des données/mesures.
L'apprentissage doit, donc, être réalisé sans l'aide d'un professeur extérieur; les algorithmes, non supervisés d'adaptation de poids, sont habituellement caractérisés par une compétition globale entre les neurones.

On peut citer:
- l'apprentissage compétitif [Rhumelhart, 85],
- le système à conservation de topologie [Kohonen, 82],
- le 'counterpropagation' [Hecht, 88].

3.5.3 Les réseaux récurrents

L'ensemble des réseaux discutés précédemment ne sont pas récurrents. Toutes les données affluent dans une seule direction, et après un nombre fini et connu de pas, fonction de la profondeur du réseau, nous avons connaissance de toutes les valeurs d'activation.

D'un autre côté, les poids et la mémoire du réseau se stabilisent ou convergent vers un état souhaité à l'aide d'un apprentissage supervisé ou non.
Les réseaux récurrents ont des connexions 'feed-back' de leurs unités de sortie vers celles d'entrée et leurs valeurs d'activation (a_i) subissent un processus de relaxation durant lequel les poids restent inchangés. Lorsque le réseau est intrinsèquement stable, il évolue vers un état où la valeur des neurones ne change plus. L'exemple le plus connu de ce type de réseau est celui de Hopfield [Hopfield, 82]. Dans le cas inverse, les modifications des valeurs d'activation ont une durée infinie. La figure 3.5 montre un réseau récurrent complètement connecté.

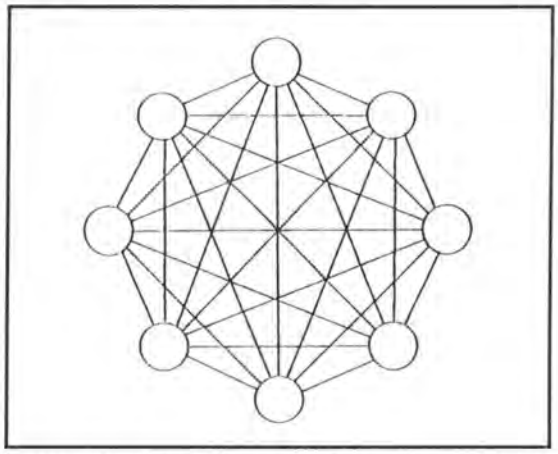


Figure 3.5 Réseau récurrent dans lequel tous les neurones sont à la fois des neurones d'entrée et de sortie.

Une application du réseau Hopfield est la mémoire associative où les poids des connexions entre neurones sont tels que pour les patterns appris (règle d'apprentissage proche de celle de Hebb), le réseau est stable. Si le pattern est incomplet ou bruyant, l''output' est incorrect.

3.6 Le paradigme de la rétropropagation

Le 'back-propagation' est un réseau de neurones dont le domaine d'applications est très étendu. De nombreux exemples d'utilisations de ce réseau pour la reconnaissance de caractères, la synthèse vocale, le contrôle de robot, etc ont permis de peaufiner le fonctionnement et la structure du BPN ('Back-Propagation Network'). Il faut noter que ce qui est rétropropagé dans le BPN, n'est pas le signal, mais la correction des poids des connexions. Il n'y a donc pas de contradiction quand on parle d'un réseau 'back-propagation' (des erreurs) de type 'feed-forward' (des signaux).

3.6.1 Description de l'architecture

Le réseau à rétropropagation est de type 'feed-forward' à apprentissage supervisé. Sa typologie de type multicouches est composée d'une couche de neurones en entrée, de n couches cachées et d'une couche en sortie (figure 3.6).

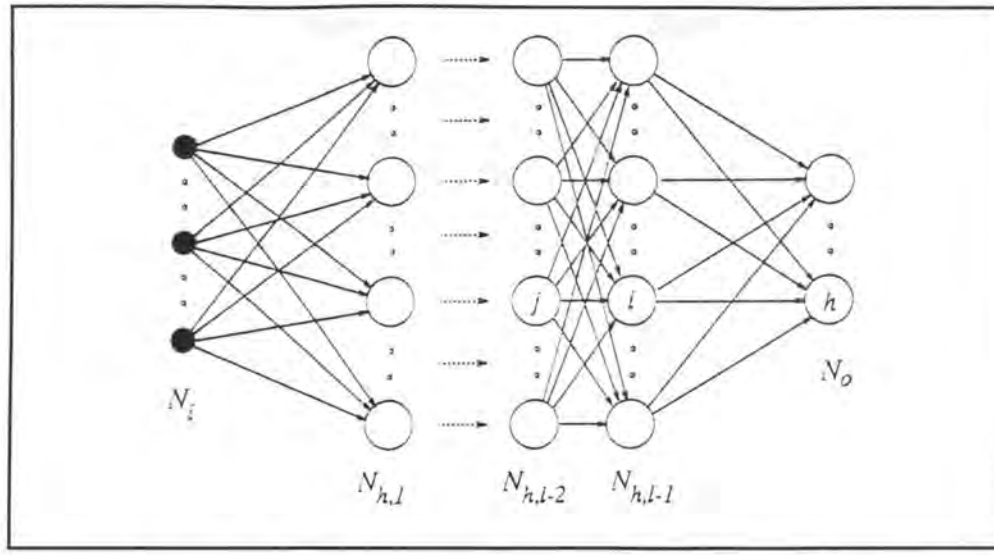


Figure 3.6 Un réseau 'back-propagation' typique avec en entrée une couche de N_i neurones, l -1 couches cachées avec N_l neurones et une couche de sortie avec N_o neurones.

La couche d'entrée ne modifie rien, il s'agit, en quelque sorte, d'une interface permettant au pattern de se propager dans le réseau. Chacune des unités de cette couche répond à la loi de propagation suivante: $a_i=I_i$!
Chaque neurone est entièrement connecté à l'ensemble des neurones de la couche supérieure. Le nombre de couches cachées est au moins égal à un. Si il y a plus d'une couche caachée on parlera de MBPN (rétropropagation à plusieurs couches internes ou Multilayer Back-propagation). Dans la plupart des cas, la fonction d'activation sigmoïde est utilisée pour la propagation du signal dans les couches cachées et de sorties.
Pour notre application, nous utiliserons le paradigme rétropropagation avec une seule couche cachée et une fonction d'activation sigmoïde. En effet, il a été démontré [Hartman, 90] qu'une seule couche intermédiaire suffisait pour déterminer approximativement une fonction avec un nombre fini de discontinuités, à partir du moment où les fonctions d'activation de la couche interne étaient non linéaires.

Mais, il nous reste à savoir comment les poids des interconnexions sont modifiés et comment le système apprend.
L'idée, indiquée par le nom du réseau, consiste à propager en arrière l'erreur des unités de sortie! Mais attention, le signal est toujours propagé vers l'avant (type 'feed-forward'), seule, la correction des poids se fait vers l'arrière.

3.6.3 La rétropropagation des erreurs

La rétropropagation des erreurs consiste à rechercher un minimum global d'erreurs pour un ensemble de patterns par la méthode de descente graduelle, ce qui revient à trouver la fonction qui épouse le mieux l'espace ou les régions représentées par les classes. Un cycle d'apprentissage avec le réseau 'back-propagation' est caractérisé par une propagation du pattern vers l'avant jusqu'à ce que les neurones de

la couche de sortie soient activées. Ensuite, l'erreur est rétropropagée. Ce mécanisme est la principale nouveauté du BPN.

Nous allons montrer comment, à partir de la loi de Hebb, la loi de la rétropropagation de l'erreur ou loi delta généralisée a été établie.

- La loi de Hebb [Hebb, 49] se résume en ces termes: « Si 2 unités i et j sont actives simultanément, leur interconnexion doit être renforcée ». Plus précisément on définit cette loi comme suit:

$$\Delta w_{ij} = \alpha.a_i.a_j$$

avec une constante positive de proportionnalité α représentant le taux d'apprentissage.

- La loi delta [Widrow, 60] est une extension de la loi de Hebb qui permet de modifier les poids des connexions entre les neurones de la couche d'entrée et ceux de la couche de sortie d'un réseau multicouches à 2 couches. Cette loi fonctionne exclusivement pour une fonction d'activation linéaire et pour un réseau avec deux couches (et donc pas un BPN). Son principe consiste à minimiser la fonction d'erreur par une méthode appelée « descente graduelle ».
Soit un réseau avec un neurone x_j de la couche d'entrée et un neurone x_i de la couche de sortie. Pour la valeur d'entrée a_j, de l'unité x_i, la modification du poids de la connexion d'un neurone x_j au neurone de sortie x_i égale le produit de cette valeur avec la différence entre l''output' désiré du neurone x_i: d_i et l''output' effectif de ce neurone x_i: a_i .

$$\Delta w_{ij} = \alpha.(d_i-a_i).a_j$$

avec une constante positive de proportionnalité α représentant le taux d'apprentissage.

- La loi delta générale [Rhumelhart, 86] est une généralisation de la règle delta pour des fonctions d'activation non linéaires et un réseau à plusieurs couches cachées.
Si on utilise une fonction d'activation linéaire, un réseau à plusieurs couches n'est pas plus efficace qu'un réseau à simple couche.
La règle delta généralisée permet de modifier les poids des connexions entre les couches intérieures dans un réseau doté de plusieurs neurones en sortie.

$$\Delta^p w_{ij} = \alpha.\delta^p_i.a^p_j$$

avec $\delta^p_i = (d^p_i-a^p_i).F'_i(i^p_i)$ pour une unité de la couche de sortie
$\delta^p_i = F'_i(i^p_i).\sum \delta^p_h w_{hi}$ pour une unité d'une couche cachée

Pratiquement, la règle delta généralisée s'exprime comme ceci:
L'application de la loi delta généralisée concerne deux phases. Au cours de la première, le pattern est présenté et propagé, vers l'avant, à travers le réseau, pour

calculer la valeur d'activation a_i^p pour chaque unité de sortie. Cette sortie est comparée avec la valeur désirée, ce qui donne un signal d'erreur δ_i^p pour chaque unité de sortie. La seconde phase consiste à un passage vers l'arrière, à travers le réseau, durant lequel le signal d'erreur est passé à chaque unité et les changements appropriés sont calculés. Le calcul de ces ajustements de poids se fait par l'ajout de la valeur de $\Delta^p w_{ij}$ à l'ancien poids de la connexion entre le neurone i et le neurone j, w_{ij}.

3.6.4 Améliorations

3.6.4.1 Les connexions entrées/sorties

Il est possible de connecter les unités de la couche d'entrée à celles de la couche de sortie du réseau. Ceci a pour but de faciliter l'ajustement des poids si le problème est linéaire.

3.6.4.2 Le momentum ('smoothing')

La descente graduelle correcte requiert des pas infinitésimaux. La constante de proportionnalité est le taux d'apprentissage α. Pour des raisons pratiques, on choisi un taux d'apprentissage aussi large que possible, tout en évitant les valeurs trop élevées, sources d'un phénomène d'oscillation.
Une façon d'éviter celle-ci avec un α élevé est de modifier les poids en tenant compte du changement précédent par l'ajout d'un momentum: β. La loi delta généralisée avec le momentum calcule la différence de poids au temps t + 1 comme ceci :

$$\Delta w_{ij}(t+1) = \alpha\, \delta_i^p\, a_i^p(t) + \beta\, \Delta w_{ij}(t)$$

Le composant $\beta\, \Delta w_{ij}(t)$ détermine l'effet du changement précédent.

La figure 3.7 montre le rôle du momentum:

a: faible taux d'apprentissage α (oscillation faible)
b: large taux d'apprentissage α (oscillation élevée)
c: large taux d'apprentissage α et momentum β ajouté (oscillation faible)

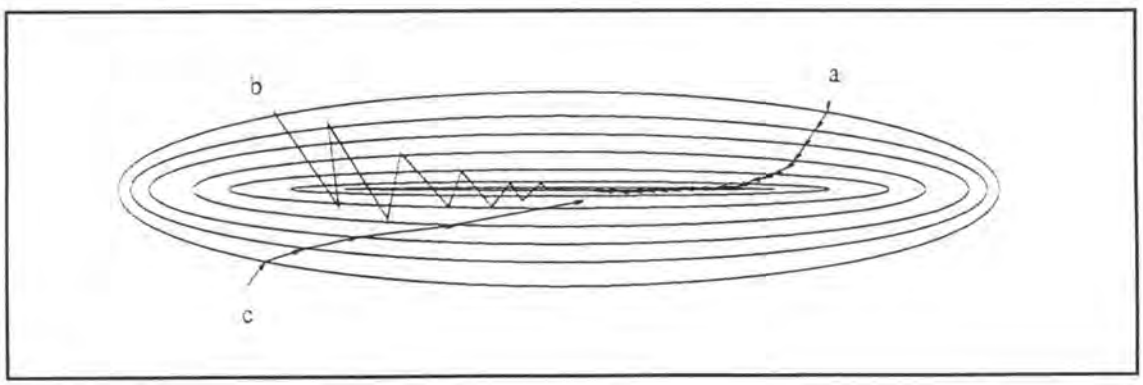


Figure 3.7 La descente dans l'espace des poids

3.6.4.3 Le 'batching'

Théoriquement, l'algorithme de rétropropagation réalise la descente graduelle sur l'erreur totale uniquement si les poids sont ajustés après que l'entièreté de l'ensemble d'apprentissage ait été présenté. La plupart du temps, le vecteur d'erreur pour chaque unité de sortie est calculé séparément pour chaque pattern et les poids sont également adaptés mais il y a une indication empirique d'accélération de la convergence. Il existe une méthode évitant de changer les poids de chaque pattern présenté: il s'agit du 'batching'. Le principe de celui-ci est de calculer l'erreur moyenne sur un ensemble de patterns et de ne réaliser la modification des poids qu'à la fin de la présentation de cet ensemble au réseau. Toutefois, il faut prendre garde à l'ordre dans lequel les patterns sont présentés; ainsi, l'utilisation systématique de la même séquence peut générer une focalisation vers les premiers patterns.
Ce problème peut, malgré tout, être résolu par une méthode d'apprentissage où l'on modifie l'ordre des patterns au fur et à mesure des présentations!

3.6.5 Avantages et inconvénients

Parmi les points positifs du paradigme de la rétropropagation, nous retiendrons que ce type de réseau est bien connu, qu'il y a beaucoup d'exemples de bon fonctionnement pour la reconnaissance d'images et qu'il existe beaucoup de raffinements et d'améliorations permettant d'optimiser son utilisation selon les applications. De plus, la structure du 'back-propagation' est proche de certaines associations de neurones biologiques présentes dans la partie du cerveau qui décode l'information sensorielle. L'apprentissage supervisé et sa structure modifiable s'avèrent être des avantages déterminants pour beaucoup d'applications, comme par exemple le traitement d'images.

D'un autre côté, on peut regretter la lenteur du processus d'apprentissage de ce paradigme mais l'augmentation exponentielle de la vitesse des ordinateurs minimise cet inconvénient. Cependant, deux problèmes majeurs peuvent apparaître lors de l'utilisation du BPN:

1. « La paralysie du réseau »

En effet, lorsque le réseau s'entraîne, les valeurs des poids peuvent être très importantes. L'entrée effective d'un neurone d'une couche cachée ou de sortie peut atteindre des valeurs d'autant plus élevées. Or, avec la fonction d'activation, on obtient des valeurs proche de 1 ou 0; les ajustements des poids ont, à cet endroit de la courbe, un effet très faible, ce qui peut mener à un blocage virtuel du réseau! Dès lors, on peut conseiller de déceler cet état de paralysie par une fonction d'activation linéaire en sortie.

2. Le minimum local

La surface des erreurs d'un réseau complexe est remplie de montagnes et de vallées. Au cours de la descente graduelle, le réseau peut-être piégé dans un minimum local alors qu'un minimum plus profond est peut-être proche! Deux solutions sont envisageables:

- une méthode probabiliste qui testerait les minimums locaux afin de trouver le minimum global mais sa lenteur serait un inconvénient,
- une augmentation modérée du nombre de neurones de la couche cachée qui diminuerait le risque de tomber dans un minimum local.

3.7 L'entraînement et la théorie des conseils

La mémoire d'un réseau de neurones est répartie dans les poids de ses connexions. Apprendre, c'est mémoriser!
Mais certains problèmes se caractérisent par un apprentissage difficile. Aleksander et Morton décrivent ce qu'ils appellent les 'Hard learning problems' [Aleksander, 90]. Ce sont les problèmes qui nécessitent des structures neuronales plus complexes que le perceptron pour être solutionnés. On peut citer:

- les patterns avec connexions,
- les patterns avec parité.

Cependant, ces problèmes sont souvent résolus par un réseau suffisamment complexe.

Nous n'avons pas encore décrit « une politique d'apprentissage » dont le but serait de déterminer les exemples utilisés pour l'entraînement du réseau. Le principe de l'apprentissage est de fournir des exemples formateurs et d'engendrer la capacité d'apprendre à partir de ces exemples plutôt que de donner, au programme, le détail des solutions qu'il doit mettre en oeuvre pour exécuter la tâche qui lui est confiée.

Selon Abu-Mostafa, « Les systèmes d'intelligence artificielle apprennent mieux et plus vite quand on leur fournit des informations sous la forme de conseils intelligents » [Abu-Mostafa, 1995]. Quand les exemples utilisés pour l'apprentissage ne contiennent pas suffisamment d'informations vitales, la machine ne parvient pas à apprendre correctement. On peut alors, souvent, lui apporter les informations requises en lui donnant quelques conseils intelligents.
Ces conseils sont des «règles d'invariance » intéressantes pour résoudre une multitude de problèmes intermédiaires entre les problèmes structurés et entièrement définis, auxquels aucun exemple n'est nécessaire, et les problèmes aléatoires, totalement indéfinis, pour lesquels on n'a que des exemples d'apprentissage. Si le pattern d'entrée est une image, ces invariances sont, par exemple, des distorsions géométriques (rotation, échelle, translation).

Le comportement d'un système est représenté mathématiquement par une fonction qui associe à des valeurs d'entrée (qui spécifient le problème à traiter) des valeurs de sortie (par exemple, une décision à prendre). Par l'apprentissage, on cherche à faire coïncider cette fonction et une fonction cible: autrement dit, on veut que le système associe des entrées particulières à des sorties spécifiques. On peut utiliser des exemples d'apprentissages dérivés de la fonction cible pour guider la sélection des valeurs des paramètres libres de la machine.

A chaque exemple, la machine affine ses réglages afin de mieux associer ses entrées et ses sorties. Quand le système parvient à un réglage qui correspond d'aussi près que possible à la fonction cible, on considère qu'il l'a effectivement apprise. Ainsi l'apprentissage est simplement la recherche des réglages optimaux. Comme cette recherche est guidée par les exemples d'apprentissage, cette méthode est appelée l'apprentissage par l'exemple.

Les exemples d'apprentissages contiennent-ils assez d'informations pour que la machine réponde correctement à des questions nouvelles ?
L'enregistrement des exemples d'apprentissage ne garantit pas nécessairement que la machine réagit correctement quand on lui présente un cas qu'elle n'a jamais rencontré auparavant. Si les données sont déficientes, qu'il y a trop peu d'exemples ou que les informations sont mauvaises la machine risque de prendre beaucoup de temps avant d'apprendre.

Comment donner des conseils?

Toutes les représentations des conseils doivent être standardisées, afin de permettre à l'algorithme d'apprentissage de les traiter sur un pied d 'égalité. Les conseils peuvent donc être présentés sous une forme analogue à celle des entrées-sorties utilisées pour l'apprentissage.
Les exemples utilisés pour représenter les conseils n'ont pas besoin d'être réels car nous ne demandons pas à la machine de prendre une décision pour un cas réel mais plutôt d'agir conformément au conseil ou à la règle. Dans le cas du système de vision, on peut représenter les règles d'invariance en utilisant des images sans aucun lien avec la fonction réelle.

L'étape d'équilibrage

Il est difficile de trouver un bon équilibre entre les conseils et les exemples réels. Le système ne parvient généralement pas à tenir compte des exemples d'apprentissage et de tous les exemples associés aux conseils.
De plus, pendant le processus d'apprentissage, certains conseils sont appris plus vite que d'autres mais si l'on parvient à détecter les règles, les plus difficilement apprises on peut rectifier la situation à l'itération suivante, il s'agit de la minimisation adaptative.

Nous pouvons conclure cette section par un survol des difficultés de l'apprentissage:

- Un surapprentissage qui survient lorsque la machine mémorise les exemples d'apprentissages au dépens des généralisations,

- Un allongement rédhibitoire des temps de calcul: lorsque le système cherche un minimum global, il peut être piégé dans un minimum local. Par chance, les études ont montré que l'on obtient une performance satisfaisante même si l'optimum n'est que local.

Une autre solution, opposée à celle proposée ci-dessus, est de ne pas faire apprendre les invariants par l'apprentissage mais de construire un réseau « absorbant » les invariants [Reid, 94]. L'invariance structurée directement dans l'architecture ne doit plus être apprise. Le principe consiste à combiner les paires ou triplets de neurones en entrée du réseau, le signal d'entrée de la couche suivante étant la somme pondérée des produits des paires ou triplets.

4. Application à la reconnaissance de cellules

Nous proposons ici d'expliquer la démarche qui va être suivie dans ce mémoire: nous allons décrire les bases d'un système de reconnaissance de cellules, aussi appelé système expert. Le cadre théorique décrit dans les chapitres 2 et 3 va être utilisé avec comme exemple les fibroblastes.

Une cellule donnée (fibroblaste), fixée sur une boîte de Pétri, constitue un objet, identifié dans notre travail par un numéro et appartenant à une et une seule classe, selon la théorie du vieillissement cellulaire. Nous disposons d'un pattern de cette cellule, en l'occurrence une image digitalisée de 512 x 512 pixels, c'est-à-dire d'un vecteur de 262.144 mesures. Sachant qu'il n'existe pas de modèle mathématique basé sur ce vecteur, mais que cette information suffit à un expert pour classifier la cellule, nous sommes partis à la recherche d'un moyen pour classifier ces images.

Lors de la discussion avec les experts (chapitre 1), il est apparu que le moyen principal de classification visuelle des cellules était la forme de ces dernières. Cette forme, rappelons-le étant délimitée par la membrane cytoplasmique. Cependant, dans des cas litigieux, les renseignements sur la densité cytoplasmique et sur la forme du noyau étaient utilisés pour lever une ambiguïté entre deux classes. Enfin, il est possible que des éléments extérieurs à la cellule, comme le fait que les cellules soient plus ou moins proches l'une de l'autre, ou la présence de bras cellulaires séparés de la cellule, puissent intervenir dans la décision finale.
Malgré cela, sans simplifier à l'extrême, nous pouvons dire que la forme de la cellule est, dans la majorité des cas, discriminante. Nous n'utiliserons donc pas les informations en niveaux de gris et nous considérerons la cellule comme noire sur un fond blanc. De plus, il n'y aura qu'une seule cellule par image.

Nous allons mettre en application la méthode de traitement d'images, décrite dans la section 2.2, au système de reconnaissance de cellules. Nous décrirons ci-dessous les options prises pour chacune des phases de la méthode. Compte tenu que notre choix s'est porté sur le réseau de neurones, il nous sera possible de supprimer la phase d'analyse et de la remplacer par une phase de prétraitement dépendant du réseau de neurones.

Les fibroblastes, fixés sur la boîte de Pétri, constituent les objets du monde réels qui sont traités par la phase I du système expert.

Phase I: L'acquisition des données

Cette phase est caractérisée par l'acquisition des images, les cellules. Ces données ont le format suivant:

- La taille des images: 512 x 512 pixels
- Quantification en niveaux de gris (256 niveaux de gris)

Le dispositif utilisé ne réalise pas d'échantillonnage mais l'intensité lumineuse sera quantifiée en niveau de gris, de 0 à 255 niveaux.

Phase II: Le prétraitement indépendant de l'application

Le prétraitement indépendant de l'application comprend différentes tâches telles que:

1. La réduction du bruit qui consiste à effacer de l'image:
   - les 'taches', les bras cytoplasmiques, les zones de délimitations. Ceci peut être réalisé par un seuillage ou une fonction de réduction de bruit (élimination d'un pixel sans voisin).
   - les cellules de type I, VI et VII car nous ne cherchons pas à créer des régions avec ces classes.
   - les cellules pathologiques, c'est-à-dire les cellules qui ne sont pas classifiables par l'expert, parce qu'elles sont dans un état non habituel (dégénérées). Il ne s'agit évidemment pas ici de discuter du modèle du vieillissement.

2. L'homogénéisation des données par le processus de binarisation (mise en noir et blanc).

3. La réduction du format des données de 512 x 512 pixels à 100 x 100 pixels.

Phase III: Le prétraitement dépendant de la méthode de reconnaissance

Cette troisième phase effectue un prétraitement qui cette fois dépend de la méthode de reconnaissance, en l'occurrence les réseaux de neurones. Divers traitements sont réalisés tels que:

- L'homogénéisation des données comprenant différentes fonctionnalités
- La normalisation des données, dont les fonctionnalités sont la rotation et la translation. Il n'y a pas de mise à l'échelle parce que nous utilisons toujours le même oculaire.

Il importe de savoir qu'il n'y a pas de phase d'analyse, présentant des données 'intelligentes' telles que la surface, le périmètre, ... Ceci constitue l'originalité de la démarche dont l'objectif est de percevoir la manière dont un système peut se structurer face à une information à l'état brute, parallèlement au cerveau humain qui interprète et donne une signification à une image.

Phase IV: La reconnaissance

Cette dernière phase, au cours de laquelle l'image est reconnue, fait appel au paradigme de rétropropagation, caractérisé par un réseau de neurones à plusieurs couches et un algorithme d'apprentissage supervisé.

A. Blum décrit un exemple d'utilisation d'un réseau de neurones de type 'rétro-propagation' [Blum, 92]. Il s'agit d'une application de reconnaissance d'images dans laquelle des chiffres (0 à 9) sont écrits à la main. Dans cet exemple, le réseau apprend des associations entre les patterns en entrée (images de chiffres, formées de pixels) et les 'outputs' (valeurs de chiffres). Cet exemple nous incite à utiliser le réseau de type 'back-propagation' pour la reconnaissance de cellules.

Pratiquement, le réseau est structuré comme suit (figure 4.1):
- une couche d'entrée de 200 ou 400 neurones ou 'NI' (selon la taille du pattern-image présenté au réseau).
- une couche cachée dont la taille varie entre 1 et 80 neurones ou 'NH'.
- une couche de sortie composée de 5 neurones ou 'NO'. Chacune des valeurs d'activations de ceux-ci, comprises entre 0 et 1, représentent une classe. Par exemple, si la valeur de sortie du neurone numéro 3 est la plus élevée, c'est la classe III qui est proposée comme solution. Le neurone 1 représente une classe non déterminée, autre que les classe II, III, IV et V.

Pour rappel, les neurones de la couche d'entrée ne servent que d'interface et ne font donc pas de calcul. Chacun de ces neurones est connecté à chaque neurone de la couche cachée, et pareillement pour les connexions des unités de la couche cachée vers la couche de sortie. La fonction d'activation est une fonction sigmoïde dont la pente (λ) est instanciée à 1.

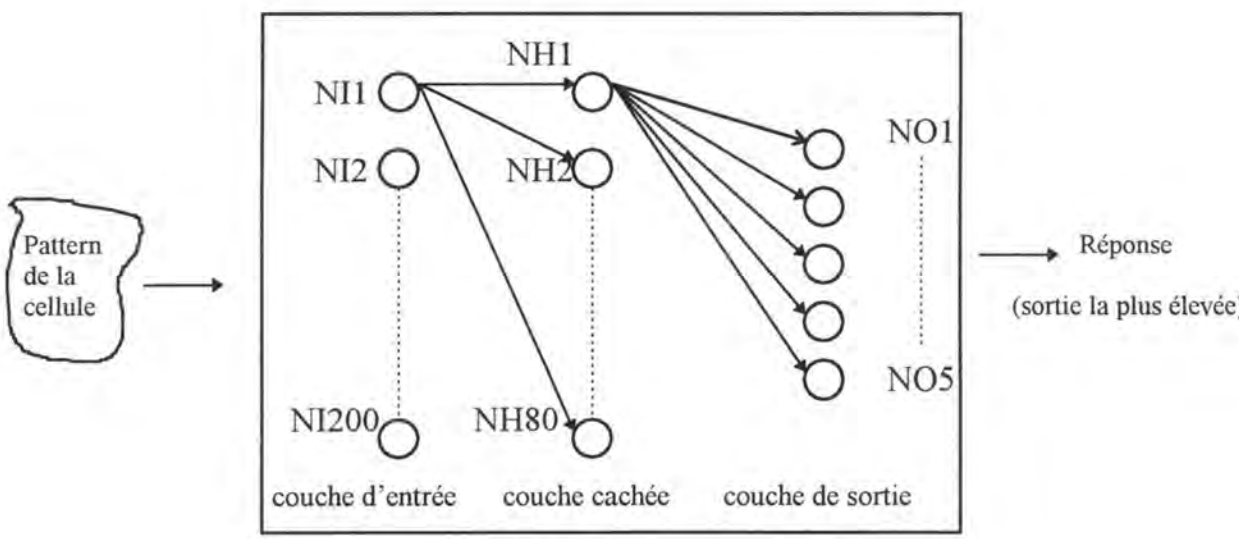


Figure 4.1 Schéma du réseau utilisé pour la reconnaissance des cellules

Ce réseau fonctionne de la manière suivante:

1. Etape d'apprentissage

La présentation d'un ensemble de couples (pattern, 'output' désiré) au réseau permet la modification des poids en vue de l'apprentissage. Le pattern est l'image de la cellule, et l''output' désiré est un vecteur de taille 5 avec 4 éléments de valeur 0, et un seul élément de valeur 1, celui-ci représentant la classe de la cellule présentée.

2. Etape de test

Lors de la présentation d'un pattern, le réseau, par 'feed-forward', calcule l'état d'activation des unités de sortie. La valeur la plus élevée de ces activations est celle du neurone représentant la classe choisie par le réseau. Il ne reste plus qu'à comparer ce choix avec celui de l'expert pour tester l'apprentissage.

5. Traitements automatiques

5.1 Introduction

Notre travail a pour objet d'étudier la faisabilité d'une méthode de reconnaissance d'images dans le cas particulier d'une classification de cellules. Pour ce faire, nous avons développé et utilisé des outils logiciels. Après avoir présenté et justifié nos choix dans le chapitre IV, nous détaillerons les procédures créées et les logiciels choisis. Cette partie se terminera par la présentation d'une architecture globale d'un possible système de classification automatique de cellules.

Le diagramme de flux (figure 5.1) dégage les quatre phases du système. Chacune d'entre elles réalise un ensemble de traitements sur des données de différents types:

A: Les cellules à classer se trouvent dans la boîte de pétri. Ces cellules sont des objets du monde réel.

B: L'ensemble des images (256 niveaux de gris, 512 x 512 pixels) acquises par l'opérateur au moyen de la caméra. Ces images sont sauvegardées dans un format TIFF.

C: Les images sont en noir et blanc et ont subi un traitement de 'nettoyage'.

D: Un fichier d'images est disponible pour mettre en entrée du réseau de neurones.

E: Le réseau donne une réponse telle que le nombre de cellules dont le classement par cette méthode est identique à celui proposé par l'expert.

Phase I: **Prise d'images / codage**
Phase II: **Prétraitement indépendant du RN** ou 'nettoyage'
Phase III: **Prétraitement dépendant du RN** ou prétraitement
Phase IV: **Reconnaissance d'images (le réseau de neurones)**

Si ces quatre phases ont été réalisées tout au long de ce mémoire, ce sont essentiellement les phases III et IV qui ont fait l'objet d'une étude plus approfondie, compte tenu de leur significative interférence avec l'étude de faisabilité du système.

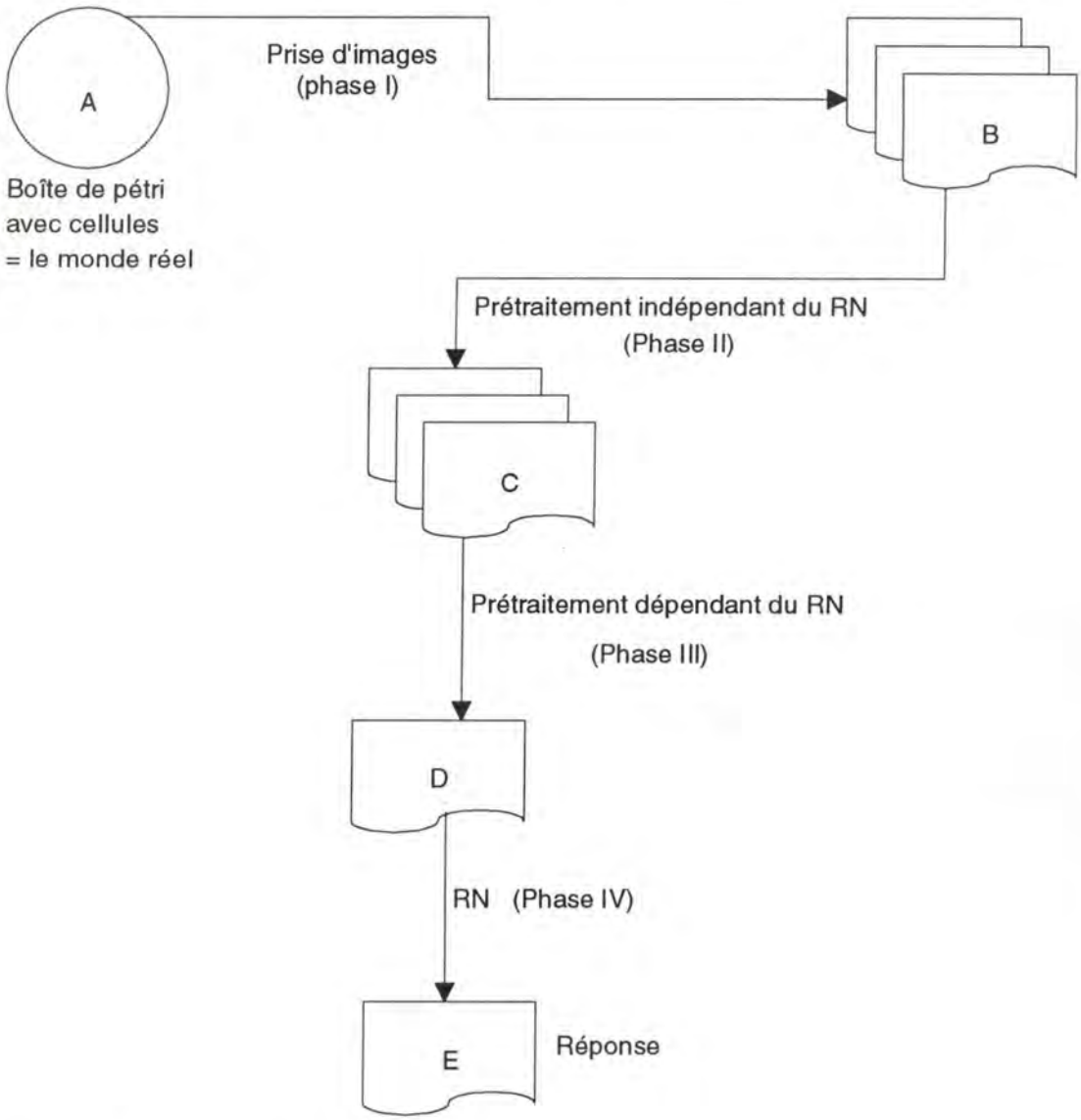


Figure 5.1 Diagramme de flux du système expert

5.2 Phase I: Acquisition et digitalisation de l'image

Le but de ce traitement manuel est d'acquérir, avec un expert, les cellules de différentes classes, en l'occurrence des fibroblastes de classe II, III, IV et V. L'expert choisit des cellules dont la classe d'appartenance n'est pas ambiguë et obtient un panel d'images qui totalise 77 cellules.

Les boîtes de Pétri contenant les cellules fixées et colorées sont placées sous un microscope (Leitz, Labovert FS) sans contraste de phase, ce dernier s'avérant superflu pour des cellules colorées. Le grossissement de ce microscope est de 100 fois (luminosité 7 lum) au niveau des oculaires. Le microscope est couplé, grâce à une bague pourvue d'une lentille de grossissement 0.55 fois, à une caméra CCD noir et blanc de marque Videk (Kodak).

La caméra est reliée à une station SUN Sparc Station II, Visage 110 (Millipore). Sur cette dernière tourne un programme d'analyse de gels d'électrophorèse à une ou deux dimensions (Bioimage, Millipore). Les parties intéressantes de ce logiciel pour notre application sont les sous-programmes de scan et d'acquisition d'images.
Les images scannées sont sauvées sous le format Sun Raster et ont la dimension de 512 x 512 pixels. Ensuite, elles sont traduites en un format TIFF (Tagged Image File Format). Pourquoi un tel format TIFF [Rimmer, 93] ? Son intérêt est qu'il gère des images de taille quelconque et en monochrome comme en 24 bits; les fichiers TIFF peuvent migrer sans problème d'une architecture vers une autre, depuis le PC jusqu'au Macintosh. De plus, il gère les niveaux de gris. Enfin, sa plus importante caractéristique est que ce format n'est pas propriétaire, donc dépendant d'une application, mais au contraire est supporté par une grande gamme de logiciels, dont le logiciel Image 1.31.
Une description moins laconique de la phase I d'acquisition d'images se trouve dans le mémoire de F. Montaigne [Montaigne, 94].

5.3 Phase II: Prétraitement indépendant du réseau de neurones ou 'nettoyage'

Cette phase, réalisée manuellement, comprend différents traitements qui ont été effectués à l'aide du logiciel Image 1.31 sur Macintosh:

1. La réduction du bruit ('nettoyage' au sens strict), soit:

- Elimination des poussières ou déchets cellulaires (petites taches noires),
- Conservation d'une seule cellule par image (*reduce noise* + *gommage manuel*),
- Le seuillage: séparation des cellules du milieu intercellulaire (fonction *threshold*).

2. La binarisation de l'image (mise en noir et blanc): les pixels appartenant à la cellule sont transformés en noir (255) et les autres pixels en blanc (0). A cette fin, on utilise la fonction *make binary*.

3. Le changement de format: le passage du format 500 x 500 pixels au format 100 x 100 pixels se fait grâce à la fonction *scale and rotate* (figure 5.2).

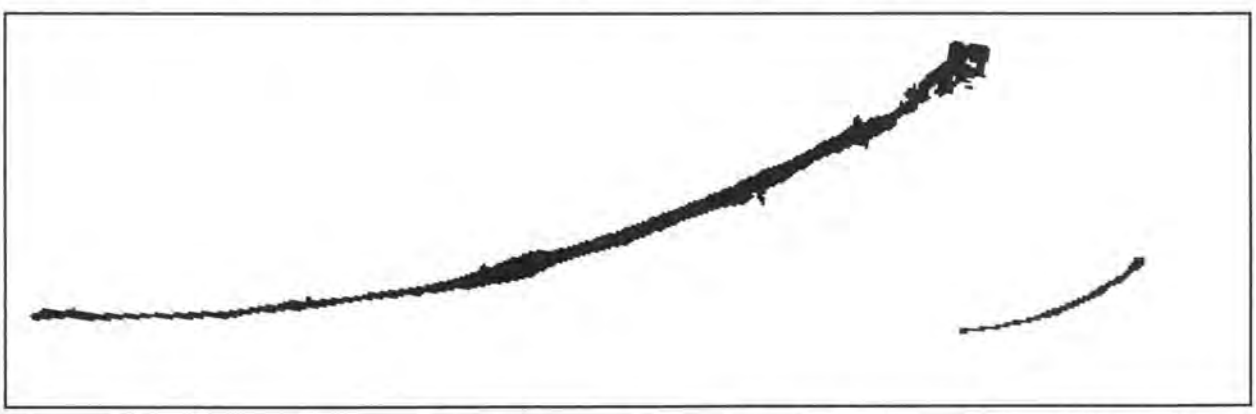


Figure 5.2 Image d'une cellule de format 500 x 500 pixels (à gauche) et 100 x 100 pixels (à droite)

4. La traduction du format TIFF en un format TXT. Dans ce format, chaque pixel est représenté par un caractère égal à '0' si le pixel est blanc et 3 caractères ('255') si le pixel est noir. Cette transformation en un format simplifié se justifie par la faible dimension des images et par la facilité du traitement des fichiers textes.

L'automatisation de cette phase ne constitue pas un des objectifs de ce mémoire car elle n'est pas critique dans l'étude de faisabilité. Cependant, on soulignera dans la conclusion les quelques difficultés que posent cette phase:
. cellules collées
. déchets cellulaires

5.4 Phase III: Prétraitement de l'image dépendant du réseau de neurones ou 'prétraitement'

5.4.1 Introduction

Nous avons vu dans l'introduction (section 3.7) qu'il était primordial de présenter «correctement» les données en entrée du réseau de neurones. Dans ce chapitre nous décrirons, dans un premier temps, la démarche de transformation de données utilisées et sa raison d'être. Dans un second temps, nous détaillerons les fonctionnalités issues de l'analyse de cette partie du problème. Le code de ces fonctionnalités, en Turbo Pascal, se trouve en annexe (A18-A33).

5.4.2 Les transformations

Le Diagramme de flux (figure 5.3) montre les actions successives des traitements sur un ensemble d'images de type 1 fournies par la phase II du système. Nous obtiendrons, finalement, un fichier qui pourra servir de 'set' d'apprentissage ou de 'set' de test (voir section 5.5).
Le format des «images», tout au long ,des traitements est le suivant:

image 1: codée par une matrice de 100 x 100 pixels de type caractère ('0' signifie blanc et '255' signifie noir)
image 2: codée par une matrice de 100 x 100 pixels de type caractère ('0' signifie blanc et '1' signifie noir)
image 3: codée par une matrice de 100 x 100 pixels de type caractère ('0' signifie blanc et '1' signifie noir)
image 4: codée par 2 lignes de 100 chiffres représentés par des caractères
image 5: codée par 2 lignes de 100 chiffres représentés par des caractères
image 6: codée par 2 lignes de 100 chiffres représentés par des caractères
image 7: codée par 2 lignes de 100 chiffres représentés par des caractères
image 8: codée par 4 lignes de 100 chiffres représentés par des caractères
image 9 ou 9': image 8 ou 5 à laquelle s'ajoute les variables d'entraînements (5 variables) ainsi que la classe de la cellule (1 variable)
fichier 10: suite d'image 9 ou 9'

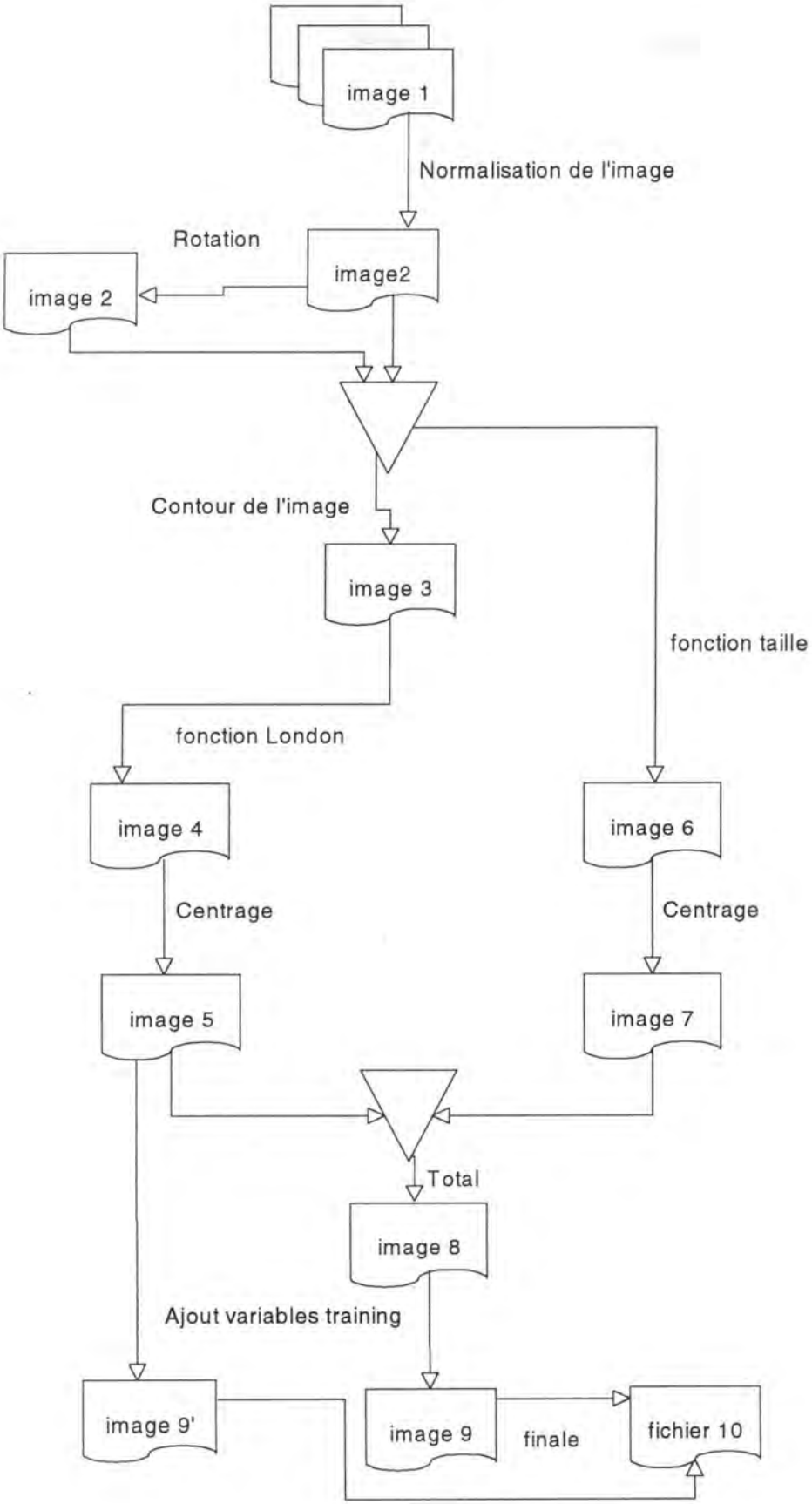


Figure 5.3 Diagramme de flux des fonctionnalités de prétraitement

Notons que nous utilisons le terme 'image' au sens large c'est-à-dire comme une information sur un objet du réel qui peut avoir une signification dans le cadre de notre système de reconnaissance, en l'occurrence le réseau de neurones. Il est clair qu'une image de type 8 est difficilement décodée par un être humain.

Le terme 'fichier' a lui aussi une signification particulière; il constitue une suite d'images dans laquelle chacune d'entre elles est représentée par un enregistrement. Nous conserverons cette nomenclature par la suite.

Chaque cellule est représentée par une image binaire (donc noir et blanc) de format 100 x 100 pixels. Ce format arbitraire permet de conserver suffisamment d'informations sur la forme de la cellule tout en évitant une surcharge de données. Cette image binaire correspond à l'image 1 de la figure 5.3.

Description des fonctionnalités

1. La fonctionnalité 'normalisation de l'image' transforme celle-ci en un format plus simple à traiter pour les algorithmes qui suivent.

2. La fonctionnalité 'rotation' émule trois rotations de 90° : à partir d'une image, cette fonctionnalité en fournit 3 nouvelles. L'image initiale subit une rotation de 0°, les trois autres subissent respectivement une rotation de 90°, 180° et 270°.
Cette fonctionnalité a pour but de donner un conseil intelligent au réseau: « quelque soit l'orientation de la cellule, celle-ci fait partie de la même classe ». Ainsi, le réseau apprend à ne pas tenir compte de l'orientation de la cellule.
Cependant, les tests préliminaires ont montré qu'il fallait affiner les rotations pour parvenir à rendre indépendant le système à cette transformation géométrique. Pour cela, nous effectuerons (en phase II), une rotation de 45° grâce à Image 1.31. Chacune des 2 images obtenues est traitée par la fonctionnalité rotation. A partir d'une image, nous en obtenons 7 autres qui correspondent à la même cellule mais dans 7 orientations différentes. Le schéma 5.4 montre les 8 orientations d'une cellule de classe IV.

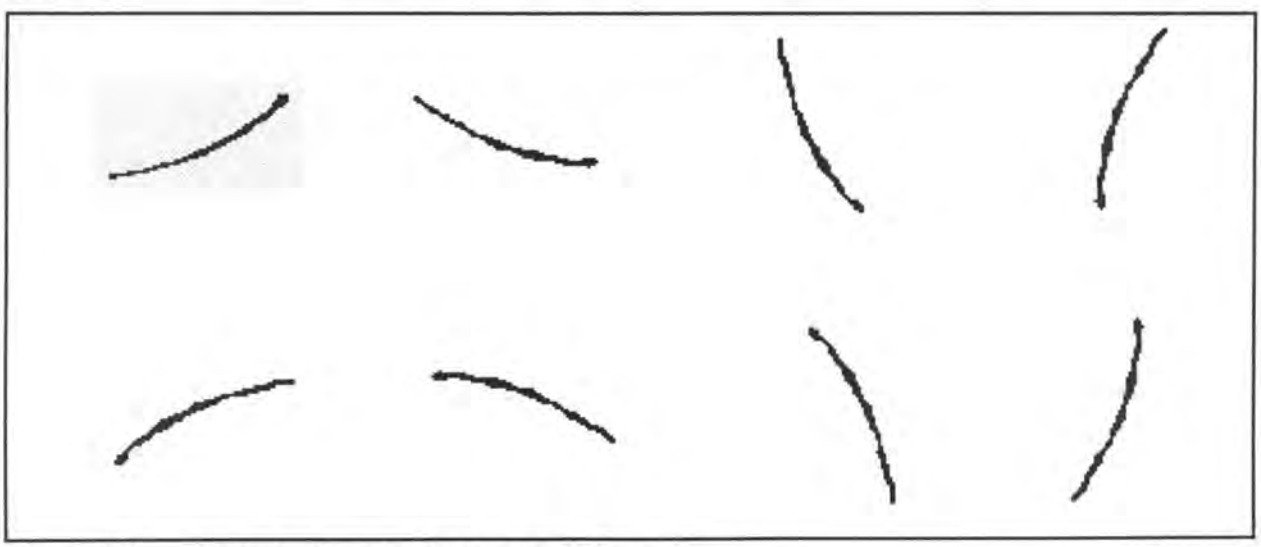


Figure 5.4 Cellule dans les huit orientations

3. La fonctionnalité 'contour de l'image' ne conserve que le contour de l'image (figure 5.5.e). Cette fonctionnalité est créée pour des raisons plutôt méthodologiques, elle permet de mieux comprendre la fonctionnalité 'London' (qui peut être implantée telle quelle). De toute façon, il est clair que l'on ne perd pas d'informations en ne conservant que le contour d'une forme noir sur fond blanc.

4. La fonctionnalité 'London' nous a été suggérée par une équipe de Londres (communication de J. Johnson) travaillant sur la reconnaissance des caractères par la méthode des réseaux de neurones. Son utilité est double:

a. Diminuer la taille des données en entrée du réseau afin de prendre en compte:
- les contraintes techniques du logiciel utilisé (pas plus de 500 entrées)
- les contraintes de temps calcul avec cette génération de PC

b. Proposer, en entrée du réseau, des données pertinentes et non redondantes. Ceci est vivement recommandé pour faciliter un apprentissage (section 3.7).

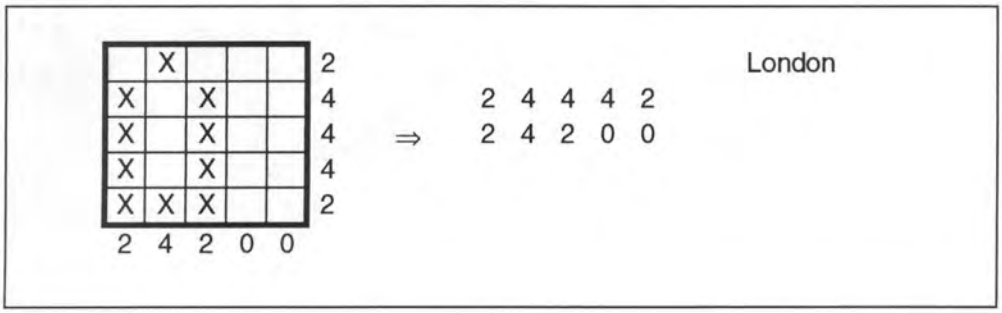
La méthode consiste à compter les changements horizontaux et verticaux et utiliser ces comptages comme données d'entrée du réseau. Le nombre à côté de chaque ligne correspond au nombre de changements du blanc vers le noir et du noir vers le blanc sur cette ligne. C'est le même principe pour le nombre en dessous de chaque colonne. (figure 5.5.a)

Remarque
La partie extérieure de l'image est, par définition, blanche ce qui implique que le nombre de changements est toujours pair. Ce nombre sera d'ailleurs divisé par deux.

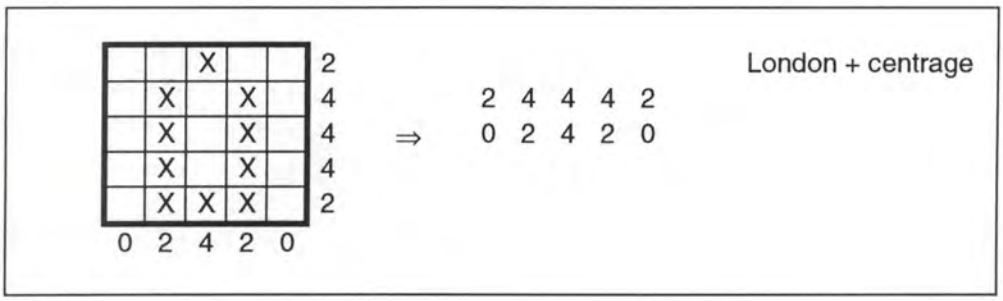
5. La fonctionnalité 'centrage' permet de rendre le système indépendant du cadrage de la cellule dans l'image. Chacune des deux lignes de 'l'image 4' sera centrée (figure 5.5.b et 5.5.d). L'intérêt de cette transformation est d'enlever une des deux transformations géométriques (pour rappel: la translation et la rotation), en l'occurrence la translation. Nous préservons donc l'invariance de la translation. Par opposition à la théorie des conseils (section 3.7), il n'y a pas d'apprentissage d'invariance de translation. On utilise une astuce pour passer outre cette transformation géométrique. On peut constater, sur la figure 5.3, que la fonctionnalité 'centrage' est utilisée soit après la transformation 'London', soit après la transformation 'taille'. Dans ces deux cas le but est le même.

6. La fonctionnalité 'taille' a pour but d'utiliser l'information perdue par la transformation 'London' et qui pourrait s'avérer nécessaire à une optimalisation des résultats. L'algorithme proposé est proche de celui de 'London'; au lieu de compter les changements horizontaux et verticaux, on compte le nombre de pixels appartenant à la cellule sur une ligne ou une colonne. Le format obtenu est identique à celui acquis par la transformation 'London' et est présenté sur la figure 5.5.c.

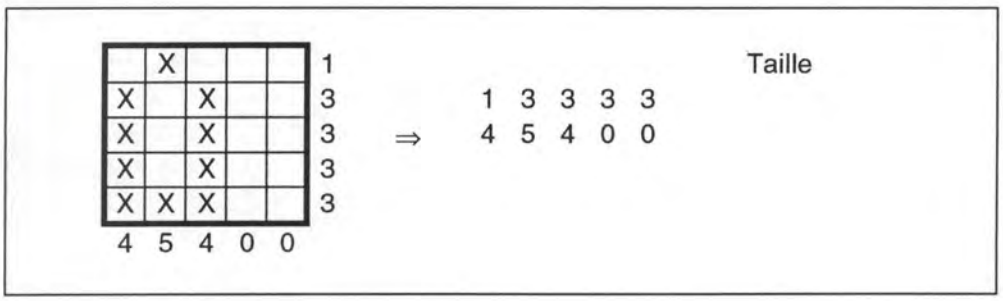
(a)


2
4
4
4
2
2 4 2 0 0
⇒
2 4 4 4 2
2 4 2 0 0
London

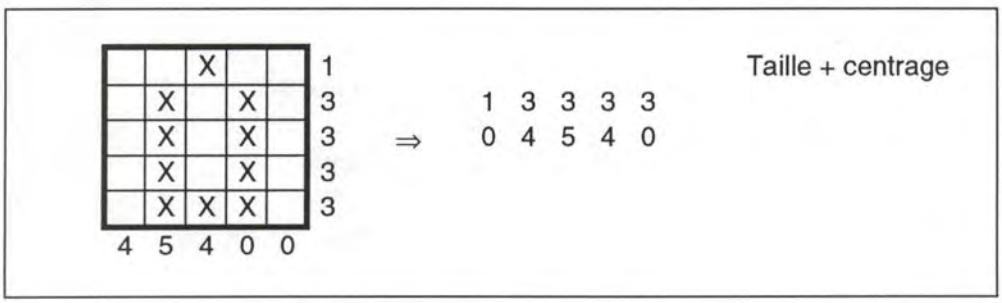
(b)


2
4
4
4
2
0 2 4 2 0
⇒
2 4 4 4 2
0 2 4 2 0
London + centrage

(c)


1
3
3
3
3
4 5 4 0 0
⇒
1 3 3 3 3
4 5 4 0 0
Taille

(d)


1
3
3
3
3
4 5 4 0 0
⇒
1 3 3 3 3
0 4 5 4 0
Taille + centrage

(e)

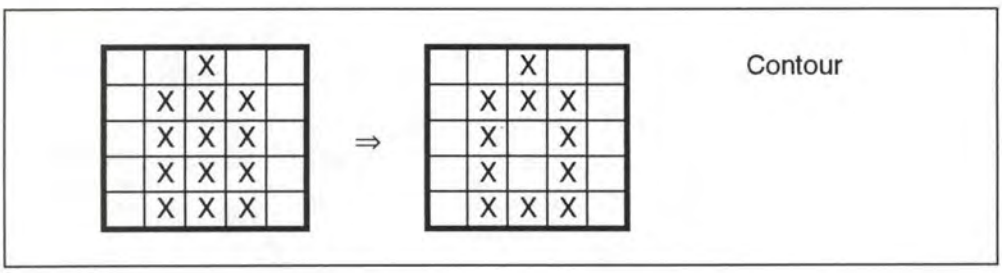

⇒
Contour

Figure 5.5 Présentation schématique des modifications de l'image effectuées par les fonctionnalités (a) 'London', (b) 'London' + 'centrage', (c) 'taille', (d) 'taille' + 'centrage' et (e) 'contour'.

7. La fonctionnalité 'total' met à la suite l'une de l'autre les 'images 5' et '7' pour obtenir 'l'image 8' (figure 5.3). Cette fonctionnalité est implémentée afin d'ajouter une information sur la taille de la cellule à l'information sur sa forme. L'information principale sur la forme de la cellule est celle fournie par la transformation 'London'. L'algorithme ajoute à la suite de cette information, les renseignements issus de la fonctionnalité 'taille'. Il est important de constater que l'ordre des renseignements ne peut influencer le réseau à partir du moment où il reste cohérent au cours d'un cycle apprentissage/test.

Remarque
Nous verrons dans les résultats, l'importance de mettre sur le même pied les informations en entrée du réseau de neurones, qui devra lui-même se structurer pour prendre en compte l'information qui maximisera la reconnaissance. Il ne s'agit pas de présenter au réseau une seule donnée sur la taille et 200 données sur la forme ('London') car il y a peu de chances qu'il puisse donner une signification à cette seule donnée.

8. La fonctionnalité 'ajout variables training' vise à ajouter les variables d'apprentissages et la solution à l'image de type 8: par exemple une cellule de classe IV aura comme variables d'apprentissages: '00010' (le $4^{ème}$ caractère est à un et les autres à 0) et comme solution le caractère '4'. Ceci est vrai pour les deux types d'utilisation d'images (test ou apprentissage). Nous verrons, dans la section 5.5, que les variables d'apprentissages ne seront pas utilisées pour l'étape de test. On obtiendra une image de type 9 ou 9'.

9. La fonctionnalité 'finale' ajoute simplement les images de type 9 ou 9' une à la suite de l'autre et forme un fichier de **taille = [200 (ou 400) + 6] caractères x n images**. Les 200 caractères correspondent à l'information de type 'London' et les 400 caractères correspondent à l'information issue de 'total'. Les 6 derniers caractères de l'image sont les 5 variables d'entraînement et la classe de la cellule.

Les Algorithmes

Nous décrivons, ci-dessous, les algorithmes des principales fonctionnalités utilisées par notre système. Les algorithmes des fonctionnalités 1, 7, 8, 9, les plus élémentaires, ne sont pas décrits dans cette section mais le code de l'ensemble des fonctionnalité se trouve en annexe A18 à A33.

1. 'normalisation' de l'image
IN: n 'image 1'
OUT: n 'images 2'

Cette fonctionnalité transforme les 255 en 1, ce qui simplifie le traitement de l'image, nous obtenons ainsi l'image 2.

2. 'rotation'
IN: 'image 2'
OUT: 4 'images 2'

Soit un tableau à 2 dimensions ('image 2') de taille 100 x 100

```
For ligne : = 1 to 100
        For colonne : =  1 to 100
        Begin
                t 90° (colonne, ti - ligne) : = t O° (ligne, colonne)
                t 180° (colonne, ti - ligne) : = t 90° (ligne, colonne)
                t 270° (colonne, ti - ligne) : = t 180° (ligne, colonne)
        end.
```

3. 'contour'
IN: 'image 2'
OUT: 'image 3'

La règle de l'algorithme est la suivante: soit 1 pour noir, 0 pour blanc et -1 pour noir remplacé par blanc. Un 1 devient -1 si et seulement si ses huit plus proches voisins sont soit à 1 ou à -1. Une dernière boucle transforme les -1 en 0.

4. 'London'
IN: 'image 4'
OUT: 'image 5'

Le principe de l'algorithme 'London' est de compter le nombre de changements (passage de 1 à 0 et de 0 à 1) qui surviennent dans une ligne. Les itérations de l'algorithme vont calculer dans un premiers temps le nombre de changements pour chacune des lignes, par la suite le nombre de changements pour chacune des colonnes de la matrice. En sortie, 'l'image' sera constituée de 2 lignes de 100 chiffres de 0 à 9 (nombre de changements sur une ligne). Une ligne pour les changements verticaux et une seconde ligne pour les horizontaux.

```
(*taille image (ti) = 99*)
(*BOUCLE1*)
      for compy:=0 to ti do
      begin
      change:=0;couleur:=0;
      for compx:=0 to ti do
```

```
        begin
        if (proj[compx,compy]='1')then
        begin
            if couleur=0 then
            begin
              change:=change+1;
              couleur:=1;
            end;
        end;
        if ((proj[compx,compy]='0') and (couleur=1))then
        begin
              change:=change+1;
              couleur:=0;
              end;
        end;
        if proj[ti,compy]='1' then change:=change+1;
        change:=change div 2;
        if change >= 9 then lond[compy,1]:=(suite[9])
        else lond[compy,1]:=(suite[change]);
    end;

    (*BOUCLE2*)
    for compx:=0 to ti do
    begin
    change:=0;couleur:=0;
    for compy:=0 to ti do
        begin
        if ((proj[compx,compy]='1') and (couleur=0))
          then
              begin
              change:=change+1;
              couleur:=1;
              end;
        if ((proj[compx,compy]='0') and (couleur=1))
          then begin
              change:=change+1;
              couleur:=0;
              end;
        end;
    if proj[compx,ti]='1' then change:=change+1;
    change:=change div 2;
    if change >= 9 then lond[compx,2]:=(suite[9])
    else lond[compx,2]:=(suite[change]);
    end;
end;
```

5. 'centrage'
IN: 'image 4' ou 'image 6'
OUT: 'image 5' ou 'image 7'

```
      mi:=ti div 2;
      for compli:=1 to 2 do
      begin
          ga:=0;dr:=ti;
          while lond[ga,compli] ='0' do
              ga:=ga+1;
          while lond[dr,compli] ='0' do
              dr:=dr-1;
          decalage:=mi-((ga+dr) div 2);
          for compco:=ga to dr do
             centr[decalage+compco,compli]:=lond[compco,compli];
          for compco:=0 to ga-1+decalage do centr[compco,compli]:='0';
          for compco:=dr+1+decalage to ti do centr[compco,compli]:='0';
      end;
  end;
```

6. 'taille'
IN: 'image 6'
OUT: 'image 7'

```
(*BOUCLE1*)
for compy:=0 to ti do
      begin
      tot:=0;
      for compx:=0 to ti do
         begin
         if (proj[compx,compy]='1') then
             tot:=tot+1;
         end;

      case tot of  0..0:sc:='0';
                   1..5:sc:='1';
                   6..10:sc:='2';
                   11..15:sc:='3';
                   16..20:sc:='4';
                   21..25:sc:='5';
                   26..40:sc:='6';
                   41..55:sc:='7';
                   56..73:sc:='8';
                   74..99:sc:='9';
      end;
```

```
    tail[compy,1]:=sc;
    end;
    (* FIN1*)


(*BOUCLE2*)
for compx:=0 to ti do
    begin
    tot:=0;
    for compy:=0 to ti do
      begin
      if (proj[compx,compy]='1') then
         tot:=tot+1;
      end;

    case tot of  0..0:sc:='0'; (*  codification de la taille*)
                 1..5:sc:='1';
                 6..10:sc:='2';
                 11..15:sc:='3';
                 16..20:sc:='4';
                 21..25:sc:='5';
                 26..40:sc:='6';
                 41..55:sc:='7';
                 56..73:sc:='8';
                 74..99:sc:='9';
    end;
    tail[compx,2]:=sc;
    end;
    end;
    (*FIN2*)
```

Chaque information sur le nombre de pixels d'une ligne ou d'une colonne prend une valeur de 0 à 100. Comme nous utilisons des caractères, ce type d'information est codé sur 10 caractères (0 à 9) selon la codification utilisée dans la partie « case of » de l'algorithme.

L'implémentation de ces fonctionnalités (annexe A18 à A33) subit quelques petites modifications car il n'y a pas de fonctionnalités de coordination, d'interface homme-machine et de base de données.

Le programme celltot permet:

si **option 1**, de transformer les 'images 1' en 'images 2'

si **option 2**, de transformer les 'images 2' et de les mettre en séquences dans 4 fichiers: un fichier d'images contour (contrôles), un fichier de noms, 1 fichier d'image de type 9 (fichier 10) et un fichier d'images de type 9' (fichier 10).

Le programme cellrot permet d'émuler la rotation de 90°: 90-180-270 à partir d'images de type 2 et de donner quatre fois plus d'images de type 2 en sortie.

5.4.3 Le choix des cellules

Toutes les fonctionnalités de transformations permettent de sélectionner le type de données que l'on va mettre en entrée du réseau ('taille', 'London', ...) afin d'aider celui-ci à devenir indépendant de la variabilité due aux transformations géométriques et de faciliter la généralisation, c'est-à-dire l'apprentissage en proposant des données aussi pertinentes et minimales que possible.

Mais une dernière fonctionnalité ('finale') relativement cachée dans le diagramme de flux est plus importante qu'elle n'y paraît. Elle rassemble toutes les images, dans un ordre donné, dans un ficher qui va servir de 'set' d'apprentissage ou de test.

Deux possibilités s'ouvrent à nous: quelles cellules choisir et quel ordre pour ces cellules dans la réalisation du 'set' d'apprentissage ou de test?
En ce qui concerne le 'set' de test, l'ordre n'a pas d'importance et le choix des cellules est quelconque, excepté qu'il ne peut se trouver de cellules appartenant au 'set' d'apprentissage de la session courante et que les cellules dont la classe est ambiguë ne font pas partie du 'set' des tests.

Pour la réalisation du 'set' d'apprentissage, différentes sessions présentées dans les résultats permettront de répondre à ces questions.
Il est clair que cette fonctionnalité centrale devrait se trouver à un niveau supérieur de l'architecture mais dans notre travail, nous l'avons simplement intégrée dans le programme celltot.

L'utilisation d'une nomenclature dans l'identification des cellules permet de faciliter l'automatisation de ces fonctionnalités. Voici la nomenclature utilisée pour identifier les images (cellules): par exemple, l'image N210 représentant une cellule de classe II avec

N: groupe de cellules
2: classe de la cellule
1: numéro d'ordre de la cellule
0: rotation de la cellule de 0 à7 (0 correspondant à la position d'origine de la cellule et 1 à une rotation de + 45° etc, jusqu'à 315°).

5.5 Phase IV: HNC, un logiciel de simulation de réseaux de neurones

5.5.1 Définitions

Comme nous l'avons vu dans la section 3.4, l'utilisation classique d'un réseau de neurones de type 'back-propagation' comprend trois étapes: apprentissage, test et production. Ces trois étapes constituent une session du réseau. Lors des essais du système de reconnaissance de cellules nous réaliserons différentes sessions en changeant les paramètres, le 'set' d'apprentissage, ... Ceci nous permettra d'évaluer le système. Dans notre étude de faisabilité nous ne réaliserons pas l'étape de production, car celle-ci entre en compte dans une utilisation pratique du réseau qui a appris et a été testé. L'étape de test est capitale car elle permet de connaître le pourcentage d'erreurs d'un 'RN' ayant appris à reconnaître un pattern.

Quelques définitions

'Set' d'apprentissage ou de test: D'un point de vue logique, il s'agit d'un ensemble d'images, représentant chacune une cellule, choisi pour réaliser un apprentissage ou un test sur un réseau. Le fichier (type 10) est la matérialisation de cet ensemble; chaque enregistrement de celui-ci étant un vecteur/image avec des variables d'entraînement. Nous utiliserons le même type de fichier pour les étapes d'apprentissage ou de test. Toutefois, les variables d'entraînement ne seront pas fournies en entrée du 'RN' pour une étape de test.

Cycle: Le cycle est l'unité élémentaire d'une session réseau de neurones. Le vecteur/image ou pattern est fourni en entrée du réseau. La première partie d'un cycle consiste en une propagation vers l'avant de cet 'input', jusqu'au calcul de 'l'output'. Ensuite, l'erreur est calculée par comparaison à 'l'output' attendu ('training'). Dans la deuxième partie du cycle (uniquement pour un apprentissage), l'erreur est rétro-propagée dans les couches de sortie et cachée.

'Run': Un 'run' comprend de 1 à n cycles; le nombre de cycles correspond au nombre d'images du fichier. Un 'run' consiste donc à présenter l'ensemble des cellules d'un 'set' au réseau.

'Loop': Un 'loop' comprend de 1 à n répétitions du 'run'; il s'agit donc du nombre de fois que le 'set' sera présenté au réseau de neurones. Pour un test, il faut le présenter une fois, pour un apprentissage il est nécessaire de le présenter plusieurs fois.

Etape: Une étape est définie par un nombre de 'loop' et les paramètres du réseau pour cette étape ('run-time' et 'load-time' du réseau de neurones). L'étape est l'unité fonctionnelle de base de notre système. Elle est de type apprentissage ou test.

Session: Suite d'étapes permettant d'évaluer les paramètres du réseau de neurones et/ou le 'set' d'apprentissage. Une session comprend au moins une étape d'apprentissage et une étape de test.

5.5.2 Introduction

La phase IV du diagramme des flux, figure 5.1, est la phase durant laquelle une suite d'images vont être proposées en entrée d'un logiciel de réseau de neurones. Cette phase utilise le logiciel ExploreNet 3000 release 2.12. Ce logiciel tourne sur un système Windows 3.1.

ExploreNet est un ensemble unifié d'outils logiciels pour la création et l'application de réseaux de neurones. Non seulement, il supporte la création de différents types de réseaux de neurones, mais il fournit des outils de transformations, d'affichages et d'analyses de données. ExploreNet est utilisé pour construire et faire fonctionner des systèmes qui contiennent un réseau. Les différents types d'outils proposés par ExploreNet sont appelés modules. Ces modules ont une représentation graphique et effectuent diverses opérations de calculs ou affichages sur les données. Ces modules peuvent être connectés entre eux, ces connexions étant représentées par des flèches. Il s'agit donc de construire sa propre application en choisissant des modules pertinents et en les paramètrant de manière adhoc, mais aussi en réalisant des connexions qui permettront de visualiser un véritable flux de données.

L'interface homme-machine est décrite de la manière suivante (figure 5.6):
- Le titre de la fenêtre principale de l'application: ExploreNet - Nom fichier
- Une barre de menu, au dessus, avec diverses fonctions générales (items Help, File, Create, ...)
- Une boîte à outils à gauche dans laquelle se trouve l'ensemble des modules utilisables ainsi que leur liaisons.
- Le reste de la fenêtre constitue la zone de travail dans laquelle l'application sera construite.

Parmi l'ensemble des modules disponibles, on trouve un module d'interface avec un fichier de données en entrée (File Modules), un module de présentation de données (Graph Modules), ...
Cependant un module sera décrit plus en détail, c'est le module du réseau proprement dit: le 'Network Modules'. Ce dernier module sert à piloter le logiciel Neurosoftware. Neurosoftware est composé d'une structure de données et des opérations effectuées sur cette structure. Le détail de ces structures, pour un réseau de type 'back-propagation' est décrit ci-après.

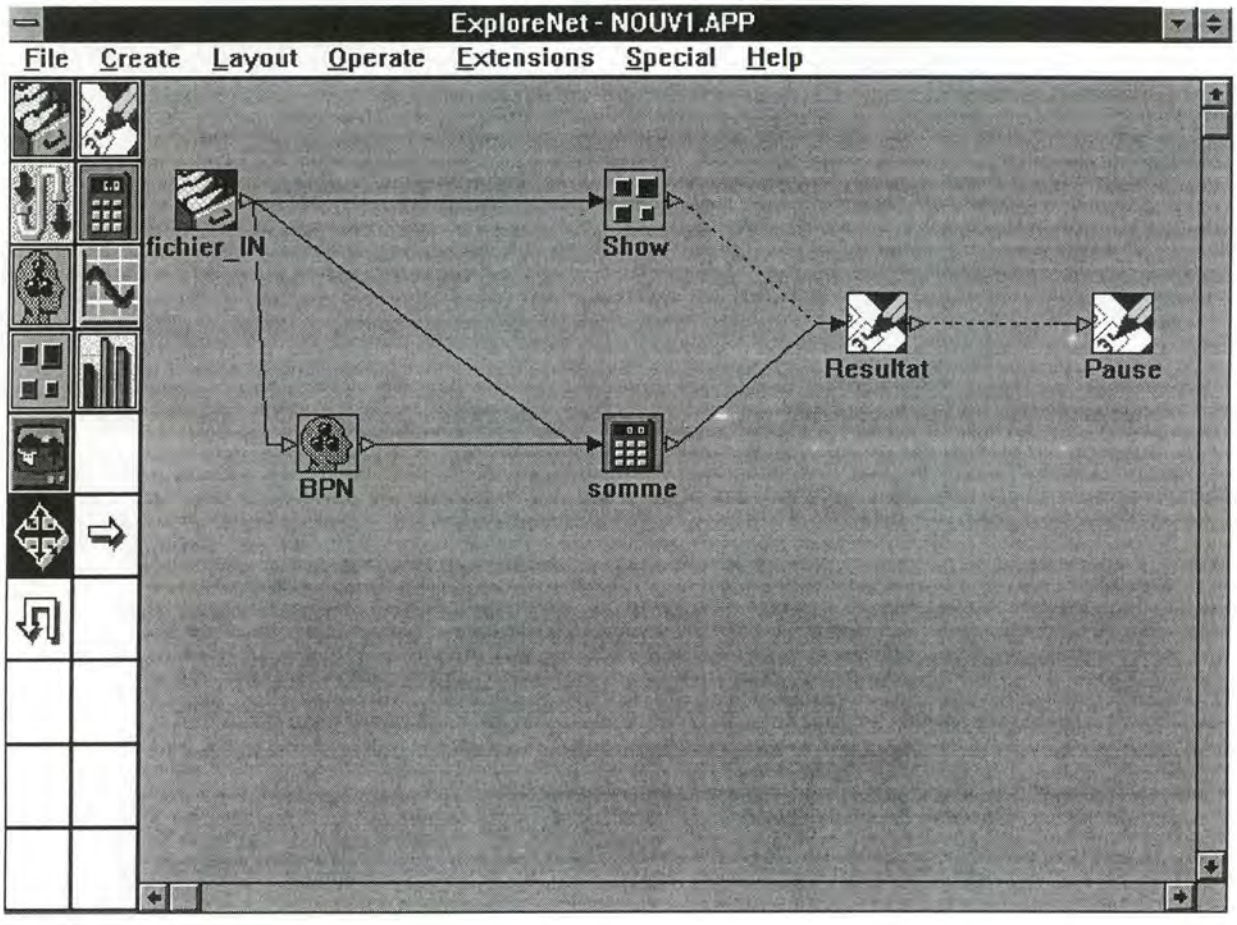


Figure 5.6 Présentation de l'interface du logiciel HNC

Structure de données:

Quatre types de données sont utilisées par l'application. Les paramètres de structure et d'apprentissage du réseau sont appelés 'Constantes'.

Constantes (CTS) 'Load-time': ces paramètres sont toujours les mêmes au cours d'une session, ils concernent l'architecture globale du réseau mais aussi la gestion des paramètres d'activation des neurones.

'Run-time': ces paramètres peuvent changer durant une session, ils concernent les règles d'apprentissage.

Etats (STS) Chacune des sorties de chaque neurone au moment de la sauvegarde.

Poids (WTS) Ensemble des valeurs des poids de chacune des connexions inter-neurones du réseau.

Données locales (LCL) Chacune des variables locales des neurones autres que le poids de la connexion et l'état du neurone.

Pour le paradigme du 'back-propagation', il suffit d'enregistrer à la fin de chaque étape les constantes et les poids. Les autres données ne sont utiles que dans les cycles d'une même étape. Les poids représentent la mémoire du réseau. Initialement, avant toute étape, ils ont des valeurs au hasard. Après une étape d'apprentissage, ils représentent l'acquis du système et sont donc enregistrés. Ils ne sont plus modifiés dans les étapes de test.

'Load-time' (I, H, O, C, M) (figure 5.7)

1. Les paramètres de la taille du RN: nombre de neurones pour la **couche d'entrée** (**I**), la **couche intermédiaire** (**H**) (BPN) ou les couches intermédiaires (H1, H2, ...) (MBPN), la **couche de sortie** (**O**).

2. Les paramètres *miscellaneous*: la **connexion** des entrées aux sorties (**C**) permet la structuration d'une relation de linéarité entre la couche **I** et la couche **O**.
Les paramètres d'initialisation des poids (initial weight maximum et random seed) seront toujours instanciés respectivement à 0,5 et à 0.
La **méthode d'apprentissage** (**M**) sera soit 'batching', soit 'smoothing'; l'intérêt de ces méthodes a été discuté dans la section 3.6 .

3. Les paramètres de la fonction d'activation des neurones: fonction d'activation de type logistique (sigmoïde) répondant à la formule suivante: $f(x) = (p2\text{-}p3/1+e^{-p1x}) + p3$ avec p3: la limite inférieure, p2: la limite supérieure et p1: la pente (λ). Ces valeurs sont instanciées à 0 por p3 et à 1 pour p2 et à 1 pour p1.
Il est possible d'utiliser d'autres fonctions d'activation.

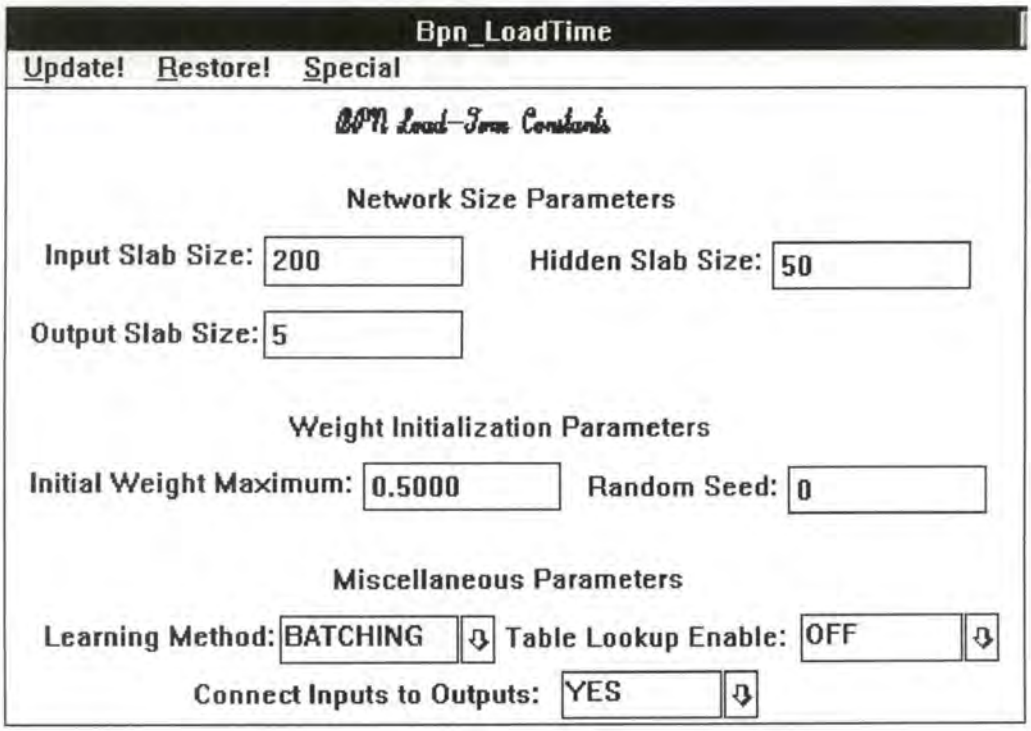


Figure 5.7 Fenêtre d'acquisition des paramètres 'load-time'

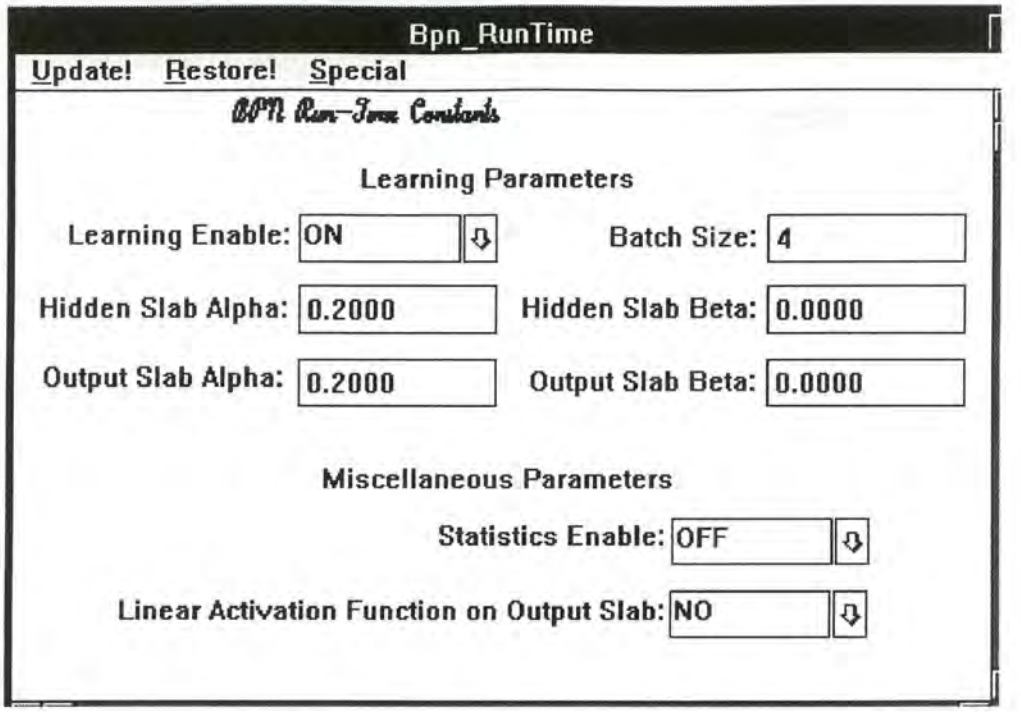


Figure 5.8 Fenêtre d'acquisition des paramètres 'run-time'

Run-time (T, AH, BH, AO, BO) (figure 5.8)

1. les paramètres de contrôle d'apprentissage:
   - La possibilité d'apprentissage, est à 'ON' si c'est une étape d'apprentissage et à 'OFF' si c'est une étape de test.
   - La méthode d'apprentissage (cf 'load-time').
   - La **taille du batching** (**T**) doit être indiquée: de 2 à n en mode 'batching'.

2. les paramètres du taux d'apprentissage pour chacune des couches, sauf la couche d'entrée (interface, pour rappel): alpha (**AH** pour la couche cachée, **AO** pour la couche de sortie). La façon dont les poids changent est déterminée par les lois d'apprentissages des neurones. Si l'apprentissage est possible, le vecteur de sortie courant est comparé au vecteur désiré, et l'erreur entre les deux vecteurs est calculée. Les valeurs d'erreurs sont utilisées pour calculer de nouveaux poids pour tous les neurones de sortie et des couches intermédiaires et en conséquence pour réduire l'erreur en sortie du réseau. Ce processus est répété jusqu'à ce que l'apprentissage soit réalisé ou que le 'RN' ait appris aussi loin qu'il le peut. L'idée est de trouver un ensemble de poids qui minimisent l'erreur globale. Le critère d'erreur utilisé pour le BPN est le « Mean Squared Error » (MSE). Pour un ensemble donné d'entrées/variables d'entraînements, le MSE est la moyenne de toutes les paires des carrés des différences entre l''output' désiré (training) et l''output' courant.
L'équation du 'feed-forward' c'est-à-dire de la progression vers l'avant est incrémentée à chaque cycle, que l'étape soit de type apprentissage ou test. Par contre, lors de l'apprentissage la rétropopagation des erreurs suit le principe discuté.

Trois formules sont présentées pour les trois méthodes d'apprentissage:

Normal	$w_{lij}^{new} = w_{lij}^{old} + \alpha\, d_{li}\, z_{(l-1)j}$	
	avec $d_{li} = f'(I_{li})\,(t_i - z_{li})$	
Batching	$w_{lij}^{new} = w_{lij}^{old}$	/ if count ≠ taille du 'batching'
	or	
	$w_{lij}^{new} = w_{lij}^{old} + (\alpha\, d_{li}\, z_{(l-1)j} + w_{lij}^{old})$	/ if count = taille du 'batching'
	avec $d_{li} = f'(I_{li})\,(t_i - z_{li})$	
Smoothing:	$w_{lij}^{new} = w_{lij}^{old} + (1-\beta)\,\alpha\, d_{li}\, z_{(l-1)j} + \beta\, \Delta w_{lij}^{old}$	
	avec $d_{li} = f'(I_{li})\,(t_i - z_{li})$	

Les constantes α et β de ces formules sont choisies par l'utilisateur. Le taux d'apprentissage α est utilisé pour toutes les méthodes d'apprentissage tandis que β est utilisé pour le 'smoothing' (**BH** pour la couche cachée, **BO** pour la couche de sortie). Le choix adéquat d'une méthode d'apprentissage et de valeurs de constantes permet un apprentissage rapide et ne s'arrêtant pas dans n'importe quel minimum local.

3. Les paramètres divers ne concernent pas l'application.

Les calculs effectués par Neurosoftware sur la structure de données ne seront pas précisés, cependant ils sont semblables à ceux décrits dans la partie théorique sur l'approche réseau de neurones.

5.5.3 Description de l'application

Nous avons construit une application type que nous utiliserons dans tous nos tests, cette application est donc un raffinement de la phase IV (figure 5.1). La zone de travail de l'application *Nouv1* (figure 5.6) comprend les modules suivants:

- Le module *fichier_IN* (type File) met en relation l'application et le fichier d'entraînement ou de test. Les paramètres de ce module sont le nom du fichier d'entrée (pour rappel le fichier 10 de la figure 5.2) et la taille des enregistrements (images) de ce fichier.

Un fichier de type 10, qui servira d'entrée au réseau, comprend, par exemple, 8 enregistrements ou images de type 9. Chaque enregistrement a la structure suivante:

- les 200 premiers caractères représentent les données issues de la fonctionnalité 'London',
- les 200 suivants sont les données issues de la fonctionnalité 'taille',
- le 401ème caractère représente la classe de la cellule
- les variables d'entraînements sont codées sur les caractères 402 à 406.

La taille de ce fichier d'apprentissage est de 406 x 8 (3248 caractères).

- Le module *Show* (type Tile) permet un affichage graphique des données sous la forme de rectangles codés. Ceci a pour but de contrôler visuellement les images.

- Le module *BPN* (type Network) traite les enregistrements un par un selon un algorithme qui est fonction du paradigme du 'RN' utilisé.

- Le module *Somme* (type Data) permet de comptabiliser les réponses fournies par le réseau et éventuellement de les comparer avec la solution correcte.

- Le module *Resultat* (type Form) permet d'afficher les résultats.
Les flèches continues symbolisent le flux des données entre les différents modules de traitements. De plus, différentes options gèrent la façon dont l'application va se dérouler (par ex: 5 cycles, ...).

La fenêtre *Resultat* (figure 5.9) indique les scores obtenus par les 5 sorties du réseau (score 1 à 5), le meilleur score est choisi par l'application qui inscrit la classe correspondante dans la case proposition. On peut comparer visuellement la proposition du réseau à la réponse correcte de l'expert (correction).
Le nombre total d'images, et le nombre d'images correctement classées par le réseau de neurones permettent d'évaluer l'apprentissage.

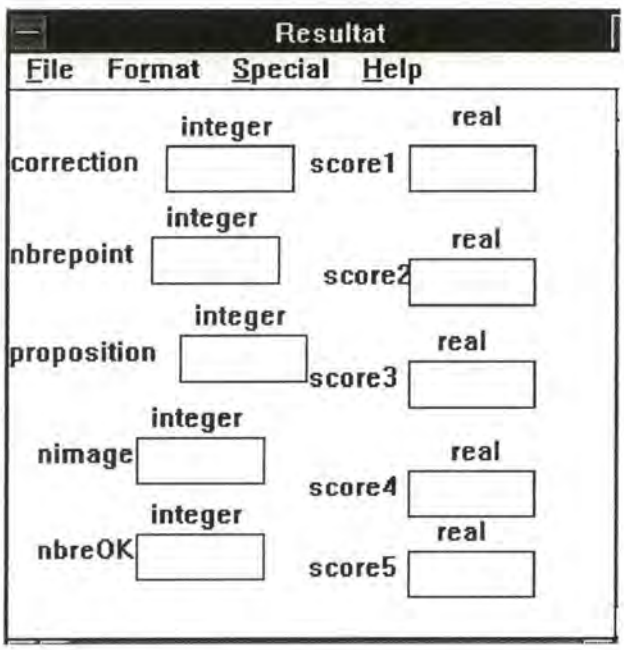


Figure 5.9 Présentation de la fenêtre résultat de l'application *Nouv1*

Cette application est l'application minimale pour faire fonctionner un système neuronal. Il est possible d'ajouter des graphiques et des informations sur l'état du réseau, mais ça n'apporte plus beaucoup aux tests.

5.6. Présentation des résultats

Nom1 (N1)	Etape	Type	'set'	run (nbre)	loops (nbre)	param RN	temps (min.)
	1	Apprentis.					
	2	test					

(a)

(b)

N1	Etape	classe II	classe III	classe IV	classe V	total
	1					
	2					

Figure 5.10 Tableaux de présentation des résultats

La figure 5.10 montre les deux tableaux *a* et *b* qui seront utilisés pour présenter les résultats. Pour chaque session, les deux tableaux seront montrés.
Le tableau *a* décrit la session: son nom, le numéro de l'étape, le type de l'étape, le 'set' mis en entrée du réseau (liste en annexe), le 'run' correspondant à la taille du fichier, le 'loop' correspondant au nombre de présentation du fichier au réseau, les paramètres du système correspondant aux valeurs des 'run-time' et 'load-time' (liste à la section 5.5.2) et enfin le temps en minute pris par l'application.
Le tableau *b* présente les résultats de la session: avec les pourcentages d'apprentissage ou de reconnaissance selon le type d'étape pour chacune des quatre classes de cellules à reconnaître et le pourcentage total.

5.7 Description d'une architecture pour le système expert

La figure 5.11 montre une architecture probable d'un système expert de reconnaissance automatique de cellules. Cette architecture pourrait être généralisée à la reconnaissance de n'importe quel objet représenté par une image digitalisée.

Selon la description du cours de méthodes de développement de logiciels de E. Dubois, l'architecture logique d'un logiciel comprend cinq niveaux. Le plus haut niveau, ou niveau 5, rassemble les fonctionnalités qui coordonnent des fonctionnalités de plus bas niveau. Le niveau 1, le plus bas niveau, rassemble les fonctions élémentaires du logiciel. Une fonctionnalité d'un niveau supérieur peut toujours utiliser les services d'une fonctionnalité d'un niveau inférieur. Un module est une unité logique qui rassemble des fonctionnalités.

Nous présentons les niveaux 4 et 5, parce qu'ils suffisent à donner une bonne image de l'architecture de l'application et parce que ce sont ces parties de l'application qui nous ont le plus intéressés durant ce travail.

Au niveau 5, on trouve un module *Coordinateur* pour la gestion des images de cellules et de la présentation ainsi que la coordination entre les modules.

Au niveau 4, le module *Préparation* (qui correspond aux phases I et II) comprendrait des fonctionnalités d'acquisition et de digitalisation des images et de prétraitement/reformage. Ce module devrait permettre le transfert du système Sun à un PC. De plus dans ce module *Préparation* beaucoup de choses restent à mettre au point: la prise en compte de plusieurs cellules sur une image, la gestion de la taille de l'image et le 'nettoyage'.
Pour ce qui est du module *Image* (phase III), toutes ses fonctionnalités sont décrites dans la section 5.3 . Cependant, afin de construire un fichier d'apprentissage adéquat, une fonction permettant de choisir et sélectionner des images de la base de données serait à ajouter.
Le module *HNC* pourrait soit intégrer un logiciel existant (par exemple ExploreNet 3000 release 2.12), soit utiliser des bibliothèques de réseaux de neurones proposant le paradigme 'back-propagation'.

Le niveau 3, gestion de l'interface homme-machine (IHM), n'est pas développé dans notre architecture et se confond dans le programme *celltot*. Cependant, l'IHM est une partie importante pour la réalisation de ce système car de sa qualité dépendra l'acceptation ou non de l'utilisateur.

Le niveau 2, gestion de la base de données (BD), n'est pas non plus développé. La BD devrait permettre de conserver des images en différents formats mais aussi d'enregistrer les paramètres du réseau ainsi que les poids d'un réseau de neurones structuré (dont les poids des connexions après l'apprentissage).

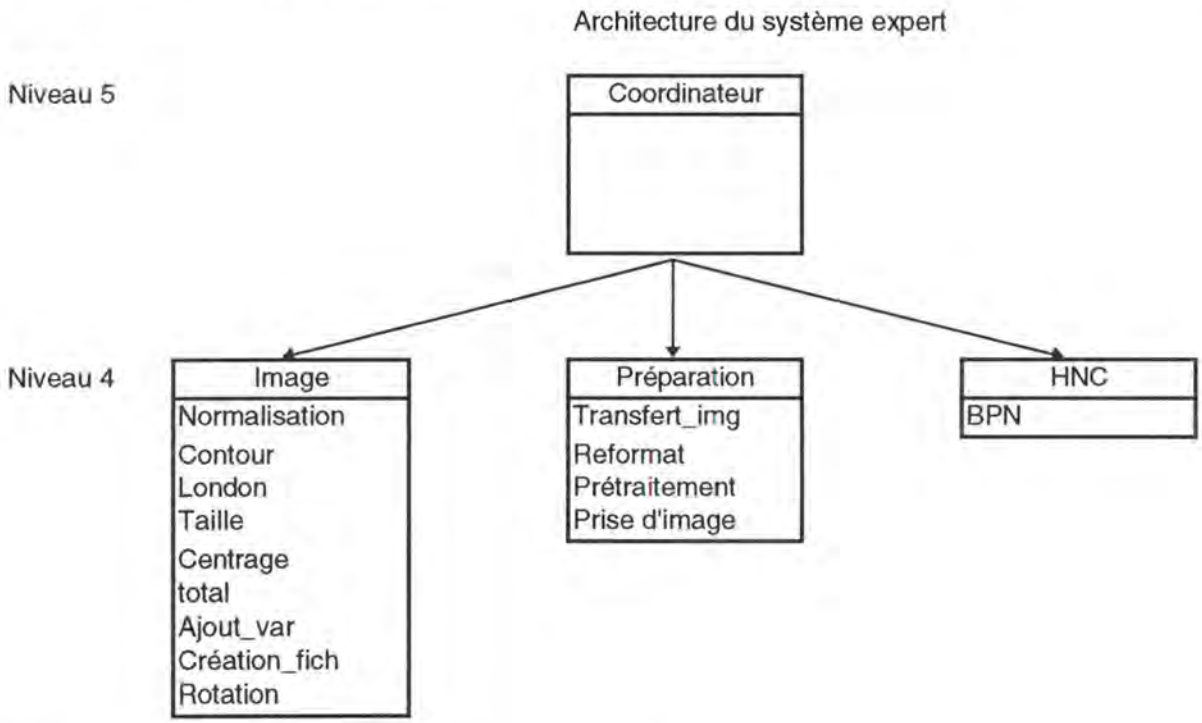


Figure 5.11 Architecture du système expert

6. Résultats

6.1 Introduction

Dans ce chapitre, nous poserons un ensemble de questions auxquelles les tests réalisés vont permettre de répondre le plus objectivement possible, avec comme objectif prioritaire: la minimisation du 'bruit' dans notre système de reconnaissance de cellules (voir paragraphe 2.2.5). Les conditions d'acquisition d'images (Phase I), de culture de cellules (densité, milieu, colorants,...) et de 'nettoyage' d'images (Phase II), d'une importance capitale pour le bon fonctionnement du système de reconnaissance, sont une source de 'bruits'. L'élimination de ceux-ci est un problème relativement technique qui doit pouvoir être résolu. Par contre, les questions porteront sur le prétraitement dépendant de la méthode de reconnaissance (Phase III) et l'utilisation des réseaux de neurones (Phase IV). En effet, les phases III et IV sont essentielles quant à l'efficacité potentielle du système.

Ces questions sont réparties en trois catégories:

Quel type de réseau faut-il utiliser (Phase IV) ?

Ce type de tests permettra de déterminer les paramètres du réseau qui optimisent l'apprentissage. Les tests préliminaires ont montré que le paradigme 'back-propagation' convenait à la reconnaissance de cellules. Selon ces tests, le réseau 'back-propagation' à une seule couche cachée semble suffisamment puissant pour reconnaître les cellules. Un réseau à plusieurs couches cachées a également fait l'objet de notre étude, sans pour autant, améliorer les résultats.

Les autres types de réseaux de neurones tel que celui de Hopfield, ou le réseau à 'counterpropagation' n'ont pas été testés, étant donné qu'ils ne répondaient pas ou peu au but recherché. Cependant dans des études ultérieures, l'utilisation d'un réseau à auto-organisation (apprentissage non supervisé) pourrait apporter des renseignements complémentaires sur la classification des cellules. Définissant lui-même ses classes, le réseau pourrait délimiter des sous-classes supplémentaires et représenter de manière plus adéquate la population des cellules.

Dans un premier temps, nous testerons l'influence de la structure du réseau (taille de la couche cachée, connexions I/O) sur l'apprentissage et dans un second temps, nous évaluerons les paramètres d'apprentissage tels que les constantes d'apprentissage, le 'batching' et le 'smoothing'. Enfin, les tests sur la dynamique d'apprentissage en fonction du nombre de 'loops' permettront de déterminer les conditions de l'apprentissage minimum.

Quelles cellules doivent faire partie du 'set' d'apprentissage (phase III) ?

Cette question est déterminante. En effet, s'agit-il de réaliser un 'set' d'apprentissage avec un nombre élevé de cellules, afin de représenter une grande diversité de cas (particuliers et généraux) ou bien, au contraire, le restreindre à un faible nombre de cellules représentatives (les 'prototypes') ? Ne serait-il pas intéressant que l'expert construise un modèle de la cellule typique, un sorte de portrait robot ?

En ce qui concerne l'invariance de rotation, nous avons choisi de présenter la cellule dans huit orientations différentes (cf. section 5.4.2, description de la fonctionnalité 'rotation'). Une question surgit: « Est-il indispensable de présenter pour chaque cellule, les huit rotations de celle-ci ? ».

Enfin, l'ordre du 'set' d'entraînement influence t'il l'apprentissage ?

Quel prétraitement appliquer aux cellules (phase III)?

Dans la section 5.4, nous proposons une série de fonctionnalités dont l'enchaînement présenté sur la figure 5.3 sera respecté. Pour rappel, nous obtenons, après la phase III, deux types de fichiers d'entraînements ou de tests: un fichier avec des « images 9' » (issues de la fonctionnalité 'London') et un fichier avec des « images 9 » (issues de la fonctionnalité 'total'). L'« image 9' » contient des informations sur le nombre de changements de pixels des lignes et des colonnes de l'image. L'« image 9 » a, en plus des informations sur le nombre de changements, des informations sur le nombre de pixels en noir de chaque ligne et de chaque colonne de l'image. La qualité de l'apprentissage, réalisé avec ces deux types d'images, fera l'objet d'une comparaison. Dorénavant, nous aurons soin d'identifier par une particule *lon* et par une particule *tot*, les fichiers contenant respectivement des images de type 9' et des images de type 9.

Cet enchaînement de fonctionnalités représenté par la figure 5.3 est la résultante d'un nombre important de tests préliminaires, dont l'aboutissement a permis de peaufiner ce prétraitement.

Nous proposons d'en décrire un bref historique:

1. Le test du réseau sans prétraitement

Les images en format 100 x 100 pixels (« image 1 ») sont testées avec le 'BPN'. Malheureusement, aucun apprentissage n'est réalisé en raison d'un problème technique: le logiciel utilisé ne supporte pas une telle quantité d'informations.

2. Le choix du format d'image

Les images d'origine (512 x 512 pixels) sont transformées en format 100 x 100 pixels et 40 x 40 pixels. Ce second format ne nous satisfait pas vu la perte importante d'informations.

3. La solution 'London'

Nous obtenons des résultats encourageants avec ce type d'images (« image 4 ») mais nous rencontrons, néanmoins, beaucoup d'erreurs dans la classification des cellules. Le système expert reconnaît moins de la moitié des cellules qui lui sont présentées lors des tests. Le fait que les lois d'invariances n'ont pas été apprises par le réseau semble être une des raisons de ce faible nombre de classifications correctes.

4. La solution 'centrage'

Les résultats obtenus sont positifs, le pattern est identique pour une cellule donnée quelle que soit sa situation sur l'image. Par exemple, grâce à la fonctionnalité de 'centrage', une cellule qui se trouve en haut de l'image possède le même vecteur de mesure que cette même cellule, se trouvant en bas de l'image. Il n'y a plus d'apprentissage de la transformation de translation à effectuer par le réseau.

5. L'apprentissage de la 'rotation'

En présentant au réseau des cellules dans des orientations différentes, on lui apprend la loi d'invariance de la rotation. Cet apprentissage a permis une amélioration de la reconnaissance des cellules. Les sessions, qui évaluerons cette amélioration, seront présentées dans le paragraphe 6.6.2.1 .

6. Un supplément d'informations concernant la surface de la cellule

L'ajout d'une donnée analytique, telle que la surface de la cellule, aux 200 mesures de la fonction 'London' n'améliore pas le résultat. Le mélange de caractéristiques et de mesures (section 2.1) ne semble pas convenir à l'approche réseau de neurones, d'autant plus que il n'y a qu'un seul neurone portant l'information analytique sur les 201 neurones de la couche d'entrée du réseau.

6.2 Descriptions des cellules et des paramètres choisis pour les tests

6.2.1 Les cellules

En annexe A1 à A14 se trouve le catalogue des cellules de classe II, III, IV et V. Il comprend 77 images (une image correspond à une cellule) en format 512 x 512 pixels.
Les trois premières lettres du nom de l'image représentent la qualité et la destination de la cellule. Ainsi, on distingue 4 pools d'images:

- NTR: cellules utilisées pour l'entraînement,
- NTE: cellules utilisées pour le test,
- NTESP: cellules utilisées pour le test mais dont la classe est ambiguë,
- NTS: cellules plus difficiles à classer.

Le premier chiffre représente la classe de la cellule et le second indique le numéro de série de la cellule.

6.2.2 Les 'sets' de cellules (apprentissages et tests)

La liste des cellules sélectionnées pour chacun des 'sets' d'apprentissage et de tests se trouve en annexe A15 à A17. La nomenclature des cellules de ces listes est semblable à celle présentée ci-dessus. Cependant, un troisième chiffre (de 0 à 7), représentant l'orientation de la cellule, est ajouté. Ce chiffre représente le coéfficient par lequel il faut multiplier 45° pour obtenir la rotation.
Par exemple, l'image NTR210 est une cellule utilisée pour l'entraînement, de classe II, de série 1 et n'ayant pas subi de rotation. L'image NTR211 est la même cellule ayant subi une rotation de + 45°.

Les 'sets', définis ci-dessous, sont identifiés par un nom et seront accompagnés d'une particule *lon* ou *tot* selon qu'il s'agit de fichiers rassemblant des images de type 9' ou 9.

'Bas1' (A15): Il s'agit du 'set' d'apprentissage de référence (128 images) qui sera comparé à d'autres 'sets'. Il est composé des 4 premières cellules (série 1 à 4) de chacune des 4 classes du pool NTR (A 1 à 4). A chaque cellule de ce 'set' correspond huit images (les 8 rotations).
L'ordre des cellules de ce 'set' sera déterminé d'abord par le numéro de la série, puis celui de la rotation et enfin celui de la classe. Cet ordre sera noté: « série ⇒ rotation ⇒ classe ».

'Bas0' (A15): Ce 'set' est construit comme 'Bas1' mais il ne comprend pas les cellules ayant subi une rotation. Ce 'set' comprend 16 images.

'Bas2' (A15): Ce 'set' est construit comme 'Bas1' mais l'ordre des cellules est différent: classe ⇒ série ⇒ rotation (128 images).

'Bas3' (A15): Ce 'set' est construit comme 'Bas1' mais l'ordre des cellules est différent: rotation ⇒ série ⇒ classe (128 images)

'Long1' (A16): Ce 'set' est composé des 9 cellules (série 1 à 9) de chacune des 4 classes du pool NTR (A 1 à 4).
L'ordre des cellules de ce 'set' est le suivant: série ⇒ rotation ⇒ classe (288 images).

'All1' (A16): Ce 'set' est construit comme 'Long1' mais il ne comprend pas les cellules ayant subi une rotation (36 images).

'Ts1R' (A17): Cellules de la série NTS de classe II avec rotations (16 images)
'Ts2R'(A17): Cellules de la série NTS de classe III avec rotations (8 images)
'Ts3R' (A17): Cellules de la série NTS de classe III avec rotations (8 images)
'Ts4R' (A17): Cellules de la série NTS de classe II avec rotations (32 images)

'F1' (A17): Fichier contenant la première cellule de chaque classe de la série NTR (32 images). L'ordre est le suivant: série ⇒ rotation ⇒ classe.

'F2': Fichier composé du fichier 'Long1' suivi des fichiers 'Ts1R', 'Ts2R', 'Ts3R' et 'Ts4R'. (352 images)

Test2 (A16): Cellules de la série NTE de classe II (10 images)
Test3 (A16): Cellules de la série NTE de classe III (4 images)
Test4 (A16): Cellules de la série NTE de classe IV (6 images)
Test5 (A17): Cellules de la série NTE de classe V (8 images)
Testsp (A17): Cellules de la série NTESP de classe ambiguë (5 images)
ntesp 20: classe IV ou V
ntesp 21: classe III ou IV
ntesp 22: classe IV ou V
ntesp 23: classe III ou IV
ntesp 24: classe IV ou V

6.2.3 Les paramètres du réseau

Les paramètres de base utilisés, décrits dans le paragraphe 5.5.2, pour les sessions, sont indiqués ci-dessous. Ces paramètres de référence sont issus du savoir-faire des utilisateurs de réseaux de neurones, de la littérature [Blum, 92] et de notre propre expérience.

'Load-time'

1. La taille du réseau:

Couche d'entrée (I): 200 neurones (prétraitement 'London': fichier d'images 9')
400 neurones (prétraitement 'total': fichier d'images 9)

Couche cachée (H): 50 neurones (prétraitement 'London': fichier d'images 9')
30 neurones (prétraitement 'London + taille': fichier d'images 9)

Couche de sortie (O): 5 neurones

2. Méthode d'apprentissage (M): 'batching'

3. Connexion (C): oui

'Run-time'

Pour ce qui est des paramètres d'apprentissage, la possibilité d'apprentissage est à 'ON' si c'est une étape d'apprentissage et à 'OFF' si c'est une étape de test.

1. Taille du 'batching'(T): 4 cycles

2. Taux d'apprentissage:
pour les neurones de la couche cachée (AH): 0,2
pour les neurones de la couche de sortie (AO): 0,2

3. Taux de 'smoothing'
pour les neurones de la couche cachée (BH): 0
pour les neurones de la couche de sortie (BO): 0

Si une modification d'un de ces paramètres a lieu, elle sera mentionnée dans le tableau à la rubrique 'param RN'. Par exemple, si la couche cachée comporte 80 neurones, on indiquera H=80.

6.3 Description de la session de référence

Court1Lon (C1L)	Etape	Type	'set'	run (nbre)	loops (nbre)	param RN	temps (min.)
	1	Apprentis.	Bas1lon	128	150	/	50
	2	test	testlon 2/3/4/5	28	1	/	1
	3	test	testsplon	5	1	/	1

C1L	Etape	classe II	classe III	classe IV	classe V	total
	1					128/128
	2	9/10	4/4	5/6	7/8	25/28

L'apprentissage (étape 1) est effectué à 100 % et le test (étape 2) montre que 89 % des cellules testées sont classifiées correctement par le système. Les cellules incorrectement classifiées sont les cellules NTE 29, NTE 44 et NTE 57. On peut comprendre ces erreurs de reconnaissance sans toutefois les accepter. En effet, la cellule NTE 29 est 'nettoyée' imparfaitement, la cellule NTE 44 est une cellule de classe IV de longueur relativement faible et la cellule NTE 57 est relativement petite par rapport à son appartenance à la classe V.
Le tableau 6.1 exprime les scores obtenus pour chacune des cellules du test. Les scores de faibles valeurs (inférieure à 0,05) ne sont pas indiqués. Pour rappel, la fonction d'activation des neurones de la couche de sortie n'est pas linéaire mais sigmoïde. Les scores ont des valeurs comprises entre 0 et 1. En conséquence, un score de 0,99 représente un score très élevé. Il ne s'agit donc pas d'un pourcentage de la probabilité d'appartenance à une classe ou à une autre.

Ce tableau nous inspire deux remarques:

- Les valeurs des scores sont dans 21 cas sur 28, correctes et supérieures à 0,75. Dans pareil cas, la réponse du réseau est sans appel. Par contre, on trouve 5 cas où le réseau répond correctement mais avec un score inférieur à 0,75.

- Lorsque le réseau commet une erreur, elle est dans 2 cas sur 3, très nette.

Nom de la cellule	Classe	Score classe II	Score classe III	Score classe IV	Score classe V	Erreur
NTE 20	II	0,99				
NTE 21	II	0,92				
NTE 22	II	0,76				
NTE 23	II	0,99				
NTE 24	II	0,31	0,16			
NTE 25	II	0,11				
NTE 26	II	0,67				
NTE 27	II	0,89				
NTE 28	II	0,95	0,12			
NTE 29	II	0,02	0,19		0,65	oui
NTE 30	III		0,54		0,12	
NTE 31	III		0,85			
NTE 32	III		0,99			
NTE 33	III		0,8			
NTE 40	IV			0,99		
NTE 41	IV			0,99		
NTE 42	IV			0,99		
NTE 43	IV			0,98		
NTE 44	IV			0,0207	0,91	oui
NTE 45	IV			0,99	0,31	
NTE 50	V				0,99	
NTE 51	V				0,99	
NTE 55	V				0,82	
NTE 53	V				0,99	
NTE 54	V		0,73		0,93	
NTE 55	V		0,23		0,8	
NTE 56	V				0,92	
NTE 57	V		0,92		0,0293	oui
NTESP 20	IV / V				0,99	
NTESP 21	III / IV			0,99	0,85	
NTESP 22	IV / V			0,5		
NTESP 23	III / IV				0,99	oui
NTESP 24	IV / V			0,99		

Tableau 6.1 Détails des résultats obtenus par la session C1L

Le test de l'étape 3, dont les résultats sont présentés sur le tableau 6.1, montre que le réseau reconnaît 80% des cellules de type 'NTESP', dont la classification est ambiguë. Cependant sa réponse consiste souvent en un choix net d'une classe; contrairement à l'expert, le réseau, avec cet apprentissage, n'a pas assimilé cette notion d'ambiguïté.

Globalement cette session montre le bon fonctionnement de la méthode. Cependant la faible taille de l'échantillon testé justifiera notre réserve dans les conclusions. De plus, les cellules utilisées pour l 'étape de test sont relativement typiques.

6.4 L'architecture du réseau

Avant d'examiner les sessions sur le prétraitement des données, il convient de déterminer les paramètres du réseau qui optimisent l'apprentissage.

6.4.1 Structure du réseau

6.4.1.1 Nombre de neurones de la couche cachée (H)

C1LS1	Etape	Type	'set'	run (nbre)	loops (nbre)	param RN	temps (min.)
	1	Apprentis.	Bas1lon	128	150	H = 1	
	2	test	testlon 2/3/4/5	28	1	H = 1	1

C1LS1	Etape	classe II	classe III	classe IV	classe V	total
	1					123/128
	2	7/10	3/4	5/6	5/8	20/28

C1LS5	Etape	Type	'set'	run (nbre)	loops (nbre)	param RN	temps (min.)
	1	Apprentis.	Bas1lon	128	150	H = 5	
	2	test	testlon 2/3/4/5	28	1	H = 5	1

C1LS5	Etape	classe II	classe III	classe IV	classe V	total
	1					124/128
	2	8/10	3/4	5/6	7/8	23/28

C1LS10	Etape	Type	'set'	run (nbre)	loops (nbre)	param RN	temps (min.)
	1	Apprentis.	Bas1lon	128	150	H = 10	
	2	test	testlon 2/3/4/5	28	1	H = 10	1

C1LS10	Etape	classe II	classe III	classe IV	classe V	total
	1					127/128
	2	9/10	4/4	5/6	7/8	25/28

C1LS30	Etape	Type	'set'	run (nbre)	loops (nbre)	param RN	temps (min.)
	1	Apprentis.	Bas1lon	128	150	H = 30	
	2	test	testlon 2/3/4/5	28	1	H = 30	1

C1LS30	Etape	classe II	classe III	classe IV	classe V	total
	1					128/128
	2	9/10	4/4	5/6	7/8	25/28

C1LS80	Etape	Type	'set'	run (nbre)	loops (nbre)	param RN	temps (min.)
	1	Apprentis.	Bas1lon	128	150	H = 80	
	2	test	testlon 2/3/4/5	28	1	H = 80	1

C1LS80	Etape	classe II	classe III	classe IV	classe V	total
	1					128/128
	2	9/10	4/4	5/6	7/8	25/28

Nom de la session	C1LS1	C1LS5	C1LS10	C1LS30	C1L	C1LS80
Param RN	H = 1	H = 5	H = 10	H = 30	H = 50	H = 80
Total	20/28	23/28	25/28	25/28	25/28	25/28

Tableau 6.2 Synthèse des sessions dont l'étude porte sur le nombre de neurones de la couche cachée (H)

Interprétation

Ces sessions montrent une influence du nombre de neurones de la couche cachée sur la qualité de l'apprentissage du réseau (tableau 6.2). Pour cette application, un minimum de 30 neurones doit être atteint pour assurer un apprentissage optimal. En effet, en deçà de cette valeur, l'apprentissage est imparfait et la reconnaissance correcte des cellules tests chute, jusqu'à 25 %, pour un seul neurone dans la couche cachée. Par contre, les résultats ne s'améliorent plus au delà de 30 neurones. Il n'y a donc pas d'intérêt à utiliser un réseau avec une couche cachée de 80 neurones , étant donné l'augmentation du temps de calcul. Nous utiliserons, dorénavant, un réseau avec une couche cachée de 40 neurones, ce qui était l'option de départ.

6.4.1.2 Connexions directes des entrées aux sorties (C)

C1LW	Etape	Type	'set'	run (nbre)	loops (nbre)	param RN	temps (min.)
	1	Apprentis.	Bas1lon	128	150	C = non	
	2	test	testlon 2/3/4/5	28	1	C = non	1

C1LW	Etape	classe II	classe III	classe IV	classe V	total
	1					127/128
	2	6/10	4/4	5/6	7/8	22/28

Interprétation

Bien que l'apprentissage (étape 1) semble presque complet, l'étape de reconnaissance (22 cellules reconnues) se déroule nettement moins bien que dans la session C1L de base. Il est donc profitable de laisser ces connexions directes entre les unités des couches d'entrée et de sortie du réseau.

6.4.2 Les paramètres d'apprentissage

6.4.2.1 La dynamique de l'apprentissage

Nous allons essayer de déterminer le nombre de 'loops' nécessaire à un apprentissage complet. Autrement dit, combien de fois faut-il présenter le 'set' d'entraînement au réseau pour que le système ait appris les exemples proposés.

C1LL0	Etape	Type	'set'	run (nbre)	loops (nbre)	param RN	temps (min.)
	1	Apprentis.	Bas1lon	128	0	/	
	2	test	testlon 2/3/4/5	28	1	/	1

C1LL0	Etape	classe II	classe III	classe IV	classe V	total
	1					37/128
	2	1/10	4/4	0/6	1/8	6/28

C1LL1	Etape	Type	'set'	run (nbre)	loops (nbre)	param RN	temps (min.)
	1	Apprentis.	Bas1lon	128	1	/	
	2	test	testlon 2/3/4/5	28	1	/	1

C1LL1	Etape	classe II	classe III	classe IV	classe V	total
	1					82/128
	2	9/10	0/4	4/6	6/8	19/28

C1LL5	Etape	Type	'set'	run (nbre)	loops (nbre)	param RN	temps (min.)
	1	Apprentis.	Bas1lon	128	5	/	
	2	test	testlon 2/3/4/5	28	1	/	1

C1LL5	Etape	classe II	classe III	classe IV	classe V	total
	1					102/128
	2	7/10	1/4	6/6	7/8	21/28

C1LL10	Etape	Type	'set'	run (nbre)	loops (nbre)	param RN	temps (min.)
	1	Apprentis.	Bas1lon	128	10	/	
	2	test	testlon 2/3/4/5	28	1	/	1

C1LL10	Etape	classe II	classe III	classe IV	classe V	total
	1					116/128
	2	8/10	3/4	6/6	7/8	24/28

C1LL30	Etape	Type	'set'	run (nbre)	loops (nbre)	param RN	temps (min.)
	1	Apprentis.	Bas1lon	128	30	/	
	2	test	testlon 2/3/4/5	28	1	/	1

C1LL30	Etape	classe II	classe III	classe IV	classe V	total
	1					123/128
	2	8/10	4/4	5/6	7/8	24/28

C1LL50	Etape	Type	'set'	run (nbre)	loops (nbre)	param RN	temps (min.)
	1	Apprentis.	Bas1lon	128	50	/	
	2	test	testlon 2/3/4/5	28	1	/	1

C1LL50	Etape	classe II	classe III	classe IV	classe V	total
	1					125/128
	2	9/10	4/4	5/6	7/8	25/28

C1LL500	Etape	Type	'set'	run (nbre)	loops (nbre)	param RN	temps (min.)
	1	Apprentis.	Bas1lon	128	500	/	
	2	test	testlon 2/3/4/5	28	1	/	1

C1LL500	Etape	classe II	classe III	classe IV	classe IV	total
	1					128/128
	2	9/10	4/4	5/6	7/8	25/28

Nom de la session	C1LL0	C1LL1	C1LL5	C1LL10	C1LL30	C1LL50	C1L	C1LL500
Loops	0	1	5	10	30	50	150	500
Total (étape 1)	37/128	82/128	102/128	116/128	123/128	125/128	128/128	128/128
Total (étape 2)	6/28	19/28	21/28	24/28	24/28	25/28	25/28	25/28

Tableau 6.3 Synthèse des sessions étudiant la dynamique d'apprentissage

Interprétation

Au regard des résultats de la session C1LL0, nous constatons un taux de reconnaissance d'environ 20 %. Ceci n'a rien d'étonnant vu le nombre de sorties du réseau (5 neurones); celui-ci a une chance sur cinq de reconnaître par hasard la cellule. De plus, le réseau apprend très vite: apprentissage en un 'loop' (session C1LL1) de 82 cellules sur les 128 du 'set' d'entraînement. Il reconnaît 68 % des cellules du fichier de test. Ce résultat quelque peu surprenant est tempéré par le fait que chaque cellule est présentée 8 fois (dans des positions différentes) par 'loop'.
Bien que l'apprentissage soit rapide, il faut attendre 50 'loops' pour obtenir un taux de reconnaissance équivalent à la session de référence. Le réseau se stabilise donc relativement lentement.

Il faut aussi souligner que l'apprentissage n'est réalisé complètement (128/128) qu'à partir de 100 'loops'. L'intérêt de réaliser un apprentissage complet suscite quelques polémiques. En effet, celui-ci pourrait être perçu comme une tentative de forcer le réseau à apprendre des exemples s'éloignant trop du prototype. Cependant, cette hypothèse est infirmée à la fois par l'expérimentation et la théorie. En effet, le taux de reconnaissance dans une étape de test ne s'est jamais dégradé suite à un apprentissage complet du 'set' d'entraînement. De plus, il est fort probable que le réseau trouve rapidement un minimum local (après environ 10 loops) et il est, dés lors, très difficile de forcer le réseau à apprendre des exemples nouveaux.

6.4.2.2 Le taux d'apprentissage (AH et AO)

Pour rappel, le taux d'apprentissage ou α permet de modifier l'importance du changement des poids des connexions (5.5.2). Il s'agit de choisir un taux d'apprentissage suffisamment élevé pour permettre un apprentissage rapide et suffisamment bas pour éviter de tomber dans un minimum 'trop' local. Le nombre de 'loops' est de 30, ce qui est insuffisant pour un apprentissage complet mais il permet d'évaluer le taux d'apprentissage des neurones de la couche cachée (AH) et de la couche de sortie (OH). On peut noter que les taux AH et OH sont identiques.

C1LE0,1	Etape	Type	'set'	run (nbre)	loops (nbre)	param RN	temps (min.)
	1	Apprentis.	Bas1lon	128	30	AH = AO = 0,1	
	2	test	testlon 2/3/4/5	28	1	/	1

C1LE0,1	Etape	classe II	classe III	classe IV	classe V	total
	1					123/128
	2	7/10	4/4	5/6	7/8	23/28

C1LE1	Etape	Type	'set'	run (nbre)	loops (nbre)	param RN	temps (min.)
	1	Apprentis.	Bas1lon	128	30	AH = AO = 1	
	2	test	testlon 2/3/4/5	28	1	/	1

C1LE1	Etape	classe II	classe III	classe IV	classe V	total
	1					125/128
	2	9/10	4/4	5/6	7/8	25/28

C1LL30				123/128
8/10	4/4	5/6	7/8	24/28

Interprétation

En comparant, ces résultats avec ceux de la session C1LL30 (paragraphe 6.4.2.1), on remarque une croissance du taux de reconnaissance en fonction du taux d'apprentissage. Il est clair que la situation se stabilise après une centaine de 'loops' supplémentaires. Mais il est intéressant de voir qu'un taux d'apprentissage élevé ne conduit pas, dans notre étude, à une mauvaise reconnaissance des cellules (session C1LE1). Cependant, par mesure de prudence nous préférons conserver un taux d'apprentissage de 0,2, qui correspond mieux aux valeurs proposées dans la littérature.

6.4.2.3 Le 'smoothing', le 'batching' et le mode normal

La session de référence a une équation de propagation d'erreur de type 'batching' (T = 4), c'est-à-dire que les poids ne sont changés (par rétropropagation) que tous les 4 cycles (paragraphe 5.5.2). Cette amélioration du 'back-propagation' est-elle utile ? Le 'smoothing' a-t'il une influence sur la qualité de l'apprentissage ?
Le nombre de 'loops' est de 30, ce qui est insuffisant pour un apprentissage complet mais il permet d'évaluer ces différents changements. La session de référence est C1LL30 (paragraphe 6.4.2.1), avec un 'batching' de 4, dont nous apercevons ici les résultats:

Le mode 'batching' (T = 4)

C1LL30				123/128
8/10	4/4	5/6	7/8	24/28

Le mode 'smoothing'

SB1	Etape	Type	'set'	run (nbre)	loops (nbre)	param RN	temps (min.)
	1	Apprentis.	Bas1lon	128	30	BH = BO = 0,2	
	2	test	testlon 2/3/4/5	28	1	/	1

SB1	Etape	classe II	classe III	classe IV	classe V	total
	1					124/128
	2	9/10	4/4	5/6	7/8	25/28

La méthode de 'smoothing' permet un apprentissage plus rapide.

Le mode normal

NB1	Etape	Type	'set'	run (nbre)	loops (nbre)	param RN	temps (min.)
	1	Apprentis.	Bas1lon	128	30	T = 0	
	2	test	testlon 2/3/4/5	28	1	/	1

NB1	Etape	classe II	classe III	classe IV	classe V	total
	1					124/128
	2	9/10	4/4	5/6	6/8	24/28

L'absence d'amélioration ne détériore pas l'apprentissage.

Le mode 'batching' (T = 128)

B128	Etape	Type	'set'	run (nbre)	loops (nbre)	param RN	temps (min.)
	1	Apprentis.	Bas1lon	128	30	T = 128	
	2	test	testlon 2/3/4/5	28	1	/	1

B128	Etape	classe II	classe III	classe IV	classe V	total
	1					86/128
	2	6/10	0/4	6/6	7/8	19/28

Le fait de modifier les poids, tous les 128 cycles (= taille du 'set' d'apprentissage), ralentit considérablement l'apprentissage sans pour autant le mettre en péril. On peut comparer les résultats de la session 'B128' avec ceux de la session 'C1LL1' (paragraphe 6.4.2.1) où, après un seul 'loop', le réseau obtenait un apprentissage de 82/128 et reconnaissait 19 cellules sur les 28 présentées.

6.4.3 Conclusions

Ces améliorations ne changent pas beaucoup la qualité de l'apprentissage. Il convient de prendre garde aux extrêmes: un apprentissage trop court, une couche cachée avec peu de neurones et un taux d'apprentissage trop faible peuvent être source de 'bruit' pour le système.

Nous constatons, aussi, que les paramètres de la session de référence (C1L) sont adéquats pour notre tâche de reconnaissance.

6.5 Le choix du 'set' d'apprentissage

6.6.1 L'ordre du 'set' d'apprentissage

C2L	Etape	Type	'set'	run (nbre)	loops (nbre)	param RN	temps (min.)
	1	Apprentis.	**Bas2lon**	128	150	/	
	2	test	testlon 2/3/4/5	28	1	/	1

C2L	Etape	classe II	classe III	classe IV	classe V	total
	1					126/128
	2	7/10	4/4	5/6	7/8	23/28

C3L	Etape	Type	'set'	run (nbre)	loops (nbre)	param RN	temps (min.)
	1	Apprentis.	**Bas3lon**	128	150	/	
	2	test	testlon 2/3/4/5	28	1	/	1

C3L	Etape	classe II	classe III	classe IV	classe V	total
	1					125/128
	2	7/10	4/4	5/6	6/8	22/28

Interprétation

Par comparaison, avec la session 'C1L' (paragraphe 6.3), les résultats obtenus par les sessions 'C2L' et 'C3L' sont moins fructueux. L'ordre optimal des cellules consiste, donc, en la présentation au réseau de quatre exemplaires de chacune des 4 classes à la suite l'un de l'autre. On peut en déduire l'hypothèse suivante. Les lois que le réseau doit apprendre sont la différenciation des classes et l'invariance de la rotation; il faut, donc, donner au système des exemples, exprimant ces lois, l'un à la suite de l'autre. La moins bonne solution serait de présenter au réseau de neurones toutes les cellules d'une classe, à la suite l'une de l'autre, et puis toutes les cellules d'une autre classe. La loi ne vient pas d'un exemple mais de la comparaison d'exemples successifs.

Une autre session, réalisée à partir de cellules placées au hasard, dans le 'set' d'entraînement, obtient un taux de reconnaissance de 24 cellules sur 28, ce qui est à la fois mieux que les résultats obtenus avec les 'sets' 'Bas2Lon' et 'Bas3Lon' et moins bien que la session de référence. Ce qui confirme l'hypothèse présentée dans ce paragraphe.

On peut émettre une remarque intéressante: l'apprentissage s'effectue presque totalement, quel que soit l'ordre des cellules; seule l'étape de test permet de déceler des différences d'apprentissage.

6.6.2 La taille du 'set' d'apprentissage

6.6.2.1 L'invariance de la rotation

COL	Etape	Type	'set'	run (nbre)	loops (nbre)	param RN	temps (min.)
	1	Apprentis.	Bas0lon	16	150	/	
	2	test	testlon 2/3/4/5	28	1	/	1

COL	Etape	classe II	classe III	classe IV	classe V	total
	1					15/16
	2	6/10	2/4	5/6	6/8	18/28

Interprétation

Cet exemple montre l'importance de l'apprentissage de la rotation: il augmente les résultats d'au moins 25 %. Il importe de constater que cette amélioration concerne surtout les classes II et III.

6.6.3.2 Le nombre de cellules du 'set'

L'utilisation d'un nombre plus élevé de cellules

L1L	Etape	Type	'set'	run (nbre)	loops (nbre)	param RN	temps (min.)
	1	Apprentis.	Long1lon	288	150	/	100
	2	test	testlon 2/3/4/5	28	1	/	1

L1L	Etape	classe II	classe III	classe IV	classe V	total
	1					280/288
	2	7/10	3/4	5/6	7/8	22/28

VL1L	Etape	Type	'set'	run (nbre)	loops (nbre)	param RN	temps (min.)
	1	Apprentis.	F2lon	352	150	/	
	2	test	testlon 2/3/4/5	28	1	/	1

VL1L	Etape	classe II	classe III	classe III	classe V	% long1
	1	TSR1:16/16	TSR2:8/8	TSR3:8/8	TSR4:16/16	270/288
	2	6/10	3/4	5/6	7/8	21/28

Interprétation

Dans les sessions 'L1L' et 'VL1L', nous avons essayé de réaliser un apprentissage plus « fin », en ajoutant davantage d'exemples au 'set'. Nous remarquons une détérioration des résultats, d'autant plus élevée que l'on ajoute des cas particuliers.

L'utilisation d'un faible nombre de cellules

VC1L	Etape	Type	'set'	run (nbre)	loops (nbre)	param RN	temps (min.)
	1	Apprentis.	F1lon	32	150	/	
	2	test	testlon 2/3/4/5	28	1	/	1

VC1L	Etape	classe II	classe III	classe IV	classe V	total
	1					32/32
	2	7/10	4/4	5/6	7/8	23/28

Interprétation

Cet exemple est très intéressant dans le sens où l'utilisation d'une seule cellule par classe pour l'étape d'apprentissage permet d'obtenir une reconnaissance de 23 cellules sur les 28 du test. Nous avons choisi, pour cette session, l'approche « prototype », en sélectionnant des cellules qui nous semblaient bien adhérer à la définition des experts.

Bien que, le total de l'étape 2 soit inférieur d'environ 7% à celui obtenu par le 'set' de référence (C1L), l'approche orientée « prototype » semble mieux convenir que l'approche « exceptions »; le but étant d'apprendre, au réseau, des lois et des conseils plutôt que des exemples particuliers (section 3.7). Dés lors, le 'set' idéal serait composé d'un faible nombre de cellules représentatives de l'image mentale de l'expert.

6.6.3.3 La recherche d'un 'set' idéal

La recherche d'un 'set' idéal requiert des informations sur l'entraînement du réseau. Par exemple, dans l'étape 1 de la session 'L1L' (paragraphe 6.6.3.2), l'apprentissage de certaines configurations rotationnelles des cellules NTR32, NTR33, NTR34, NTR56 et NTR38 n'est pas réalisé.
Mais ce n'est qu'en utilisant un nombre plus important de cellules pour les tests et l'apprentissage que l'on pourra donner les caractéristiques du 'set' idéal.

6.6 Le prétraitement des images

Les sessions 'C1T', 'L1T', 'VL1T', 'VC1T' ont été réalisées avec des images ayant subi le prétraitement 'total'. Les images de ces 'sets' ont des informations sur la taille de la cellule, en plus de celles sur les changements ('London').

Court1Tot (C1T)	Etape	Type	'set'	run (nbre)	loops (nbre)	param RN	temps (min.)
	1	Apprentis.	Bas1tot	128	150	/	
	2	test	testtot 2/3/4/5	28	1	/	1

C1T	Etape	classe II	classe III	classe IV	classe V	total
	1					128/128
	2	8/10	3/4	5/6	3/8	23/28

Long1Tot (L1T)	Etape	Type	'set'	run (nbre)	loops (nbre)	param RN	temps (min.)
	1	Apprentis.	Long1tot	288	150	/	
	2	test	testtot 2/3/4/5	28	1	/	1

L1T	Etape	classe II	classe III	classe IV	classe V	total
	1					272/288
	2	5/10	4/4	6/6	7/8	22/28

VL1T	Etape	Type	'set'	run (nbre)	loops (nbre)	param RN	temps (min.)
	1	Apprentis.	F2tot	352	150	/	
	2	test	testtot 2/3/4/5	28	1	/	1

VL1T	Etape	classe II	classe III	classe IV	classe V	total
	1	TSR1:16/16	TSR2:8/8	TSR3:8/8	TSR4:24/32	288/288
	2	5/10	4/4	5/6	6/8	20/28

VC1T	Etape	Type	'set'	run (nbre)	loops (nbre)	param RN	temps (min.)
	1	Apprentis.	Bas1tot	32	150	/	
	2	test	testtot 2/3/4/5	28	1	/	1

VC1T	Etape	classe II	classe III	classe IV	classe V	total
	1					31/32
	2	8/10	4/4	6/6	4/8	22/28

Interprétation

D'une manière générale, le prétraitement, avec les informations sur la taille en plus de celles sur les changements diminue la capacité du réseau à reconnaître les cellules. Nous le constatons sur ce tableau comparatif:

Nom du 'set' d'apprentissage	bas1	long1	F2	F1	bas1*
London	25/28	22/28	21/28	23/28	100/160
Total	23/28	22/28	20/28	22/28	113/160

Les différences s'estompent avec les 'sets' 'long1', 'F2' et 'F1'. Ceci ne permet pas de décider définitivement de la meilleure méthode. Cependant, la méthode basée sur le prétraitement 'London' uniquement, pourrait être choisie car elle est plus simple à implémenter, et obtient des résultats égaux ou supérieurs à l'autre méthode. Ce résultat est étonnant car l'ajout d'informations semble perturber le réseau, mais n'est-ce pas là une étape vers un apprentissage plus « raffiné »?

Il serait, de toute façon, utile de réaliser des tests complémentaires. La session 'bas1*', dont l'étape d'apprentissage est identique à la session de référence 'C1L', est testée avec un 'set' de test constitué à partir du fichier 'long1' auquel on a retiré les images représentant les cellules du 'set' 'bas1'. Cette étape de test, dont le résultat est synthétisé sur le tableau ci-dessus, montre que la méthode de prétraitement 'total' classifie correctement 8 % des cellules en plus que la méthode de référence (prétraitement 'London'). Il est donc difficile de décider lequel de ces deux prétraitements est optimal.

6.7 Conclusions et recommandations

La structure et les paramètres du réseau doivent être choisis de manière adéquate. Dans le cas de notre système de reconnaissance de cellules, les paramètres utilisés pour notre session de référence sont à retenir. La variabilité des résultats est très faible, pour autant que l'on respecte certaines normes précisées dans la section 6.4.

Le meilleur résultat obtenu, à savoir 89 % de reconnaissance, est une indication du potentiel de la méthode. Les différents tests montrent que le système devrait reconnaître en moyenne plus de 70 % des cellules. Le choix des cellules utilisées pour l'apprentissage est d'une importance capitale. D'ailleurs, nous conseillons de rechercher un faible nombre de cellules (10 cellules par classe et par rotation est un maximum) plutôt qu'un nombre élevé de cas particuliers car nous voulons privilégier l'idée que l'expert se fait de la définition de la classe.
Pour ce qui est de l'ordre de présentation des cellules pour l'apprentissage, nous conseillons l'option choisie dans notre session de référence, en l'occurrence: « série $\Rightarrow$ rotation $\Rightarrow$ classe ».

Nous avons constaté dans nos tests préliminaires que l'apprentissage de l'invariance de la translation était complet. Par contre, nous n'avons pas pu montrer que la loi d'invariance de rotation était acquise parfaitement. Mais, les résultats ont, tout de même, montré une nette amélioration entre un apprentissage sans exemple de rotation et un apprentissage où chaque cellule est présentée dans huit orientations différentes.

Enfin, le prétraitement donnant des informations sur les changements ('London') semble adéquat. Cependant, nous ne pouvons pas éliminer la méthode 'total' sans obtenir des résultats complémentaires.

7. Comparaison avec une méthode statistique: l'analyse discriminante

7.1 Calcul d'une fonction

Le but de l'analyse discriminante est de trouver la fonction de n variables qui rendra le mieux compte de la séparation des observations en k groupes. Cette fonction peut être représentée par un hyperplan (si k=2) à n-1 dimensions, séparant linéairement les 2 groupes dont les observations sont disséminées dans un espace à n dimensions. Il est nécessaire de connaître, à l'avance, à quel groupe appartient chacune des observations! Par la suite, cette fonction peut être testée sur des observations non classées.
L'analyse discriminante peut, donc, constituer une alternative à la méthode neuronale, pour la phase de reconnaissance des cellules (phase IV de notre système). En effet, cette méthode statistique comprend, par analogie aux méthodes connexionistes, une étape d'apprentissage (calcul de la fonction discriminante) et une étape de test où l'instanciation de la fonction discriminante, par les valeurs des mesures de nouvelles cellules, permet d'évaluer la qualité de l'apprentissage.

7.2 Méthodes

Afin de pouvoir comparer les deux approches, nous avons conservé les trois premières phases du système de reconnaissance, seule la phase IV a été modifiée. Au cours de cette phase d'analyse discriminante, nous utiliserons des 'sets' d'images de type 'London', pour les étapes d'apprentissage et de test. La procédure 'DISCRIM' du 'package' de statistique 'SAS' a été utilisée sur le système VAX-VMS.

La présentation des résultats sera semblable à celle observée dans le chapitre 6 (résultats). Nous réaliserons deux sessions avec des 'sets' d'apprentissage différents afin de voir les éventuelles défaillances ou qualités de la méthode.

7.3 Analyse des résultats

AD1Lon (AD1L)	Etape	Type	'set'	run (nbre)	loops (nbre)	param RN	temps (min.)
Analyse discriminante	1	Apprentis.	**All1lon**	36	/	/	15
Analyse discriminante	2	test	testlon 2/3/4/5	28	/	/	1

AD1L	Etape	classe II	classe III	classe IV	classe V	total
Analyse discriminante	1					36/36
Analyse discriminante	2	4/10	1/4	5/6	6/8	16/28

AD2L	Etape	Type	'set'	run (nbre)	loops (nbre)	param RN	temps (min.)
Analyse discriminante	1	Apprentis.	**F1lon**	32	150	/	50
Analyse discriminante	2	test	testlon 2/3/4/5	28	1	/	1

AD2L	Etape	classe II	classe III	classe IV	classe V	total
Analyse discriminante	1					32/32
Analyse discriminante	2	3/10	4/4	4/6	7/8	18/28

Ces deux sessions montrent que l'apprentissage a été effectué à 100 %. Il existe, donc, une fonction linéaire qui permet de classer toutes ces cellules! Ce résultat étonnant peut être biaisé par le nombre relativement faible d'observations, par rapport aux 200 variables.

Pour la session AD1L, l'étape 2 montre que cette nouvelle méthode permet de reconnaître 16 cellules sur les 28 présentées. Pour ce qui est de la session AD2L, 18 cellules sur les 28 du test sont reconnues.

Cette approche donne des résultats satisfaisants mais est-elle aussi bonne que la méthode neuronale?

7.4 Comparaison avec l'approche réseau de neurones

RN1Lon (RN1L)	Etape	Type	'set'	run (nbre)	loops (nbre)	param RN	temps (min.)
	1	Apprentis.	**All1lon**	36	150	/	/
	2	test	testlon 2/3/4/5	28	1	/	/

RN1L	Etape	classe II	classe III	classe IV	classe V	total
	1					36/36
	2	4/10	1/4	5/6	8/8	18/28

RN2Lon (RN2L)	Etape	Type	'set'	run (nbre)	loops (nbre)	param RN	temps (min.)
	1	Apprentis.	**F1lon**	32	150	/	/
	2	test	testlon 2/3/4/5	28	1	/	/

RN2L	Etape	classe II	classe III	classe IV	classe V	total
	1					32/32
	2	7/10	4/4	5/6	7/8	23/28

En conclusion, l'utilisation de l'analyse discriminante, comme 4ème phase du système de reconnaissance de cellules, diminue d'environ 18 % la capacité du système à reconnaître les cellules, par rapport à l'approche réseau de neurones. De plus, le logiciel SAS ne peut calculer une fonction de discrimination si il y a plus de 100 images dans le 'set' d'entraînement ('overflow' des tableaux de données). Pour cette raison, il n'a pas été possible de tester l'analyse discriminante avec le 'set' 'bas1', qui a donné le meilleur résultat avec la méthode des réseaux de neurones.

Un autre problème mérite d'être souligné, il s'agit du temps calcul nécessaire pour les sessions. Il importe de prendre en ligne de compte deux paramètres:

- le temps d'apprentissage: plus long avec l'approche réseau de neurones (environ 50 minutes) qu'avec l'analyse discriminante (environ 10 minutes),
- la durée du test: plus longue avec l'analyse discriminante (environ 10 minutes) comparativement au réseau de neurones (environ 30 secondes).

La durée du test nous préoccupe davantage que le temps d'apprentissage. En effet, l'apprentissage n'est sensé être réalisé qu'une seule fois, par contre les tests et la production de résultats s'effectueront de nombreuses fois. Dés lors, nous accorderons notre préférence aux réseaux de neurones par rapport à la méthode d'analyse discriminante.

8. Conclusion

Arrivé au terme de ce mémoire, il y a lieu d'évaluer les critères de faisabilité du système de reconnaissance automatique de cellules. Si les résultats obtenus sont plus qu'encourageants, il ne faut toutefois pas oublier la possibilité de modifier certaines phases du système: en y ajoutant une analyse (calcul de caractéristiques tels que la surface, la longueur des cellules, ...), en utilisant des méthodes statistiques (analyse discriminante) pour la phase de reconnaissance.

Le critère d'évaluation est sans aucun doute la minimisation des « bruits »; un « bruit » étant une circonstance non idéale entravant la classification des objets. En effet, moins il y aura de « bruits », mieux nous pourrons reconnaître les cellules.
Le système de reconnaissance de cellules choisi génère certains « bruits », lors de certaines de ses phases.

La phase I, le codage, n'engendre aucune erreur de mesure sensible par le système.

Le prétraitement indépendant de la méthode de reconnaissance, phase II, dans le cas où il serait automatisé, pourrait être générateur de « bruits »: les problèmes de séparation de cellules collées, de 'nettoyage' de déchets cellulaires, ... Dans le cadre de notre travail, le 'nettoyage' manuel nous a déchargé de cette difficulté.

Quant au prétraitement dépendant de la méthode de reconnaissance, phase III, il nous satisfait. Cependant, il est difficile d'évaluer si il est une source importante de bruits car il existe d'autres techniques permettant la création de vecteur de mesures (pattern) en entrée du réseau de neurones, qui n'ont pas fait l'objet d'une évaluation. Bien que, nous ayons évalué deux types de prétraitement, 'London' et 'total', les résultats ne nous ont pas permis de prendre position pour l'un d'entre eux. Les informations sur les changements dans l'image ('London'), qui donnent une représentation de la forme, occasionnent, apparemment, peu de perte d'informations vitales pour la reconnaissance des cellules.

Ayant eu recours à une méthode connexioniste, la phase d'analyse n'a plus été nécessaire. Dès lors, le risque de perdre des informations importantes, durant cette phase, a été supprimé.

Le réseau de neurones et plus précisément l'algorithme de 'rétropropagation' est la méthode de reconnaissance utilisée pour la 4ème phase de notre système expert. Nous avons constaté qu'à partir du moment où les paramètres du réseau de neurones sont adéquats, le paradigme de 'rétropropagation' est un bon support pour l'apprentissage.
Mais, les erreurs dans le choix des patterns, peuvent être source de bruits. Il s'agit donc de choisir correctement les cellules qui vont servir d'exemples. Même si nous

avons réussi à faire apprendre des invariants géométriques tel que la rotation, des améliorations sont encore possibles par un meilleur choix des patterns d'entrée.
L'ordre de présentation des cellules pendant l'apprentissage a aussi son importance, et les résultats montrent qu'il convient de présenter les cellules de classes différentes, l'une à la suite de l'autre.

Quels sont les avantages de la méthode des réseaux de neurones ?

Le réseau utilise des données 'brutes', faciles à obtenir. En conséquence, la phase d'analyse peut être supprimée, ce qui réduit le processus de développement.

De plus, la relative indépendance des résultats par rapport aux paramètres du réseau constitue un argument en faveur de l'adaptabilité du réseau et de sa capacité à se structurer en fonction des données.

Malgré un apprentissage relativement lent (maximum 2 heures), cette méthode est rapide pour les étapes de test et de production.

Par rapport à l'analyse discriminante, le réseau de neurones offre un meilleur taux de classifications correctes et une plus grande rapidité pour l'étape de reconnaissance de cellules. Le fait que le réseau traite des problèmes non linéairement séparables, par opposition à l'analyse discriminante, semble être un avantage déterminant pour résoudre un problème aussi complexe que la classification de cellules.

Toutefois, malgré les résultats positifs obtenus, il faut souligner la complexité de la méthode et les difficultés à percevoir comment les connexions entre neurones se structurent et se modifient. N'oublions pas la principale limite du système: l'absence de réponse floue. En effet, la fonction de calcul des valeurs de sortie n'est pas linéaire, il est donc difficile d'estimer le degré de certitude de la réponse du réseau.

Ces quelques inconvénients sont largement dominés par les qualités du réseau de neurones: la souplesse et l'adaptabilité.

Enfin, quelles sont les perspectives à envisager pour un travail ultérieur ?

Il serait utile: - d'automatiser la phase I (codage) et la phase II ('nettoyage'),
- d'implémenter le système,
- d'affiner la méthode de reconnaissance par de multiples tests,
- de développer l'interface homme-machine afin de contribuer à une interaction entre l'utilisateur et le système,
- d'utiliser un module de base de données permettant la gestion d'un ensemble d'images de cellules typiques pour les tests.

Pourquoi ne pas approfondir, au niveau cellulaire, la reconnaissance de cellules collées, des « taches »? Pourquoi ne pas étendre l'apprentissage aux classes I, VI et VII ?

Pourquoi ne pas imaginer de mettre deux réseaux de neurones en série, le second reprenant, en entrée, la sortie du premier avec des renseignements tels que la surface, les niveaux de gris, ...?

Pourquoi ne pas avoir recours à d'autres méthodes pour faire 'apprendre' au réseau de neurones la loi d'invariance de la rotation (un réseau de neurones structuré différemment pouvant apprendre plus rapidement)?

Pourquoi ne pas tester la méthode sur d'autres types d'images: caractères, larves (biologie), autres cellules, ... Le système de reconnaissance automatique de cellules tel qu'il est présenté devrait être capable de traiter d'autres objets.

Telles sont l'ensemble des suggestions qu'il y aurait peut-être lieu de prendre en considération pour la réalisation d'un travail ultérieur dans ce domaine en pleine expansion qu'est la reconnaissance d'images.

BIBLIOGRAPHIE

[Abu-Mostafa, 95] : Y. Abu-Mostafa, « L'apprentissage en Informatique », Pour la Science, n° 212, pp 62-68, juin 1995.

[Aleksander, 89] : I. Aleksander, « Neural Computing Architectures », North Oxford Academy, 1989.

[Aleksander, 90] : I. Aleksander, H. Morton, « An introduction to Neural Computing », Chapman Edition, 1990.

[Anderson, 87] : J. A. Anderson, E. J. Wisniewski et S. R. Viscuso, « Software for Neural Networks ». Departement of Cognitive and Linguistic Sciences and Departement of Psychology, Brown University Providence, R1 02912.

[Belaïd, 92] : A. et Y. Belaïd, « Reconnaissance de formes. Méthodes et applications », Inter-édition, 1992.

[Blum, 92] : A. Blum, « Neural Networks in C++: an object oriented framwork for Building Connexionnist Systems », Wyley and Sons, 1992.

[DARPA, 88] : Defense Advanced Research Project Agency, « DARPA Neural Network Study (U.S.) », AFCEA International Press, 1988.

[Dayhoff, 90] : J. Dayhoff, « Neural Networks Architecture. An introduction. », 1990.

[Hartman, 90] : E. J. Hartman, J. D. Keeler et J. M. Kowalski, « Layered Neural Networks with Gaussian hidden units as universal approximations », Neural Computation, pp 210-215, (summer) 1990.

[Hebb, 49] : D. O. Hebb, « The Organization of behavior », J. Wiley and Sons, 1949.

[Hecht, 88] : R. Hecht-Nielsen, « Neurocomputing: Picking the human brain », IEEE Spectrum, March 1988, pp 36-41.

[Herault, 93] : J. Herault et C. Jutten, « La mémoire des réseaux neuromimétiques », La Recherche, Vol.25, n° 267, pp 824-830, juillet-août 1994.

[Hopfield, 82] : J. J. Hopfield, D. I. Feinstein et R. G. Palmer, «' Unlearning' has a stabilizing effect in collective memories », Nature n°304, pp 159-169, 1982.

[Kohonen, 84] : T. Kohonen, « Self-Organization and Associative Memory », Springer-Verlag, Berlin, Heidelberg, New-York, Tokyo, 1984.

[Kröse, 93] : B. J. A. Kröse et P. P. van der Smagt, « An introduction to Neural Networks », University of Amsterdam, Departement of Computer Systems, Fifth edition, 123p., january, 1993.

[Mc Culloch, 43] : W. S. Mc Culloch and W. Pitts, « A logical calculus of ideas immanent in nervous activity »,The Bulletin of Mathematical Biophysics, december 1943.

[Minsky, 69] : M. Minsky et S. Papert, « Perceptrons: An introduction to Computational Geometry », The Mit Press, 1969.

[Montaigne, 94] : F. Montaigne, « Application des techniques de convexité à la classification d'images de cellules », FUNDP Namur, Département des Sciences, mémoire de licence en Sciences Mathématiques, 1993-1994.

[Reid, 94] : M. B. Reid et L. Spirkovska, « Higher Order Neural Networks Applied to 2D and 3D Object Recognition », Machine Learning, 15(2), pp 169-199, 1994.

[Rimmer, 93] : S. Rimmer, « Les images Bit-Map », édition Dunod, 1993.

[Rosenblatt, 59] : F. Rosenblatt, « Principles of Neurodynamics », Spartan Books, New-York, 1959.

[Rumelhart, 86] : D. E. Rumelhart, G. E. Hinton et R. J. Williams, « Learning representations by Back-Propagating errors », Nature n°323, pp 533-536, 1986.

[Schalkoff, 89] : R. J. Schalkoff, « Pattern Recognition: Statistical, Structural and Neural Approaches », Wiley and Sons, pp2-29; pp 204-283, 1989.

[Toussaint, 92] : O. Toussaint, A. Houbion et J. Remacle, « Aging as a multi-step process characterized by a lowering of entropy production leading the cell to a sequence of defined stages.II.testing some predictions on aging human fibroblasts in culture », Mechanisms of Aging and Development, 65, pp 66-83, 1992.

[Widrow, 60] : B. Widrow et M. E. Hoff, « Adaptive Switching Circuits », in 1960 Ire Wescon Convention Record, New-York, pp 96-104, 1960.

Annexe

A01 à A14: Catalogue des cellules (format 512 * 512 pixels)
A15 à A17: Présentation des fichiers ou 'set' d'apprentissage et de test
A18 à A29: Programme celltot
A30 à A33: Programme cellrot

A1 Cellules de la série NTR (classe II)

ntr21

ntr22

ntr 23

ntr24

ntr25

ntr26

ntr27

ntr28

ntr29

A2 Cellules de la série NTR (classe III)

ntr31

ntr32

ntr 33

ntr34

ntr35

ntr36

ntr37

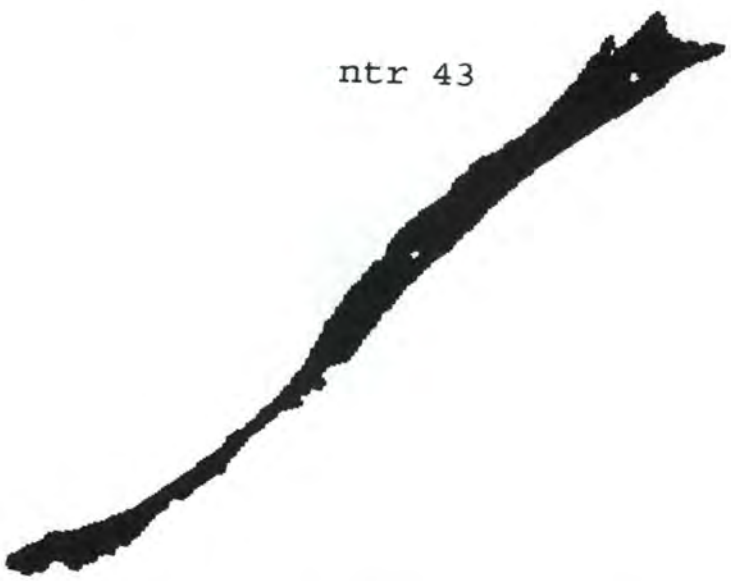
ntr38

ntr39

A3 Cellules de la série NTR (classe IV)

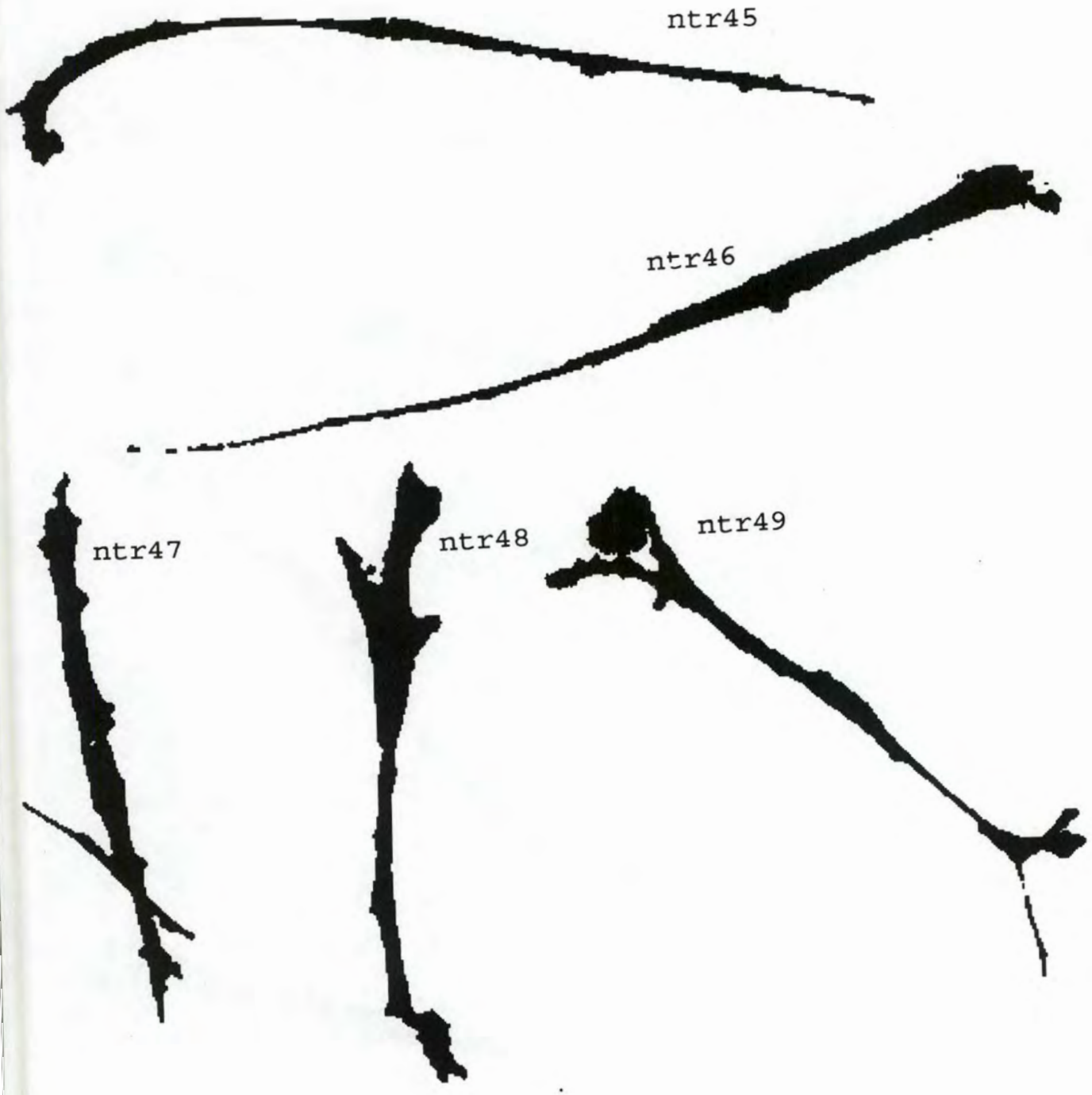
ntr41

ntr 43

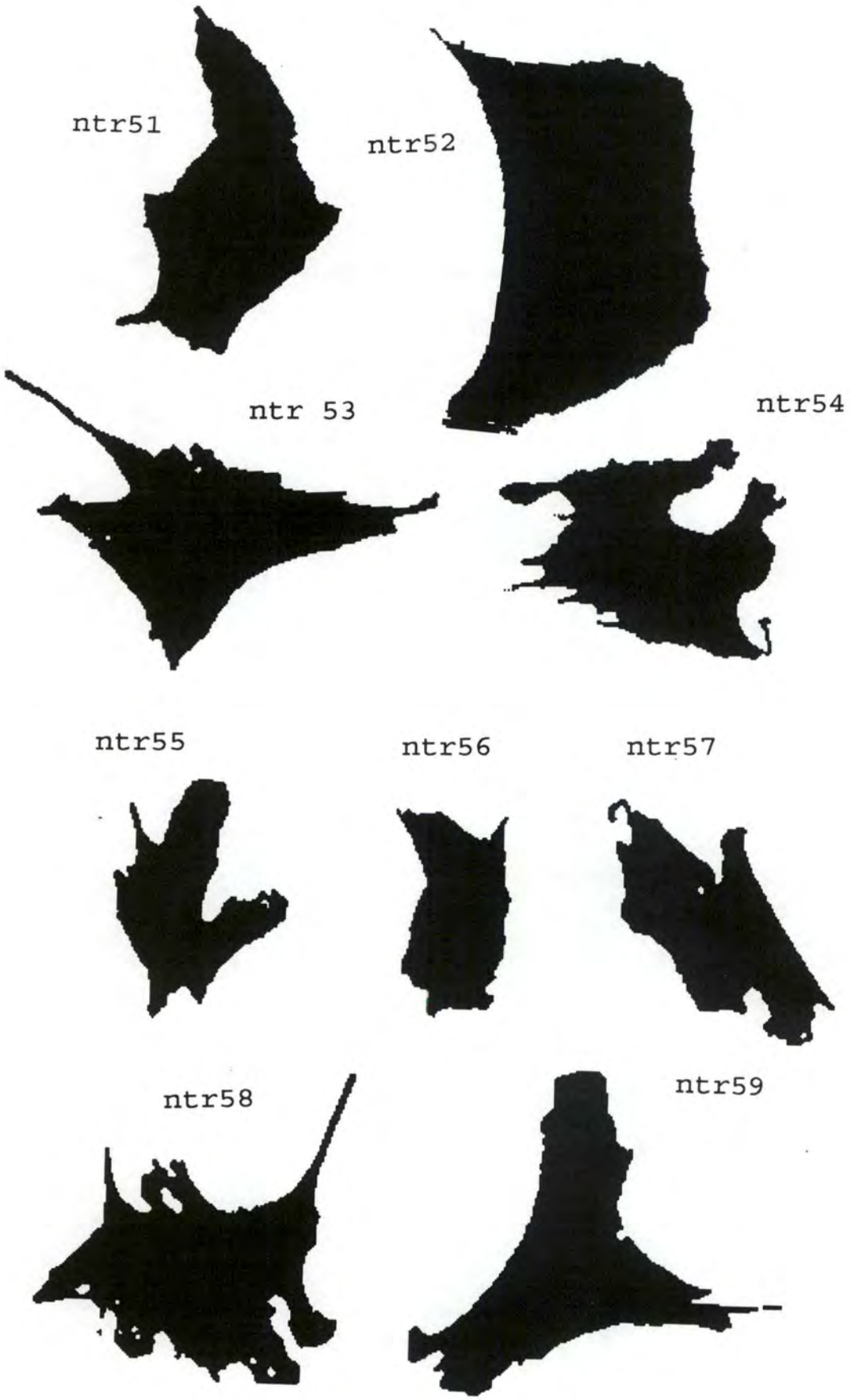


ntr44

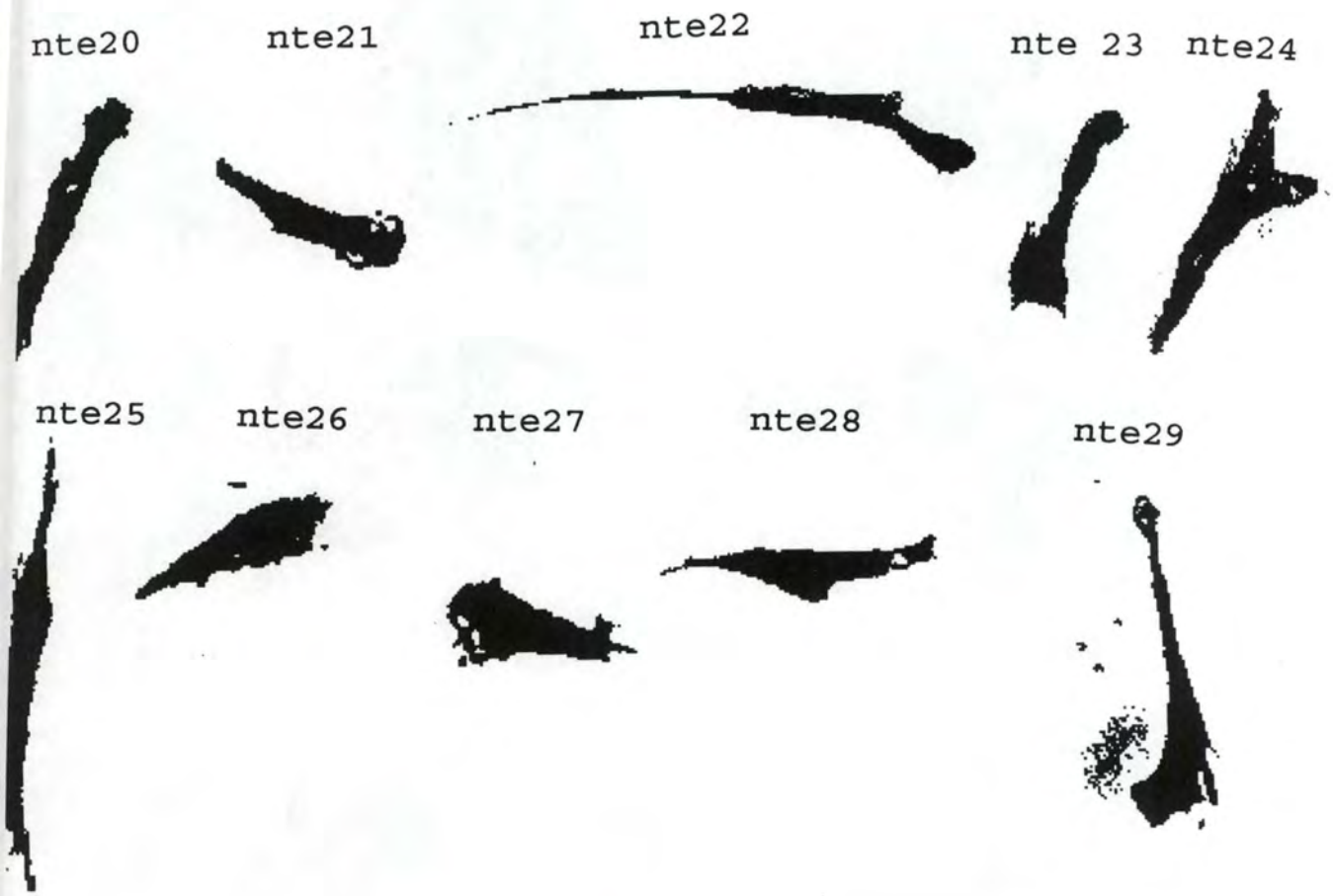
A4 Cellules de la série NTR (classe IV)



A5 Cellules de la série NTR (classe V)



A6 Cellules de la série NTE (classe II)



A7 Cellules de la série NTE (classe III)

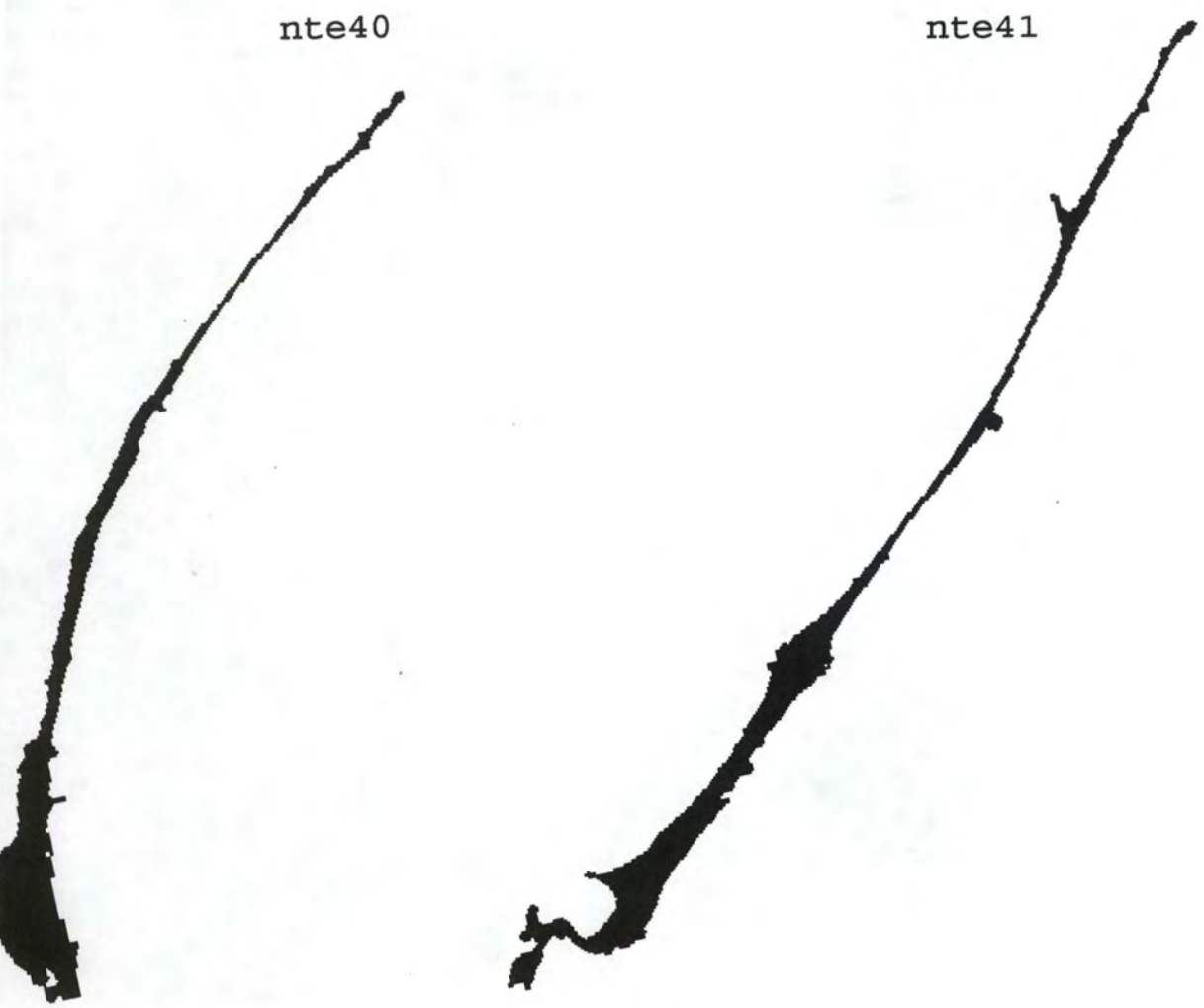
nte 30

nte31

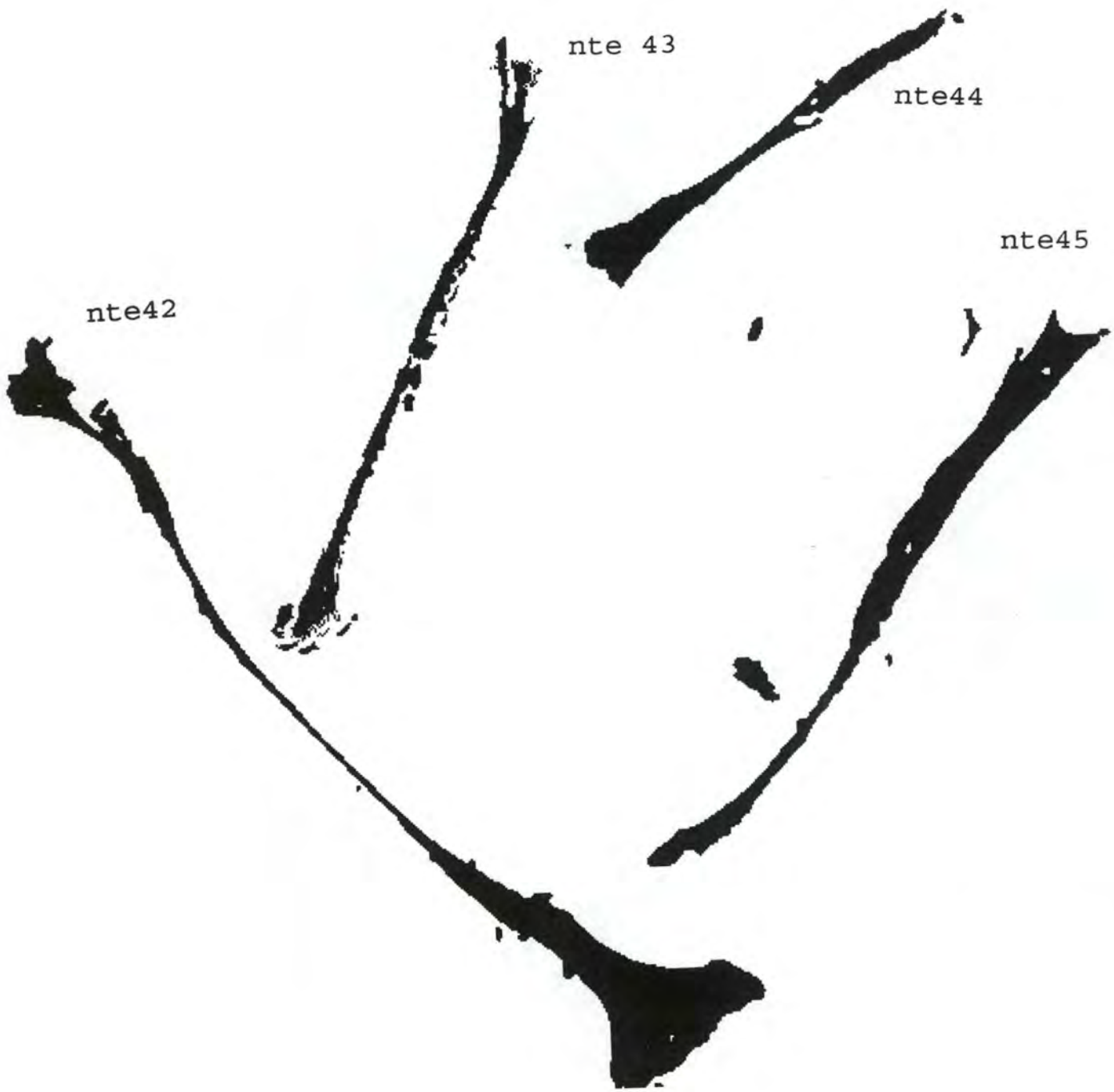
nte32

nte 33

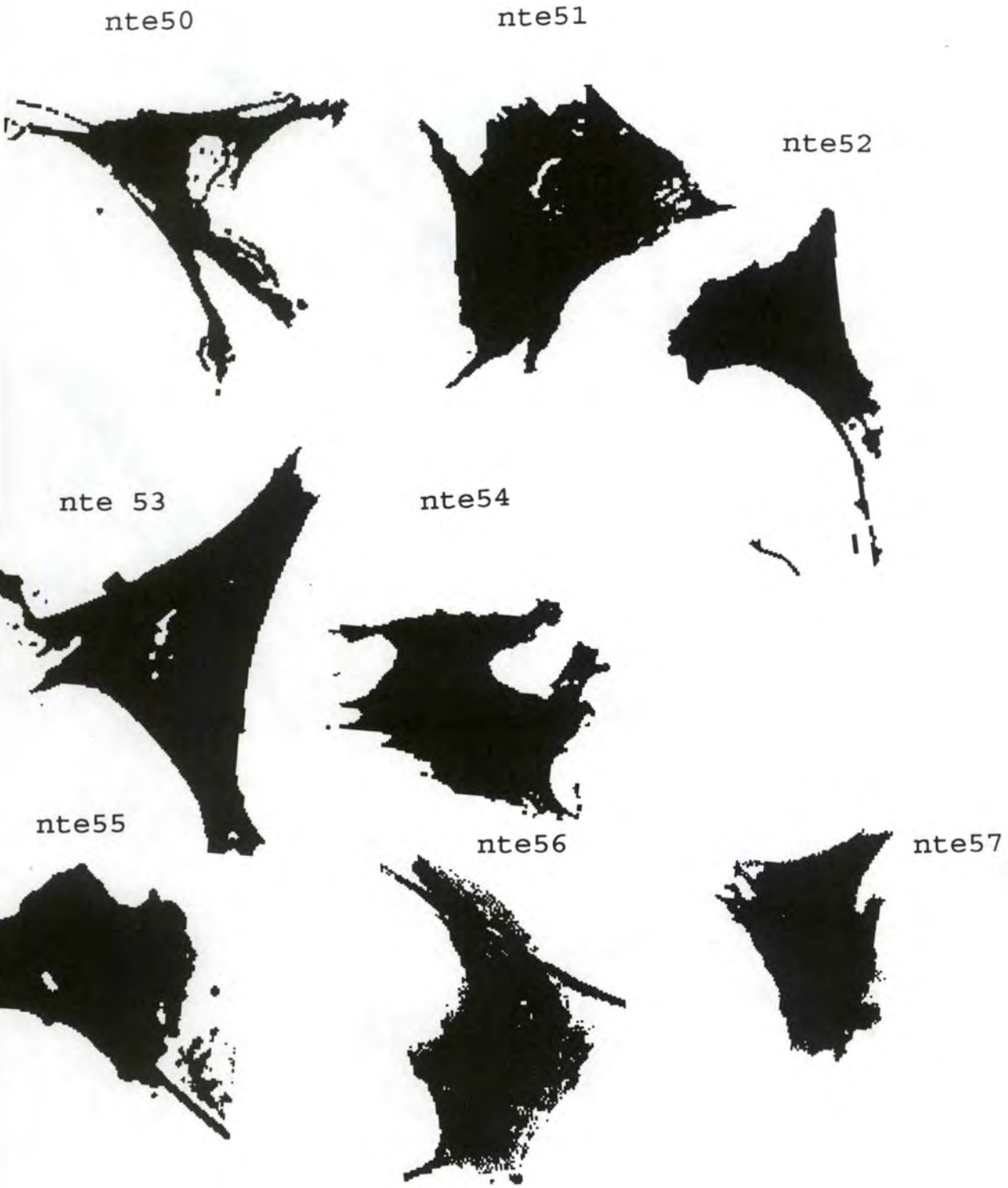
A8 Cellules de la série NTE (classe IV)



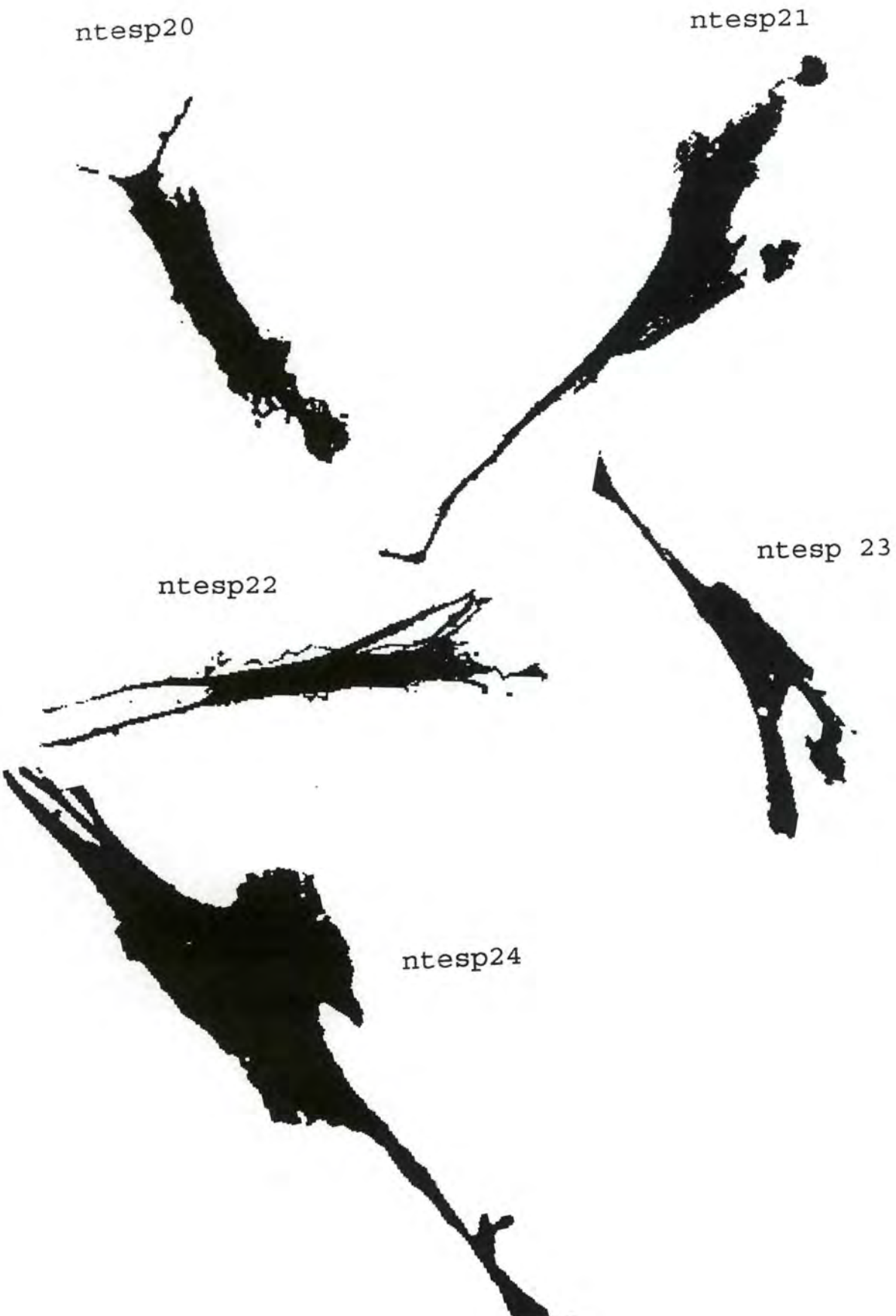
A9 Cellules de la série NTE (classe IV)



A10 Cellules de la série NTE (classe V)



A11 Cellules de la série NTESP (classe III, IV ou V)



A12 Cellules de la série NTS (classe II)

nts20

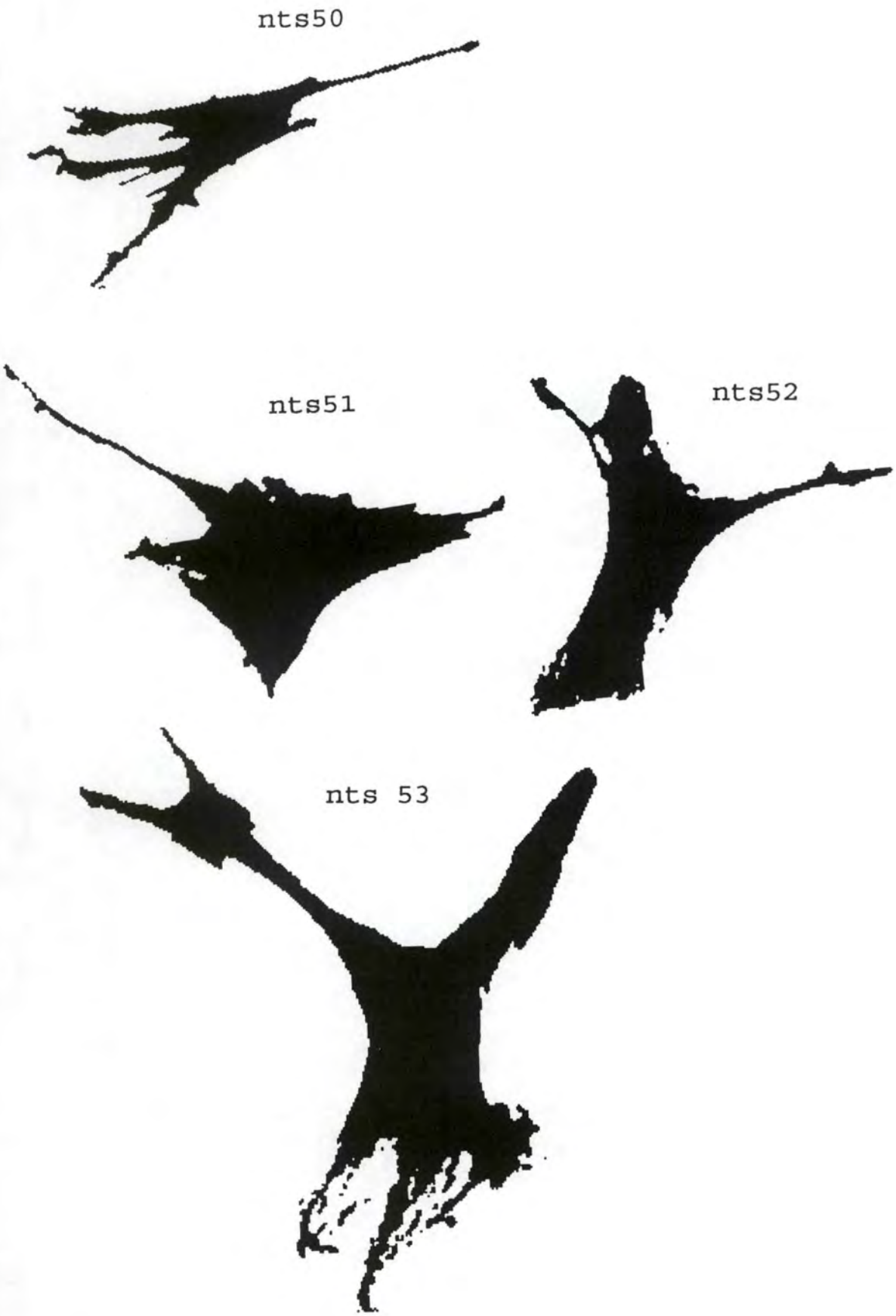
nts21

A13 Cellules de la série NTS (classe III)

nts32

nts38

A14 Cellules de la série NTS (classe V)



A15 Liste des fichiers ('sets') utilisés

1. Bas1

01-10:	NTR210	NTR310	NTR410	NTR510	NTR211	NTR311	NTR411	NTR511	NTR212	NTR312
11-20:	NTR412	NTR512	NTR213	NTR313	NTR413	NTR513	NTR214	NTR314	NTR414	NTR514
21-30:	NTR215	NTR315	NTR415	NTR515	NTR216	NTR316	NTR416	NTR516	NTR217	NTR317
31-40:	NTR417	NTR517	NTR220	NTR320	NTR420	NTR520	NTR221	NTR321	NTR421	NTR521
41-50:	NTR222	NTR322	NTR422	NTR522	NTR223	NTR323	NTR423	NTR523	NTR224	NTR324
51-60:	NTR424	NTR524	NTR225	NTR325	NTR425	NTR525	NTR226	NTR326	NTR426	NTR526
61-70:	NTR227	NTR327	NTR427	NTR527	NTR230	NTR330	NTR430	NTR530	NTR231	NTR331
71-80:	NTR431	NTR531	NTR232	NTR332	NTR432	NTR532	NTR233	NTR333	NTR433	NTR533
81-90:	NTR234	NTR334	NTR434	NTR534	NTR235	NTR335	NTR435	NTR535	NTR236	NTR336
91-100:	NTR436	NTR536	NTR237	NTR337	NTR437	NTR537	NTR240	NTR340	NTR440	NTR540
101-110:	NTR241	NTR341	NTR441	NTR541	NTR242	NTR342	NTR442	NTR542	NTR243	NTR343
111-120:	NTR443	NTR543	NTR244	NTR344	NTR444	NTR544	NTR245	NTR345	NTR445	NTR545
121-128:	NTR246	NTR346	NTR446	NTR546	NTR247	NTR347	NTR447	NTR547		

2. Bas0

01-10:	NTR210	NTR310	NTR410	NTR510	NTR220	NTR320	NTR420	NTR520	NTR230	NTR330
11-16:	NTR430	NTR530	NTR240	NTR340	NTR440	NTR540				

3. Bas2

01-10:	NTR210	NTR211	NTR212	NTR213	NTR214	NTR215	NTR216	NTR217	NTR220	NTR221
11-20:	NTR222	NTR223	NTR224	NTR225	NTR226	NTR227	NTR230	NTR231	NTR232	NTR233
21-30:	NTR234	NTR235	NTR236	NTR237	NTR240	NTR241	NTR242	NTR243	NTR244	NTR245
31-40:	NTR246	NTR247	NTR310	NTR311	NTR312	NTR313	NTR314	NTR315	NTR316	NTR317
41-50:	NTR320	NTR321	NTR322	NTR323	NTR324	NTR325	NTR326	NTR327	NTR330	NTR331
51-60:	NTR332	NTR333	NTR334	NTR335	NTR336	NTR337	NTR340	NTR341	NTR342	NTR343
61-70:	NTR344	NTR345	NTR346	NTR347	NTR410	NTR411	NTR412	NTR413	NTR414	NTR415
71-80:	NTR416	NTR417	NTR420	NTR421	NTR422	NTR423	NTR424	NTR425	NTR426	NTR427
81-90:	NTR430	NTR431	NTR432	NTR433	NTR434	NTR435	NTR436	NTR437	NTR440	NTR441
91-100:	NTR442	NTR443	NTR444	NTR445	NTR446	NTR447	NTR510	NTR511	NTR512	NTR513
101-110:	NTR514	NTR515	NTR516	NTR517	NTR520	NTR521	NTR522	NTR523	NTR524	NTR525
111-120:	NTR526	NTR527	NTR530	NTR531	NTR532	NTR533	NTR534	NTR535	NTR536	NTR537
121-128:	NTR540	NTR541	NTR542	NTR543	NTR544	NTR545	NTR546	NTR547		

4. Bas3

01-10:	NTR210	NTR310	NTR410	NTR510	NTR220	NTR320	NTR420	NTR520	NTR230	NTR330
11-20:	NTR430	NTR530	NTR240	NTR340	NTR440	NTR540	NTR211	NTR311	NTR411	NTR511
21-30:	NTR221	NTR321	NTR421	NTR521	NTR231	NTR331	NTR431	NTR531	NTR241	NTR341
31-40:	NTR441	NTR541	NTR212	NTR312	NTR412	NTR512	NTR222	NTR322	NTR422	NTR522
41-50:	NTR232	NTR332	NTR432	NTR532	NTR242	NTR342	NTR442	NTR542	NTR213	NTR313
51-60:	NTR413	NTR513	NTR223	NTR323	NTR423	NTR523	NTR233	NTR333	NTR433	NTR533
61-70:	NTR243	NTR343	NTR443	NTR543	NTR214	NTR314	NTR414	NTR514	NTR224	NTR324
71-80:	NTR424	NTR524	NTR234	NTR334	NTR434	NTR534	NTR244	NTR344	NTR444	NTR544
81-90:	NTR215	NTR315	NTR415	NTR515	NTR225	NTR325	NTR425	NTR525	NTR235	NTR335
91-100:	NTR435	NTR535	NTR245	NTR345	NTR445	NTR545	NTR216	NTR316	NTR416	NTR516
101-110:	NTR226	NTR326	NTR426	NTR526	NTR236	NTR336	NTR436	NTR536	NTR246	NTR346
111-120:	NTR446	NTR546	NTR217	NTR317	NTR417	NTR517	NTR227	NTR327	NTR427	NTR527
121-128:	NTR237	NTR337	NTR437	NTR537	NTR247	NTR347	NTR447	NTR547		

A16 Liste des fichiers ('sets') utilisés (suite)

5. Long1

01-10:	NTR210	NTR310	NTR410	NTR510	NTR211	NTR311	NTR411	NTR511	NTR212	NTR312
11-20:	NTR412	NTR512	NTR213	NTR313	NTR413	NTR513	NTR214	NTR314	NTR414	NTR514
21-30:	NTR215	NTR315	NTR415	NTR515	NTR216	NTR316	NTR416	NTR516	NTR217	NTR317
31-40:	NTR417	NTR517	NTR220	NTR320	NTR420	NTR520	NTR221	NTR321	NTR421	NTR521
41-50:	NTR222	NTR322	NTR422	NTR522	NTR223	NTR323	NTR423	NTR523	NTR224	NTR324
51-60:	NTR424	NTR524	NTR225	NTR325	NTR425	NTR525	NTR226	NTR326	NTR426	NTR526
61-70:	NTR227	NTR327	NTR427	NTR527	NTR230	NTR330	NTR430	NTR530	NTR231	NTR331
71-80:	NTR431	NTR531	NTR232	NTR332	NTR432	NTR532	NTR233	NTR333	NTR433	NTR533
81-90:	NTR234	NTR334	NTR434	NTR534	NTR235	NTR335	NTR435	NTR535	NTR236	NTR336
91-100:	NTR436	NTR536	NTR237	NTR337	NTR437	NTR537	NTR240	NTR340	NTR440	NTR540
101-110:	NTR241	NTR341	NTR441	NTR541	NTR242	NTR342	NTR442	NTR542	NTR243	NTR343
111-120:	NTR443	NTR543	NTR244	NTR344	NTR444	NTR544	NTR245	NTR345	NTR445	NTR545
121-130:	NTR246	NTR346	NTR446	NTR546	NTR247	NTR347	NTR447	NTR547	NTR250	NTR350
131-140:	NTR450	NTR550	NTR251	NTR351	NTR451	NTR551	NTR252	NTR352	NTR452	NTR552
141-150:	NTR253	NTR353	NTR453	NTR553	NTR254	NTR354	NTR454	NTR554	NTR255	NTR355
151-160:	NTR455	NTR555	NTR256	NTR356	NTR456	NTR556	NTR257	NTR357	NTR457	NTR557
161-170:	NTR260	NTR360	NTR460	NTR560	NTR261	NTR361	NTR461	NTR561	NTR262	NTR362
171-180:	NTR462	NTR562	NTR263	NTR363	NTR463	NTR563	NTR264	NTR364	NTR464	NTR564
181-190:	NTR265	NTR365	NTR465	NTR565	NTR266	NTR366	NTR466	NTR566	NTR267	NTR367
191-200:	NTR467	NTR567	NTR270	NTR370	NTR470	NTR570	NTR271	NTR371	NTR471	NTR571
201-210:	NTR272	NTR372	NTR472	NTR572	NTR273	NTR373	NTR473	NTR573	NTR274	NTR374
211-220:	NTR474	NTR574	NTR275	NTR375	NTR475	NTR575	NTR276	NTR376	NTR476	NTR576
221-230:	NTR277	NTR377	NTR477	NTR577	NTR280	NTR380	NTR480	NTR580	NTR281	NTR381
231-240:	NTR481	NTR581	NTR282	NTR382	NTR482	NTR582	NTR283	NTR383	NTR483	NTR583
241-250:	NTR284	NTR384	NTR484	NTR584	NTR285	NTR385	NTR485	NTR585	NTR286	NTR386
251-260:	NTR486	NTR586	NTR287	NTR387	NTR487	NTR587	NTR290	NTR390	NTR490	NTR590
261-270:	NTR291	NTR391	NTR491	NTR591	NTR292	NTR392	NTR492	NTR592	NTR293	NTR393
271-280:	NTR493	NTR593	NTR294	NTR394	NTR494	NTR594	NTR295	NTR395	NTR495	NTR595
281-288:	NTR296	NTR396	NTR496	NTR596	NTR297	NTR397	NTR497	NTR597		

6. All1

01-10:	NTR210	NTR310	NTR410	NTR510	NTR220	NTR320	NTR420	NTR520	NTR230	NTR330
11-20:	NTR430	NTR530	NTR240	NTR340	NTR440	NTR540	NTR250	NTR350	NTR450	NTR550
21-30:	NTR260	NTR360	NTR460	NTR560	NTR270	NTR370	NTR470	NTR570	NTR280	NTR380
31-36:	NTR480	NTR580	NTR290	NTR390	NTR490	NTR590				

7. Test2

01-10:	NTE200	NTE210	NTE220	NTE230	NTE240	NTE250	NTE260	NTE270	NTE280	NTE290

8. Test3

01-04:	NTE300	NTE310	NTE320	NTE330

9. Test4

01-06:	NTE400	NTE410	NTE420	NTE430	NTE440	NTE450

A17 Liste des fichiers ('sets') utilisés (fin)

10. Test5

01-08: NTE500 NTE510 NTE520 NTE530 NTE540 NTE550 NTE560 NTE570

11. Testsp

01-05: NTESP200 NTESP210 NTESP220 NTESP230 NTESP240

12. Ts1R

01-10: NTS200 NTS201 NTS202 NTS203 NTS204 NTS205 NTS206 NTS207 NTS210 NTS211
11-16: NTS212 NTS213 NTS214 NTS215 NTS216 NTS217

13. Ts2R

01-08: NTS320 NTS321 NTS322 NTS323 NTS324 NTS325 NTS326 NTS327

14. Ts3R

01-08: NTS380 NTS381 NTS382 NTS383 NTS384 NTS385 NTS386 NTS387

15. Ts4R

01-10: NTS500 NTS501 NTS505 NTS503 NTS504 NTS505 NTS506 NTS507 NTS510 NTS511
11-30: NTS512 NTS513 NTS514 NTS515 NTS516 NTS517 NTS520 NTS521 NTS522 NTS523
31-30: NTS524 NTS525 NTS526 NTS527 NTS530 NTS531 NTS532 NTS533 NTS534 NTS535
31-32: NTS536 NTS537

16. F1

01-10: NTR210 NTR310 NTR410 NTR510 NTR211 NTR311 NTR411 NTR511 NTR212 NTR312
11-20: NTR412 NTR512 NTR213 NTR313 NTR413 NTR513 NTR214 NTR314 NTR414 NTR514
21-30: NTR215 NTR315 NTR415 NTR515 NTR216 NTR316 NTR416 NTR516 NTR217 NTR317
31-40: NTR417 NTR517

A18 Implémentation en TurboPasal pour Windows de CELLTOT (1/12)

```
program celltot;
uses wincrt;
(*par vincent van der Kaa pour Mémoire en Informatique
        Listen the man said
        comprend: fonctionnalités transformation, contour, 'london',
        taille, centrage et création/gestion d'un fichier
        d'apprentissage*)

(*ti définit la longueur en pixel de l'image, cette image DOIT être carrée
et avoir un ti inférieur ou égal à 100*)
const ti=99;
const surf=9999; (*surf=tiXti est simplement la surface en pixel de l'image*)
const out=5; (*nbre de sorties du réseau*)
const csdebut=1;const csfin=9; (*nbre de série*)
const crdebut=0;const crfin=7; (*nbre de rotation*)
const ccdebut=2;const ccfin=5; (*nbre de classe*)

type visiongr1d=array [0..23500] of char;(*val max de vis 24000 et 3Xsurf*)
type vision1d=array [0..surf] of char;
type vision2d=array [0..ti,0..ti] of char;
type visionlon=array [0..ti,1..2] of char;
type training=array [0..out] of char;
type pix=record carac:char;end;
type img=file of pix;
type nom=string[30];
type sol=integer;

var pixel:pix;
var image:img;
var suite:array [0..9] of char;
var comser,comrot,comclas,longimg,numrot,choix:integer;

(*var pour phase 1 ; *)
var nomred,nomimg,nomnorm:nom;
var tabimggr1d:visiongr1d;
var tabimg1d:vision1d;

(*var pour phase 2;*)
var nomfichier1,nomfichier2,nomfichier3,nomfichier4,nomcour:nom;
var fichier1,fichier2,fichier3,fichier4:img;
var i,j:integer;
var tabtrain:training;
var tabimg:vision2d;
var tablon,tabcentr,tabtail,tabcent2:visionlon;
var a,b:char; (*prise en compte de la taille*)
```

A19 Implémentation en TurboPasal pour Windows de CELLTOT (2/12)

```
(*var pour phase 3;*)
var nomfinit1,nomfinit2,nomfinit3,nomfinit4:nom;
var finit1,finit2,finit3,finit4:img;

procedure modiftab2(var tabin:vision1d;var tabout:vision2d);
(*transforme un tableau vecteur d'une longueur de surf en un tableau matrice
d'une taille de de ti * ti)
var compi,compj,compt:integer;
begin
    for compt:=0 to surf do
    begin
        compi:=compt div (ti+1);   (*division entière*)
        compj:=compt mod (ti+1);    (*reste*)
        tabout[compi,compj]:=tabin[compt];
    end;
end;

procedure modiftab1(var tabin:vision2d;var tabout:vision1d);
(*transforme un tableau matrice ti * ti en un tableau vecteur
d'une taille de surf*)
var compi,compj,compt:integer;
begin
    for compi:=0 to ti do
    begin
        for compj:=0 to ti do
        begin
            compt:=compi*(ti+1)+compj;
            tabout[compt]:=tabin[compi,compj];
        end;
    end;
end;

procedure met_img_tabgr1D (var nominit:nom;var tab1d:visiongr1d;
                    var longtab:integer);
(*ouvre un fichier/img de taille inconnue et le place dans un tableau 1D-vecteur*)
var imgcour:img;
begin
    assign (imgcour,nominit);
    reset (imgcour);
    longtab:=-1;
    while not eof(imgcour) do
    begin
        longtab:=longtab+1;
        read (imgcour,pixel);
        tab1d[longtab]:=pixel.carac;
    end;
```

A20 Implémentation en TurboPasal pour Windows de CELLTOT (3/12)

```
    close (imgcour);
end;

procedure met_img_tab1d (var nominit:nom;var tab1d:vision1d);
(*ouvre un fichier/img de taille surf et le place dans un tableau1d-vecteur*)
var imgcour:img;
var long :integer;
begin
    assign (imgcour,nominit);
    reset (imgcour);
    for long:=0 to surf do
    begin
        read (imgcour,pixel);
        tab1d[long]:=pixel.carac;
    end;
    close (imgcour);
end;

procedure trans_tabGr1d_tab1D (var tab1dold:visiongr1d;
                               var tab1dnew:vision1d;var longtab:integer);
(*transforme un tableau1D de taille inconnue avec 0 et 255 en un tableau 1d avec des
0 pour blanc et 1 pour noir de taille surf*)
var comptnew,comptold:integer;
var test:char;
begin
    comptnew:=-1;
    for comptold:=0 to longtab do
    begin
        test:=(tab1dold[comptold]);
        if test='0'then
        begin
          comptnew:=comptnew+1;
          tab1dnew[comptnew]:='0';
        end
        else if test='2' then
        begin
          comptnew:=comptnew+1;
          tab1dnew[comptnew]:='1';
        end;
    end;
end;

procedure insc_tab1D_fich (var nominit:nom;var tab1d:vision1d);
(*inscrit un tableau 1d de taille surf ds un fichier*)
var imgcour:img;
var pixel:pix;
```

A21 Implémentation en TurboPasal pour Windows de CELLTOT (4/12)

```
var compix,comtrain:integer;
begin
    assign (imgcour,nominit);
    rewrite (imgcour);
    for compix:=0 to surf do
    begin
        pixel.carac:=tab1d[compix];
        write (imgcour,pixel);
    end;
    close (imgcour);
end;

procedure cree_va_train (var solinit:sol;var train:training);
(*crée le train d'entraînement selon la classe connue*)
var comtrain:integer;
begin
    for comtrain:=1 to out do
       train[comtrain]:='0';
    train[solinit]:='1';
    train[0]:=suite[solinit];
end;

procedure contour (var projxy:vision2d);
(*A partir d'une image codée sur une matrice ti*ti, cet algorithme ne retient que les pixels qui
appartiennent au contour de l'objet représenté*)
var compx,compy:integer;
begin
for compx:=1 to ti-1 do
        begin
        for compy:=1 to ti-1 do
           begin
           if projxy[compx,compy]='1' then
        if (((projxy[compx-1,compy-1]= '1')or (projxy[compx-1,compy-1]= '2'))
        and ((projxy[compx,compy-1]= '1')or (projxy[compx,compy-1]= '2'))
        and ((projxy[compx+1,compy-1]= '1')or (projxy[compx+1,compy-1]= '2'))
        and ((projxy[compx-1,compy]= '1')or (projxy[compx-1,compy]= '2'))
        and ((projxy[compx+1,compy]= '1')or (projxy[compx+1,compy]= '2'))
        and ((projxy[compx-1,compy+1]= '1')or (projxy[compx-1,compy+1]= '2'))
        and ((projxy[compx,compy+1]= '1')or (projxy[compx,compy+1]= '2'))
        and ((projxy[compx+1,compy+1]= '1')or (projxy[compx+1,compy+1]= '2')))
           then projxy[compx,compy]:='2';
           end;
        end;
```

A22 Implémentation en TurboPasal pour Windows de CELLTOT (5/12)

```
for compx:=1 to ti-1 do
        begin
        for compy:=1 to ti-1 do
           begin
           if projxy[compx,compy] = '2' then projxy[compx,compy] := '0';
           end;
        end;
   end;

procedure london (var proj:vision2d;var lond:visionlon);
(* compte le nombre de changements verticaux et horizontaux d'un objet
        représenté sur une image ti * ti*)
var change,couleur,compx,compy:integer;
begin
(*BOUCLE1*)
      for compy:=0 to ti do
      begin
      change:=0;couleur:=0;
      for compx:=0 to ti do
         begin
         if (proj[compx,compy]='1')then
         begin
            if couleur=0 then
            begin
              change:=change+1;
              couleur:=1;
            end;
         end;
         if ((proj[compx,compy]='0') and (couleur=1))
           then begin
              change:=change+1;
              couleur:=0;
              end;
         end;
      if proj[ti,compy]='1' then change:=change+1;
      change:=change div 2;
      if change >= 9 then lond[compy,1]:=(suite[9])
      else lond[compy,1]:=(suite[change]);
      end;
```

A23 Implémentation en TurboPasal pour Windows de CELLTOT (6/12)

```
(*BOUCLE2*)
      for compx:=0 to ti do
      begin
      change:=0;couleur:=0;
      for compy:=0 to ti do
        begin
        if ((proj[compx,compy]='1') and (couleur=0))
          then begin
              change:=change+1;
              couleur:=1;
              end;
        if ((proj[compx,compy]='0') and (couleur=1))
          then begin
             change:=change+1;
              couleur:=0;
              end;
        end;
      if proj[compx,ti]='1' then change:=change+1;
      change:=change div 2;
      if change >= 9 then lond[compx,2]:=(suite[9])
      else lond[compx,2]:=(suite[change]);
      end;
end;

procedure centrer (var lond:visionlon;var centr:visionlon);
(* centre l'objet sur une image de type ti * ti*)
var ga,dr,mi,decalage,compli,compco:integer;
        begin
        mi:=ti div 2;
        for compli:=1 to 2 do
        begin
            ga:=0;dr:=ti;
            while lond[ga,compli] ='0' do
                ga:=ga+1;
            while lond[dr,compli] ='0' do
                dr:=dr-1;
            decalage:=mi-((ga+dr) div 2);
            for compco:=ga to dr do
               centr[decalage+compco,compli]:=lond[compco,compli];
            for compco:=0 to ga-1+decalage do centr[compco,compli]:='0';
            for compco:=dr+1+decalage to ti do centr[compco,compli]:='0';
        end;
    end;
```

A24 Implémentation en TurboPasal pour Windows de CELLTOT (7/12)

```
procedure taille (var proj:vision2d;var tail:visionlon);
(*calcule le nombre de pixels noir (0) sur chaque ligne et chaque colonne de l'image ti*ti*)
var change,compx,compy,tot,comp:integer;
var sc:char;
(*BOUCLE1*)
  begin
    for compy:=0 to ti do
    begin
    tot:=0;
    for compx:=0 to ti do
      begin
      if (proj[compx,compy]='1') then
          tot:=tot+1;
      end;

    case tot of  0..0:sc:='0';
                 1..5:sc:='1';
                 6..10:sc:='2';
                 11..15:sc:='3';
                 16..20:sc:='4';
                 21..25:sc:='5';
                 26..40:sc:='6';
                 41..55:sc:='7';
                 56..73:sc:='8';
                 74..99:sc:='9';
    end;
    tail[compy,1]:=sc;
    end;

(* BOUCLE 2*)
 for compx:=0 to ti do
    begin
    tot:=0;
    for compy:=0 to ti do
      begin
      if (proj[compx,compy]='1') then
          tot:=tot+1;
      end;
```

A25 Implémentation en TurboPasal pour Windows de CELLTOT (8/12)

```
case tot of  0..0:sc:='0';
                    1..5:sc:='1';
                    6..10:sc:='2';
                    11..15:sc:='3';
                    16..20:sc:='4';
                    21..25:sc:='5';
                    26..40:sc:='6';
                    41..55:sc:='7';
                    56..73:sc:='8';
                    74..99:sc:='9';
      end;
      tail[compx,2]:=sc;
      end;
      end;

(**********************DEBUT PROGRAM*********************************)
begin
(*initialisation d'un tableau de transfert char <==> integer*)
suite[0]:='0';suite[1]:='1';suite[2]:='2';suite[3]:='3';
      suite[4]:='4';suite[5]:='5';suite[6]:='6';suite[7]:='7';
      suite[8]:='8';suite[9]:='9';
writeln ('Welcome to the cell treatment module');
writeln ('Writen by Vincent van der Kaa');
writeln ('1 pour normaliser, 2 pour transformer, 3 pour ajouter');
readln (choix);
if choix = 1 then

begin
(*Option 1, mettre en format 0/1 sans var d'entraînement dans des
fichiers qui portent comme prénom n pour normalisé*)
for comser:=csdebut to csfin do
begin
   for comrot:=crdebut to crfin do
   begin
      for comclas:=ccdebut to ccfin do
      begin
         nomred:=('tr'+suite[comclas]+suite[comser]+'m');
         nomimg:=((nomred)+suite[comrot]);
         writeln (nomimg);
         met_img_tabgr1d (nomimg,tabimggr1d,longimg);
         trans_tabGr1d_tab1d (tabimggr1d,tabimg1d,longimg);
         if crfin=1 then numrot:=comrot*4 else numrot:=comrot;
         nomnorm:= ('ntr'+suite[comclas]+suite[comser]+'m'+suite[numrot]);
         insc_tab1d_fich (nomnorm,tabimg1d);
      end;
   end;
```

A26 Implémentation en TurboPasal pour Windows de CELLTOT (9/12)

```
end;
end;

if choix>=2 then
(*Option 2 :
(*transforme un ensemble de fichiers-images
(0 et 1): normalisé en un fichier .dat. Ce fichier peut être utilisé
comme apprentissage ou test pour un réseau de neurones. *)

begin
write ('Ecris 4 fichiers 1 pour contour l''autre pour london centre ');
writeln ('Le 3 pour london et taile centré et le 4 pour les noms ');

Write ('nom du fichier contour?');
readln (nomfichier1);
assign (fichier1,nomfichier1);
rewrite (fichier1);

Write ('nom du fichier london?');
readln (nomfichier2);
assign (fichier2,nomfichier2);
rewrite (fichier2);
Write ('nom du fichier taille?');
readln (nomfichier3);
assign (fichier3,nomfichier3);
rewrite (fichier3);

Write ('nom du fichier de noms?');
readln (nomfichier4);
assign (fichier4,nomfichier4);
rewrite (fichier4);

If choix=3 then
begin
    Write ('nom du fichier ancien contour?');
    readln (nomfinit1);
    assign (finit1,nomfinit1);
    reset (finit1);
    while not eof(finit1) do
        begin
        read (finit1,pixel);
        write (fichier1,pixel);
        end;
    close (finit1);

    Write ('nom du fichier ancien centre?');
```

A27 Implémentation en TurboPasal pour Windows de CELLTOT (10/12)

```
readln (nomfinit2);
    assign (finit2,nomfinit2);
    reset (finit2);
     while not eof(finit2) do
        begin
        read (finit2,pixel);
        write (fichier2,pixel);
        end;
    close (finit2);

    Write ('nom du fichier ancien contour?');
    readln (nomfinit3);
    assign (finit3,nomfinit1);
    reset (finit3);
    while not eof(finit3) do
        begin
        read (finit3,pixel);
        write (fichier3,pixel);
        end;
    close (finit3);

    Write ('nom du fichier ancien contour?');
    readln (nomfinit4);
    assign (finit4,nomfinit4);
    reset (finit4);
    while not eof(finit4) do
        begin
        read (finit4,pixel);
        write (fichier4,pixel);
        end;
    close (finit4);
end;

for comser:=csdebut to csfin do
begin
   for comrot:=crdebut to crfin do
   begin
      for comclas:=ccdebut to ccfin do
      begin
      nomcour:=('ntr'+suite[comclas]+suite[comser]+'m'+suite[comrot]);
      cree_va_train (comclas,tabtrain);
      writeln (nomcour);
      met_img_tab1d (nomcour,tabimg1d);
      modiftab2(tabimg1d,tabimg);
      taille (tabimg,tabtail);
      contour (tabimg);
```

A28 Implémentation en TurboPasal pour Windows de CELLTOT (11/12)

```
for i:=0 to ti do
        begin
          for j:=0 to ti do
          begin
              pixel.carac:=tabimg[i,j];
              write (fichier1,pixel);
          end;
        end;
        for i:=0 to out do
        begin
           pixel.carac:=tabtrain[i];
           write (fichier1,pixel);
        end;

        london (tabimg,tablon);
        centrer (tablon,tabcentr);

        centrer (tabtail,tabcent2);

        pixel.carac:='N';
        write (fichier4,pixel);
        pixel.carac:=suite[comclas];
        write (fichier4,pixel);
        pixel.carac:=suite[comser];
        write (fichier4,pixel);
        pixel.carac:=suite[comrot];
        write (fichier4,pixel);

        for j:=1 to 2 do
        begin
          for i:= 0 to ti do
          begin
              pixel.carac:=tabcentr[i,j];
              write (fichier2,pixel);
              write (fichier3,pixel);
          end;
        end;

        for j:=1 to 2 do
        begin
          for i:= 0 to ti do
          begin
              pixel.carac:=tabcent2[i,j];
              write (fichier3,pixel);
          end;
```

A29 Implémentation en TurboPasal pour Windows de CELLTOT (12/12)

```
        for i:=0 to out do
        begin
            pixel.carac:=tabtrain[i];
            write (fichier2,pixel);
            write (fichier3,pixel);
        end;
        end;
    end;
end;
    close (fichier1);
    close (fichier2);
    close (fichier3);
    close (fichier4);

end;
end.
(************************FIN PROGRAM*********************************)
```

A30 Implémentation en TurboPasal pour Windows de CELLROT (1/4)

```
program cellrot;
(*Par vincent van der Kaa*)
(* émulation de rotation à partir d'un fichier-image de taille surf*)
90° X 3 *)
(*utilise une image 1d avec 0/1 d'un nom donné et l'enregistre dans un fichier
ainsi que trois autres fichiers/img calculées par trois rotation successive de 90°
  *)
uses wincrt;

(*ti définit la longueur en pixel de l'image, cette image DOIT être carrée
et avoir un ti inférieur ou égal à 100*)
const ti=99;
const surf=9999; (*surf=tiXti est simplement la surface en pixel de l'image*)

type vision1d=array [0..surf] of char;
type vision2d=array [0..ti,0..ti] of char;
type pix=record carac:char;end;
type img=file of pix;
type nom=string[30];
type sol=integer;

var t1d:vision1d;
var t0,t1,t2:vision2d;
var nomimg,nomred,nomnorm,nomrot,nomnew:nom;
var pixel:pix;
var image:img;
var suite:array [0..9] of char;
var comser,comrot,comclas:integer;
var rotj,roti,nrmaj,numrot,longimg:integer;

procedure modiftab2(var tabin:vision1d;var tabout:vision2d);
(*transforme un tableau 1d vecteur en un tableau 2d matrice
taille de 0 à n-1*)
var compi,compj,compt:integer;
begin
    for compt:=0 to surf do
    begin
        compi:=compt div (ti+1);
        compj:=compt mod (ti+1);
        tabout[compi,compj]:=tabin[compt];
    end;
end;
```

A31 Implémentation en TurboPasal pour Windows de CELLROT (2/4)

```
procedure modiftab1(var tabin:vision2d;var tabout:vision1d);
(*transforme un tableau matrice en un tableau vecteur
taille de 0 à n-1*)
var compi,compj,compt:integer;
begin
    for compi:=0 to ti do
    begin
        for compj:=0 to ti do
        begin
            compt:=compi*(ti+1)+compj;
            tabout[compt]:=tabin[compi,compj];
        end;
    end;
end;

procedure met_img_tab1d (var nominit:nom;var tab1d:vision1d);
(*ouvre un fichier/img de taille surf et le place ds un tableau*)
var imgcour:img;
var long :integer;
begin
    assign (imgcour,nominit);
    reset (imgcour);
    for long:=0 to surf do
    begin
        read (imgcour,pixel);
        tab1d[long]:=pixel.carac;
    end;
    close (imgcour);
end;

procedure insc_tab1D_fich (var nominit:nom;var tab1d:vision1d);
var imgcour:img;
var pixel:pix;
var compix,comtrain:integer;
begin
    assign (imgcour,nominit);
    rewrite (imgcour);
    for compix:=0 to surf do
    begin
        pixel.carac:=tab1d[compix];
        write (imgcour,pixel);
    end;
    close (imgcour);
end;
```

A32 Implémentation en TurboPasal pour Windows de CELLROT (3/4)

```
(*****************DEBUT PROGRAM*******************)
begin
(*initialisation d'un tableau de transfert char <==> integer*)
suite[0]:='0';suite[1]:='1';suite[2]:='2';suite[3]:='3';
suite[4]:='4';suite[5]:='5';suite[6]:='6';suite[7]:='7';
 suite[8]:='8';suite[9]:='9';

writeln ('Welcome to the cell rotation treatment module');

(* mettre en format 0/1 avec var d'entraînements dans des
fichiers qui porte comme prénom n pour normalisé
posibilité d'émuler trois rotation de 90°*)

for comser:=5 to 9 do
begin
    for comrot:=4 to 4  do
    begin
        for comclas:=2 to 5 do
        begin
            nomred:=('ntr'+suite[comclas]+suite[comser]+'m');
            nomimg:=(nomred+suite[comrot]);
            writeln (nomimg);
            met_img_tab1d (nomimg,t1d);
            modiftab2(t1d,t0);

            for rotj:= 0 to ti do
            begin
                for roti:=0 to ti do
                begin
                    t1[rotj,ti-roti]:=t0[roti,rotj];
                end;
            end;

            for rotj:= 0 to ti do
            begin
                for roti:=0 to ti do
                begin
                    t2[rotj,ti-roti]:=t1[roti,rotj];
                end;
            end;

            nrmaj:=comrot+1;
            nomnew:=(nomred+suite[nrmaj]);
            modiftab1(t1,t1d);
            insc_tab1D_fich (nomnew,t1d);
```

A33 Implémentation en TurboPasal pour Windows de CELLROT (4/4)

```
        nrmaj:=nrmaj+1;
        nomnew:=(nomred+suite[nrmaj]);
        modiftab1(t2,t1d);
        insc_tab1D_fich (nomnew,t1d);

        for rotj:= 0 to ti do
        begin
            for roti:=0 to ti do
            begin
            t1[rotj,ti-roti]:=t2[roti,rotj];
            end;
        end;

        nrmaj:=nrmaj+1;
        nomnew:=(nomred+suite[nrmaj]);
        modiftab1(t1,t1d);
        insc_tab1D_fich (nomnew,t1d);
        end;
    end;

(**************************FIN PROGRAM**********************)
end.
```