ORSAY
N° d'ordre

UNIVERSITE DE PARIS-SUD
CENTRE D'ORSAY

THESE

présentée
pour obtenir

LE TITRE DE DOCTEUR EN SCIENCES

Spécialité Informatique

par

Emmanuel VIENNET

Architectures Connexionnistes Multi-Modulaires
Application à l'Analyse de Scène

Soutenue le **18 juin 1993** devant la commission d'examen composée de

M.	Yves Kodratoff	President
M.	Christoph von der Malsburg	Rapporteur
M.	Bernard Victorri	Rapporteur
Mme.	Françoise Fogelman	Examinateur
M.	Michel Weinfeld	Examinateur

Architectures Connexionnistes Multi-Modulaires
Application à l'Analyse de Scène

Thèse présentée à l'Université de Paris-Sud
pour obtenir
LE TITRE DE DOCTEUR EN SCIENCES

Spécialité Informatique

par

Emmanuel VIENNET

Signatures des membres du jury :

M. Yves Kodratoff
Président

...

M. Christoph von der Malsburg
Rapporteur

M. Bernard Victorri
Rapporteur

Mme. Françoise Fogelman
Examinateur

M. Michel Weinfeld
Examinateur

... ...

Fait à Orsay le 18 juin 1993.

Connectionist Multi-Modular Architectures
Application to Scene Analysis

---

by
Emmanuel VIENNET

Table des matières

Table des figures

Abréviations

ACP	Analyse en Composantes Principales
ISR	Integrated Segmentation and Recognition
kNN	k Nearest Neighbor (k plus proches voisins)
LVQ	Learning Vector Quantization
MLP	Multi Layer Perceptron (Perceptron multi-couche)
NN	Neural Network
QV	Quantification Vectorielle
RBF	Radial Basis Functions
TDNN	Time Delay Neural Network

Remerciements

Je remercie Monsieur Y. Kodratoff, Directeur de Recherches au CNRS, qui m'a fait l'honneur de présider ce jury.

Monsieur le Professeur C. von der Malsburg et Monsieur le Professeur B. Victorri ont accepté d'évaluer mon travail de thèse, et m'ont adressé des critiques instructives. Je tiens à les en remercier vivement.

Je remercie également Monsieur M. Weinfeld, Directeur de Recherches au CNRS, qui a bien voulu marquer son intéret pour mon travail en acceptant d'être membre de ce jury.

Ce travail a été dirigé par Madame le Professeur F. Fogelman, qui m'a accueilli dans son équipe. Je tiens à lui exprimer mes remerciements les plus sincères pour avoir guidé mes travaux de ses conseils avisés, et pour m'avoir consacré une part importante de son temps.

L'ambiance amicale régnant au sein de l'équipe Réseaux Connexionnistes a joué un grand rôle dans ce travail. J'en remercie chaleureusement tous les membres, et particulièrement P. Gallinari qui dirige l'équipe avec dynamisme.

Les commentaires constructifs de Y. Bennani, M. de Bollivier, L. Bottou, X. Driancourt, B. Lamy, C. Mejia et de tous les thésards, que je ne peux citer tous ici, m'ont été infiniment utiles.

Je dois à J.C. Feauveau mon intérêt pour les analyses multirésolution par ondelettes, et je le remercie vivement du temps qu'il m'a consacré au début de cette thèse.

Enfin, j'adresse mes remerciement aux membres du LRI, laboratoire qui a accueilli mes travaux.

Résumé

Cette thèse est consacrée a l'étude d'approches connexionnistes pour l'analyse de scène, c'est à dire la localisation puis l'identification d'objets dans des images. Pour cette tâche, il est essentiel de disposer de systèmes de classification précis, rapides et dotés de capacité d'apprentissage. Les réseaux connexionnistes, systèmes adaptatifs reposant sur des traitements numériques massivement parallélisables, sont par conséquent d'excellents candidats.

La premiere partie de la these décrit les modèles connexionnistes les plus intéressants pour le traitement et la classification d'images. Nous indiquons les liens étroits unissants ces modèles aux approches statistiques usuelles.

La construction d'une architecture connexionniste à partir de modules algorithmiques simples facilite le traitement de tâches complexes. La décomposition en modules distincts permet en effet d'introduire des connaissances a priori sur la solution, de mieux contrôler l'apprentissage, et débouche sur des systèmes plus efficaces. Nous proposons plusieurs nouvelles architectures multi-modulaires pour la classification.

Dans une deuxième partie, nous étudions le problème de la reconnaissance et de la segmentation de chiffres manuscrits. Des études experimentales detaillées montrent l'apport des systèmes modulaires, qui permettent une très nette acceleration des calculs à performances comparables.

Ces résultats nous suggèrent une deuxième application, la localisation et l'identification de visages dans des images de scènes naturelles, pour laquelle peu de solutions performantes ont ete proposées. Nous utilisons une analyse multiresolution et un classifieur constitué de deux reseaux en cascade. Les visages détectés sont identifiés par un troisième réseau, qui permet de distinguer les personnes connues des visages inconnus. Les performances obtenues autorisent des applications réelles.

Mots Clés

Réseaux connexionnistes, Architectures modulaires, Reconnaissance des formes, Classification, Segmentation d'image, Reconnaissance d'écriture manuscrite, Identification de visages, Communication homme-machine.

Abstract

This thesis deals with connectionist approachs to scene analysis, ie localisation and identification of objects in images. For this task, we need accurate classification systems. Connectionist networks, which are adaptative systems based upon parallel computations are a natural choice.

The first part of the thesis describe the most interesting connectionist models for image processing and pattern classification. We focus on links with standard statistical methods.

For complex tasks, it is easier to build large architectures from simple algorithmic modules. This task decomposition allows to use a priori knowledge about the solution. We propose several new multi-modular architectures for classification tasks.

In a second part, we study the optical hand-written digits recognition problem. Experimental comparisons show the benefit of using multi-modular systems, especially for segmentation tasks.

This results suggested us to study another application, localisation and identification of human faces in natural scenes. For this problem, very few solutions have been proposed. We use a multiresolution analysis and a classifier built from two neural networks. Detected faces are then identified by a third network, which also distinguish known faces from unknown ones. The performances obtained are suitable for a real world application.

Keywords

Connectionists networks, Modular architectures, Pattern Recognition, Classification, Image segmentation, Handwritten digits recognition, Face identification, Man-Machine Communication.

Chapitre 1

Introduction

Les modèles connexionnistes offrent une panoplie de techniques pour la classification, la détection d'évènements et le traitement de signal. Leur application à des problèmes de reconnaissance de formes a récemment suscité de nombreux travaux, qui ont parfois débouché sur des systèmes extrèmement performants, notamment en traitement de la parole ou en reconnaissance d'écriture manuscrite.

Ce travail porte sur l'utilisation des techniques connexionnistes pour l'analyse de scène. Par analyse de scène, nous entendons ici les tâches de détection (ou segmentation) et identification d'objets dans des images. Nous ignorons donc les aspects de plus haut niveau, comme l'interprétation du contenu sémantique des images.

Si l'on ne compte plus les applications performantes des réseaux connexionnistes à des tâches de classification, leurs utilisations en analyse d'images sont encore assez rares. Nous avons tenté d'explorer les possibilités de ces techniques sur des problèmes difficiles, comme la segmentation d'écriture manuscrite ou la localisation de visages. Pour résoudre de tels problèmes, il est nécessaire d'intégrer la reconnaissance de l'objet au processus de segmentation.

1.1 Modèles connexionnistes et classification

Le connexionnisme recouvre une large classe de modèles et d'algorithmes, qui ont en commun d'utiliser des représentations *distribuées* de l'information et de reposer sur des traitements numériques *parallélisables*. Ces modèles sont nés de la confrontation d'idées venues de différentes disciplines : biologie, cybernétique, informatique, physique, statistique etc.

Ces dernières années, les liens unissant les réseaux connexionnistes et les

approches statistiques ont fait l'objet d'une attention particulière. De nombreux résultats ont pu être démontrés, établissant des équivalences entre certains modèles connexionnistes simples et des techniques statistiques bien connues : analyse discriminante ou analyse en composantes principales par exemple. D'autre part, les modèles connexionnistes sont des approximateurs universels de fonctions : pour une fonction donnée, il existe toujours un modèle capable de l'approcher à la précision voulue [103]. Ce type de résultat garanti qu'il existe une architecture permettant de résoudre le problème posé.

L'apprentissage d'un système connexionniste consiste à optimiser les valeurs des paramètres du réseau (poids), à l'aide d'un ensemble d'exemples du problème à résoudre. Lorsque l'on utilise un réseau pour une tâche de classification avec un critère du type «moindres carrés», on modèlise les surfaces de séparation entre classes. Il s'agit d'un apprentissage discriminant, qui prend en compte les structures inter et intra-classes des données pour diminuer l'erreur de classification.

Les méthodes d'optimisation par descente stochastique de gradient sont très bien adaptées aux réseaux connexionnistes. La mise à jour des poids ne nécessite qu'un exemple à la fois, ce qui permet l'emploi d'ensembles d'apprentissage de très grandes tailles. Des résultats théoriques récents prouvent la convergence de ce type d'algorithmes sous certaines conditions réunies en pratique [18].

En résumé, les avantages principaux des techniques connexionnistes sont les suivants :

— Puissance d'approximation ;
— Algorithmes d'apprentissage permettant de traiter des volumes importants de données ;
— Robutesse : comme nous le verrons, les systèmes connexionnistes sont souvent très résistants aux bruits et aux déformations des données ;
— Flexibilité : lors de la conception d'une architecture connexionniste, il est aisé d'introduire des connaissances a priori sur le problème à résoudre. Par exemple, nous verrons qu'il est possible de jouer sur les schémas de connexions (locales, masques) pour obtenir un traitement invariant à certaines transformations des entrées. Il est souvent délicat d'utiliser ce type d'information dans les techniques de classification traditionnelles, à moins de concevoir un module séparé de pré-traitement.
— Efficacité et parallèlisation : les temps d'apprentissage des réseaux connexionnistes sont souvent importants. Par contre, une fois les poids figés, l'utilisation du classifieur obtenu est très aisé et repose sur des traitements simples (produits de matrices, convolutions) facilement parallèlisables lors d'une implémentation sur circuit dédié (e.g. VLSI).

1.2 Approches multi-modulaires

Lorsque l'on aborde un nouveau problème, il est fréquent de disposer d'idées a priori sur la décomposition du problème en tâches distinctes. Pour certaines de ces tâches, on dispose parfois déja d'algorithmes satisfaisants. Un avantage important des approches connexionnistes est qu'elles se prètent bien à la conception de systèmes modulaires ou hybrides. Les systèmes modulaires, combinant plusieurs algorithmes (connexionnistes ou non) de natures différentes ont fait l'objet de nombreux travaux récents [22, 108, 11, 4].

Nous nous intéressons dans ce travail à l'utilisation d'approches connexionnistes multi-modulaires dans le but d'obtenir des systèmes de taille réduite, donc très efficaces, offrant des performances comparables ou supérieures aux systèmes existants. Les systèmes que nous proposons sont basés sur des modules simples (réseaux MLP, LVQ ou RBF) et assez largement utilisés. Nous montrons comment l'utilisation de techniques d'optimisation globales permet de construire des architectures multi-modulaires performantes.

1.3 Analyse de scène : segmentation et reconnaissance

La reconnaissance d'objets dans une image peut se décomposer en deux étapes : localisation de l'objet, puis identification de l'objet isolé. Cette approche n'est envisageable que si des critères simples permettent la localisation ; c'est par exemple le cas de la reconnaissance de caractères imprimés s'ils sont isolés les uns des autres, ce qui est rarement le cas en pratique. Dans des situations plus complexes, on doit intégrer l'identification au processus de localisation. Ainsi, il est impossible de séparer les différentes lettres d'un mot manuscrit avant d'avoir reconnu ce mot.

Les méthodes connexionnistes sont de bonnes candidates pour la construction de systèmes intégrant segmentation et reconnaissance. En effet, ces techniques se prêtent bien à la construction de systèmes modulaires, mèlant extraction de caractéristiques bas niveau de l'image (contours etc) et classification de ces caractéristiques. Dans ces systèmes, les modules sont optimisés simultanément, ce qui permet l'obtention d'une solution optimale garantissant l'adéquation des différents traitements les uns aux autres.

1.4 Applications : chiffres manuscrits et visages humains

Tout au long de nos travaux, nous avons pris le parti de chercher des techniques applicables à des problèmes réels de grande taille. Nous nous sommes d'abord intéressés au problème la reconnaissance de chiffres manuscrits. Il s'agit d'une application très actuelle : la demande industrielle de systèmes de reconnaissance d'écriture est aujourd'hui très forte (traitement automatique de documents, ordinateurs sans claviers, ...) et les solutions disponibles sont souvent insuffisantes. De notre point de vue, un grand intérêt de cette application est qu'elle a fait l'objet d'un très grand nombre de publications récentes, proposant ou comparant différentes approches. De plus, de nombreuses bases de données de tailles importantes sont accessibles.

La reconnaissance d'écriture manuscrite pose les problèmes fondamentaux de l'analyse de scène : elle requiert une interaction étroite entre l'interprétation sémantique (haut niveau) et l'extraction de caractéristiques (bas niveau) de l'image. La compréhension d'un mot dans une phrase fait ainsi souvent appel à une interprétation du sens de la phrase. Nous nous sommes contenté dans ce travail d'étudié l'intégration des processus de reconnaissance de chiffres et de segmentation.

La seconde application de nos travaux, plus originale, concerne la localisation et l'identification de visages humain dans des scènes. Au début de nos travaux, cette application n'avait pratiquement pas été étudiée, et le problème était très ouvert. Nous avons utilisé certaines approches développées sur les chiffres manuscrits et montré comment utiliser des architectures connexionnistes pour obtenir un système de localisation multi-échelle performant.

1.5 Plan de la thèse

Cette thèse est divisée en huit chapitres principaux.

Au cours des chapitres 2 et 3, nous passons en revue les outils statistiques et les modèles connexionnistes utilisés au cours de ce travail. Nous nous efforçons d'insister sur les liens entre les différentes approches, un grand nombre de modèles connexionnistes étant finalement très proches de techniques «classiques» beaucoup mieux connues.

Le chapitre 4 passe en revue quelques approches possibles en segmentation d'image, et détaille certains algorithmes d'analyse multirésolution d'images par ondelettes, sur lesquels nous avons travaillé au début de cette thèse.

Les chapitres 5 et 6 abordent le problème de la reconnaissance de chiffres manuscrits. Le chapitre 5 est une revue de la bibliographie récente dans le domaine, et le chapitre 6 décrit les systèmes que nous avons proposés et discute de leurs performances.

Nous abordons avec le chapitre 7 l'application principale de nos travaux, le traitement d'images de visages. Ce chapitre passe en revue la littérature dans ce domaine, décrit le problème posé et discute des performances des différents systèmes que nous avons mis au point.

Le chapitre 8 est consacré à la mise au point d'un module de localisation de visages dans des scènes naturelles. C'est l'occasion de discuter de différentes approches simples, puis de proposer une architecture en plusieurs étages offrant d'excellentes performances en segmentation.

Enfin, nous discutons dans le chapitre 9 les performances d'un système complet intégrant la localisation et l'identification de visages. Nous verrons que ce système offre des performances utilisables dans le monde réel.

Chapitre 2

Méthodes statistiques

2.1 Introduction

Nous décrivons dans ce chapitre différents outils statistiques de classification, dont la connaissance est indispensable à la compréhension des techniques connexionnistes développées dans la suite de cette thèse.

Le chapitre débute par une brève introduction à la théorie statistique de la décision, qui formalise les problèmes de classification.

Nous présentons ensuite l'approche paramétrique et la règle de décision de Bayes, et indiquons comment y traiter le cas du rejet. Nous évoquerons au passage les méthodes du maximum de vraisemblance et d'estimation bayesienne.

Nous abordons ensuite les méthodes non paramétriques standards, et présentons les classifieurs «plus proches voisins» (kNN) et de Parzen.

La section 2.6 évoque les techniques de classification basées sur le calcul direct des frontières entre classes et détaille les fonctions linéaires discriminantes. Nous présentons alors le modèle du Perceptron, qui a joué en grand rôle dans l'élaboration des modèles connexionnistes.

La section 2.7 aborde diverses techniques de quantification vectorielle (QV). De nombreux algorithmes d'apprentissage connexionnistes utilisent une quantification vectorielle (en particulier pour l'initialisation des paramètres). Nous détaillons en particulier l'algorithme des k-moyennes (k-means ou KMS).

Enfin, nous présentons la technique d'analyse en composantes principales dans la section 2.8. Cette technique d'analyse de données fournit un outil simple et puissant, nécessaire à la compréhension de plusieurs algorithmes connexionnistes présentés dans la suite de la thèse.

2.2 Théorie de la décision

Cette partie donne quelques définitions permettant d'introduire les bases théoriques de la discrimination et de la classification.

On considère des formes (patterns) x dans un espace de représentation de dimension d. On désire classer x dans l'une des M classes connues $\omega_1, \omega_2, \cdots, \omega_M$.

Une *règle de décision* est une fonction qui partitionne l'espace de représentation en régions $\Omega_i, i = 1, \cdots, M$ ou‘ M est le nombre de classes (les régions Ω_i ne sont pas nécessairement connexes). Une forme est classée dans la classe ω_k si sa représentation vectorielle x appartient à la région Ω_k. Les frontières entre les régions sont appelées *surfaces de décisions*.

On suppose en général que l'on connait les probabilités $P(\omega_i)$ $i = 1, \cdots, M$ qu'une forme appartienne à Ω_i. Ces probabilités sont indépendantes de la forme x, et connues avant son observation ; on les nomme donc *probabilités a-priori*. On a evidemment :

$$\sum_{i=1}^{M} P(\omega_i) = 1 \tag{2.1}$$

A ce stade, si l'on ne dispose d'aucune autre information, la meilleure règle de décision est de classer la forme dans la classe ω_k telle que :

$$P(\omega_k) > P(\omega_i),\ i = 1, \cdots, M;\ i \neq k \tag{2.2}$$

Cette règle triviale minimise la probabilité de commettre une erreur, et partitionne l'espace en une seule région.

Une règle de décision d traduit la décision d'affecter x à la classe ω_i : cette règle associe à la forme x un nombre i, indice de la classe :

$$d(x) = i \qquad \text{décision d'affecter } x \text{ à } \omega_i.$$

2.3 Discrimination paramétrique avec rejet

Dans le cadre paramétrique, les lois de probabilités suivantes sont supposées complètement connues :

— Loi de x dans la classe ω_i : $f(x|\omega_i)$

— Probabilité a-priori de chaque classe : $P(\omega_i)$.

Afin de formaliser le cas du rejet, ou‘ il n'y a pas d'affectation à une classe proprement dite, il est pratique [40, 41] d'introduire une classe supplémentaire, dite classe de rejet et notée ω_0. On affectera à cette classe les formes ambiguës ou trop lointaines de celles connues. La décision de rejeter x se note $d(x) = 0$.

2.3.1 Règle de Bayes (coût minimum)

Avant de rechercher la règle de décision optimale, il convient de définir un *coût* associé à chaque décision. Pour cela, on introduit une matrice de coût $C(\omega_i|\omega_j)$. Cette matrice, arbitraire (i.e. c'est une donnée du problème), donne le coût de classer une forme de la classe ω_j dans la classe ω_i, $i = 0, \cdots, M$ et $j = 1, \cdots, M$.

On va choisir la règle de décision $d(x)$ telle que le coût moyen sur toutes les formes soit le plus faible possible.

On associe le coût $C(\omega_{d(x)}|\omega_j)$ à la décision $d(x)$ prise pour une forme de la classe ω_j. Pour une forme x, l'espérance mathématique du coût associé à la décision $d(x)$ s'écrit :

$$C(x) = \sum_{j=1}^{M} C(\omega_{d(x)}|\omega_j) P(\omega_j|x) \tag{2.3}$$

Le coût (ou encore risque) moyen est donné par l'espérance mathématique prise sur toutes les formes x :

$$C = \int_{\Omega} C(x) f(x)\, dx \tag{2.4}$$

ou‘ Ω est l'espace de définition de x et $f(x)$ la probabilité non conditionnelle de x, parfois appelée densité de mélange et donnée par :

$$f(x) = \sum_{i=1}^{M} f(x|\omega_i) P(\omega_i) \tag{2.5}$$

On cherche la règle de décision qui minimise C. Comme $f(x)$ est une fonction positive, il suffit de minimiser $C(x)$.

La règle optimale d^* minimise $\sum_{j=1}^{M} C(\omega_{d^*(x)}|\omega_j) P(\omega_j|x)$. Le coût minimum s'écrit :

$$C^*(x) = \min_{i=0,\cdots,M} \sum_{j=1}^{M} C(\omega_i|\omega_j) P(\omega_j|x) \tag{2.6}$$

La règle d^* est appelée *règle de Bayes* et C^* le risque de Bayes.

Pour obtenir d^*, il faut exprimer $P(\omega_j|x)$, ce que fait la formule suivante (également de Bayes) :

$$P(\omega_j|x) = \frac{f(x|\omega_j) P(\omega_j)}{\sum_{i=1}^{M} f(x|\omega_i) P(\omega_i)} \tag{2.7}$$

La formule de Bayes permet d'exprimer la probabilité a posteriori en fonction des probabilités a priori. On retrouve la densité de mélange $f(x)$ au dénominateur.

Dans le cas le plus simple, ou' toutes les erreurs ont le même coût et ou' l'on ne rejette jamais, on voit qu'il suffit d'affecter x à la classe dont la probabilité a posteriori (donnée par (2.7)) est la plus grande.

2.3.2 Régle des coûts $[0, 1]$

Afin d'expliciter la règle de Bayes dans un cas usuel simple, on va fixer des valeurs particulières aux différents coûts. La règle dite «des coûts $[0, 1]$» est très simple : elle accorde la même importance à tous les types d'erreurs, et affecte un coût nul aux décisions correctes :

$$\begin{cases} C(\omega_i, \omega_i) = 0, & i = \ 1, \ldots, M \\ C(\omega_i, \omega_j) = 1, & i, j = \ 1, \ldots, M, i \neq j \end{cases} \tag{2.8}$$

Enfin, on fixe une valeur constante C_r au coût de rejet :

$$C(\omega_0, \omega_j) = C_r \tag{2.9}$$

L'espérance du coût devient alors :
— si $d(x) = 0$ (x rejeté)

$$C_0(x) = \sum_{j=1}^{M} C_r P(\omega_j|x) = C_r \tag{2.10}$$

— si $d(x) = i$ (on accepte de classer)

$$C_i(x) = \sum_{j=1, j\neq i}^{M} P(\omega_j|x) = 1 - P(\omega_i|x) \tag{2.11}$$

Cette expression est appelée probabilité conditionnelle d'erreur de classification ; on la notera $e_i(x) = 1 - P(\omega_i|x)$. La probabilité conditionnelle d'erreur de classification minimum pour x est :

$$e^*(x) = \min_{i=1,M} e_i(x) = 1 - \max P(\omega_i|x) \tag{2.12}$$

On a la règle de décision suivante :

$$d(x) = \begin{cases} i \ (x \text{ classé dans } \omega_i) & \text{si } e^*(x) = e_i(x) \leq C_r \\ 0 \ (x \text{ rejeté}) & \text{si } C_r < e^*(x) \end{cases} \tag{2.13}$$

La décision de rejeter est prise si la probabilité conditionnelle d'erreur est supérieure au coût de rejet.

On peut exprimer la règle de décision en termes de probabilité a posteriori :

$$d(x) = \begin{cases} i & \text{si } P(\omega_i|x) \geq 1 - C_r \\ 0 & \text{si } \max_{j=1,M} P(\omega_j|x) < 1 - C_r \end{cases} \tag{2.14}$$

Comme la somme des $P(\omega_i|x)$ vaut 1, leur maximum est compris entre $1/M$ et 1. On a donc :

$$0 \leq 1 - \max_{i=1,M} P(\omega_i|x) \leq 1 - 1/M \tag{2.15}$$

Pour qu'il y ait possibilité de rejet, la relation suivante doit être vérifiée :

$$0 \leq C_r \leq \frac{M-1}{M} \tag{2.16}$$

Relation rejet - erreur

Si l'on applique la règle de décision optimale (eq.(2.13)), on peut démontrer [41] la relation suivante entre la probabilité d'erreur de classification P_E et la probabilité de rejet P_R :

$$P_E(C_r) = -\int_0^{C_r} u \, dP_R(u) \tag{2.17}$$

P_R peut se calculer en sommant sur la zone de rejet :

$$P_R(C_r) = \int_{\omega_0} f(x) \, dx \tag{2.18}$$

2.3.3 Rejet de distance

Le rejet que nous venons d'introduire a pour but de réduire l'erreur de classification. On appelera ce type de rejet un *rejet d'ambiguïté* : selon ce critère, on rejette les formes proches des surfaces de décision, pour lesquelles plusieurs décisions pourraient être prises (voir figure 2.1).

Dans bien des cas, ce critère est insuffisant. En particulier, il ne permet pas de détecter une forme n'appartenant à aucune des classes connues. Dans la pratique, de telles formes (*outliers*), absentes par définition de l'ensemble d'apprentissage, risquent de contribuer significativement à l'erreur de classification. Par exemple, dans un système de reconnaissance de chiffres manuscrits composé d'un module de segmentation localisant et découpant les chiffres suivi d'un module de reconnaissance, il sera nécessaire que ce dernier

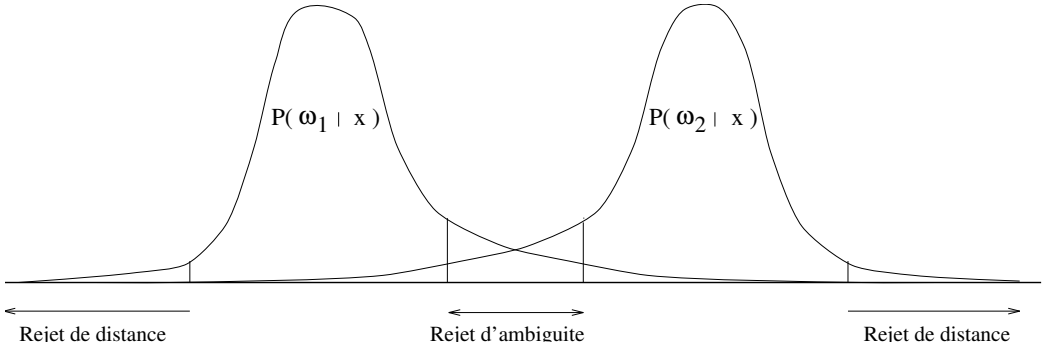


FIGURE 2.1 – Zones de rejet d'ambiguïté et de distance.

rejette les formes sans signification provenant d'erreurs de segmentation. Il convient donc d'ajouter au rejet d'ambiguïté définit plus haut un deuxième cas, que l'on appellera *rejet de distance*. On refusera de classer les formes trop éloignées de celles rencontrées durant l'apprentissage.

Le rejet de distance doit naturellement être fondé sur une notion de distance aux classes. En terme de densité de probabilités, on peut l'exprimer sous la forme suivante : x rejeté en distance si

$$f(x) = \sum_{i=1}^{M} f(x|\omega_i)P(\omega_i) \; < \; C_d \tag{2.19}$$

Ou' l'on introduit un deuxième seuil de rejet C_d.

2.4 Estimation Paramétrique

Dans cette section, nous rappelons brièvement les bases statistiques des procédures d'estimation paramétrique.

Dans les approches paramétriques, on suppose connue la forme des lois de probabilités $p(x|\omega_j)$. Ces lois sont entièrement déterminées par un jeu de paramètres inconnus Θ. On écrit alors $p(x \,|\, \omega_i, \Theta_i)$. Il s'agit de déterminer la valeur optimale de Θ.

2.4.1 Méthode du Maximum de Vraisemblance

La méthode du maximum de vraisemblance consiste à maximiser la probabilité d'observation de l'échantillon $\mathcal{X}$ dont on dispose pour l'apprentis-

sage [90]. Cette probabilité s'exprime comme

$$p(\mathcal{X}|\Theta) = \prod_{k=1}^{n} p(x_k|\Theta) \tag{2.20}$$

ou' n est le nombre d'exemples utilisés. $p(\mathcal{X}|\Theta)$ est appelé la *vraisemblance* de Θ compte tenu de l'échantillon $\mathcal{X}$. On introduit généralement le logarithme de la vraisemblance

$$l(\Theta) = \log p(\mathcal{X}|\Theta) \tag{2.21}$$

ce qui permet de caractériser le maximum de vraisemblance par le jeu d'égalités :

$$\nabla_\Theta l = \sum_{k=1}^{n} \log p(x_k|\Theta) = 0 \tag{2.22}$$

On peut alors calculer la valeur optimale des paramètres, notée $\hat{\Theta}$.

2.4.2 Approche Bayesienne

La méthode d'estimation paramétrique bayesienne introduit une distribution de probabilité sur les paramètres Θ (*priors*). Cette distribution permet d'introduire des connaissances a priori dans la solution $p(x|\Theta)$ cherchée.

On suppose que le jeu de paramètres Θ suit la loi $p(\Theta)$. Après observation de l'échantillon $\mathcal{X}$, on obtient la distribution $p(\Theta|\mathcal{X})$, et on cherche à calculer $p(x|\mathcal{X})$. Pour cela, on écrit

$$p(x|\mathcal{X}) = \int p(x, \Theta \,|\, \mathcal{X}) d\Theta \tag{2.23}$$

d'ou'

$$p(x|\mathcal{X}) = \int p(x|\Theta)\, p(\Theta|\mathcal{X}) d\Theta \tag{2.24}$$

Si la distribution $p(\Theta)$ est piquée en $\hat{\Theta}$, l'intégrale s'évalue simplement et l'on retrouve la solution du maximum de vraisemblance :

$$p(x|\mathcal{X}) \approx p(x|\hat{\Theta}) \tag{2.25}$$

2.5 Discrimination non paramétrique

Dans la section précédente, nous avons supposé connues les lois de probabilité de chaque classe. En reconnaissance des formes, on dispose rarement d'une telle information ; les distributions de probabilité $f(x|\omega_i)$ sont le plus

souvent inconnues et de forme compliquée : il est par exemple irréaliste de les représenter par des gaussiennes.

Les techniques de discrimination non paramétriques visent à *estimer* les lois de probabilité $f(x|\omega_i)$ à partir des données disponibles (exemples d'apprentissage).

2.5.1 Estimation à partir d'exemples

Nous nous intéressons ici à l'estimation d'une loi de probabilité f à partir d'exemples (appartenant tous à la même classe).

On dispose de n vecteurs indépendants $x_1, x_2, \cdots, x_n$ de $\mathbb{R}^d$ constituant l'ensemble d'apprentissage. On cherche à estimer f au point x_0.

Soit $\mathbb{A}$ une région de $\mathbb{R}^d$ et p la probabilité de trouver x dans $\mathbb{A}$

$$p = \int_{\mathbb{A}} f(x)\, dx \tag{2.26}$$

Notons $k_{\mathbb{A}}$ le nombre d'échantillons (exemples) trouvés dans $\mathbb{A}$. C'est une variable binomiale de p et n :

$$P(k_{\mathbb{A}} = k) = C_n^k\, p^k\, (1-p)^{n-k} \tag{2.27}$$

et

$$E(k_{\mathbb{A}}) = n\, p \tag{2.28}$$

Un estimateur du maximum de vraisemblance de p est

$$\hat{p} = \frac{k_{\mathbb{A}}}{n} \tag{2.29}$$

On va supposer que la loi f est continue autour de x_0, et construire une suite de régions emboitées $\mathbb{A}_n$ autour de ce point. On note alors p_n la probabilité de trouver x dans $\mathbb{A}_n$ et V_n le volume de $\mathbb{A}_n$:

$$\frac{\text{Proba}(x \in \mathbb{A}_n)}{\text{Volume}(\mathbb{A}_n)} = \frac{p_n}{V_n} \tag{2.30}$$

On a

$$\lim_{n\to\infty} V_n = 0 \quad \text{et} \quad \lim_{n\to\infty} \frac{p_n}{V_n} = f(x_0) \tag{2.31}$$

Ce qui donne l'estimateur de $f(x_0)$ cherché :

$$\hat{f}(x_0) = \frac{\hat{p}_n}{V_n} = \frac{k_n}{n\, V_n} \tag{2.32}$$

ou‘ k_n est le nombre d'échantillons dans $\mathbb{A}_n$.

La convergence est assurée sous les trois conditions suivantes :

$$\begin{array}{l} \lim_{n\to\infty} V_n = 0 \\ \lim_{n\to\infty} k_n = \infty \\ \lim_{n\to\infty} k_n/n = 0 \end{array} \tag{2.33}$$

En pratique, on dispose d'un nombre limité d'exemples et l'on doit se fixer un volume V. Ce volume doit être assez grand pour contenir suffisament d'exemples, mais rester petit pour obtenir la meilleure précision possible.

Deux méthodes standards sont utilisées pour régler ce problème [59] : la première, qui donne l'*estimateur de Parzen* consiste à fixer les volumes V_n ; la seconde fixe le nombre d'échantillons utilisés k_n, c'est la méthode des k *plus proches voisins* (kPPV ou kNN [1])

2.5.2 Estimateur de Parzen

L'estimateur de Parzen est construit en fixant le volume de la région $\mathbb{A}_n$. Il suffit d'utiliser un paramètre scalaire, nommé *largeur* :

$$V_n = h_n^d \tag{2.34}$$

L'estimateur de $f(x_0)$ s'écrit alors, d'après (2.32),

$$\hat{f}_n(x_0) = \frac{1}{n\, h_n^d}\, k_n \tag{2.35}$$

Le nombre d'échantillons k_n s'exprime en introduisant la fonction indicatrice de la région $\mathbb{A}_n$, ou plus généralement une fonction noyau ϕ quelconque (i.e. localisée autour de 0, positive et sommant à un). On obtient alors un estimateur de la forme :

$$\hat{f}_n(x_0) = \frac{1}{n\, h_n^d} \sum_{i=1}^{n} \phi(\frac{x_0 - x_i}{h_n}) \tag{2.36}$$

Le choix du noyau ϕ est libre. On utilisera le plus souvent une gaussienne.

Le choix de la largeur h_n est crucial. Ce paramètre contrôle la «*localité*» de l'estimateur. S'il est très grand, l'approximation sera très lisse, et l'on perdra souvent des détails. S'il est trop petit, l'estimation sera très sensible aux bruits. Fukunaga [72] suggère d'utiliser $h_n = n^{\alpha/d}$, avec $0 < \alpha < 1$.

1. kNN pour *k nearest neighbors*.

2.5.3 k plus proches voisins

Dans l'approche «plus proches voisins», on fixe le nombre d'échantillons k_n et l'on cherche la région $\mathbb{A}_n$ correspondante. On centre la région en x_0, on en fait croître le volume jusqu'a contenir les k_n plus proches échantillons voisins de x_0. Le volume obtenu permet une estimation directe de la densité de probabilité grâce à l'équation (2.32).

Appliquons cette approche aux problèmes de classification.

Règle du plus proche voisin

La règle du plus proche voisin consiste simplement à affecter la forme x à la classe de l'exemple (échantillon) le plus proche. On montre aisément [58] que lorsque le nombre d'exemples tend vers l'infini, le plus proche voisin de x tend vers x. On peut aussi montrer que le risque d'erreur de cette méthode est borné par deux fois le risque de Bayes [52, 53].

Dans cette approche, il n'est pas possible d'introduire naturellement la notion de rejet.

Règle des plus proches voisins (avec rejet)

En général, on gagne à considérer plusieurs voisins. On procède alors à un *vote*, de façon à selectionner la classe la plus représentée parmi les k voisins de x.

En cas d'égalité (plusieurs classes également représentées), différentes options sont possibles. Une des plus simples, que nous avons retenu pour nos simulations, est de choisir la classe de l'exemple le plus proche.

Rejet d'ambiguïté La manière la plus directe pour rejeter certaines formes avec les règles des k plus proches voisins est d'utiliser une majorité qualifiée [100]. On affecte la forme à la classe majoritaire si, parmi les k plus proches voisins, au moins k' appartiennent à cette classe. Le seuil k' peut dépendre de la classe [52].

Rejet de distance Le rejet basé sur une majorité qualifiée permet de rejeter les formes ambiguës. Il ne prend pas en compte la distance absolue aux classes, c'est à dire de la distance de x à ses voisins. Ce point est rarement évoqué dans la littérature, ou‘ tous les résultats sont obtenus asymptotiquement, dans le cas ou‘ les échantillons sont aussi proches les uns des autres qu'on le désire. En pratique, on a vu qu'il est souvent important de détecter les formes nouvelles, non représentées dans l'ensemble d'apprentissage.

Dubuisson [58] propose l'introduction d'un second critère, permettant de rejeter la forme x si la distance moyenne de ses k plus proches voisins est supérieure à un seuil T :

$$\sum_{i=1}^{k} d(x, x_{\nu(i)}) \; > \; k\,T \tag{2.37}$$

ou' $\nu(i)$ donne l'indice du i-ème voisin.

Toutefois, il est souvent irréaliste de mettre en pratique ces critères de rejets avec la règle des k plus proches voisins. En effet, la valeur optimale pour k est souvent très basse (par exemple, dans les simulations sur les chiffres manuscrits (chap. 6, section 6.4.1), on verra que $k = 3$ est optimal). Dans ces conditions, le concept de majorité qualifié perd son intérêt.

Pour des classes obéissant à des lois normales, on peut calculer le nombre optimal k^* de voisins à utiliser lorsque l'on dispose de N exemples dans un espace de dimension n [72] :

$$k^* = \left[\frac{(n+2)^2\,(n-2)^{2+n/2}}{\Gamma^{4/n}(\frac{n+2}{2})\;n^{1+n/2}\,(n^2-6n+16)}\right]^{\frac{n}{n+4}} \times N^{\frac{4}{n+4}} \tag{2.38}$$

ou' Γ est la fonction gamma habituelle (telle que $\Gamma(x+1) = x\Gamma(x)$).

n	4	8	16	32	64	128
k^*	$0.75N^{1/2}$	$0.94N^{1/3}$	$0.62N^{1/5}$	$0.34N^{1/9}$	$0.17N^{1/17}$	$0.09N^{1/33}$
N for $k^* = 5$	4.4 10	$1.5\ 10^2$	$3.4\ 10^4$	$3.2\ 10^{10}$	$9.2\ 10^{24}$	$3.8\ 10^{57}$

TABLE 2.1 – Cette table (reproduite d'après [72]) indique le nombre optimal k^* de voisins pour différentes dimensions n de l'espace. La troisième ligne donne le nombre d'exemples nécessaires pour obtenir $k^* = 5$.

La table 2.1 indique quelques valeurs de k^*. Dès que la dimension de l'espace dépasse 16, le nombre d'exemples nécessaires excède les possibilités habituelles. On est donc le plus souvent contraint d'utiliser un très faible nombre de voisins.

2.6 Fonctions linéaires discriminantes

Les méthodes non paramétriques présentées ci-dessus visent à estimer la densité de probabilité des différentes classes pour construire un classifieur.

Une autre approche consiste à calculer directement les frontières entre classes. Cette approche est basée sur le calcul de *fonctions discriminantes*. On ne parle plus ici des distributions de probabilité, mais on contraint la forme des fonctions discriminantes caractérisant les classes.

Du fait de leur simplicité, de nombreux travaux se sont intéressés aux fonctions discriminantes linéaires [59].

2.6.1 Cas de deux classes

Une fonction discriminante linéaire des composantes de x s'écrit :

$$g(x) = w^t x + b \tag{2.39}$$

w est le «vecteur de poids», et b une valeur appelée *seuil*.

La classifieur prend la décision selon le signe de g :

$$d(x) = \begin{cases} 1 \ (x \text{ classé dans } \omega_1) & \text{si } g(x) > 0 \\ 2 & \text{si } g(x) < 0 \end{cases} \tag{2.40}$$

La frontière entre les deux classes est définie par $g(x) = 0$. Dans le cas linéaire, c'est un hyperplan. Le vecteur de poids w est perpendiculaire à ce plan. Notons que la distance d'un point x à l'hyperplan séparateur est donnée par :

$$r = \frac{g(x)}{\|w\|} \tag{2.41}$$

2.6.2 Extensions à plus de 2 classes

Il existe différentes façons d'étendre l'approche précédente à plus de deux classes [59]. La méthode la plus employée consiste à utiliser une fonction discriminante g_i par classe.

On affecte x à ω_i si $g_i(x) > g_j(x)$ pour tout $j \neq i$.

2.6.3 Critère du Perceptron

Le calcul des fonctions discriminantes à partir des exemples invoque les minimisation d'un critère. Le critère le plus simple imaginable est le nombre d'exemples mal classés (i.e. tombant du mauvais coté du plan séparateur). Ce critère est cependant difficile à utiliser car la fonction correspondante est très discontinue : il est impossible d'en calculer le gradient par rapport aux poids.

Le critère du perceptron [178, 179, 156] cherche à minimiser la distance des points mal classés au plan séparateur. Il prend la forme :

$$J(x, w) = -\sum_{x \in \Upsilon} wx + w_0 \tag{2.42}$$

ou‘ Υ est l'ensemble des exemples mal classés par g.

Ce critère est nul lorsque tous les points sont bien classés. Il ne prend pas en compte la position des autres exemples.

2.6.4 Optimisation du critère

La détermination des paramètres w de la fonction discriminante g est faite par otimisation du critère $J(w)$.

L'approche standard consiste à descendre le gradient de J :

$$w(t+1) = w(t) - \epsilon \nabla J(w) \tag{2.43}$$

Dans le cas du critère du perceptron, on a :

$$\nabla J(w) = -x \tag{2.44}$$

d'ou‘

$$w(t+1) = w(t) + \epsilon\, x \tag{2.45}$$

L'équation (2.45) est connue sous le nom de *règle du perceptron*. On montre aisément [59] que ce processus converge toujours si les exemples sont linéairement séparables. Si ce n'est pas le cas, les poids w vont osciller indéfiniment. En général, on fait lentement décroître le gain ϵ vers zero, de façon à forcer la convergence.

2.6.5 Critère de l'erreur quadratique

Les critères présentés plus haut ne prennent en compte que les exemples mal classés. Intuitivement, la frontière entre classes devrait dépendre de tous les points disponibles. D'autre part, il est souhaitable d'arriver à une solution (même sous-optimale) quelle que soit la forme des classes (linéairement séparables ou non).

Pour cela, on fixe une valeur «désirée» y_i à la fonction discrimante $g(x_i)$ en chaque point de l'ensemble d'apprentissage. Le problème se ramène alors à résoudre une équation linéaire (matricielle) de la forme

$$g(X) = Y \tag{2.46}$$

Cette équation n'a en général pas de solution, le nombre de contraintes excédant le nombre de paramètres libres. On utilise une procédure de moindres carrés pour minimiser

$$J_{\text{mse}}(w) = \|Y - g(X)\|^2 \tag{2.47}$$

La minimisation de J_{mse} peut s'effectuer à l'aide du calcul de la pseudo-inverse [164, 85, 173]. Pour les problèmes de taille importante, on préférera utiliser une descente de gradient stochastique [18]. On retrouve alors la règle d'adaption de Widrow-Hoff [219, 101] :

$$w(t+1) = w(t) + \epsilon \left(y_t - g(x_t)\right) g(x_t) \tag{2.48}$$

ou' (x_t, y_t) est le couple exemple-valeur désirée présenté à l'instant t de l'apprentissage. Cette règle caractérise des premiers modèles d'automate adaptatif, l'Adaline [219].

2.6.6 Problème de généralisation

L'apprentissage à l'aide du critère du Perceptron prend fin dès que toutes les classes sont séparées. N'importe quel plan séparateur fait l'affaire. Par conséquent, rien ne garantit que l'on obtient une solution optimale pour classer de nouveaux exemples de chaque classe. Par exemple, le plan trouvé peut passer très près d'une classe et très loin de l'autre (voir figure 2.2).

D'un autre coté, le critère des moindres carrés, s'il règle les problèmes de convergence dans le cas non linéairement séparable, ne garanti pas d'obtenir la frontière optimale pour la classification.

Une idée intéressante consiste à rechercher les solutions maximisant la *marge*, c'est à dire la distance minimale entre les exemples et la frontière trouvée [13].

2.6.7 Rejet

La notion de rejet est rarement introduite dans les méthodes basées sur les fonctions discriminantes.

En fait, le rejet des formes ambiguës ne pose pas de problèmes particuliers. Ces formes sont par définition proches des frontières entre classes. On doit rejeter x si :

$$|g(x)| < \theta \tag{2.49}$$

Cette équation donne une «bande» autour du plan séparateur, dans laquelle on refuse de classer x.

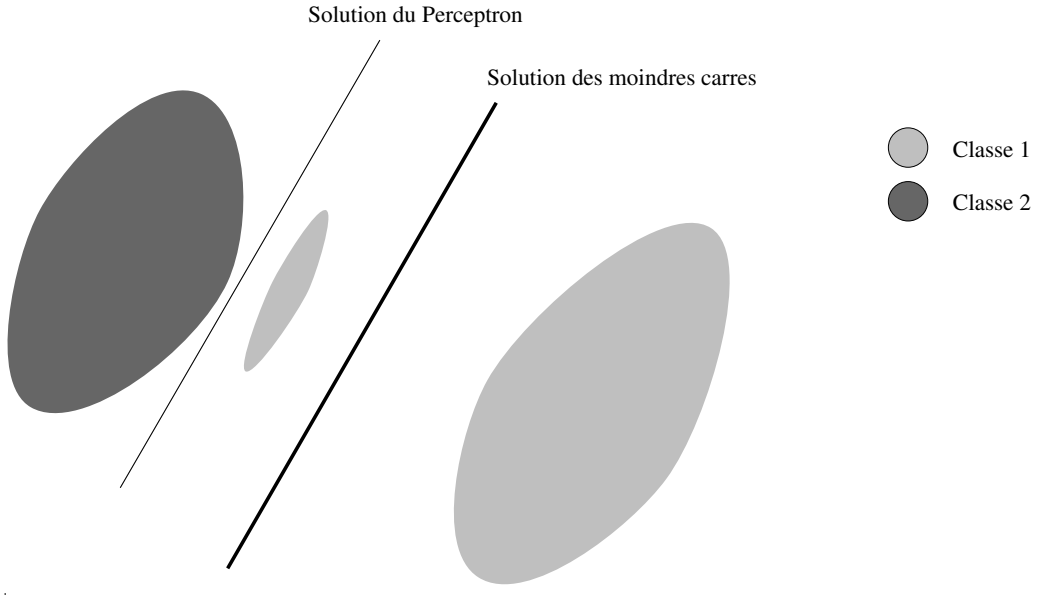


FIGURE 2.2 – Frontières linéaires entre classes obtenues suivant le critère des moindres carrés et celui du perceptron. Les classes 1 et 2 sont ici linéairement séparables. Le perceptron classe correctement tous les exemples, tandis que le critère des moindres carrés reflette la répartition générale des classes.

Le rejet «de distance» (des formes éloignées) est plus délicat. Le critère du perceptron ne considère que les formes mal classées. On ne mémorise par conséquent aucune information sur (par exemple) la distance moyenne des formes de chaque classe aux frontières. Il semble par conséquent que cette méthode soit incapable de rejeter seule les formes inconnues.

2.7 Quantification Vectorielle

Nous abordons dans cette section diverses techniques de quantification vectorielle (QV). Ces techniques sont souvent utilisées en classification pour extraire les éléments pertinents de l'ensemble d'apprentissage, par exemple. Ainsi, lorsque l'usage d'un classifieur k plus proches voisins n'est plus envisageable faute de puissance de calcul, on pourra envisager une technique de QV (ou *clustering*) pour sélectionner des prototypes pertinents. Ces techniques ont bien d'autres applications, en particulier en codage de données (transmissions, etc), sur lesquelles nous ne nous étendrons pas ici.

Le but de la quantification vectorielle est de représenter des données à l'aide d'un ensemble réduit de valeurs (scalaires ou vectorielles), que l'on

appelle souvent *codebook*, ou dictionnaire. Une donnée x sera alors codée par l'élément y du dictionnaire le plus proche de x.

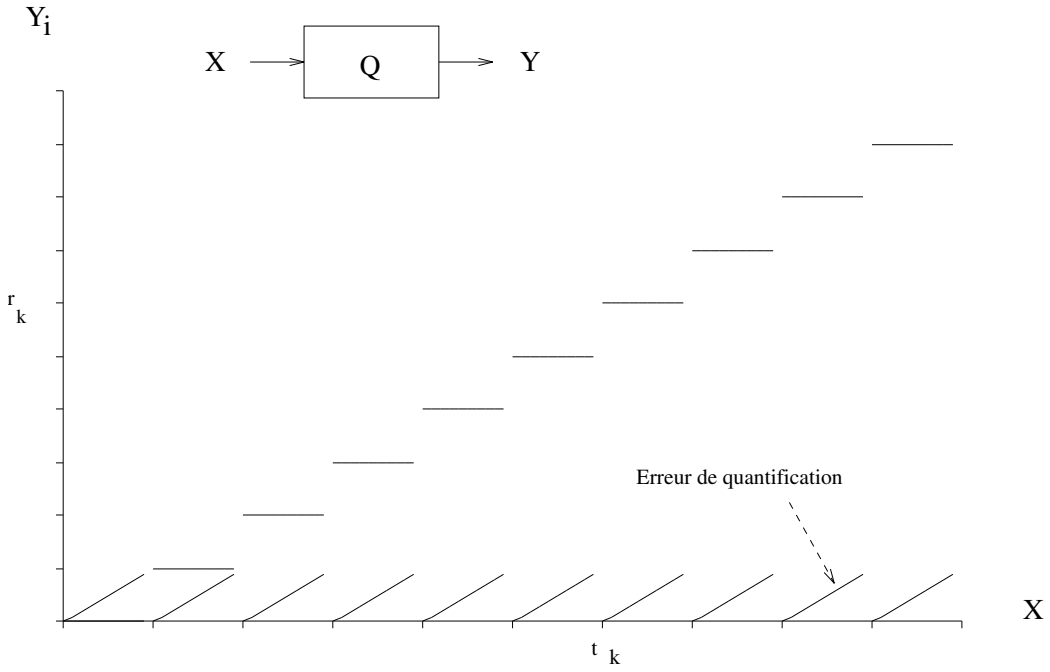


FIGURE 2.3 – Opération de quantification scalaire.

2.7.1 Cas uni-dimensionnel

Nous indiquons dans cette section comment construire un quantificateur scalaire optimal au sens des moindres carrés. Nous avons représenté sur la figure 2.3 l'opération de quantification scalaire.

La construction du quantificateur consiste à choisir le nombre L de niveaux de quantification, les niveaux de décisions t_k et les niveaux de reconstruction r_k [174, 109]. On cherche ces niveaux tels que l'erreur quadratique moyenne de reconstruction soit minimisée :

$$\mathcal{E} = E[(x-y)^2] = \int_{t_1}^{t_{L+1}} (x-y)^2 \, p(x) \, dx \tag{2.50}$$

ou‘ $p(x)$ est la densité de probabilité de x. On peut écrire :

$$\mathcal{E} = \sum_{i=1}^{L} \int_{t_i}^{t_{i+1}} (x-r_i)^2 \, p(x) \, dx \tag{2.51}$$

On obtient alors facilement, en annulant les dérivées partielles de $\mathcal{E}$ par rapport aux t_k et r_k, les conditions suivantes :

$$t_k = \frac{1}{2}\,(r_k + r_{k-1}) \tag{2.52}$$

$$r_k = \frac{\int_{t_k}^{t_{k+1}} x\,p(x)\,dx}{\int_{t_k}^{t_{k+1}} p(x)\,dx} = E[\,x \,|\, x \in [t_k, t_{k+1}]\,] \tag{2.53}$$

L'équation (2.52) est peu surprenante : le choix des niveaux de reconstruction suffit à déterminer le quantifieur. Les équations (2.52) et (2.53) forment un système d'équations non linéaires, que l'on peut résoudre itérativement (par la méthode de Newton par exemple) une fois les bornes extrèmes t_1 et t_{L+1} déterminées. Ces bornes fixent la *dynamique* du quantificateur.

2.7.2 Cas multi-dimensionnel

Méthode en ligne

Les méthodes «en ligne» ont l'avantage de ne pas nécessiter la présence simultanée de toutes les données pendant l'apprentissage.

Nous présentons ici une méthode en ligne très simple, qui est très facile à mettre en œuvre et donne des résultats qui ne sont pas si mauvais. On n'utilise qu'un paramètre, qui est une distance seuil D. Contrairement à la plupart des méthodes globales dicutées plus loins, la taille du dictionnaire n'est pas fixée à l'avance et résulte du choix de D et des données rencontrées lors de l'apprentissage. On procède comme suit :

1. Placer le premier exemple dans le dictionnaire ;
2. Pour chaque nouvel exemple, rechercher le représentant le plus proche dans le dictionnaire. Si la distance entre ce représentant et l'exemple excède le seuil D, ajouter l'exemple au dictionnaire.

Notons qu'un avantage de cette méthode est qu'il n'y a pas d'adaptation des vecteurs du dictionnaire. La mesure de similarité entre les formes peut donc être quelconque sans modification de l'algorithme.

Algorithmes des k-moyennes

L'algorithme des K-moyennes (dit *k-means*, KMS en abrégé), est une méthode de quantication vectorielle globale (nécessitant la présence de tous les exemples) proposée par MacQueen en 1967 [145], et très utilisée depuis en analyse de données.

On dispose de N exemples et on recherche K vecteurs w_k les représentant au mieux. Comme dans le cas uni-dimensionnel, cela revient à chercher une partition de l'espace en K régions. On désire minimiser l'erreur quadratique :

$$\mathcal{E}_W = \sum_{k=1}^{K} \int m_k(x) \, \|x - w_k\|^2 \, p(x) \, dx \tag{2.54}$$

ou' $m_k(x)$ vaut 1 si w_k est le plus proche voisin de x, et zero sinon. On peut aussi écrire :

$$\mathcal{E}_W = \int \min_k \|x - w_k\|^2 \; p(x) \, dx \tag{2.55}$$

Au cours de l'apprentissage, on déplace les centres w de façon à minimiser $\mathcal{E}$. Pour cela, après chaque présentation d'exemple, on déplace le centre le plus proche pour le placer au centre de gravité des exemples qu'il réprésente et du nouvel exemple. Il s'agit donc d'une descente de gradient stochastique, suivant la dynamique :

$$w_k(t+1) = w_k(t) \; + \; \eta(t)(x(t) - w_k(t)) \tag{2.56}$$

ou' $x(t)$ est l'exemple présenté à l'étape t de l'apprentissage, et k l'indice du centre le plus proche de $x(t)$. Les autres centres sont inchangés. η donne le gain de l'apprentissage.

Cet algorithme présente plusieurs inconvénients :

1. le choix des positions initiales des centres est très important. Si ce choix n'est pas approprié, l'apprentissage reste bloqué dans un minimum local de $\mathcal{E}$. Une solution proposée [161] consite à déplacer simultanément tous les centres (et non plus seulement le plus proche), avec des gains différents.
2. comme dans toutes les méthodes de descente du gradient de l'erreur, le choix gain η est important. Une trop faible valeur entraine des temps de convergence très longs, mais une valeur trop forte empèche la convergence. Darken et Moody [48] proposent d'utiliser un gain variant dans le temps selon la loi

   $$\eta(t) = \frac{\eta_0}{1 + t/\tau} \tag{2.57}$$

   Cette évolution permet de garder un gain proche de η_0 au début de l'apprentissage (pendant un durée τ), lorque les centres sont mal positionnés, puis de tendre vers 0.

   On trouve dans [39] une discussion de l'effet du gain sur l'apprentissage k-means.

Malgré ces défauts, l'algorithme des K-moyennes est très utile en pratique. Il converge généralement assez rapidement (quelques présentations successives de l'ensemble des exemples). Nous l'emploierons à de nombreuses reprises dans les chapitres suivants, pour initialiser divers classifieurs.

Algorithme LBG

L'algorithme LBG [139] est une méthode de quantification vectorielle très connue, qui évite d'avoir à choisir l'ensemble initial des vecteurs de référence comme dans KMS. LBG n'est pas basé sur la minimisation de l'erreur par descente du gradient.

On procède comme suit :

1. On se fixe la taille finale L du dictionnaire, qui doit être une puissance de deux. On se donne aussi un vecteur de *perturbation* ϵ, qui va être utilisé pour générer les centres.
2. Choisir comme premier centre w_0 le centre de gravité de l'ensemble d'apprentissage.
3. Remplacer chaque centre w_i par deux vecteurs, obtenus à l'aide de la perturbation ϵ :

$$\begin{aligned} w'_{2i} &= w_i + \epsilon \\ w'_{2i+1} &= w_i - \epsilon \end{aligned}$$

   D'autres types de perturbations sont envisageables [139].
4. Partitionner l'ensemble d'apprentissage en utilisant la règle du plus proche voisin sur les nouveaux centres. Recalculer les positions des centres obtenus, en les remplaçant par les barycentres des groupes d'exemples formés.
5. Si le nombre L de centres n'est pas encore atteind, recommencer les étapes 3, 4 et 5.

Nous n'avons pas travaillé systématiquement avec cet algorithme dans le cadre de cette thèse. Le lecteur pourra consulter [38] pour des références et des résultats expérimentaux comparant LBG à d'autres méthodes de quantification.

2.8 Analyse en Composantes Principales

L'analyse en composantes principales (ACP) est une technique d'analyse factorielle qui permet de réduire la dimension de l'espace de représentation des données de manière à perdre le moins d'information possible.

On dispose d'un ensemble de formes $\mathcal{X} = \{x_1, x_2, \cdots, x_n\}$ de $\mathbb{R}^d$. On cherche un sous-espace W de $\mathbb{R}^d$, caractérisé par sa base orthonormée $(u_1, u_2, \cdots, u_{d'})$ et son origine O', sur lequel on va projeter orthogonalement les points x_i. On exprime la qualité de cet espace par un critère de déformation minimale du type

$$J = \frac{1}{n} \sum_{i=1}^{n} d^2(x_i, x_i^p) \tag{2.58}$$

ou' x_i^p est le résultat de la projection de x_i sur W. d est une distance.

On montre sans peine [58] que l'origine optimale n'est autre que le centre de gravité de $\mathcal{X}$ défini par

$$\hat{\mu} = \frac{1}{n} \sum_{i=1}^{n} x_i \tag{2.59}$$

On peut éliminer ce paramètre en utilisant des données centrées.

Les vecteurs de base sont les vecteurs propres de la matrice de variance-covariance :

$$\hat{\Sigma} = \frac{1}{n} \sum_{i=1}^{n} x_i x_i^t \; - \; \hat{\mu}\hat{\mu}^t \tag{2.60}$$

définis par

$$\hat{\Sigma}\, u_i = \lambda_i u_i \tag{2.61}$$

Les valeurs propres donnent l'importance de chaque *axe principal*. Elles représentent la variance de la projection des données sur l'axe correspondant.

La transformation d'un vecteur x se calcule comme

$$x^p = Ux \tag{2.62}$$

ou' la matrice U regroupe les vecteurs propres u_i.

Notons que dans le domaine du traitement d'images, la transformation donnée par (2.62) est nommée *transformation de Karhunen-Loève* [109].

Chapitre 3

Modèles connexionnistes

3.1 Introduction

Dans ce chapitre, nous décrivons les modèles de réseaux connexionnistes que nous avons été amené à utiliser durant l'élaboration de cette thèse. Nous présentons d'abord rapidement les «grands classiques» les plus connus, puis détaillons successivement les modèles que nous avons utilisé durant nos travaux.

Nous présentons enfin plusieurs nouvelles architectures modulaires simples basées sur la coopération de différents algorithmes connexionnistes. L'utilisation de plusieurs modules, chacun adapté à un type de traitement (extraction de caractéristiques, réduction de dimension, classification, ...) et optimisés globalement permet souvent un gain important de performances et une nette diminution de la complexité du système complet.

Ces architectures seront testées en détail dans le chapitre 6, qui décrit les simulations menées sur le problème de la reconnaissance des chiffres manuscrits.

3.2 Définitions et rappels historiques

3.2.1 Historique

Le connexionisme peut être décrit comme l'étude d'une large classe de modèle mathématiques simples, inspirés à leur origine de l'étude des systèmes nerveux animaux [37].

En 1943, Mc Culloch et Pitts [153] ont proposé un modèle formel du neurone biologique, basé sur un automate à seuil doté de plusieurs entrées et d'un état interne. Ce *neurone formel* est l'unité de base des modèles connexion-

nistes. Il s'est avéré que les réseaux de neurones formels permettaient de rendre compte d'un certain nombre de propriétés du système nerveux, ce qui a encouragé la poursuite de recherches dans ce domaine, tant par des biologistes que par des cybernéticiens.

En 1958, le Biological Computer Laboratory (BCL), dirigé par le physicien autrichien von Foerster [68] a mené un important programme d'études visant à faire progresser la connaissance du système nerveux par l'étude des réseaux de neurones formels. Ces recherches ont été axées sur l'interprétation de la perception en termes de calculs d'invariants par des réseaux, en vue de l'élaboration d'automates capables de simuler un comportement intelligent.

Les recherches du BCL ont abouti à une première modélisation rudimentaire des propriétés du cerveau avec, par exemple, la mise en évidence du rôle fondamental des configurations de notre système nerveux lors de la perception du monde extérieur. Mais c'est la notion de calcul qui est restée au centre des préocupations de ces chercheurs, et ce sont les capacités calculatoires des réseaux plutôt que leur aptitude à modéliser le système nerveux qui ont fait l'objet des études ultérieures.

En 1957, Rosenblatt [178, 179] propose un modèle de machine adaptative, le perceptron. Il s'agit d'une machine possédant trois couches d'unités, qui effectuent simultanément des traitements ; c'est donc une machine parallèle dont certaines connexions sont ajustables, ce qui la dote d'une capacité d'apprentissage. Nous avons décrit ce modèle, d'un point de vue «classique», dans la section 2.6.3 du chapitre 2. Le perceptron a démontré qu'une machine, en modifiant sa structure, est capable de mémorisation.

Le modèle du perceptron a joué un grand rôle dans le développement des idées connexionnistes. Ses limites ont été décrites par Minsky et Papert [156] en 1969 : seuls des problèmes «simples» (i.e. formes linéairement séparables) peuvent être appris par un perceptron. Cette constatation a poussé au développement de l'intelligence artificielle symbolique, et a se détourner des modèles d'apprentissage numériques au profit de la logique et des traitements symboliques de l'information (systèmes experts etc).

L'étude des réseaux connexionnistes a connu une période d'arrêt au cours des années 70, jusqu'au début des années 80.

En 1982, le physicien J. Hopfield [102] s'est interessé à une classe de modèles connexionnistes inspirés par les modèles de systèmes désordonnés élaborés en physique statistique (verres de spins etc). Ces travaux ont suscité un grand regain d'intérêt dans le domaine, et sont à l'origine des progrès spectaculaires enregistrés ces dernières années.

3.2.2 Le connexionnisme

Le connexionnisme étudie des modèles de calculs *massivement parallèles*, permettant l'élaboration d'algorithmes *adaptatifs* pour l'apprentissage numérique. Les modèles obtenus sont appliqués à la solution de divers problèmes d'intelligence artificielle, en général concernant des tâches de *bas niveau*, comme la perception.

Les applications les plus étudiées concernent entre autres la reconnaissance des formes (RDF), la classification, l'approximation, la prévision, le contrôle, etc...

La variété d'algorithmes et d'outils théoriques utilisés dans ce domaine est si grande qu'il parait impossible de définir plus précisément le connexionnisme. La définition la plus honnête semble celle proposée par Poggio [171] : les réseaux connexionistes ne sont rien d'autre qu'une notation graphique pour une très large classe d'algorithmes...

Nous allons brièvement décrire dans les paragraphes suivants la terminologie en usage pour la description des algorithmes connexionnistes.

3.2.3 Neurone formel

La description des algorithmes connexionnistes s'appuie sur une modélisation des «neurones» ou *unités* (ou encore «cellules») et de leurs interconnexions. L'unité la plus simple est celle de l'*automate à seuil*, introduite par McCulloch et Pitts [153] en 1943.

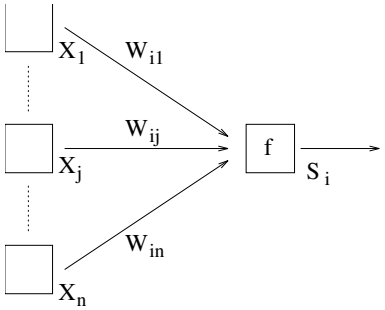


FIGURE 3.1 – Réprésentation d'un neurone formel, doté de ses entrées x_i, des poids synaptiques w_i et calculant l'état S.

Un neurone formel est un automate possédant plusieurs entrées scalaires $\{x_i\}$ et une sortie, ou *état*, S (voir figure 3.1).

3.2.4 Fonction de transfert

La sortie S de l'automate est une fonction des entrées appelée fonction de transfert du neurone :

$$S = f(x_1, x_2, \cdots, x_n) \tag{3.1}$$

On distingue deux grandes classes d'unités :
— unités *produit scalaire*, définies par

$$S = f(\sum_j w_{ij} x_j) \tag{3.2}$$

ou' f est peut être de l'un des types suivants (voir figure 3.2) :
— fonction identité : on parle alors d'unité linéaire (e.g. perceptron) ;
— fonction à seuil, type Heaviside : on a alors un automate *booléen* ;
— fonction sigmoïde : on parle alors d'unité sigmoïde. Ce type d'unité est très employé dans les perceptrons multi-couches décrits plus loin. f est typiquement de la forme :

$$f(x) = 1.71 \ \tanh(\frac{2}{3}x) \tag{3.3}$$

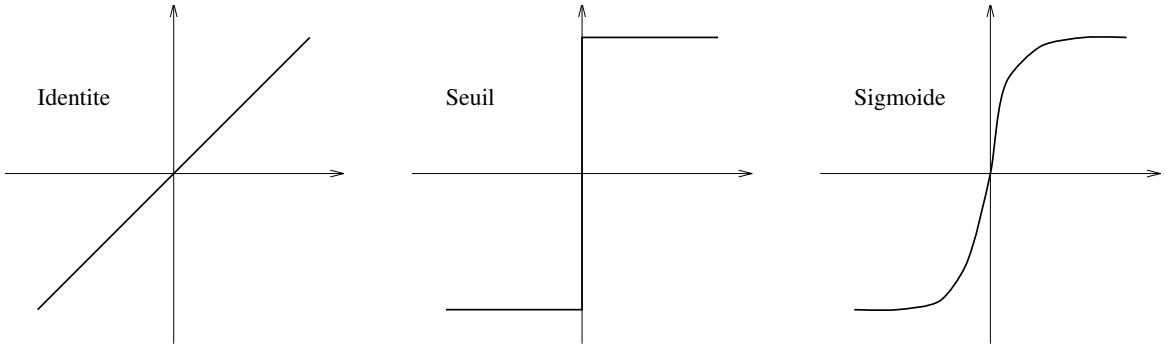


FIGURE 3.2 – Les fonctions de transfert les plus courantes pour les neurones «produit scalaire».

— unités *distance*, définies par

$$S = f(\sum_j \|X_j - w_j\|^2) \tag{3.4}$$

ou' f est une fonction identité ou un noyau (gaussien le plus souvent). La distance utilisée est presque toujours la distance euclidienne.

La fonction de transition permet de déduire l'état de l'automate à l'instant $t + 1$, connaissant ses entrées à l'instant t. Notons que cette définition ne couvre pas les automates probabilistes pour lesquels la fonction de transfert intègre un aspect aléatoire.

3.2.5 Architecture d'un réseau

Les unités sont généralement utilisées au sein d'un *réseau de neurones*, dans lequel les sorties de certaines unités servent d'entrées à d'autres. La valeur du poids associé à la connexion entre deux unités réflète la relation entre ces deux unités. On associe fréquement les poids aux connexions et non aux unités elles-mêmes comme nous l'avons fait plus haut.

Connexions complètes

Il existe de nombreuses façons de connecter les unités d'un réseau. La plus simple est de relier la sortie de chacune à toutes les autres ; on parle alors de connexions complètes.

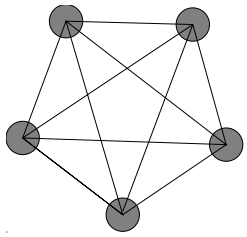


FIGURE 3.3 – Un réseau à connexions complètes.

Boucles

On dit qu'un réseau est bouclé lorsque le graphe des connexions possède des cycles. C'est évidemment le cas des connexions complètes.

Dans ce cas, il faut définir un *séquencement* utilisé lors du calcul des états. On utilise en général l'un des deux modes suivants :

séquencement asynchrone : les unités sont mises à jour indépendamment. Il est aisé de simuler ce mode sur une machine séquentielle en tirant au hasard à chaque instant la prochaine unité à recalculer ;

séquencement synchrone : les unités sont mises à jour simultanément, en utilisant les états de l'instant précédent.

Le problèmes de séquencement sont généralement contournés en utilisant des schémas de connexion sans boucles.

Connexions en couches

On utilise couramment des architectures organisées en couches de neurones. Les unités sont alors regroupées en couches successives. Chaque unité est connectée aux unités de la couche suivante uniquement (figure 3.4).

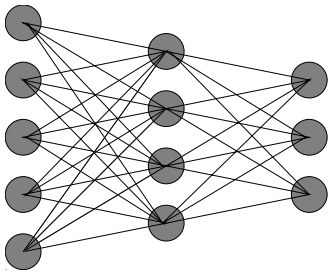


FIGURE 3.4 – Architecture multi-couches.

La première couche est appelée *couche d'entrée*, et la dernière *couche de sortie*. Les couches intermédiaires sont dites *couches cachées*.

La couche d'entrée est utilisée pour introduire des données extérieures dans le réseau. On l'appelle souvent *rétine* du réseau. Les «unités» de cette couche n'ont pas de connexions amont, et pas de poids synaptiques.

Notons qu'il existe de multiples variantes des réseaux en couches, la plus utilisée étant celle des *réseaux récurrents* (voir figure 3.5).

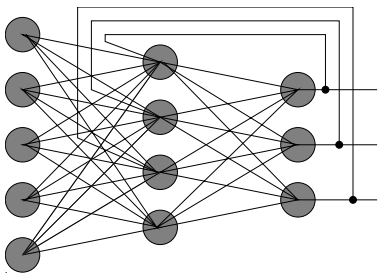


FIGURE 3.5 – Architecture multi-couches *récurrente*. Il s'agit d'un réseau bouclé, dans lequel une couche sert d'entrée à une couche située en amont. Ces réseaux sont souvent utilisés pour la prévision de séries temporelles, par exemple.

Connexions locales

Les connexions *locales* sont un cas particulier d'architecture multi-couches. Dans schéma de connexion (voir figure 3.6), une cellule est connectée à un sous-ensemble des cellules de la couche précédente (parfois appelé *receiptive field* ou champ recepteur).

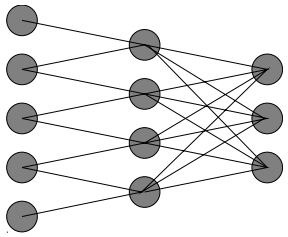


FIGURE 3.6 – Architecture multi-couches avec des connexions locales entre les deux premières couches.

Les connexions locales permettent des traitements locaux, souvent utiles en traitement d'image par exemple.

Architecture à poids partagés

On appelle architecture à *poids partagés* un réseau multi-couches dans lequel on introduit des contraintes d'égalité sur les poids de certaines connexions.

La plupart du temps, il s'agit de connexions locales (sans quoi la contrainte n'a aucun intérêt). Par exemple sur l'architecture représentée en figure 3.6, on peut partager les poids de la première couche de connexions locales. Le traitement effectué par cette couche dépend alors de deux valeurs (poids) au lieu de huit.

Un ensemble de connexions locales partageant les même poids est souvent appelé un *masque*.

L'utilisation de masques permet d'imposer un traitement *invariant par translation*. La couche de cellules calcule alors un filtrage non-linéaire de ses entrées. D'autre part, le partage des poids permet de réduire significativement le nombre de paramètres libres dans le réseau, ce qui permet d'augmenter la faculté de *généralisation* [128] d'un classifieur, par exemple.

Les premières architectures importantes basées sur l'utilisation de masques de poids partagés ont été proposée pour des systèmes de reconnaissance de phonèmes [126, 212, 213]. On appelle ces architectures *réseaux à délais* ou TDNN (Time Delay Neural Network). Les réseaux TDNN ont ensuite été très utilisé en reconnaissance de caractères [129], comme on le verra dans les chapitres suivants.

Calculabilité

Il est intéressant de déterminer le type de fonctions calculables par un réseau connexionniste. Il serait en effet vain de chercher à resoudre un problème avec un modèle de toute façon insuffisamment puissant. Par exemple, on a vu que le modèle du perceptron était incapable de classer sans erreurs des données non linéairement séparables.

Pour les réseaux multi-couches à fonction de transfert sigmoïde, Cybenko [47] a démontré un résultat très satisfaisant : toute fonction continue de $[-1,+1]^n$ dans $\mathbb{R}$ peut être approchée uniformément par un réseau multi-couches avec une couche d'unité cachées sigmoïdes et couche de sortie linéaire. Plus précisément, l'ensemble des fonctions de la forme

$$G(x) = \sum_{k=1}^{N} w_k^1 \; f\left(\sum_{i=1}^{n} w_{ik}^0 \, x_i \; + \theta_k\right) \tag{3.5}$$

ou' f est une fonction sigmoïde (voir eq. 3.3), est dense dans l'ensemble des fonctions continues sur $[-1,+1]^n$.

N est le nombre de cellules cachées. Ce résultat ne permet malheureusement pas de déterminer le nombre d'unités cachées nécessaire.

Cette propriété, qui fait du modèle MLP un approximateur universel, a été discutée par de nombreux auteurs [99, 103, 104] en s'appuyant sur le théorème de Kolmogorov-Arnold-Sprecher [124] ou sur le théorème de représentation de Riesz [182].

3.3 Modèles classiques

Nous passons rapidement en revue dans cette section quelques modèles de réseaux connexionnistes très connus.

Les premiers modèles, le perceptron et l'adaline, permettant l'apprentissage d'une surface de décision linéaire, ont été présenté dans la section 2.6.3 du chapitre 2.

3.3.1 Mémoires associatives linéaires

Les mémoires associatives linéaires ont été étudiées dès 1973 par T. Kohonen [120, 117, 121].

Un *associateur linéaire* est un réseau composé d'une seule couche d'unités linéaires, connectées à toutes les entrées. On note x les entrées et y les sorties. La fonction de transfert du système est donnée par

$$y = Wx \tag{3.6}$$

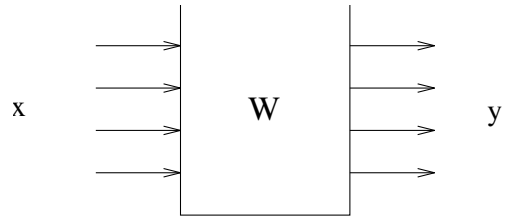


FIGURE 3.7 – Mémoire associative.

On dispose de N exemples (couples (x_i, y_i)), et l'on cherche la matrice de poids W optimale. Si l'on regroupe les exemples dans deux matrices X et Y, le problème s'écrit simplement

$$WX = Y \tag{3.7}$$

La matrice X est en général rectangulaire et non inversible. On utilise alors la matrice *pseudo-inverse* [164, 85] X^+ telle que

$$\begin{cases} XX^+X & = X \\ X^+XX^+ & = X^+ \\ X^+X \text{ et } XX^+ & \text{ sont symétriques.} \end{cases} \tag{3.8}$$

On démontre [85] que X^+ est unique, et que la solution suivante

$$W = YX^+ + Z(I - XX^+) \tag{3.9}$$

minimise l'erreur quadratique moyenne $|WX - Y|^2$. Z est une matrice quelconque. W est de norme minimale si $Z = 0$.

En pratique, on calcule rarement explicitement la pseudo-inverse et l'on préfère minimiser l'erreur quadratique par une procédure de descente de gradient (e.g. règle de Widrow-Hoff [219, 18]).

Kohonen [121] discute en détail des propriétés des mémoires associatives linéaires. Il montre en particulier qu'elles peuvent être utilisées pour débruiter des images. Par exemple, après apprentissage sur un ensemble d'images de visages, on peut reconstruire l'image d'un visage dont on cache une partie.

3.3.2 Réseaux de Hopfield

Le modèle de Hopfield, proposé en 1982 [102, 165] a suscité un grand intérêt de la part de la communauté des physiciens. Ce modèle est en partie à l'origine du renouveau de la recherche en connexionnisme dans les années 1980.

Les réseaux de Hopfield sont attrayant car basés sur un modèle simple, proche des modèles de physique des milieux désordonnés. La simplicité du modèle permet de déduire un certain nombre de propriétés théoriques (capacité de mémorisation, convergence etc).

Un réseau de Hopfield est un ensemble d'unités booléennes totalement connectées. On fixe l'état des unités à l'instant $t = 0$ et on laisse évoluer le système. Si la matrice de poids W est symétrique, on montre [102, 101] que la fonction «énergie» définie par

$$H = -S^t\, W\, S = -\sum_{i,j} w_{ij}\, s_i s_j \tag{3.10}$$

ou' S est le vecteur booléen formé par l'état des unités, est toujours décroissante (en séquencement *synchrone* ou *asynchrone*). Les unités arrivent donc toujours dans un état stable (minimum local de H). Les états stables sont appelés *attracteurs* du réseau.

On peut chercher les valeurs des poids W telles que le réseau admette K attracteurs déterminés S^k. On réalise ainsi une mémoire auto-associative. Il suffit de choisir W de telle sorte que les points S^k soient des minima de H.

Pour cela, on peut utiliser la *règle de Hebb* [98]

$$\Delta w_{ij} = -\frac{\partial}{\partial w_{ij}} H(s^k) = s_i^k s_j^k \tag{3.11}$$

ce qui revient à minimiser la fonction de coût

$$C = -\frac{1}{K} \sum_{k=1}^{K} \sum_{i,j} w_{ij} s_i^k s_j^k \tag{3.12}$$

On obtient alors en sommant sur les attracteurs

$$w_{ij} = \sum_{k=1}^{K} s_i^k s_j^k \tag{3.13}$$

Si le nombre d'exemples K est faible devant le nombre d'automates (inférieur à 1/10 [216]), l'énergie possède un minimum local par exemple. Ce n'est plus possible lorsque K augmente ; il apparait alors des attracteurs parasites.

3.4 Perceptrons multi-couches

Les modèles présentés ci-dessus sont tous incapables de traiter le problème de la classification de données non linéairement séparables. C'est d'ailleurs

cette difficulté, exposée clairement par Minsky et Papert [156] qui a limité l'intêret porté aux modèles connexionnistes dans les années 1960.

L'introduction des réseaux multi-couches permet de s'affranchir de cette contrainte. L'avancée majeure dans ce domaine est la mise au point de l'algorithme de *rétro-propagation du gradient*. Le principe de cet algorithme est très simple et découle des règles de dérivation des fonctions composées. De nombreux auteurs sont à l'origine de la formulation «moderne» de la rétro-propagation du gradient [183, 217, 133].

3.4.1 Algorithme de rétro-propagation du gradient

L'algorithme de rétro-propagation du gradient permet de déterminer les poids associés aux cellules des couches cachées d'un réseau multi-couches. Ce problème est souvent évoqué sous l'appellation de «*credit assignement problem*». La question est d'obtenir une règle permettant l'adaptation des poids de chaque couche lorsque la sortie calculée par le réseau diffère de la valeur désirée.

On considère un réseau sans boucle d'unités sigmoïdes. On notera [1] $\mathcal{I}$ l'ensemble des unités d'entrée (rétine) et $\mathcal{O}$ l'ensemble des unités de sortie. On notera également $\text{Amont}(k)$ l'ensemble des unités dont les sorties sont connectées à l'unité k, $\text{Aval}(k)$ l'ensemble des unités connectées à la sortie de l'unité k. On a par conséquent

$$\forall i \in \mathcal{O}, \text{Aval}(i) = \emptyset \text{ et } \forall i \in \mathcal{I}, \text{Amont}(i) = \emptyset$$

Enfin, on notera w_{ij} le poids de la connexion liant l'unité i à l'unité j.

Calcul des dérivées

Soit $(s_k)_{k\in\mathcal{I}}$ un exemple et $(s^*{}_k)_{k\in\mathcal{O}}$ les sorties désirées associées. On souhaite minimiser le coût C donné par la distance entre les sorties réelles et les sorties désirées

$$C_x = \sum_{k\in\mathcal{O}} (s^*{}_k - s_k)^2 \tag{3.14}$$

Pour cela, il nous faut calculer les dérivées partielles de C_x par rapport à chacun des poids. On note a_i la somme pondérée des entrées de l'unité i

$$a_i = \sum_{k\in\text{Amont}(i)} w_{ki}\, s_k \tag{3.15}$$

$$s_i = f(a_i) \tag{3.16}$$

1. La dérivation de l'algorithme exposée ici utilise les notations de Léon Bottou [18].

On calcule

$$-\frac{\partial C_x}{\partial w_{ij}} = -\frac{\partial C_x}{\partial a_j}\,\frac{\partial a_j}{\partial w_{ij}} = -\frac{\partial C_x}{\partial a_j}\,s_i = b_j\,.\,s_i \tag{3.17}$$

ou' l'on a posé

$$b_j = -\frac{\partial C_x}{\partial a_j} \tag{3.18}$$

Il reste à évaluer b_j. Notons que les variables introduites (s, a, C) dépendent toutes implicitement des poids w. Pour calculer b_j, on doit distinguer [133] deux cas, selon que l'unité j appartient à la sortie ou non. Si $j \in \mathcal{O}$

$$b_j = -\frac{\partial C_x}{\partial a_j} = -f'(a_j)\,\frac{\partial C_x}{\partial s_j} = 2f'(a_j)(s^*{}_j - s_j) \tag{3.19}$$

Si j est une unité cachée

$$\begin{aligned} b_j &= -f'(a_j)\,\frac{\partial C_x}{\partial s_j} \\ &= f'(a_j) \sum_{k \in \mathrm{Aval}(j)} -\frac{\partial C_x}{\partial a_k}\,\frac{\partial a_k}{\partial s_j} \\ &= f'(a_j) \sum_{k \in \mathrm{Aval}(j)} b_k\, w_{jk} \end{aligned} \tag{3.20}$$

L'algorithme de rétro-propagation du gradient se résume aux trois équations suivantes :

$$\begin{array}{l} \Delta w_{ij} = \epsilon\; b_j\,.\,a_i \\ \text{si } j \in \mathcal{O}, \quad b_j = 2\,f'(a_j)\,(s^*{}_j - s_j) \\ \text{sinon}, \qquad b_j = f'(a_j)\,\sum_{k \in \mathrm{Aval}(j)} b_k\, w_{jk} \end{array} \tag{3.21}$$

Gradient stochastique

L'algorithme décrit ci-dessus est une descente de gradient *stochastique*. En effet, le coût que l'on désire minimiser n'est pas C_x, qui dépend de l'exemple courant, mais une erreur globale C sur toutes les formes rencontrées,

$$C = \int C_x\, p(x) dx \tag{3.22}$$

Pour éviter ce problème, les premières versions de la rétro-propagation du gradient calculaient le gradient total (sur la base d'apprentissage), c'est à dire

$$\frac{\partial C}{\partial w_{ij}} = \left\langle \frac{\partial C_x}{\partial w_{ij}} \right\rangle \tag{3.23}$$

Ce procédé présente en fait de multiples inconvénients. La convergence s'avère plusieurs dizaines de fois plus lente que dans la version stochastique. Plus grave, l'algorithme de gradient total n'a aucun moyen de s'échapper d'un minimum local : si les dérivées de C s'annulent, l'apprentissage prend fin. Dans la version stochastique, chaque exemple «tire» les poids pour minimiser l'erreur instantannée, et ce qui permet d'échapper ainsi aux minima locaux (car le coût global peut augmenter à certains moments).

Poids partagés

Dans les réseaux à poids partagés (voir section 3.2.5), certaines connexions partagent le même poids w. Cela implique une légère modification de l'algorithme d'apprentissage.

On montre aisément que la dérivée du coût par rapport à un poids partagé est égale à la somme des dérivées des poids non partagés.

3.4.2 Estimation des probabilités a posteriori

On utilise couramment les réseaux multi-couches entrainés à l'aide de la rétro-propagation du gradient pour obtenir un classifieur. Pour cela, on présente la forme x à classer sur la couche d'entrée du réseau, et l'on associe à chacune des classes une cellule de la couche de sortie. On cherche alors à minimiser un critère des moindres carrés de la forme

$$C(x, y, w) = \|s(x, w) - y\|^2 \tag{3.24}$$

ou‘ l'on note $s(x, w)$ le vecteur formé par les états des cellules de sortie et y la sortie désirée pour la sortie x. Le vecteur y code la classe de l'exemple x. Le codage habituel est défini par

$$y_i = \begin{cases} 1 & \text{si } x \in \omega_i \\ -1 & \text{sinon} \end{cases} \tag{3.25}$$

On peut montrer [23, 177, 26, 88, 136, 181] qu'un classifieur minimisant un critère de la forme 3.24 donne une approximation au sens des moindres carrés des probabilités a posteriori d'appartenance de x à chacune des classes $P(\omega_i|x)$.

Lors de l'utilisation du réseau en classification, le critère qui affecte un vecteur x à la classe associée à la cellule la plus active revient à choisir la classe ω_j telle que

$$P(\omega_j|x) = \max_i P(\omega_i|x) \tag{3.26}$$

On retrouve donc la règle de décision de Bayes. Cependant, l'approximation des probabilités est inexacte, car l'on dispose généralement d'un nombre limité d'exemples.

3.4.3 Liens avec l'Analyse Discriminante

Principe de l'analyse discriminante de Fisher

L'analyse discriminante de Fisher [64] est une technique bien connue d'analyse de données. Le principe est de projeter les données à classer dans un sous-espace de dimension réduite, calculé de façon à ce que les classes y soit le plus séparées possible (voir figure 3.8).

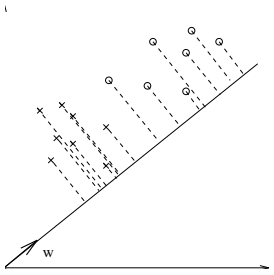


FIGURE 3.8 – Projection des données sur la droite définie par w, telle que les points projetés permettent la meilleure séparation des classes possible.

Une fois les données projetées, on peut utiliser dans le sous-espace un classifieur quelconque, qui sera plus facile à mettre en œuvre car il travaillera dans un espace de faible dimension.

Le calcul du projecteur optimal se ramène à un calcul de valeurs propres sur les matrices de variance totale, intra-classes et inter-classes [53, 97]. Cette opération est disponible dans la plupart des logiciels de statistique standards.

Equivalence entre analyse discriminante et perceptrons multi-couches linéaires

Gallinari et al. [74, 73] ont montré l'équivalence entre un réseau multi-couches linéaire et l'analyse discriminante.

Plus précisément, on considère un réseau de classification (voir 3.4.2) possédant une couche cachée, des unités à fonction de transfert linéaires, et des connexions complètes entre chaque couche. La sortie calculée par le réseau est alors donnée par les deux matrices de poids W_1 et W_2

$$s = W_2 W_1 x \tag{3.27}$$

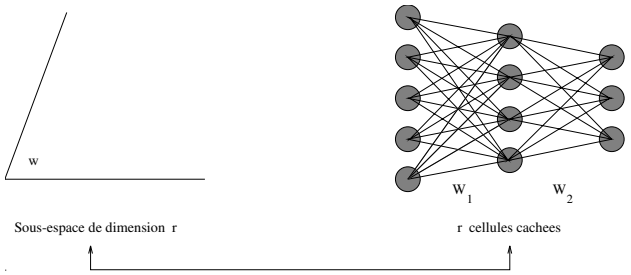


FIGURE 3.9 – Analyse discriminante (à gauche) et réseau multi-couches (à droite).

Si le réseau est entrainé pour minimiser le critère $\|y - W_2 W_1 x\|^2$, on montre que la matrice W_1 est solution du problème de l'analyse discriminante : la première couche effectue une projection dans le sous-espace optimal. La dimension de ce sous-espace est évidemment le nombre d'unités cachées du réseau.

Cas non-linéaire

L'équivalence présentée ci-dessus n'est pas démontrée dans le cas d'unités à fonction de transfert non linéaire (e.g. sigmoïde). Par contre, de nombreux résultats expérimentaux [4, 11, 1] indiquent que les réseaux multi-couches à sigmoïdes opèrent également une projection et une séparation des classes sur les couches cachées. De couche en couche, on observe une séparation des données, jusqu'a obtenir (lorque le réseau trouve une solution correcte) des groupes compacts et linéairement séparables sur la dernière couche cachée.

3.4.4 Liens avec l'Analyse en Composantes Principales

Nous avons déja présenté dans le chapitre précédent la technique d'analyse en composantes principales (ACP) (section 2.8).

Bourlard et Kamp [21] ont étudié les liens entre réseaux multi-couches et ACP. On se place ici dans le cas de réseaux à une couche de cellules cachées, utilisé en *auto-association* : on désire reproduire l'entrée sur la couche de sortie. La couche cachée, de dimension inférieure à l'entrée, doit coder les formes présentées pour pouvoir les reconstruire au mieux (voir figure 3.10).

Pour des unités linéaires, on montre [21, 2] que la couche cachée calcule alors la projection dans l'espace des composantes principales de dimension correspondante. Cependant, on perd l'information «d'ordre» obtenue par l'ACP : on calcule une projection dans le sous-espace, mais on ne dispose

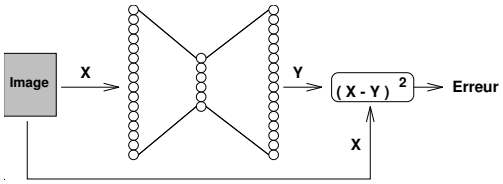


FIGURE 3.10 – Architecture auto-associative.

pas des axes principaux (ni des valeurs propres).

On peut également montrer que l'introduction de non-linéarités ne modifie pas la solution obtenue par le réseau.

Réduction de dimension non-linéaire

Le modèle présenté ci-dessus permet une approximation de l'analyse en composantes principales par un réseau MLP. Le traitement reste donc limité par son caractère linéaire, comme le montre l'exemple suivant. Soit N points disposés sur un hélice dans un espace tri-dimensionnel (cube $[-1,+1]^3$). Les points de l'hélice peuvent être repérés par un seul paramètre : c'est une variété de dimension un. L'analyse en composantes principales du nuage de points ne permet cependant pas de s'en rendre compte : les trois valeurs propres obtenues ont des amplitudes semblables. Seule une analyse *non-linéaire* permet de réduire la dimension du problème et de trouver le paramétrage optimal.

DeMers et Cottrell [49] ont montré que l'on pouvait résoudre ce problème en utilisant un réseau MLP doté de plusieurs couches cachées. La première couche cachée calcule alors un codage, et la couche centrale permet une forte réduction de dimension (ce type de réseaux posent cependant quelques problèmes de convergence : on ne trouve pas toujours la solution optimale). L'exemple de l'hélice est résolu avec une architecture totalement connectée comprenant $\{3, 10, 1, 10, 10, 3\}$ unités dans chaque couche.

D'autres types d'architectures ont été proposées, fondées par exemple sur une ACP *locale* [138].

3.4.5 Critères de rejet

Dans cette section, nous définissons différents *critères de rejet* utilisables avec des réseaux MLP classifieurs.

Nous avons évoqué en section 3.4.2 le fait que ces réseaux fournissaient, après apprentissage, une estimation des probabilités bayesiennes a posteriori d'appartenance de la forme présentée aux différentes classes apprises. Il est

naturel d'utiliser cette estimation pour classer ou rejeter chaque nouvelle forme (voir la section 2.3, chap. 2).

Nous proposons plusieurs critères de rejet. En effet, nous avons constaté expérimentalement que le critère le plus simple, basé sur l'interprétation des sorties comme probabilités bayesiennes, n'est pas toujours le plus performant, en particulier pour détecter les formes inconnues. Ces critères sont définis empiriquement (il semble difficile de dériver des relations théoriques entre le taux de rejet et le taux d'erreur pour les réseaux multi-couches). Nous les comparerons expérimentalement sur plusieurs problèmes dans le chapitre 6 (section 6.3.3).

Définition des critères de rejet MLP

Notons x la forme présentée au classifieur, S les sorties calculées par le réseau et i_1, i_2 les indices des deux cellules de sortie les plus actives. Notons aussi p_1 et p_2 les activités de ces cellules.

$$\begin{array}{lcll} i_1 & = & \arg\max_j S_j & p_1 = S_{i_1} \\ i_2 & = & \arg\max_{j \neq i_1} S_j & p_2 = S_{i_2} \end{array} \tag{3.28}$$

Le critère de décision habituel, sans rejet, consiste à affecter la forme x à la classe ω_{i_1}.

On peut maintenant définir les critères de rejet suivants :

MLP-1 si $p_1 < \theta$ alors rejeter x^k.

Ce critère est le plus simple possible. Il correspond à la règle de décison avec rejet optimale dans l'approche paramétrique (voir section 2.3.2, équation 2.14).

Une justification intuitive de ce critère est de noter que les formes ambiguës ou éloignées vont activer plusieurs cellules de sorties, ce qui implique, si les sorties somment à un, une faible valeur maximale.

MLP-2 si $p_1 - p_2 \leq \theta$ alors rejeter x^k.

Ce critère vise à améliorer le rejet en apportant de l'information sur l'activité de la deuxième cellule activée. L'idée est que les activités p_1 et p_2 sont généralement similaires pour les formes ambiguës [86].

MLP-3 si $\|S^k - D^k\| \geq \theta$ alors rejeter x^k, ou‘ D_k est un vecteur «désiré» pour la classe proposée par le classifieur, i.e. :

$$D^k_{i_1} = +1, \quad \text{et} \quad D^k_i = -1 \quad \forall i \neq i_1 \tag{3.29}$$

Ce critère prend en compte les activités de toutes les cellules de sortie. Il devrait par conséquent mieux détecter les formes inconnues (*outliers*) : même si la sortie maximale est élevée, de nombreuses classes sont généralement activées par ces formes.

MLP-4 si $(p_1, p_2) \in R_\lambda$ alors rejeter x^k, ou‘ R_λ est une région du plan définie ci dessous.

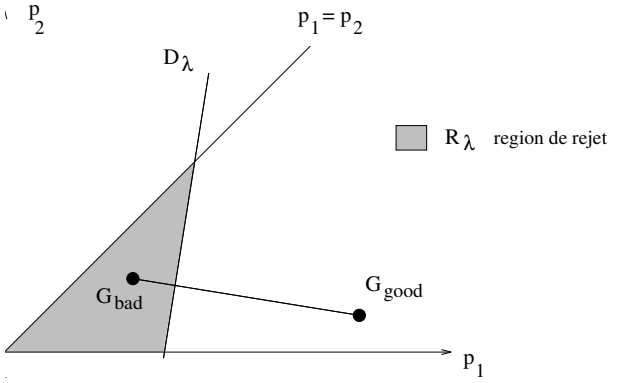


FIGURE 3.11 – Critère de rejet MLP-4. On définit une région dans le plan formé par les activités p_1 et p_2 des deux cellules maximales. G_{bad} est le barycentre des points (p_1, p_2) mal classés, et G_{good} le barycentre des points bien classés.

On considère les barycentres G_{good} et G_{bad} des points (p_1, p_2), mesurés respectivement sur les formes bien ou mal classées d'un ensemble d'évaluation. D_λ est une droite parallèle à la médiatrice du segment $(G_{\text{good}}, G_{\text{bad}})$ à distance λ (voir figure 3.11).
On rejetera les formes telles que (p_1, p_2) appartiennent à R_λ, ensemble des points placés du même coté que G_{bad} de D_λ. Le seuil de rejet ajustable est donc ici λ.
Ce critère n'utilise que les deux premières activités, et est par conséquent semblable au critère MLP-2 [19].

Remarques

— Les réseaux MLP ont souvent des problèmes pour rejeter les formes éloignées de celles rencontrées durant l'apprentissage [137, 209]. Ceci s'explique par le caractère «non local» de la fonction de transfert sigmoïde. Cette fonction est bornée et garde son activité maximale à l'infini. Durant l'apprentissage, les valeurs d'entrée sont en général proches de zero, et restent le plus souvent dans la partie linéaire de la sigmoïde. Si l'on présente ensuite une forme très différente, on rencontre des activités élevées, qui sont atténuées par la fonction de transfert : cette atténuation empèche de les distinguer des valeur normales [127, 215].

Les régions de décision associées à chaque classe sont par conséquent non bornées. Nous verrons plus loin que les classifieurs RBF permettent d'éviter ce problème en utilisant des unités locales (gaussiennes), au prix d'un accroissement du nombre d'unités nécessaires.

— L'utilisation d'une fonction de transfert de type sigmoïde joue un rôle important dans le succès des réseaux MLP. En effet, ce type de fonction permet de mêler, au sein d'un même classifieur, des traitements discr'ets (booléens) et des traitements continus linéaires. Lorsque les entrées d'une cellule sont importantes, l'état de sortie, qui correspond à la saturation de la sigmoïde, s'interprète comme une valeur booléenne. Lorsqu'elles sont petites, on reste dans le cadre linéaire évoqué plus haut.

3.5 Réseaux LVQ

L'algorithme *Learning Vector Quantization* (LVQ) a été proposé par Kohonen en 1986 [118, 119, 122]. Il s'agit principalement d'un classifieur supervisé, basé sur une quentification vectorielle adaptative. Du fait de sa simplicité de mise en œuvre et son efficacité, LVQ a été utilisé pour de nombreuses applications [152, 56].

3.5.1 Principe

Dans le formalisme connexionniste, on décrit LVQ comme un réseau à deux couches. La couche de sortie est constituée de neurones *distance* (voir section 3.2.4), dotés d'une fonction de transfert linéaire : chaque unité calcule simplement la distance entre l'entrée présentée et son vecteur de poids.

Lors de la construction du réseau, chaque unité est associée à une classe. On associe à chaque nouvelle forme la classe de l'unité d'activité maximale.

Ce classifieur (comme tous les classifieurs de type «plus proche voisin») permet de tracer des frontières de décision non linéaires, qui sont formées par les portions du diagramme de Voronoï séparant les unités associées à des classes différentes (figure 3.12).

3.5.2 Initialisation

L'initialisation du réseau LVQ consiste à déterminer les poids initiaux des unités (vecteurs de référence) et à leur attribuer une classe. Cette phase est déterminante pour les performances de l'algorithme (temps de convergence et taux d'erreur final).

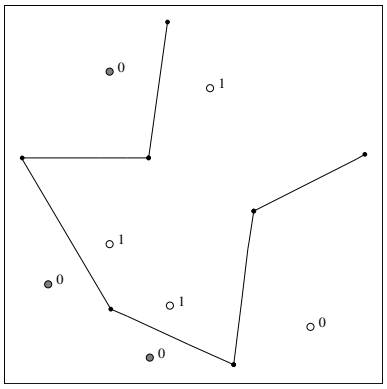


FIGURE 3.12 – Frontière de décision entre deux classes, représentées par 4 (classe 0) et 3 (classe 1) vecteurs de référence.

Le plus souvent, on utilise un algorithme simple de quantification vectoriel (type k-means, voir section 2.7.2). Notons que l'on peut procéder de plusieurs façons :

— effectuer un k-means sur tous les exemples, puis attribuer à chaque vecteur de référence la classe la plus représentée dans son entourage. On obtient alors un nombre variable d'unités dans chaque classe.

— effectuer un k-means à l'intérieur de chaque classe, de façon à placer les références associées à cette classe. Il faut alors déterminer a-priori le nombre d'unités associées à chaque classe.

Nous avons le plus souvent utilisé cette deuxième méthode, avec un nombre identique d'unités pour chaque classe.

Notons qu'il a été proposé d'utiliser des méthodes plus sophistiquées pour l'initialisation de LVQ [122]. En particulier, on peut utiliser l'algorithme des *cartes topologiques* [121], que nous ne décrirons pas ici.

3.5.3 Algorithmes d'apprentissage

Après initialisation, on optimise les positions des vecteurs de référence pour améliorer les performances de classification. Kohonen [123] a proposé plusieurs variantes pour l'apprentissage, que nous décrivons ici brièvement. Toutes sont basées sur une adaptation *stochastique* des paramètres : on présente les exemples dans un ordre aléatoire, et l'on modifie les poids après chaque présentation.

Nous adopterons les notations suivantes : x est l'exemple présenté et c sa classe. W_i est le vecteur de poids de l'unité i, et c_i sa classe. Soit i_1 l'indice

de la référence la plus proche

$$i_1 = \arg\min_i \|x - W_i\|^2 \tag{3.30}$$

i_2 est l'indice de la deuxième référence la plus proche

$$i_2 = \arg\min_{i \neq i_1} \|x - W_i\|^2 \tag{3.31}$$

LVQ1

La première version proposée est LVQ1. Seule l'unité la plus proche est modifiée selon la règle

$$\begin{array}{lll} \text{si } c = c_{i_1}, & W_{i_1}(t+1) = & W_{i_1}(t) + \alpha(t)\,[x - W_{i_1}(t)] \\ \text{si } c \neq c_{i_1}, & W_{i_1}(t+1) = & W_{i_1}(t) - \alpha(t)\,[x - W_{i_1}(t)] \end{array} \tag{3.32}$$

Si la référence la plus proche est de la bonne classe, on l'approche de l'exemple, sinon on l'en éloigne.

Le paramètre α, compris entre 0 et 1, contrôle le gain [2]. Sa valeur décroit lentement au cours de l'apprentissage, par exemple selon la loi $\alpha(t) = \alpha(0) + a/t$, avec $a \approx 0.09$.

On constate que cet algorithme représente une extension naturelle de k-means en présence de plusieurs classes (voir equation 2.56).

LVQ2

LVQ2 est une variante de LVQ1 dans laquelle on modifie simultanément les deux références les plus proches de l'exemple présenté. L'idée est de concentrer l'apprentissage sur les formes proches des frontières de décisions et incorrectement classées.

Pour cela, on définit une *fenêtre* entre chaque couple de références de classes distinctes (voir figure 3.13). Cette fenêtre est une région de l'espace située entre les deux références, définie par

$$\text{window}(i_1, i_2, \theta) = \{\ x \mid \frac{1}{\theta} > \frac{d_1}{d_2} \geq \theta\ \} \tag{3.33}$$

ou‘ θ est un seuil, et $d_k = \|x - W_{i_k}\|^2$ est la distance à la référence k.

2. Kohonen [123] distribue (sur ftp) une implémentation logicielle de LVQ (LVQ_PAK) qui contient entre autres un mécanisme plus efficace d'ajustement des gains. Je n'ai pas eu le temps d'en tester les performances.

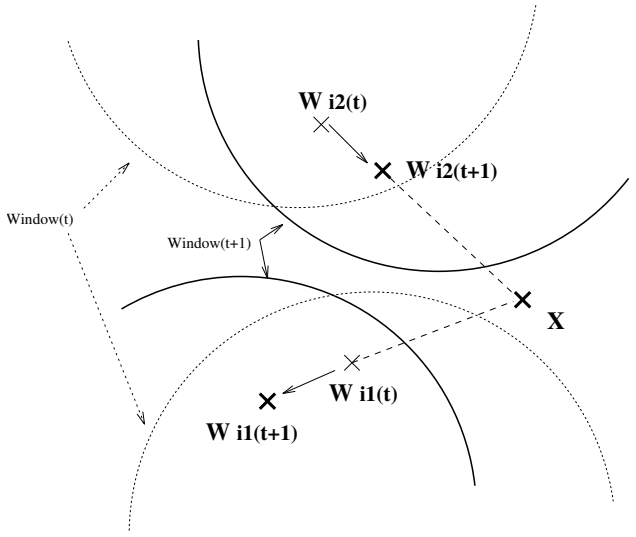


FIGURE 3.13 – Principe de LVQ2. L'exemple x est proche de i_1 et i_2, et appartient à la classe de i_2. S'il tombe dans la fenêtre (window(t)), on éloigne i_1 et l'on approche i_2. On a représenté les fenêtres avant et après adaptation.

Si l'exemple x est mal classé, mais que la deuxième référence est de la classe correcte et qu'il appartient à la fenêtre entre i_1 et i_2, alors on modifie les deux références suivant

$$\begin{aligned} W_{i_1}(t+1) &= W_{i_1}(t) - \alpha(t)\,[x - W_{i_1}(t)] \\ W_{i_2}(t+1) &= W_{i_2}(t) + \alpha(t)\,[x - W_{i_2}(t)] \end{aligned} \tag{3.34}$$

Cette modification revient à approcher de l'exemple la référence de la bonne classe tout en éloignant celle de la mauvaise classe.

LVQ2.1

L'algorithme LVQ2 est très performant et converge rapidement. Cependant, dans certains cas, on constate des instabilités empêchant la convergence. En effet, dans l'équation 3.34, les références sont déplacées d'une quantité proportionnelle à leur distance $\|x - W_i\|$ à l'exemple. Comme la référence de la bonne classe est toujours plus éloignée que celle de la mauvaise classe, on observe en moyenne un rapprochement des références : la distance $\|W_i - W_j\|$ diminue. Bien entendu, cet effet s'observe surtout dans des espaces de faible dimension ; lorsque la dimension est élevé, les corrections sont très rarement dans la même direction.

Afin d'éviter cet effet, on peut utiliser la variante suivante, que l'on appelera LVQ2.1. Dans cette variante, on supprime la condition «x mal classé» de

LVQ2. On modifie les deux références les plus proches (de classes distinctes) à chaque fois que x tombe dans la fenêtre.

Cette variante améliore, dans certains cas, la vitesse de convergence de LVQ, et ne dégrade jamais les performances. Aussi l'avons nous utilisée pour la plupart des simulations rapportées dans les chapitres suivants.

Remarques

Kohonen [122] a proposé une autre variante, dite LVQ3, qui semble poser plus de problèmes qu'elle n'en résout.

Notons enfin que d'autres versions de LVQ2 ont été proposées, par exemple pour écrire l'algorithme sous forme de la minimisation d'une fonction de coût [18, 57].

3.5.4 Critères de rejet

Comme pour les réseaux MLP, nous allons passer en revue quelques critères permettant de rejeter les formes ambigües ou inconnues avec un classifieur LVQ pour diminuer le risque d'erreur. Ici aussi, les critères seront justifiés empiriquement, et nous les comparerons expérimentalement dans le chapitre 6.

Définitions

Notons comme précédement i_1, i_2,..., i_p les indices de la plus proche, seconde, ..., p$^{\text{ieme}}$ référence, pour l'exemple x^k

$$i_p = \arg \min_{i \neq i_1, i_2, \ldots, i_{p-1}} \|x^k - W_i\|^2 \tag{3.35}$$

et $d_1, d_2, \ldots, d_p$ les carrés des distances de x^k à $W_{i_1}, W_{i_2}, \ldots, W_{i_p}$

$$d_t = \|x^k - W_{i_t}\|^2$$

dC_1 designera la distance de x^k à la classe la plus proche, et dC_2 la distance à la classe suivante

$$\begin{aligned} \mathrm{dC}_1 &= d_1 \\ \mathrm{dC}_2 &= d_p \text{ si classe}(i_1) = \text{classe}(i_2) = \ldots = \text{classe}(i_p - 1) \neq \text{classe}(i_p) \end{aligned} \tag{3.36}$$

Enfin, soit Basin(i) l'ensemble des formes pour lesquelles la référence W_i est la plus proche, et m_i son rayon moyen carré mesuré sur un ensemble d'évaluation

$$\text{Basin}(i) = \{x^k \text{ in evaluation set} |\ i_1 = i\}$$

$$m_i = \frac{1}{\text{card}(\text{Basin}(i))} \sum_{x^k \in \text{Basin}_{(i)}} \|x^k - W_i\|^2 \tag{3.37}$$

Définition des critères de rejet

Définissons maintenant les critères :

LVQ-1 si $d_1/d_2 \geq \theta$ et classe$(i_1) \neq$ classe(i_2) alors rejeter x^k.

Ce critère sélectionne simplement les formes appartenant à la fenêtre définie par l'algorithme d'apprentissage LVQ2, c'est à dire proches des frontières entre classes. On rejette donc des formes ambigües.

LVQ-2 si $\text{dC}_1/\text{dC}_2 \geq \theta$ alors rejeter x^k.

Cette condition est une extension de la précédente ; on considère ici les distances aux classes les plus proches au lieu des distances aux références. Ceci permet de rejeter des formes ambigües pour lesquelles les deux références les plus proches sont de la même classe, et que la troisième référence, d'une autre classe, est à peine plus éloignée.

LVQ-3 si $d_1/m_{i_1} \geq \theta$ alors rejeter x^k.

Ce dernier critère ne considère que la distance à la référence la plus proche, sans prendre en compte de notion de classe. On peut espérer qu'il détecte les formes éloignées, non rencontrées durant l'apprentissage ; par contre il devrait être incapable de rejeter les formes ambigües.

3.6 Réseaux RBF

3.6.1 Introduction

La méthode des *fonctions radiales de base* (RBF) est une technique d'approximation assez ancienne qui dérive de la méthode des *fonctions potentielles* [59]. Elle s'applique aussi bien à l'estimation de fonction, la prévision de séries temporelles ou aux problèmes de classification.

Le problème de d'approximation [171] se pose de la façon suivante : soit $f(X)$ une fonction continue de $\mathbb{R}^n \to \mathbb{R}$, et $F(W, X)$ une fonction approximation dépendant continuement de $W \in \mathbb{R}^p$ et X. On cherche le jeu de paramètres W^* tel que

$$\forall W,\ \ d(\ F(W^*, X),\ f(X)\ ) \leq\ d(\ F(W, X),\ f(X)\ ) \tag{3.38}$$

ou‘ d est une distance dans l'espace des fonctions.

Le problème de l'apprentissage consiste à déterminer les paramètres d'une fonction d'approximation étant donné un ensemble fini de N points $(x_i, f(x_i))$, ce qui revient à déterminer une hyper-surface interpolant les exemples. On cherche à obtenir une solution permettant une bonne *généralisation*, c'est à dire une estimation correcte de la fonction dans les zones ou' l'on ne dispose pas d'exemples. Pour cela, l'hyper-surface doit être la plus régulière possible. Durant l'apprentissage, on va minimiser un coût formé de deux termes [201] ; le premier exprime la qualité de l'interpolation (mesurée sur les exemples), tandis que le second pénalise les surfaces irrégulières. Ce terme s'exprime en général par l'intermédiaire de la norme d'un opérateur différentiel P.

$$\text{coût} = \sum_i (y_i - F(W, x_i))^2 \; + \; \lambda \| \, PF \, \|^2 \tag{3.39}$$

Typiquement, on cherche à minimiser la courbure en utilisant un opérateur laplacien [6].

L'existence et la qualité de la solution au problème d'approximation dépendent de la classe de fonctions à laquelle $F(W, X)$ appartient [176].

Nous avons déjà rencontré plus haut divers types de fonctions approximantes : les fonctions linéaires (section 2.6), ou' $F(W, X) = W.X$, et les perceptrons multi-couches

$$F(W, X) = \sigma \left(\sum_n w_n \, \sigma \left(\sum_i w_i \, \sigma \left(\; \cdots \; \sigma \left(\sum_j u_j X j \right) \; \cdots \right) \right) \right) \tag{3.40}$$

ou' σ est la fonction sigmoïde.

Un autre type d'approximation très utilisé repose sur l'emploi d'une base de fonctions Φ_i :

$$F(W, X) = \sum_i w_i \, \Phi_i(X) \tag{3.41}$$

Les interpolations par splines, les expansions en séries de polynômes orthogonaux et les classifieurs «fonctionnels» [70] entrent, parmi d'autres, dans ce cadre. Un grand intérêt de cette représentation est que l'on dispose d'un approximateur *non linéaire* en ayant à résoudre un problème linéaire (choix de W_i, les fonctions Φ_i étant déterminées).

3.6.2 Fonctions Radiales de Base

Les modèles RBF reposent sur l'équation 3.41, ou' $\Phi_i(X)$ est une fonction *radiale*, ne dépendant que de la distance de X à un centre μ_i :

$$F(W, X) = \sum_i w_i \, \Phi \left(\frac{(X - \mu_i)^2}{\sigma_i^2} \right) \tag{3.42}$$

ou‘ W regroupe les w_i, μ_i et σ_i.

Φ est en général une fonction localisée autour de zero à décroissance rapide, typiquement une exponentielle ($\Phi(r) = e^{-r}$).

On peut établir pour ce type d'approximateur le même type de résultats que pour les perceptrons multi-couches (voir section 3.2.5) : toute fonction continue peut être approchée avec une précision arbitraire [163, 170].

Notons que l'équation 3.42 est de la même forme que l'estimateur de Parzen [198] (voir section 2.5.2).

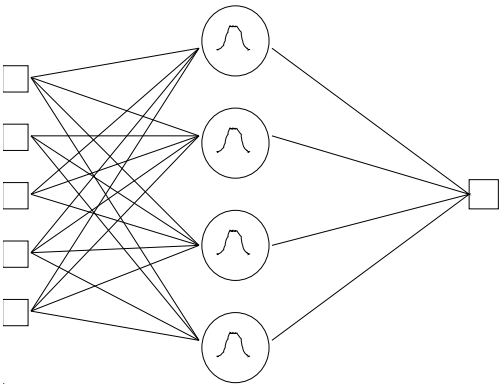


FIGURE 3.14 – Réseau RBF.

On peut représenter l'équation 3.42 sous forme d'un réseau de neurones à une couche cachée (figure 3.14). Les unités de la couche cachée sont des neurones *distance* dont la fonction de transfert est Φ.

L'apprentissage d'un réseau RBF consiste à déterminer les positions des *centres* μ, les valeurs des *largeurs* σ et les poids de la dernière couche w. Cette décomposition des paramètres en trois groupes dont l'interprétation intuitive est aisée est un net avantage des réseaux RBF sur les perceptrons multi-couches.

L'approche la plus simple consiste à attribuer à chaque exemple disponible un centre RBF, puis à fixer les largeurs de toutes les fonctions à une valeur arbitraire puis à calculer les poids w de façon à satisfaire le jeu d'égalités

$$F(W, x_i) = y_i \tag{3.43}$$

Cette approche n'est cependant pas très performante car la taille du réseau croît avec le nombre d'exemples disponibles. De plus, on accorde la même importance à toutes les données, ce qui rend le résultat très sensible aux bruits.

Avant de décrire la méthode que nous proposons, nous allons passer un revue les principales publications récentes concernant le sujet.

3.6.3 Revue bibliographique

Moody et Darken

Moody et Darken [157] ont décrit en 1989 un algorithme rapide d'apprentissage pour les réseaux RBF. La méthode proposée repose sur un apprentissage non supervisé des centres μ_i et des largeurs σ, suivie d'un apprentissage supervisé des poids w de la deuxième couche.

Le positionnement des centres est effectué à l'aide de l'algorithme k-means (voir section 2.7.2) : on fixe a-priori l'architecture du réseau (nombre d'unités k), et l'on place les centres de façon à minimiser l'erreur de quantification

$$E = \sum_{\alpha=1}^{k} \sum_{i=1}^{N} M_{\alpha i}(\mu_\alpha - x_i)^2 \tag{3.44}$$

ou' $M_{\alpha i}$ est une matrice kxN donnant l'appartenance de l'exemple au groupe (*cluster*) auquel il appartient ($M_{\alpha i} = 1$ si l'unité α est la plus proche de x_i, 0 sinon).

Les largeurs σ_α associées à chaque centre sont ensuite calculées par une heuristique type «plus proches voisins» : la largeur d'une unité est proportionnelle à la distance à l'unité la plus proche. Cette heuristique assure un certain recouvrement entre les unités voisines, qui permet d'obtenir une interpolation continue dans les régions couvertes.

Les poids w sont enfin déterminés en minimisant l'erreur quadratique sur l'ensemble d'apprentissage (on peut utiliser ici la matrice pseudo-inverse, voir section 3.3.1).

L'avantage de cette méthode est la rapidité de l'apprentissage (la phase la plus coûteuse est le k-means initial). Par contre, rien ne garantit que le paramètres obtenus soient optimaux, car ils sont optimisés séparément.

Broomhead et Lowe [28] ont montré que cet algorithme permettait de résoudre simplement le célèbre problème du «ou exclusif» (XOR, qui tracassait les connexionnistes [133] car c'est l'exemple le plus simple de problème non linéairement séparable).

Saha et Keeler [185] discutent de l'effet du choix des largeurs, et indiquent qu'un facteur de recouvrement égal à 2 est généralement optimal.

Poggio et Girosi

Poggio et Girosi [170, 171], à partir de l'étude des concepts d'approximation et de régularisation ont proposé un cadre général pour l'approximation de fonctions (HyperBF) dans lequel les RBF habituelles apparaissent comme un cas particulier. Ces articles sont la véritable référence dans ce domaine.

Une solution simple proposée est d'utiliser

$$F(W, X) = \sum_{\alpha=1}^{n} w_\alpha \Phi(\|X - \mu_\alpha\|_\nu^2) \tag{3.45}$$

La norme utilisée est pondérée : $\|x\|_\nu^2 = x^T \nu^T \nu\, x$, de façon à prendre en compte l'importance relative de chaque composante en entrée. Les coefficient de pondération de la norme sont adaptatifs.

Les paramètres w_α, μ_α et ν sont optimisés simultanément par descente de gradient pour minimiser une fonction de coût régularisée (équation 3.39). Auparavant, on les initialise à des valeurs raisonnables :

— centres μ_α positionnés sur un sous-ensemble des exemples ;
— poids w_α calculés par pseudo-inverse.

Wettschereck et Dietterich

Wettschereck et Dietterich [218] ont étudié expérimentalement l'effet de l'apprentissage des différents paramètres dans les réseaux RBF. Le réseau est initialisé de la façon proposée par Moody, puis éventuellement optimisé par une descente de gradient. Les auteurs comparent les différentes méthodes sur le problème «NETtalk» [188] (apprentissage de la prononciation de l'anglais à partir du texte).

Leurs conclusions sont les suivantes :

— le classifieur RBF simple (ala Moody) est inférieur aux autres classifieurs testés : kNN, arbres de décisions ID3 [54], réseaux multi-couches (MLP).
— le classifieur «GRBF» (optimisation des centres et des poids) améliore de 20 % le taux de reconnaissance par phonème par rapport au RBF simple. GRBF offre de meilleures performances que les réseaux MLP et est équivalent au meilleur algorithme connu pour cette tâche (arbre de décision avec codes correcteurs).
— L'optimisation supplémentaire des largeurs σ ne permet pas d'améliorer les performances.
— Enfin, les temps d'apprentissage de GRBF et du réseau MLP sont équivalents (ce qui n'est pas surprenant, l'algorithme d'optimisation étant identique), on perd la rapidité d'apprentissage de la méthode de Moody et al.

Bishop

Chris Bishop a testé l'utilisation d'un régularisateur type «courbure minimale» pour les réseaux connexionnistes : MLP [6] puis RBF [7]. On optimise

un coût de la forme

$$E = \sum_i (y_i - F(W, x_i))^2 \; + \; \lambda \frac{1}{2} \sum_{p=1}^{N} \sum_i \left\{ \sum_n \left(\frac{\partial^2 F_i(W, x_p)}{\partial x_n^2} \right)^2 \right\} \qquad (3.46)$$

ou‘ n indice les cellules d'entrée et i les sorties.

L'auteur, contrairement à ce que propose Poggio et al., n'optimise que les poids w de la dernière couche à l'aide de l'équation 3.46. L'avantage est que l'on conserve un apprentissage très rapide, les techniques d'optimisation linéaires (pseudo-inverse) pouvant encore être utilisés. L'auteur présente des résultats expérimentaux convainquants sur des problèmes simples d'interpolation.

Musavi et al.

Musavi et al. [160, 159] ont étudié les RBF pour la classification d'un point de vue orienté «statistique», et proposent des algorithmes pour la sélection des centres, largeurs et poids. L'inconvénient de ces algorithmes est qu'ils ne sont pas *adaptatifs* : l'apprentissage demande la présence simultanée de tous les exemples et des quantité importantes de calculs.

Pour la sélection des centres, les auteurs proposent [159] un algorithme itératif de *clustering* supervisé.

On estime ensuite les largeurs et la norme utilisée (comme chez Poggio et al.) à l'aide du calcul des ellipsoïdes CPS (*constant potential surface*) obtenus par une procédure d'orthogonalisation type Gram-Schmidt.

La méthode suggérée semble un peu lourde (temps et complexité des calculs) et nous manquons de résultats comparatifs sur des problèmes réels.

Carlin

Mats Carlin [36] compare différentes functions radiales pour les réseaux RBF : gaussiennes, multi-quadriques, splines TPS (*Thin Plate Splines*), splines pseudo-cubiques, logarithmiques.

L'apprentissage utilisé est un positionnement non-supervisé des centres suivi d'une optimisation conjointe de tous les paramètres par descente de gradient.

Les tests comparatifs sont menés sur cinq différents problèmes (contrôle de processus, interpolation de fonction, robotique).

La conclusion est que les RBF gaussiennes se comporte bien sur tous les problèmes. Pour certaines applications, l'utilisation d'autres types de fonctions peut permettre un gain de performances. Si l'on ne dispose pas de

connaissances a-priori pour guider ce choix, il semble qu'il vaille mieux s'en tenir aux gaussiennes (d'autant, comme le rappelle Poggio, que la gaussienne est seule fonction radiale factorisable).

Autres applications

Les réseaux RBF ont été utilisés avec succès pour de nombreuses applications. Citons la prévision de séries temporelles [110, 154, 215, 95, 96], la reconnaissance de parole [195, 196], la reconnaissance optique de caractères [137] et le diagnostic [42, 112].

Les arbres de décision sont une autre méthode de classification très performante. Certains travaux cherchent à intégrer les deux approches [78] .

On trouvera aussi une revue des premiers travaux concernant les RBF dans [172].

3.7 Formalisme modulaire des algorithmes d'apprentissage

3.7.1 Introduction

Nous présentons dans cet section un formalisme général pour la description de systèmes *modulaires* d'apprentissage. Ce formalisme est essentiellement dû à Léon Bottou et Patrick Gallinari [15, 18, 14, 16], et a été inspiré par les travaux de Michel de Bollivier [12, 11].

Lorsqu'on aborde la conception d'un système de reconnaissance des formes, on a généralement des idées sur la décomposition de la tâche. Par exemple, pour un système de reconnaissance de chiffres, on décompose habituellement le traitement en plusieurs phases : pré-traitements, extraction de caractéristiques, classification. En reconnaissance de parole, il a été proposé [22, 56] des systèmes hybrides mixant réseaux multi-couches et programmation dynamique. Cette décomposition en sous-modules facilite la conception des systèmes complexes, et elle permet une interprétation du traitement effectué à chaque étage qui aide lors de la mise au point.

La principale difficulté introduite par les systèmes multi-modulaires est la méthode d'apprentissage : il s'agit de déterminer les paramètres de chaque module pour optimiser les performances du systèmes global. Ce problème est résolu en écrivant la fonction de coût globale comme une composition de fonctions, et à appliquer une technique simple de minimisation sous contraintes.

Le formalisme que nous présentons n'apporte pas d'idées nouvelles du point de vue des algorithmes d'optimisation. La présentation modulaire permet néanmoins de décrire simplement l'algorithme d'apprentissage, dans un formalisme qui ne dépend pas de la nature des modules. Ce formalisme permet des implémentations informatiques des réseaux très efficaces, que nous avons développées pour les simulations présentées par la suite.

3.7.2 Systèmes d'apprentissage

La formalisation probabiliste des algorithmes d'apprentissage a été étudiée de longue date par de nombreux auteurs [202, 18] (voir aussi un résumé de Tsypkin dans [55]).

Soit des exemples $\tilde{x}$ du problème à apprendre (dans le cas d'une tâche de classification, $\tilde{x}$ serait la paire (forme, classe)). La loi de probabilité d'observation des exemples est $p(\tilde{x})$, inconnue.

Notre système dépend des paramètres w et l'on exprime la tâche à apprendre (but de l'apprentissage) à l'aide du *coût local*, $J(\tilde{x}, w)$ qui mesure la qualité de la réponse du système sur chaque exemple.

On cherche à minimiser le coût moyen

$$C(w) = \int J(\tilde{x}, w)\, p(\tilde{x})\, d\tilde{x} \tag{3.47}$$

On peut ici encore employer la descente de gradient stochastique

$$w_{t+1} = w_t - \epsilon_t \nabla J(\tilde{x}_t, w_t) \tag{3.48}$$

L'expression du coût local J exprime simultanément la relation paramétrique entre l'entrée et la sortie du système, et rend compte de la qualité de ce dernier.

3.7.3 Systèmes modulaires

Définitions

Dans un système multi-modulaire, on souhaite exprimer la sortie comme composition des fonctions de transferts des différents modules. Pour cela, on va décomposer l'expression du coût J.

Le système est constitué d'un ensemble de modules dont on notera les entrées x_k, les sorties y_j et les paramètres w_i (figure 3.15). Les entrées de chaque module sont reliées aux sorties d'un ou plusieurs modules amont

(on n'autorise pas de boucles). Chaque module est défini par une fonction dérivable f_n calculant les sorties en fonction des entrées et des paramètres

$$y_j = f_n(x_k, w_i) \tag{3.49}$$

Le rôle du premier module est de fournir les données, il ne possède que des sorties. Le dernier module calcule le coût J, on l'appellera *module d'erreur*.

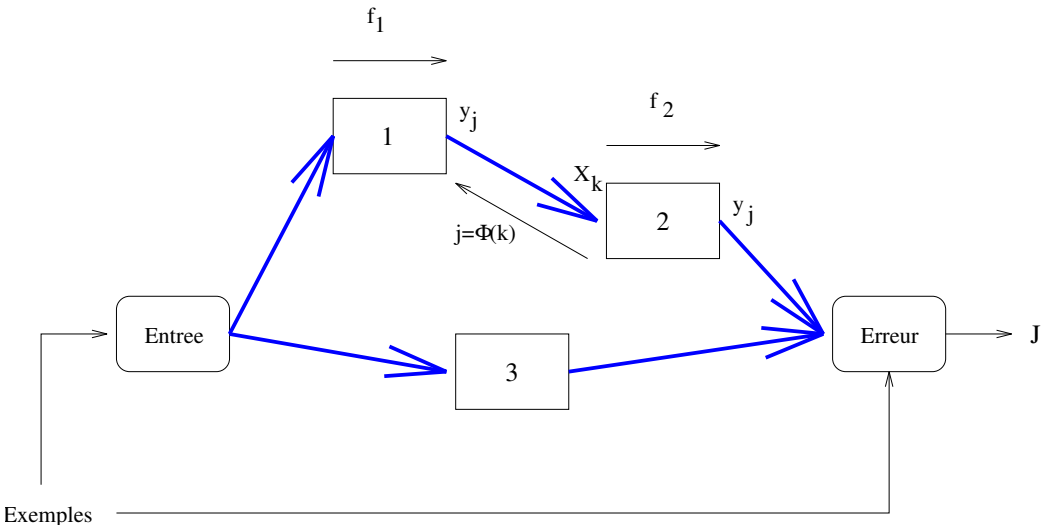


FIGURE 3.15 – Architecture multi-modulaire. Chaque module, paramétré par w_i, prend une entrée x_i et calcule la sortie y_i. Le dernier module calcule $y_{\text{last}} = J(\tilde{x}, w)$.

Si l'on veut mener les calculs sans encombres, il convient d'adopter une convention d'indiçage rigoureuse pour les différentes grandeurs considérées. On notera $Y^{-1}(n)$, $X^{-1}(n)$ et $W^{-1}(n)$ les indices associés respectivement aux sorties, entrées et paramètres du module n. Inversement, Y(j), X(k) et W(i) désigneront l'indice du module associé respectivement aux sorties y_j, entrées x_k et paramètres w_i.

La topologie des connexions entre modules sera exprimée par une fonction ϕ associant à l'indice d'une entrée d'un module l'indice de la sortie amont reliée à cette entrée (voir figure 3.15) ; on écrira

$$\forall k,\ x_k = y_{\phi(k)} \tag{3.50}$$

Nous pouvons maintenant aborder le calcul des dérivées du coût local J par rapport à tous les paramètres du système, de façon à appliquer l'apprentissage (equation 3.48).

Calcul des dérivées du coût local

Pour calculer les dérivées du coût, on écrit le Lagragien [133]

$$L = J - \sum_k \beta_k \left(x_k - y_{\phi(k)}\right) - \sum_j \alpha_j \left(y_j - f_j((x_k)_{k\in X^{-1}Y(j)}, (w_i)_{i\in W^{-1}Y(j)})\right) \tag{3.51}$$

ou' α et β sont les multiplicateurs de Lagrange. Le premier terme est le coût et les deux suivants s'annulent si les contraintes données par l'architecture du réseau (équations 3.49 et 3.50) sont vérifiées.

Les multiplicateurs de Lagrange sont donnés par les équations

$$\frac{\partial L}{\partial y_j} = 0 = -\alpha_j + \sum_{k\in\phi^{-1}(j)} \beta_k \tag{3.52}$$

et, si l'entrée k appartient au dernier module :

$$\frac{\partial L}{\partial x_k} = 0 = -\beta_k + \frac{\partial J}{\partial x_k} \tag{3.53}$$

sinon,

$$\frac{\partial L}{\partial x_k} = 0 = -\beta_k + \sum_{j\in Y^{-1}X(k)} \alpha_j \frac{\partial f_j}{\partial x_k} \tag{3.54}$$

Les multiplicateurs de Lagrange sont les quantités conjuguées des sorties et des entrées de chaque module : $\alpha_j = \partial J/\partial y_j$ et $\beta_k = \partial J/\partial x_k$. Chaque module peut calculer les β_k à partir des α_k (équation 3.54) ; ces derniers se calculent simplement grâce à l'équation 3.52. Ainsi, tous les multiplicateurs de Lagrange peuvent être calculés au cours d'une unique récurrence arrière à travers le réseau.

La dérivée Δ_i de J par rapport à un paramètre w_i est simplement

$$\Delta_i = \frac{\partial J}{\partial w_i} = \left.\frac{\partial L}{\partial w_i}\right|_{x,y,\beta,\alpha} = \sum_{j\in Y^{-1}W(i)} \alpha_j \frac{\partial f_j}{\partial w_i} \tag{3.55}$$

Résumé

Résumons les équations permettant l'apprentissage d'un système multi-modulaire :

$$\alpha_j = \begin{cases} \sum_{k\in\phi^{-1}(j)} \beta_k & \text{si la sortie } j \text{ n'appartient pas au dernier module} \\ 1 & \text{pour le dernier module} \end{cases} \tag{3.56}$$

$$\beta_k = \sum_{j \in Y^{-1}X(k)} \alpha_j \frac{\partial f_j}{\partial x_k} \tag{3.57}$$

$$\Delta_i = \frac{\partial J}{\partial w_i} = \sum_{j \in Y^{-1}W(i)} \alpha_j \frac{\partial f_j}{\partial w_i} \tag{3.58}$$

Ces trois équations décrivent tous les flux d'informations nécessaires pour effectuer l'apprentissage global du système. Les gradients $Delta_i$ doonent les modifications à apporter aux paramètres de chaque module.

Du point de vue de l'implémentation de l'apprentissage, il suffit de savoir effectuer trois opérations sur chaque module :

1. passe avant : calcul des sorties y_j en fonctions des entrées x_k et des paramètres w_i du module ;
2. passe arrière : calcul des quantités conjuguées des entrées β_k connaissant les conjugués α_j des sorties, et les x_k et w_i ;
3. passe arrière : calcul des gradients Δ_i connaissant les conjugués α_j des sorties, et les x_k et w_i ;

Le reste de l'algorithme est indépendant de la nature des modules : calcul des conjuguées des sorties (equation 3.56), modification des paramètres (équation 3.48), gestion des flux de données. Cette formulation se prête remarquablement à une implémentation *orientée objet* : chaque module est un objet doté de méthodes calculant les trois étapes énumérées ci-dessus.

3.7.4 Exemples

Nous allons maintenant passer en revue quelques exemples de modules d'apprentissage, briques de base permettant de construire tous les algorithmes connexionnistes utilisés dans cette thèse.

Le lecteur trouvera d'autres exemples dans la littérature : cartes topologiques de kohonen [187], programmation dynamique [56, 57].

Nous ne détaillons pas les calculs des dérivées pour chaque classe de modules [18], qui sont immédiats. On associe à chaque classe d'algorithme un symbole, et on donne les trois équations gouvernant la dynamique.

— Produit matriciel (Wx)

Fonction :	$y_i = \sum_k w_{ik} x_k$
Apprentissage :	$\beta_k = \sum_j \alpha_j w_{jk}$
	$\Delta_{jk} = \alpha_j x_k$

— Sigmoïde

Fonction :	$y_k = f(x_k)$
Apprentissage :	$\beta_k = f'(x_k)\alpha_k$
	(pas de paramètres)

— Distance Euclidienne ($(x-w)^2$)
  Fonction : $y_j = \sum_k (w_{jk} - x_k)^2$
  Apprentissage : $\beta_k = -2 \sum_k \alpha_j (w_{jk} - x_k)$
  $\Delta_{jk} = 2\alpha_j (w_{jk} - x_k)$
— Erreur Quadratique (MSE)
  Fonction : $J = \sum_k (d_k - x_k)^2$
  Apprentissage : $\beta_k = -2(d_k - x_k)$
  (pas de paramètres)
— Minimum
  Fonction : $J = x_{x^*} = \min\{x_k\}$
  Apprentissage : $\beta_{k^*} = 1, \beta_{k \neq k^*} = 0$
  (pas de paramètres)

A partir de ces quelques modules de base, on peut bâtir des réseaux correspondant aux algorithmes connexionnistes simples, comme le montre la figure 3.16.

La figure 3.16 montre aussi l'architecture d'un réseau RBF tel que ceux utilisés pour nos simulations. Ce réseau utilise un module «Gauss», défini comme

— Gauss
  Fonction : $y_i = \exp(-x_i / w_i^2)$
  Apprentissage : $\beta_k = -1/w_k^2 \alpha_k y_k$
  $\Delta_k = -2 x_k \beta_k / w_k$

3.7.5 Initialisation

L'algorithme d'optimisation par descente stochastique du gradient n'est efficace que si les paramètres des différents modules sont au préalable initialisés à des valeurs raisonnables. En effet, il existe en général de nombreuses configurations de paramètres triviales minimisant le coût tout en fournissant des solutions inintéressantes [11, 57].

Pour chaque type de réseau multi-modulaire, il convient de définir une stratégie efficace d'initialisation. Comme on l'a vu plus haut, les perceptrons multi-couches (MLP) ne nécessitent pas d'initialisation particulière des poids : des valeurs aléatoires font l'affaire. Cette agréable propriété provient de la nature des unités constituant ces réseaux : leur fonction d'activation couvre tout l'espace (unités non locales). Dans le cas d'unités locales (neurones distance en particulier), la convergence ne peut être obtenue que si les unités sont soigneusement placées dans des régions représentatives des exemples. On l'a vu pour l'initialisation des l'algorithmes k-means, LVQ et RBF.

Par exemple, si l'on place un module type k-means ou RBF à la sortie

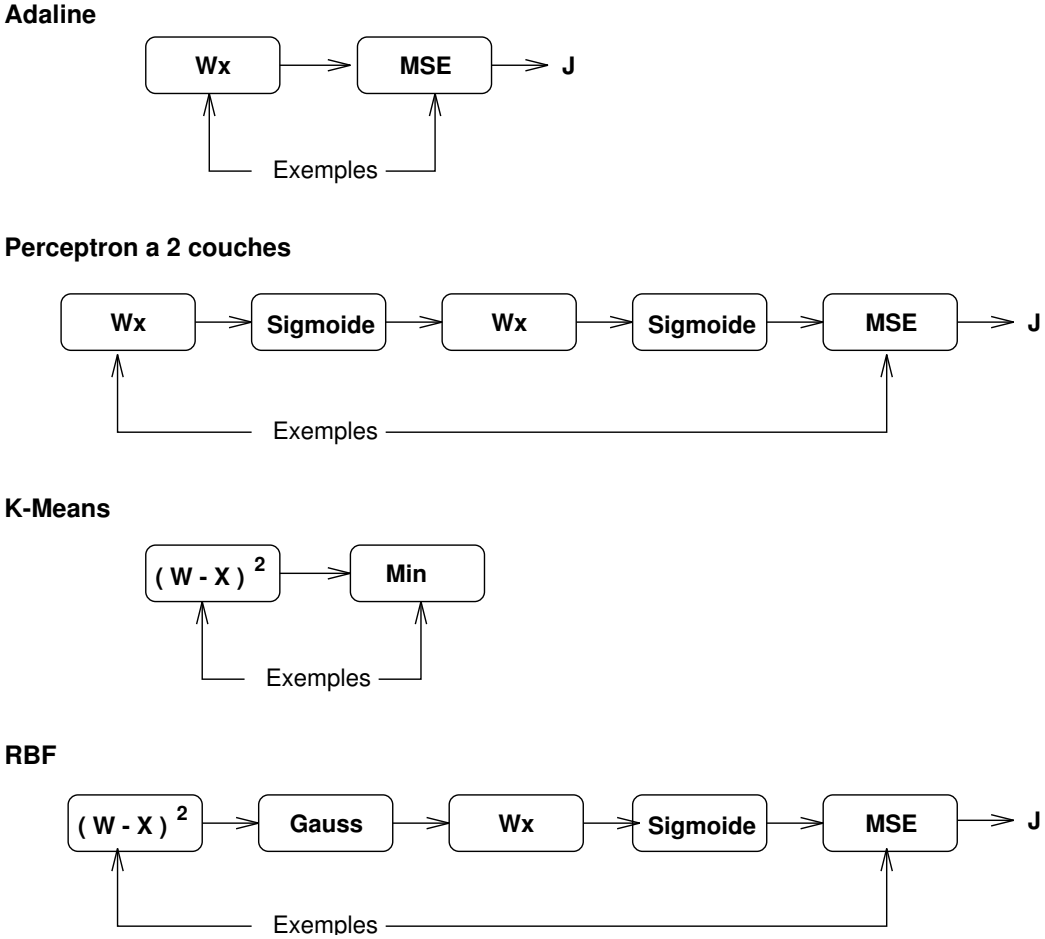


FIGURE 3.16 – Constructions de quelques algorithmes connexionnistes à partir de modules élémentaires : adaline, perceptron à 2 couches, algorithme k-means, et réseau RBF.

d'un module MLP chargé de l'extraction de caractéristiques, il est impossible d'initialiser les vecteurs de références du deuxième module sans fixer d'abord les poids du premier à des valeurs correctes.

On procédera donc souvent à un apprentissage incrémental du réseau modulaire, suivi d'une optimisation globale.

Dans la section suivante, nous détaillons l'exemple de la coopération MLP + RBF, combinaison débouchant sur des systèmes de classification très performants que nous utiliserons dans les chapitres suivants.

3.8 RBF et coopération MLP + RBF

3.8.1 Description du réseau RBF

Nous commençons par décrire l'architecture de réseau RBF que nous avons retenue. On a représenté cette architecture en bas de la figure 3.16.

Le premier module, de type *distance euclidienne*, calcule les distances entre l'entrée et N vecteurs de références de même dimension. Ces distances sont fournies au module *gauss*, paramétré par les largeurs ($w_n = \sigma_n$) associées aux gaussiennes. La suite du réseau correspond à une couche MLP à connexions complètes.

Nous sommes dans le cas ou‘ tous les paramètres du réseaux RBF sont adaptatifs.

Initialisation

Pour l'initialisation de notre RBF, nous suivons la méthode proposée par Moody et al. [157] (voir section 3.6.3). L'initialisation du module *distance euclidienne* est effectué par un k-means supervisé ; le réseau k-means est aussi représenté sur la figure 3.16. Une fois ce réseau entrainé, on déconnecte le module distance pour le placer au sein du réseau RBF.

Les largeurs σ sont ensuite estimées [185].

La matrice de poids de la dernière couche peut être estimée par un calcul de pseudo-inverse. Plus simplement, on peut l'initialiser ses coefficients à des petites valeurs aléatoires (comme dans un MLP habituel), en renforçant les coefficients reliant les centres RBF associés à une classe lors de l'initialisation à la cellule de sortie correspondant à cette classe.

Apprentissage

Après initialisation, on optimise les valeurs de tous les paramètres par descente de gradient (équation 3.56).

Contrairement à Wettschereck et al. [218], nous avons remarqué que l'apprentissage de tous les paramètres (y compris les largeurs) permettait un gain notable de performances (voir chapitre 6 pour les résultats expérimentaux).

Cette phase d'apprentissage est assez longue, d'autant que nous utilisons directement le gradient, sans informations du second ordre pour contrôler les gains. Les temps d'apprentissage sont comparables à ceux d'un réseau MLP sur le même problème.

Comme on le verra dans le chapitre 6, les réseaux RBF offrent des performances de classification semblables à celles d'un bon classifieur LVQ2.

3.8.2 Système MLP + RBF

Le modèle RBF présenté plus haut est moins puissant que celui proposé par Poggio et Girosi [170, 171] ou Musavi et al. [159] car on calcule les distances d'une forme aux références en utilisant la distance euclidienne, et non une norme pondérée (voir section 3.6.3). Ceci suppose par conséquent que l'on travaille dans un espace ou‘ les entrées sont déjà normalisées, de façon à ce qu'il soit justifié d'accorder la même importance à toutes les composantes.

Plutôt que de traiter ce problème dans le module RBF, nous proposons d'utiliser un module de pré-traitement, du type MLP.

La figure 3.17 représente l'architecture générale MLP + RBF.

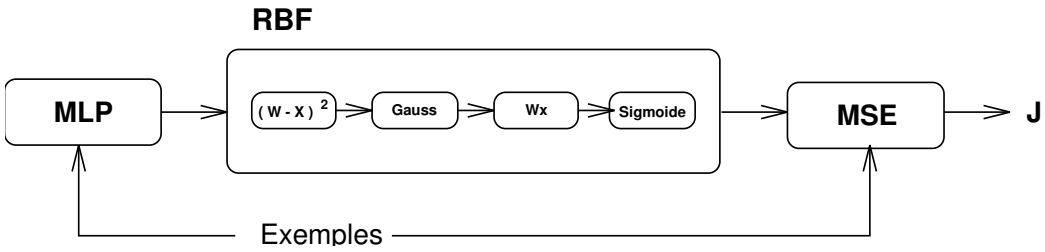


FIGURE 3.17 – Représentation modulaire d'un réseau MLP + RBF. Le module MLP est constitué une série de modules *produit de matrice* (ou *convolution*) et de modules *sigmoïde* alternés.

Cette combinaison offre de nombreux avantages : le module MLP va extraire des *caractéristiques* appropriées pour la classification, et va permettre au passage une réduction de dimension, comme on le verra au chapitre 6.

3.8.3 Apprentissage de MLP + RBF

Initialisation du module MLP

La première étape de l'apprentissage du système MLP + RBF est l'initialisation du module MLP de façon à ce qu'il permette l'initialisation du module RBF.

Pour cela, on effectue un apprentissage du MLP seul. Pour cela, on construit un système intermédiaire comprenant notre module MLP suivi d'un autre module MLP. Ce système (voir figure 3.18) constitue un réseau (perceptron multi-couches), sur lequel on peut appliquer l'algorithme d'apprentissage habituel. On introduit à cet effet un «but d'apprentissage intermédiaire», exprimé par le coût local J_{int}.

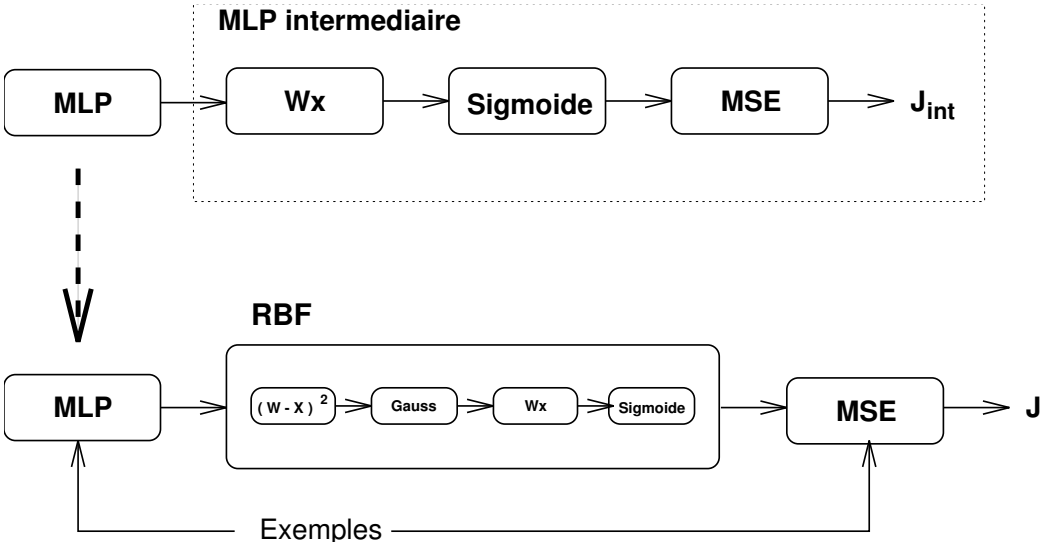


FIGURE 3.18 – L'initialisation du module MLP peut s'effectuer par l'apprentissage d'un système intermédiaire constituant un perceptron multi-couches complet (on a ici représenté une couche perceptron supplémentaire). Après apprentissage, le module MLP est connecté au RBF, et l'on n'utilise plus les modules intermédiaires (partie encadrée en pointillés).

Le réseau MLP intermédiaire est généralement un classifieur [11], mais il est aussi possible d'utiliser un système auto-associatif. Nous discuterons ces deux possibilités dans le chapitre 6.

Mise au point du module RBF

Une fois le module MLP initialisé, on l'utilise pour «pré-traiter» les exemples et l'on initialize le module RBF comme en section 3.8.1.

L'apprentissage RBF peut alors commencer. Deux solutions sont envisageables : effectuer un apprentissage *séquentiel* ou un apprentissage *coopératif*. L'apprentissage *séquentiel* optimise les paramètres du module RBF sans modifier le module MLP ; la solution atteinte a toutes les chances d'être sous-optimale, car le module MLP a été entrainé pour minimiser un coût intermédiaire J_{int} différent du coût associé au classifieur MLP+RBF complet.

L'apprentissage *coopératif* consiste à modifier simultanément tous les paramètres des deux modules pour minimiser le coût J.

Nous comparerons ces deux approches dans le chapitre 6.

3.8.4 Remarques

La méthode présentée ci-dessus s'applique de la même façon au système modulaire MLP+LVQ. Cette autre combinaison a été testée en détail par Driancourt et al. [56].

Le système MLP+LVQ pose cependant des problèmes de convergence qui oblige a modifier l'algorithme LVQ2 [57]. L'avantage du MLP+RBF est que les deux modules nécessitent chacun grossièrement le même nombre de passes d'apprentissage pour converger. Cette propriété permet, lors de l'apprentissage coopératif, à chaque module de s'adapter à l'autre pour atteindre une solution optimale.

3.9 Conclusion

Nous avons dans ce chapitre passé en revue différents modèles connexionnistes utilisés en classification, et avons détaillé les propriétés des plus importants (MLP,LVQ,RBF) en essayant de mettre l'accent sur les liens entre ces modèles et certaines techniques statistques mieux connues (Analyse Discriminante, Analyse en Composantes Principale, Estimation non paramétrique, ...).

Finalement, nous avons présenté un formalisme général permettant la construction et l'apprentissage de réseau multi-modulaires, combinant plusieurs algorithmes. L'approche multi-modulaire facilite grandement la conception de systèmes complexes pour un tâche donnée, et nous verrons par la suite qu'elle permet une nette réduction de la taille des systèmes obtenus, à performance égale.

Chapitre 4

Segmentation d'image et Analyse Multirésolution

4.1 Introduction

Nous abordons dans ce chapitre la segmentation d'image. Ce terme englobe un grande variété de techniques, axées sur l'extraction d'informations structurées à partir d'images de scène. La segmentation d'image est ainsi le premier maillon dans chaîne d'analyse de scène. Dans un tel système, on dispose d'une image «brute» (valeurs de luminance en chaque point ou pixel), et l'on cherche à construire une représentation symbolique du contenu de l'image. La sophistication de cette représentation peut grandement varier d'une application à l'autre : description en language naturel de la scène, ou plus modestement positions dans l'image de certains objets recherchés. Le cas de la reconnaissance d'écriture (OCR), abordé dans le chapitre suivant, est une application typique.

Ce chapitre mentionne rapidement les grandes classes de techniques disponibles en segmentation d'image. Nous insistons ensuite sur les analyses multirésolution par ondelettes, qui offrent d'intéressantes perspectives en traitement d'image. Nous présentons dans ce cadre un algorithme de détection de contours multi-échelles, mis au point au début de ce travail de thèse.

4.2 Techniques de segmentation d'image

Les techniques de segmentation visent à isoler dans l'image des zones correspondant à certains types d'objet. Nous passons ici en revue les principales méthodes de traitement d'image disponibles pour la segmentation. Nous renvoyons le lecteur aux ouvrages de référence [109, 81, 180] pour plus

de détails.

4.2.1 Seuillages

Dans les cas les plus simples, les régions de l'image à isoler sont caractérisées par une luminosité plus forte que le reste. On peut alors appliquer un seuillage direct de l'image (sélection des pixels dont la valeur dépasse un certain seuil). En pratique, la situation est rarement aussi simple. On applique souvent un seuillage non pas sur l'image originale mais sur une image traitée (par exemple après passage un filtre de détection de contours).

Après seuillage, on dispose d'une image binaire. L'exploitation de cette image nécessite généralement un second traitement visant à regrouper les zones homogènes pour permettre leur manipulation ultérieure. Une technique très utilisée est la recherche des composantes connexes, que nous rencontrerons souvent en OCR.

4.2.2 Détection de contours par filtrage

De façon générale, on appelle contour une courbe séparant deux régions de l'image auxquelles on attribue des propriétés différentes.

En pratique, on caractérisera souvent les régions par leur luminosité ou leur texture, caractéristiques que nous qualifierons de «bas-niveau» car définies indépendamment du contenu sémantique de la scène.

Nous allons décrire brièvement la méthode proposée par J. Canny [34] pour la détection optimale de contours dans une image. Nous l'avons employé à plusieurs reprises durant cette thèse, pour la détection des contours des visages par exemple.

Filtre optimal de Canny

Dans l'approche proposée par Canny, on désire détecter un contour sur un signal en utilisant une convolution du signal par un filtre, suivi d'un seuillage du résultat. Les maxima du résultat seront considérés comme points de contour.

L'approche de Canny s'appuie sur un modèle de contour représenté par un saut d'amplitude du signal au point x_0 noyé dans un bruit blanc gaussien de moyenne nulle. On dérive l'expression du filtre optimal en partant des trois principes suivants :

1. **Bonne détection** : on doit rater le moins possible de points de contour, et détecter le moins possible de faux points de contour.

2. **Bonne Localisation** : les points marqués comme appartenant au contour doivent être le plus près possible du centre du vrai contour ;
3. **Unicité des réponses** : on veut une seule réponse associée à chaque contour.

On exprime ensuite chacun de ces principes sous la forme de contraintes sur le filtre. On note $G(x)$ le signal, $f(x)$ la réponse impulsionnelle du filtre, supposée nulle hors de $[-W, W]$. On suppose le «vrai» contour centré en $x = 0$.

La réponse sur le contour est :

$$H_G = H_G(0) = \int_{-W}^{+W} G(0-x)f(x)\,dx = \int_{-W}^{+W} G(-x)f(x)\,dx \tag{4.1}$$

La racine carrée de la réponse moyenne carrée du filtre au bruit blanc seul est :

$$H_n = n_0 \left[\int_{-W}^{+W} f^2(x)\,dx\right]^{1/2} \tag{4.2}$$

ou‘ n_0^2 est le carré moyen du bruit par unité de longueur. Le premier critère (détection) s'écrit comme le quotient de ces deux réponses :

$$\Sigma = \frac{\left|\int_{-W}^{+W} G(-x)f(x)\,dx\right|}{n_0\sqrt{\int_{-W}^{+W} f^2(x)\,dx}} \tag{4.3}$$

Pour exprimer le deuxième critère (localisation), on va calculer la variance de la position du contour détecté. En utilisant un développement de Taylor de la réponse totale $H_n + H_G$ autour de la position x_0 du contour détecté, on calcule l'espérance de x_0^2, et on obtient :

$$E[x_0^2] \approx \frac{n_0^2 \int_{-W}^{+W} f'^2(x)\,dx}{\left[\int_{-W}^{+W} G'(-x)f'(x)\,dx\right]^2} = \delta x_0^2 \tag{4.4}$$

On définit la *localisation* comme l'inverse de δx_0 :

$$\lambda = \frac{1}{\delta x_0} \tag{4.5}$$

La recherche du filtre optimal suppose de maximiser conjointement les équations (4.3) et (4.5), qui sont les formes mathématiques associées aux critères de détection et localisation. Pour cela, on peut maximiser le produit $\Sigma\lambda$ de (4.3) et (4.5), sous la contrainte donnée par le troisième critère (unicité

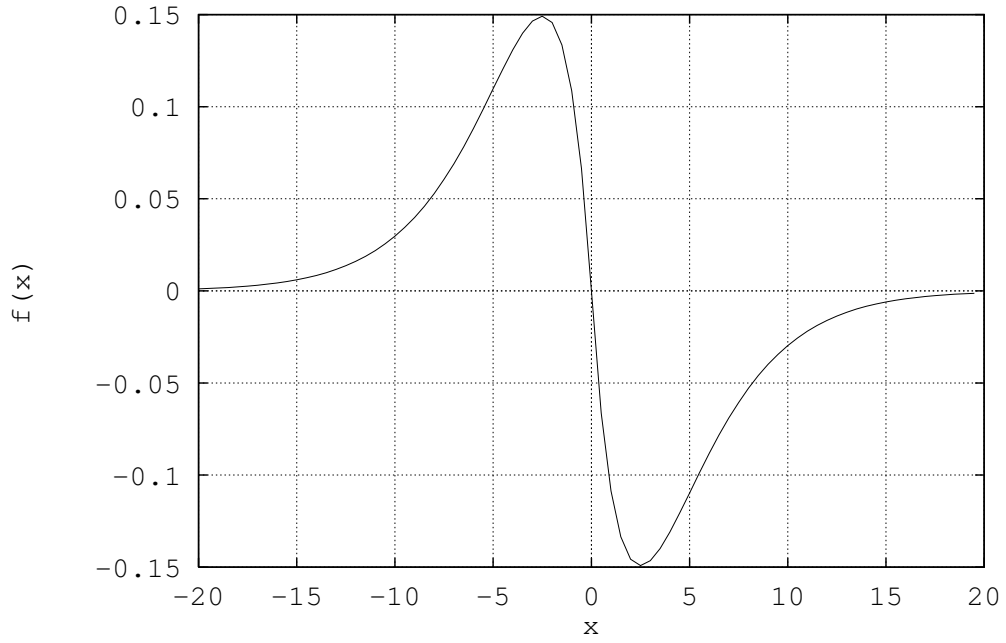


FIGURE 4.1 – Réponse impulsionnelle du filtre de Deriche, pour $\alpha = 0.4$.

des réponses). Ce dernier critère peut s'exprimer en calculant la distance moyenne entre les maxima de la réponse au bruit, notée x_{max}. On fixe cette distance à une fraction k de la largeur W du filtre : $x_{max} = kW$.

Canny a utilisé une optimisation numérique pour rechercher le filtre à réponse impulsionnelle finie f minimisant sous contrainte le produit $\Sigma\lambda$. Il a observé que la solution était proche de la dérivée d'une gaussienne (filtre DOG), très utilisé en traitement d'image [148].

Implémentation récursive

Deriche [50, 51] a appliqué la même démarche tout en cherchant à obtenir un filtre récursif (à réponse impulsionnelle infinie). Il a montré que l'on pouvait arriver ainsi à un filtre bien meilleur (au sens des critères de Canny). La réponse impulsionnelle du filtre de Deriche est donnée par :

$$f(x) = S\, x\, e^{-\alpha|x|} \tag{4.6}$$

La figure 4.1 représente $f(x)$.

On peut aisément montrer que le produit de convolution y de $f(n)$ par la suite $x(n)$ se calcule avec le système récursif suivant :

$$y^+(m) = x(m-1) - b_1 y^+(m-1) - b_2 y^+(m-2) \quad m = 1, \ldots, M \quad (4.7)$$
$$y^-(m) = x(m+1) - b_1 y^-(m+1) - b_2 y^-(m-2) \quad m = M, \ldots, 1 \quad (4.8)$$

$$y(m) = a\,(y^+(m) - y^-(m)) \quad (4.9)$$

Avec : $a = S\,e^{-\alpha}$, $b_1 = -2e^{-\alpha}$ et $b_2 = e^{-2\alpha}$. Avec la contrainte :

$$\sum_{n=0}^{+\infty} f(n) = -\sum_{n=-\infty}^{0} f(n) = -1 \quad (4.10)$$

on a : $S = -(1 - e^{-\alpha})^2/\alpha$.

Ces équations ont le grand avantage de mener à un nombre de multiplications par points très faible et ne dépendant pas de la largeur α du filtre. Cette propriété est particulièrement utile pour les approches multi-échelles, ou' l'on utilise le même filtre plus ou moins dilaté.

4.2.3 Recherche de textures

Une autre caractéristique locale de bas niveau utilisable pour segmenter une image est la *texture*. La texture d'une image est formée par la répétition (régulière ou aléatoire) de motifs de base [27]. Elle diffère évidemment suivant les régions considérées.

On pourrait par exemple tenter de caractériser les textures observées sur des images satellite de villes puis utiliser le résultat pour détecter les villes présentes dans de nouvelles images.

Pour la recherche de textures par réseaux connexionnistes, citons le travail de J. Loncelle [142].

4.2.4 *Template matching*

Dans les approches «*template matching*», on recherche un objet ou une forme dans l'image en essayant d'ajuster un modèle (template) de cet objet sur l'image.

La version la plus simple consiste à déplacer sur l'image un modèle rigide de l'objet cherché, et à compter en chaque position le nombre de pixels différents. Ce processus revient à effectuer une convolution de l'image par un filtre représentant la forme à détecter. Cette approche est susceptibles de nombreux raffinements [109] que nous ne détaillerons pas.

Une autre approche consiste à ajuster un modèle déformable (ou «élastique») sur l'image. Le modèle est alors décrit par un ensemble de fonctions paramétriques, dont on optimise les paramètres sur l'image (ou l'image de contours). Ce type de technique [111] a été appliqué par de nombreux auteurs pour l'identification de visages [222, 214, 83] (voir chapitre 7).

4.2.5 Approches multirésolution

Supposons que l'on cherche à étudier les objets contenus dans une image de scène. Suivant les conditions de prise de vue et la distnace de l'objet considéré au capteur, la taille apparente de l'objet dans l'image va varier de façon importante. Ceci a deux conséquences immédiates : premièrement, les contours de l'objet vont être plus ou moins nets ; deuxièmement, les distances entre contours vont évidemment varier.

Le problème de la netteté (ou du flou) des contours se pose lorsque l'on utilise une détection par filtrage, comme celle présentée plus haut. La taille optimale du filtre (donnée par α dans l'équation 4.6) doit être adaptée au contour que l'on cherche à détecter. Pour s'affranchir de cette difficulté, on introduit [33, 148, 220, 32] le concept «d'espace d'échelle», dans lequelle le filtrage est précédé par un lissage gaussien de l'image. On peut ensuite suivre les contours à différentes échelles [220].

Le concept d'*analyse multirésolution* permet de résoudre élégamment les difficultés liées aux changements d'échelles. L'analyse multirésolution d'une image consiste à calculer une série de vues de l'image à différentes échelles. Chaque vue permet l'étude des détails caractéristiques à cette échelle. Les décompositions multirésolutions permettent ainsi d'obtenir une interprétation de l'image insensible aux "zooms", qui décalent simplement les vues. Les vues à basses résolutions correspondent aux grandes structures donnant le contexte de l'image. Une approche naturelle en analyse de scène consiste à analyser les vues en allant de l'échelle la plus grossière vers la plus fine (approches dites «coarse to fine») [32, 33].

4.3 Transformation en Ondelettes Discrète

On peut construire une représentation multirésolution en filtrant le signal original avec une série de filtres passe-bas (par exemple gaussiens). Un problème important est alors la *redondance* de la représentation obtenue : il est souhaitable d'extraire de la vue du signal à une certaine échelle les informations ne figurant pas aux échelles plus basses. La théorie des ondelettes permet de construire des filtres tels que les signaux composant la

représentation multirésolution soient interprétables comme les coefficients du développement du signal original sur une base hilbertienne orthonormée. L'orthogonalité et la complétude de la base évitent les redondances et corrélations dans la décomposition. De plus, on peut facilement reconstruire le signal à partir de sa décomposition en ondelettes. Les fonctions de base sont obtenues par dilatation et translation d'une unique fonction, appelée ondelette de base. On choisit généralement cette fonction pour ses propriétés de localisation spatiale et fréquentielle.

La théorie des ondelettes à été introduite par Meyer [155]. Mallat [146, 147] a ensuite proposé une méthode de décomposition discrète très utile en traitement d'image.

4.3.1 Principes

Nous rappelons très rapidement dans cette section les principes des analyses multirésolution par ondelettes. Nous renvoyons le lecteurs aux références suivantes pour des explications plus complètes [155, 147, 206, 61].

L'analyse multirésolution fourni des vues du signal $S(x)$ à différentes échelles. Ces vues sont obtenues par projection du signal sur une séquence de sous-espaces emboités de $L^2(R)$, que l'on notera $(V_j)_{j\in Z}$. Ces sous-espaces constituent les différentes échelles de l'analyse. Dans l'approche ondelettes, on s'intéresse aux sous-espaces engendrés par les dilatations et translations d'une unique fonction Φ :

$$\left(\sqrt{2^{-j}}\,\Phi_j(x-2^{-j}n)\right)_{n\in Z} \text{ est une base orthonormée de } V_j.$$

On a représenté en figure 4.2 une fonction d'échelle Φ typique.

On montre facilement [147] que la projection du signal sur l'espace V_j peut se calculer par un filtrage de la projection sur V_{j+1} suivi d'un sous-échantillonnage du résultat. Cette propriété permet l'utilisation d'algorithmes pyramidaux très efficaces. Le filtre utilisé se déduit de la fonction Φ choisie et est de type passe-bas.

L'information perdue d'une échelle à l'autre peut se calculer en considérant la projection du signal $S(x)$ sur l'espace *inter-échelles* W_j, complément orthogonal de V_j dans V_{j+1} :

$$W_j \perp V_j \quad \text{et} \quad W_j \oplus V_j = V_{j+1}$$

Pour calculer la projection de $S(x)$ sur W_j, il faut trouver une base orthonormée de W_j. Cette base est obtenue par les dilatations et translations d'une unique fonction Ψ :

$$\left(\sqrt{2^{-j}}\,\Psi_j(x-2^{-j}n)\right)_{n\in Z} \text{ est une base orthonormée de } W_j.$$

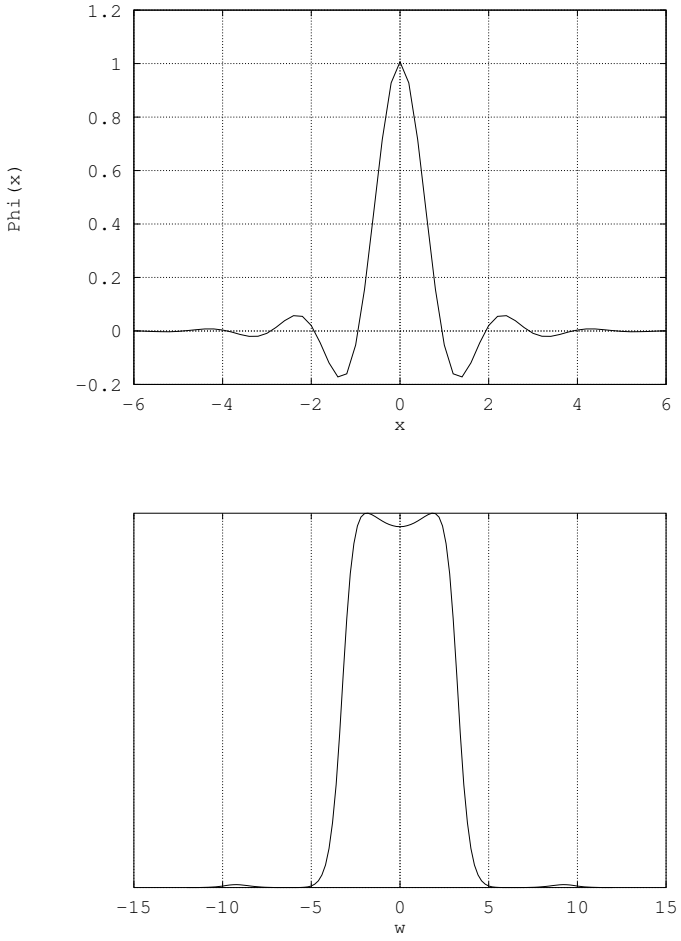


FIGURE 4.2 – Exemple de fonction d'échelle $\Phi(x)$ (en haut) et sa transformée de Fourier (en bas). Φ est un filtre passe bas, construit à partir de splines cubiques.

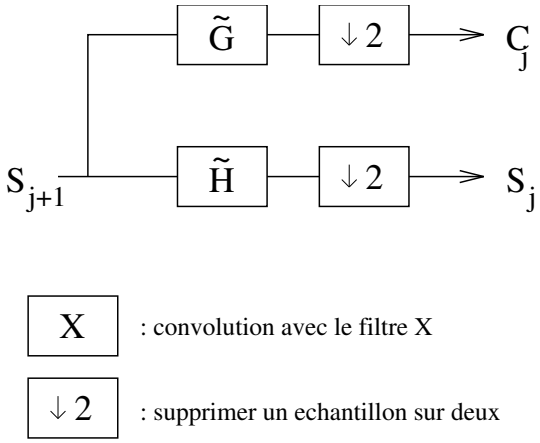


FIGURE 4.3 – Une étape de l'algorithme pyramidal de décomposition en ondelettes de Mallat. Le signal S est le signal original à l'étape 0, ou le résultat du filtrage passe-bas de l'étape précédente. Le filtre $\tilde{H}$ est passe bas, et $\tilde{G}$ passe haut. Ce dernier extrait les détails de S présents à l'échelle $j+1$ mais plus à l'échelle j.

Là aussi, la projection peut se calculer par un filtrage (passe haut) de la projection sur V_{j+1} suivi d'un sous-échantillonnage du résultat.

La décomposition du signal se calcule donc à l'aide d'un algorithme pyramidal très simple, illustré par la figure 4.3.

Le signal original est représenté par l'ensemble de signaux

$$\left\{S_J, (C_j)_{j=-1,\cdots,J}\right\}$$

Cette représentation est complète et permet de reconstruire le signal original.

4.3.2 Extensions aux images

La méthode de décomposition en ondelettes présentée ci dessus s'étend sans peine au signaux bi-dimensionnels. La façon la plus simple, proposée par Mallat [146] consiste à construire les sous-espaces de l'analyse par produits tensoriels des espaces de fonctions uni-dimensionnelles. On obtient alors un facteur de résolution égal à deux.

La figure 4.4 illustre la décomposition d'une image.

Il est souvent nécessaire en traitement d'images de disposer d'une plus grande finesse d'analyse. C'est pourquoi J.C. Feauveau a proposé une méthode avec un facteur de résolution de $\sqrt{2}$ [62].

FIGURE 4.4 – Décomposition en ondelettes d'une image 256x256, avec l'algorithme dyadique de Mallat. L'image originale est en haut. Après une étape, on obtient quatre images : une image lissée (en bas à droite), et trois images résultant de filtrages directionnels passe haut. Le côté de chacune de ces images est égal à la moitié de celui de l'image originale.

4.3.3 Une application : localisation de contours multi-échelle

Nous décrivons dans cette section une application simple des analyses multirésolution par ondelette. Nous avons développé cette application au début de ce travail de thèse, et elle a fait l'objet d'une publication [63].

Le but est d'obtenir les contours d'une image et de caractériser chaque contour par son «flou».

Pour cela, on décompose l'image à traiter en ondelettes. On retrouve les contours de l'image dans le canal «passe-haut» de la décomposition.

Le problème est maintenant de localiser chaque contour dans un seul niveau d'échelle. Chaque contour est en effet détecté, avec des intensités variables, dans toutes les échelles. On désire obtenir une représentation du signal ou‘ chaque contour n'apparaisse qu'une fois, ce qui permettrait de séparer les objets aux contours nets (détectés dans les échelles de haute résolution) des objets flous (détectés à basse résolution).

Pour cela, on utilise un réseau d'automates très simple qui fait évoluer les coefficients d'ondelettes de façon à les concentrer dans chaque niveau. Les états des automates sont initialisés par les coefficients d'ondelettes du signal étudié, puis on laisse évoluer les automates.

Evolution du réseau d'automates

Soit $N_i(t)$ l'état de l'automate i après t itérations (t entier). La dynamique est régie par :

$$N_i(t+1) = N_i(t) + \sum_{j \in \mathcal{V}} \mu_{ij}(t) N_j(t) \tag{4.11}$$

ou‘ $\mathcal{V}$ est un ensemble d'automates constituant le voisinage de i, et $\mu_{ij}(t)$ le poids associé à la connexion i-j.

Le poids μ_{ij} est fonction des états des deux automates en présence : $\mu_{ij} = f(N_i, N_j)$.

On désire localiser les coefficients d'ondelette, mais conserver la norme L^2 (énergie) de la représentation. La variation d'énergie lors d'une itération est égale à

$$\Delta E = \sum_i N_i^2(t+1) - \sum_i N_i^2(t) = 2 \sum_{i,j} \mu_{ij} N_i N_j + \sum_i \left(\sum_j \mu_{ij} N_i \right)^2 \tag{4.12}$$

Le premier terme s'annule si $\mu_{ij} = -\mu_{ji}$, et le second est négligeable si $\mu_{ij} \ll 1$.

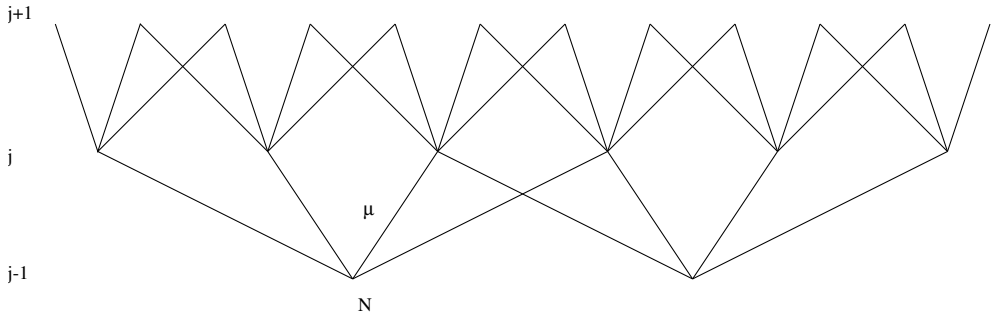


FIGURE 4.5 – Architecture du réseau d'automates pour la localisation de contours.

La fonction $f(x, y)$ doit donc être anti-symétrique. D'autre part, afin de concentrer les valeurs, on désire augmenter les forts coefficients et atténuer les faibles coefficients. Pour cela, on utilise la forme suivante :

$$\mu_{ij} = \lambda(N_i - N_j)N_iN_j \tag{4.13}$$

Il est aisé de constater que les états ou‘ seuls quelques coefficients sont non nuls sont des états stables. Par contre, il est difficile de prouver que l'on atteint toujours un état stable, comme c'est le cas en pratique.

Cas uni-dimensionnel

Nous avons appliqué ce système à des signaux uni-dimensionnels.

Il reste à fixer la forme du voisinage $\mathcal{V}$. On désire préserver la localisation spatiale des singularités. Chaque automate est donc relié uniquement à ses proches voisins dans les échelles adjacentes, comme représenté sur la figure 4.5.

Les figures 4.6 et 4.7 illustrent le fonctionnement du système. Sur la figure 4.6, on voit les valeurs initiales des automates, égales aux valeurs absolues des coefficients d'ondelettes du signal. La figure 4.7 donne l'état de l'automate après 200 itérations.

Remarques

Le système présenté ci-dessus souffre de plusieurs défauts liés à l'utilisation de la décomposition en ondelettes dyadiques (i.e. avec sous-échantillonage d'un facteur 2 entre chaque échelle). Cet algorithme est en effet sensible aux bruits, et surtout n'est pas invariant par translation : lorsque le signal original est translaté, les coefficients d'ondelette varient de façon complexe.

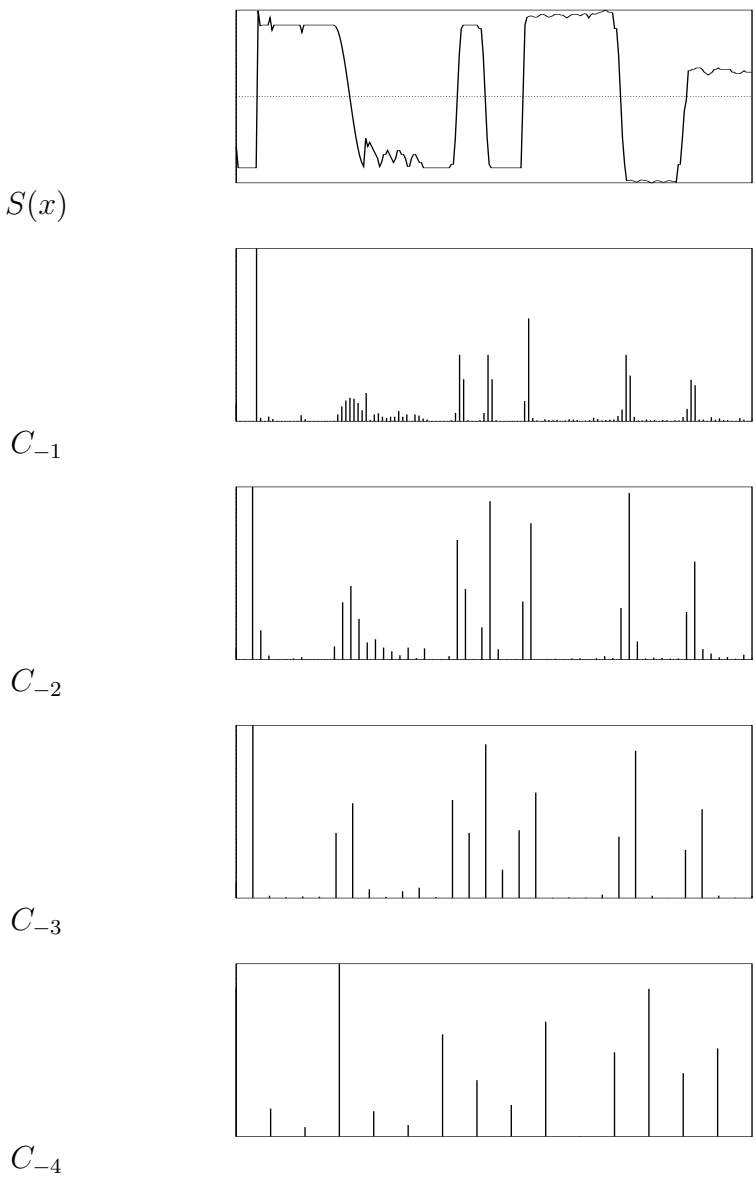


FIGURE 4.6 – Valeurs absolues des coefficients d'ondelettes. Le signal original est représenté en haut.

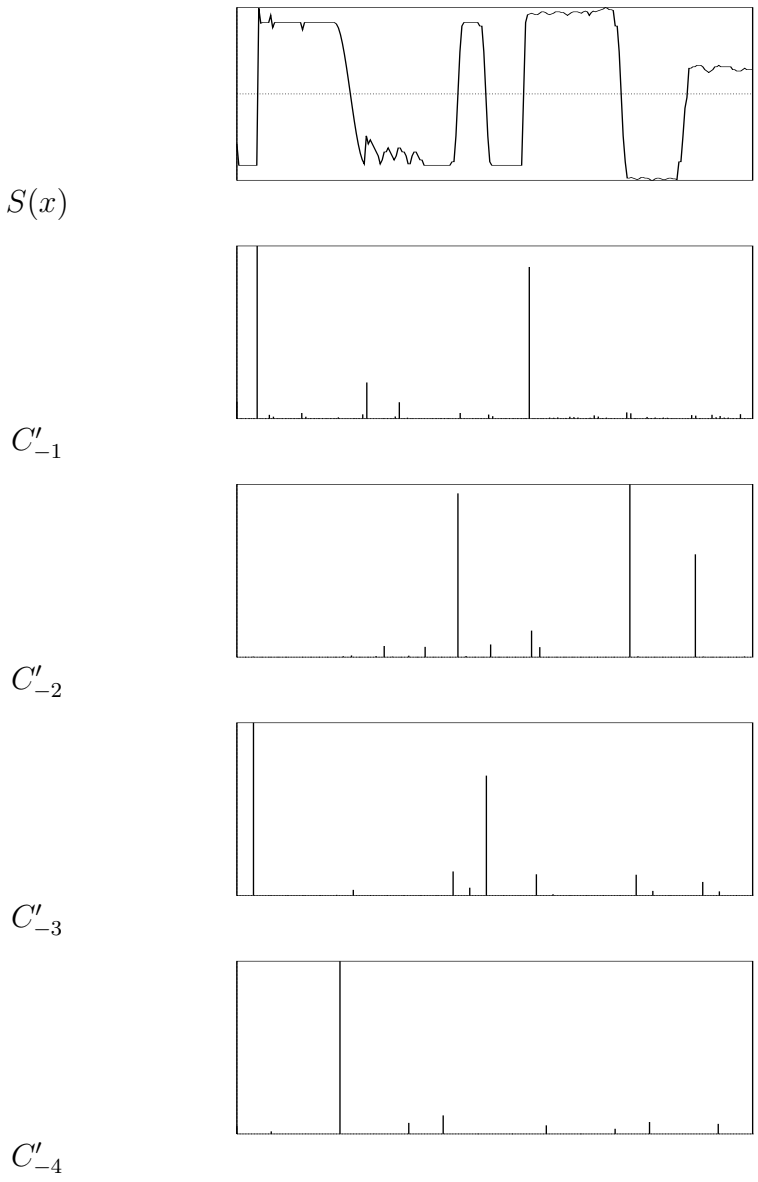


FIGURE 4.7 – Etat du réseau après 200 itérations. On voit que les valeurs se sont concentrées et permettent un repérage précis des singularités du signal original, qui est reproduit en haut. Les transitions lentes sont localisées aux niveaux inférieurs, tandis que les plus abruptes le sont aux niveaux supérieurs.

Ce phénomène n'est pas très gênant pour les signaux uni-dimensionnels, mais rend impossible la généralisation de cette méthode au traitement d'images. Une solution est d'utiliser une décomposition en ondelettes *redondante* (le même algorithme sans sous-échantillonnage à chaque niveau), qui donne des résultats acceptables au prix de temps de calculs prohibitifs. Une autre approche consisterait à utiliser un sous-échantillonnage *adaptatif* [141].

4.4 Conclusion

Les techniques de segmentation d'image évoquées dans ce chapitre sont basées sur une approche «bas niveau». Dans ce type d'approches, on cherche à utiliser des caractéristiques de l'image pour découper des régions, mais on n'intègre pas (excepté dans certaines techniques de *template matching*) de connaissance de plus haut niveau. Il est par conséquent très délicat d'intégrer dans un même système la segmentation et la reconnaissance d'objets.

Prenons l'exemple de la recherche de visages dans une image. Un détecteur de contour pourra nous fournir une carte des contour de l'image. On pourra ensuite rechercher dans cette carte les sous-ensembles de contours pouvant correspondre à des visages. Dans une telle approche, il sera cependant toujours impossible de garantir l'optimalité du module de détection de contours pour la tâche globale de localisation de visages.

Comme on l'a vu dans le chapitre précédent, les modèles connexionnistes permettent d'envisager une approche ou l'on optimise simultanément l'ensemble des modules d'un système. Dans le chapitre suivant, nous allons décrire plusieurs exemples de systèmes intégrant la segmentation et la reconnaissance de chiffres manuscrits. Nous reviendrons sur la segmentation de visages dans les chapitres 8 et 9.

Chapitre 5

Etat de l'art en OCR manuscrit

5.1 Introduction

La reconnaissance de chiffres manuscrits [1] est un problème très attrayant car d'abord aisé, d'application pratique immédiate, et posant néanmoins de très intéressants problèmes. L'importance des applications pratiques (lecture automatique de codes postaux, de chèques banquaires, de formulaires, ...) attire de nombreuses équipes de recherches et stimule le développement de nouveaux algorithmes. Les performances des systèmes de reconnaissance de chiffres ont ainsi considérablement augmenté ces dernières années, pour arriver aujourd'hui à un niveau frôlant celui de l'être humain placé dans les mêmes conditions (caractères isolés sans utilisation de contexte).

Ce chapitre présente l'état de l'art dans le domaine de la reconnaissance de chiffres manuscrits imprimés.

Cependant, les travaux dans ce domaine étant très nombreux, nous nous sommes limités à décrire quelques travaux, pour certains très récents, qui nous semblent illustrer des démarches fructueuses dans ce domaine.

La section suivante rappelle quelques généralités sur la reconnaissance automatique de l'écriture.

Nous passons ensuite en revue différents travaux portant sur la reconnaissance de chiffres isolés.

La dernière section décrit enfin plusieurs systèmes intégrant la segmentation des chiffres et leur reconnaissance (systèmes ISR, pour *Integrated Segmentation-Recognition*) proposés récemment dans la littérature.

1. On écrit souvent «OHCR», pour *Optical Handwritten Character Recognition.*

5.2 Généralités sur la reconnaissance du manuscrit

Un des buts principaux de l'intelligence artificielle est de rendre les machines capables de tâches naturelles pour les humains. Les ordinateurs devraient ainsi être capable d'interagir avec les humains de la façon la plus simple possible. Ce but est reflété dans la tendance actuelle de l'industrie informatique visant à rendre les ordinateurs «conviviaux». L'écriture manuscrite est un moyen de communication utilisé par tous et appris dès le plus jeune age. L'écriture est ainsi candidat naturel comme moyen de communication homme-machine, ne nécessitant aucun apprentissage particulier de la part de l'homme.

D'autre part, des systèmes fiables de lecture de l'écriture manuscrite seraient utiles dans maintes applications informatiques «classiques» : informatisation de fichiers (reprise d'anciennes bases de données sur papier), traitement automatique de documents, formulaires, etc...

Il est surprenant de constater combien peu d'attention a été porté, jusqu'a une date relativement récente, au problème de la reconnaissance automatique d'écriture. La compréhension de la parole a en comparaison mobilisé plus d'efforts de la part des chercheurs en intelligence artificielle.

5.2.1 Différents types d'applications et de problèmes

Nous allons diviser grossièrement les problèmes liés au traitement automatique de l'écriture manuscrite, afin d'identifier les axes de recherches principaux dans le domaine (voir figure 5.2).

Traitement «On-line» ou «off-line»

On peut d'abord distinguer entre les traitements *on-line*, c'est à dire en direct, et ceux *off-line*. Les systèmes «on-line» utilisent un capteur spécifique (tablette graphique, stylo electronique, voire gant spécial) pour acquérir des informations sur la dynamique de l'écriture. L'écriture est ici codée comme une trajectoire (de points ou de traits) $(x(t),\ y(t))$. Les informations de levée et descente du stylo facilitent grandement la segmentation. L'avènement des ordinateurs tablettes portables rend urgent la mise au point de modules fiables de lecture on-line.

Dans l'approche «off-line», on travaille sur l'image, en général fournie par un scanner. Cette approche est adaptée au traitement de documents existants sur papier. L'information dynamique étant perdue, il est souvent difficile de segmenter les mots en caractères, les traits de stylo se recouvrant souvent.

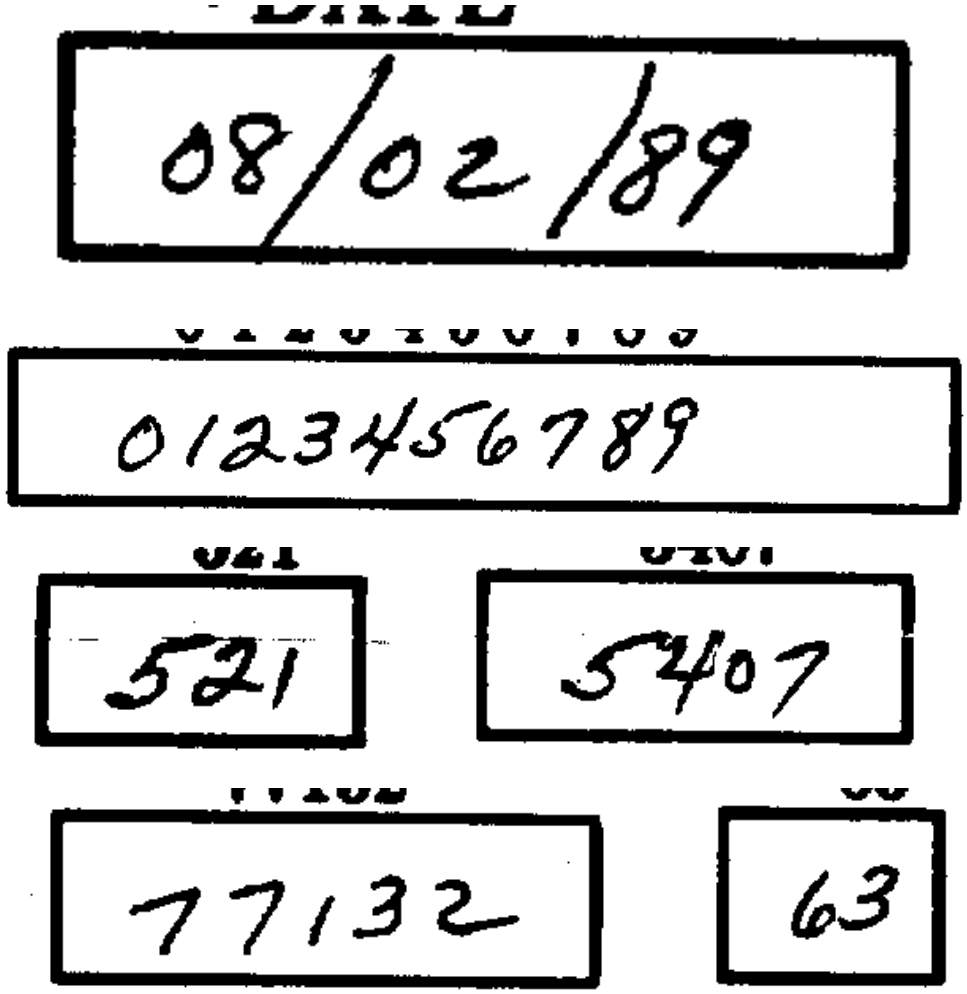


FIGURE 5.1 –]
Exemples de chiffres manuscrits américains, extraits de la base de données NIST.

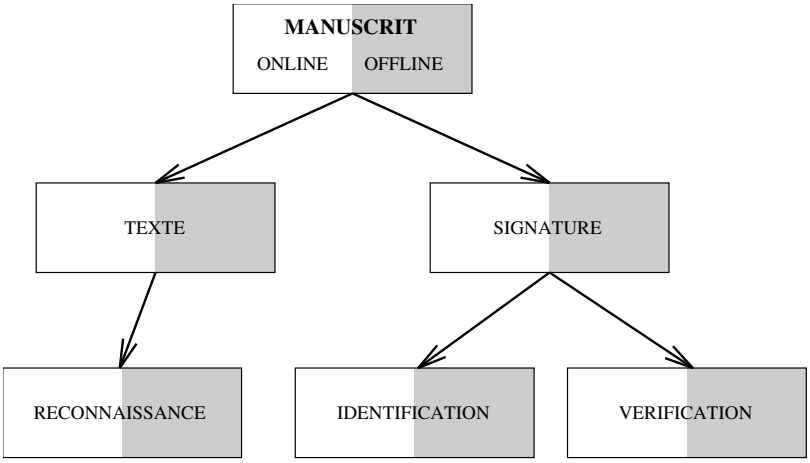


FIGURE 5.2 – Subdivisions en reconnaissance d'écriture manuscrite.

Lecture du texte ou Identification du scripteur

La finalité du système introduit une deuxième distinction, orthogonale à la première, distinguant les méthodes visant à reconnaitre l'auteur du texte (scripteur) de celles visant à lire les caractères. Les systèmes d'identification du scripteur utilisent en général les signatures. On distingue encore la vérification (ou authentification) de signature de la reconnaissance ou identification.

En authentification, on fournit simultanément au système une identité et une signature. Le système doit accepter ou refuser l'identité proclamée. Ce problème est très difficile, car on demande des taux de fiabilité très élevés.

En mode identification, le système doit proposer une identité, ce qui peut s'avérer très difficile si l'on à a connaître plusieurs milliers de scripteurs (clients d'une banque par exemple).

5.2.2 Conclusion

Dans la suite de cette thèse, nous ne nous intéresserons uniquement à l'approche *off-line* pour la reconnaissance de chiffres manuscrits : c'est le cas ou' l'on dispose d'une image scannée de chiffres ayant été écrits à la main.

Le lecteur intéressé par une bibliographie plus complete sur le traitement de l'écriture manuscrite consultera avec profit les référérences suivantes : [106], [105, numéro spécial d'IEEE sur l'OCR], [190, présentation d'un système de segmentation avec chaînes de Markov, et liste importante de références], et [184].

5.3 Reconnaissance de chiffres isolés

Le premier étage d'un module de reconnaissance de chiffres consiste classiquement en une extraction de *caractéristiques*. L'extraction de caractéristiques a pour but d'augmenter la tolérance aux bruits et aux déformations (petites translations, etc) de l'image, et de réduire la dimension des données à classer. Les caractéristiques sont ensuite utilisées par un *classifieur*, qui détermine la classe du caractère.

Ces deux étapes sont en général distinctes, et sont optimisées séparément. Un des avantages majeurs des systèmes connexionnistes est qu'ils permettent de traiter l'extraction de caractéristiques et la classification en un seul module : l'apprentissage porte simultanément sur le classifieur et l'extracteur de caractéristiques, et l'on présente directement au système une image du caractère à reconnaître.

Les travaux dans ce domaine étant très nombreux, nous avons dû sélectionner quelques exemples nous semblant pertinents quant à l'évolution récente du domaine.

Nous avons ainsi retenu un extracteur de caractéristiques «typique», proposé par Burel et al. [30], car il illustre bien l'approche consistant à extraire des caractéristiques pré-déterminées de l'image avant de procéder à la classification. Ce système est simple et néamoins relativement performant.

Nous présentons ensuite le réseau TDNN «LeNet» proposé par Y. Le Cun à AT&T [129]. C'est le réseau le plus populaire en reconnaissance de chiffres, et il a été utilisé depuis dans plusieurs applications industrielles de par le monde.

Nous décrivons ensuite deux approches très récentes, basées sur de nouveaux principes et améliorant très sensiblement les performances. Il s'agit d'une part de l'utilisation d'une nouvelle mesure de similarité (*Tangent Distance*) pour la classification [192], et d'autre part d'un nouvel algorithme de classification (*Local Classifier*) [17].

Après avoir présenté ces systèmes, nous évoquerons les résultats d'une étude récente menée par le bureau des standards américain (NIST) pour comparer les meilleurs systèmes du moment.

5.3.1 Exemple d'extracteur de caractéristiques

Nous avons choisi de décrire le système proposé par Burel et al. [30] pour illustrer l'approche utilisant deux niveaux distincts (extraction de caractéristiques puis classification) pour la reconnaissance de chiffres.

L'extracteur de caractéristiques proposé par ces auteurs utilise des algorithmes de traitement d'image pour calculer trois types de caractéris-

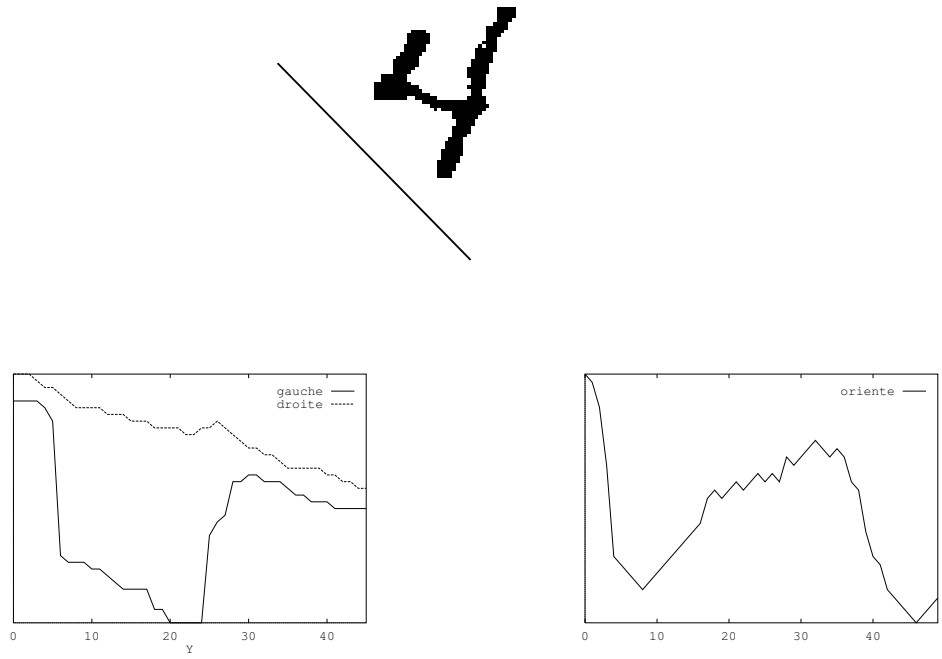


FIGURE 5.3 – Les caractéristiques métriques utilisées par Burel et al. : les profils gauches et droits sont les distance de la gauche de l'image au coté gauche (resp. droit) du caractère. Le profil orienté est la distance mesuré perpendiculairement à un axe orienté à 45 degrés (tracé sous le chiffre).

tiques, nommées *métriques*, *statistiques* et *morphologiques*. Ces algorithmes travaillent directement sur l'image (bitmap) du caractère.

Caractéristiques métriques Ce groupe de caractéristiques est constitué par les profils gauche, droite et orienté selon un axe à 45 degrés (voir figure 5.3). Les profils latéraux sont normalisés sur 32 points, et le profil orienté sur 16 points.

On obtient ainsi un vecteur de 80 valeurs réelles.

Caractéristiques statistiques Ce second groupe apporte de l'information globale sur le caractère. Pour cela, on mesure la surface relative du caractère dans plusieurs régions (voir figure 5.4). Ces mesures donnent un vecteur de 20 valeurs.

Caractéristiques morphologiques Le dernier groupe de caractéristique est constitué d'informations sur la forme du caractère, obtenue en recherchant et comptant les *cavitées*. On définit plusieurs types de ca-

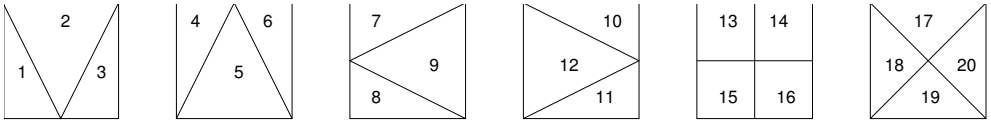


FIGURE 5.4 – Les 20 regions utilisées pour mesurer les caractéristiques sont définies par 6 masques couvrants le caractère. On mesure la surface occupée par le caractére dans chaque région.

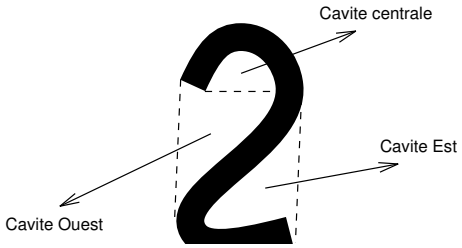


FIGURE 5.5 – Quelques types de cavités.

vités (voir figure 5.5), de différentes orientations. Le codage de ces caractéristiques est effectué sur 12 cellules.

L'ensemble des caractéristiques est regroupé dans un vecteur de dimension 112.

La classification de ce vecteur est effectuée à l'aide d'un MLP. Les auteurs ont comparé différents types de réseaux multi-couches avec un classifieur plus proches voisins (kNN), sur une base de données de 1500 chiffres (16 scripteurs). Ils concluent à une nette supériorité des réseaux à poids partagés. Par contre, les réseaux totalement connectés arrivent à de moins bons résultats que le classifieur kNN.

Du fait de la dimension relativement faible de l'espace de caractéristiques (112), le MLP utilisé est petit et rapide : environ 2000 connexions. Cependant, l'extraction des caractéristiques nécessite l'implémentation d'algorithmes relativement complexes, non parallélisables.

Les performances présentées par les auteurs semblent inférieures à celles obtenues par les systèmes TDNN que nous allons présenter plus bas : 6.4 % d'erreur sur une base de 1000 chiffres manuscrits (14 scripteurs). Les résultats des différents systèmes présentés dans ce chapitre sont regroupés dans la table 5.1 (page 120). Le jeu de données utilisé n'est cependant pas le même, aussi la comparaison est-elle délicate.

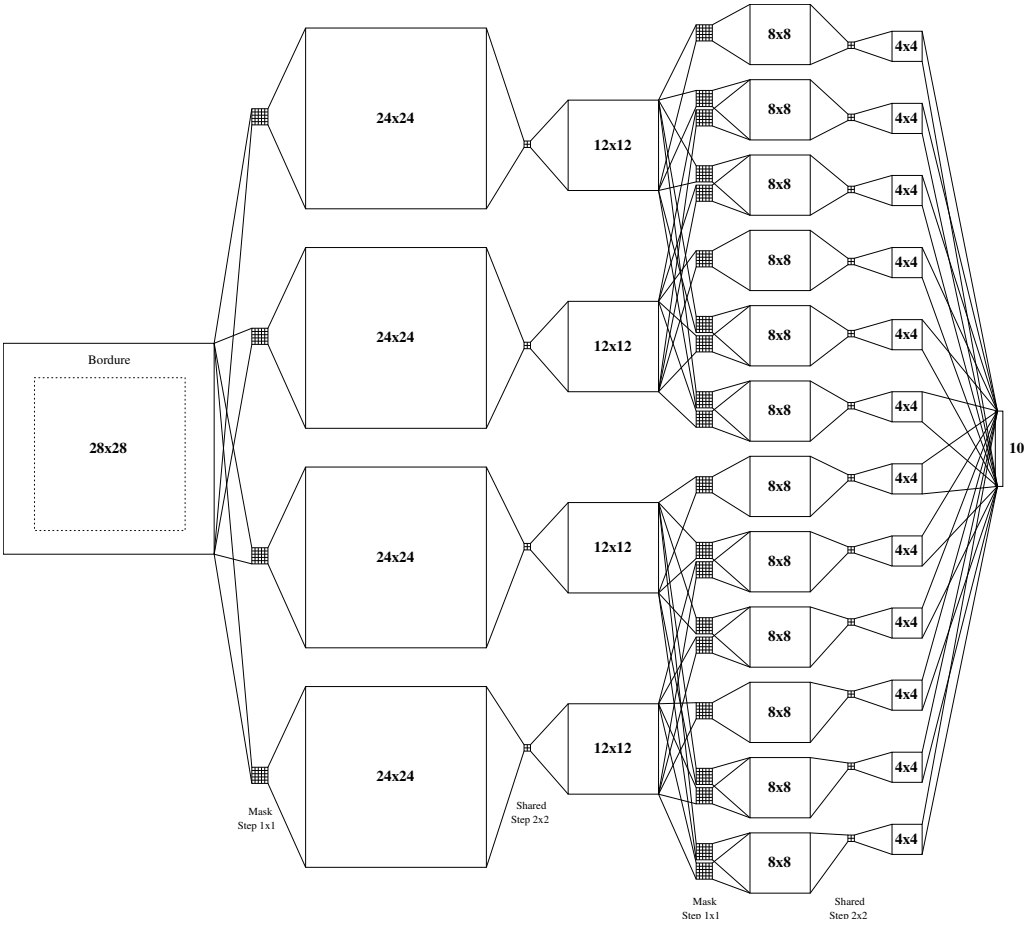


FIGURE 5.6 – Architecture du réseau LeNet.

5.3.2 Approche TDNN : LeNet

L'approche utilisant un seul réseau de neurones pour extraire et classer des caractéristiques est parfaitement illustrée par le système proposé en 1989 par Y. Le Cun [129, 132]. Le réseau proposé, de type TDNN, est devenu célèbre sous le nom *LeNet*. C'est en effet le premier système connexionniste arrivant à des performances égalant, voire dépassant, les meilleurs systèmes de reconnaissance de chiffres.

Architecture L'architecture de LeNet est représentée schématiquement sur la figure 5.6. Le chiffre à reconnaître, normalisé dans un carré de taille 16x16, est présenté centré sur la rétine (première couche). La rétine est en fait de taille 28x28, mais seules les cellules centrales (carré 16x16) sont utilisées :

les cellules extérieures forment une *bordure*. L'emploi de cette bordure évite des effets de bords gênants lorsque l'on convolue la rétine avec les masques de poids de la première couche.

La première couche cachée est constituée de quatre groupes de cellules, connectés à la rétine par des masques de poids partagés de taille 5x5. La deuxième couche effectue un lissage et un sous-échantillonnage des caractéristiques extraites sur la couche précédente. Ce lissage permet d'augmenter la résistance du réseau aux translations de l'image.

Les deux couches suivantes continuent le travail sur le même principe : extraction de caractéristiques par des masques de poids partagés suivie par un sous-échantillonnage.

Enfin, l'avant dernière couche, de taille $192 = 12 \times 16$, est totalement connectée aux 10 cellules de sortie (une par classe de chiffres). Cette couche de poids effectue une classification des caractéristiques extraites. Cette classification est quasi-linéaire.

Au total, LeNet comporte 4 634 cellules, 98 442 connexions et 2 578 paramètres libres.

On peut considérer ce réseau comme un extracteur de caratéristiques, fournissant un codage sur 192 valeurs.

Performances Les performances du réseau LeNet sont excellentes. Les résultats publiés dans [129] portent sur la base de chiffres (manuscrits et imprimés mélangés) constituée par les postes américaines (USPS) ; on a environ 10 000 chiffres pour l'apprentissage et 2 700 pour le test. Le taux d'erreur mesuré sur l'ensemble de test est de 3.4 % sans rejet. Le taux de rejet à 1 % d'erreur est de 5.7 %. Si l'on ne test que sur les chiffres manuscrits, on obtient 5.1 % d'erreur (voir table 5.1).

De nombreuses publications [128, 130] établissent la supériorité de cette architecture pour la classification de chiffres manuscrits. Elle a été mise au point après de nombreux tests et comparaisons.

La supériorité de cette architecture repose sur l'utilisation de *contraintes* sur les poids : les masques de poids partagés permettent d'une part d'implémenter un filtrage de l'image invariant par translation, et d'autre part de réduire le nombre de degrés de liberté du classifieur.

La conception de l'architecture LeNet a été guidée par l'utilisation de la technique d'*Optimal Brain Damage* [131]. Dans cette technique, on mesure «l'utilité» des connexions à l'aide d'une approximation du hessien. La suppression des connexions les moins utiles ne diminue pas les performances et permet souvent une meilleure généralisation après ré-apprentissage. En partant d'un réseau trop grand, on peut ainsi réduire sa taille (supprimer des

groupes de cellules) pour arriver à une architecture optimale.

5.3.3 Ajout de contraintes : *Tangent Prop*

Simard et al. [193] ont proposé d'ajouter de nouvelles contraintes lors de l'apprentissage d'un réseau TDNN, toujours afin d'augmenter la généralisation. L'idée est d'utiliser le fait que la classification d'un chiffre devrait rester invariante lorsque ce chiffre est légèrement transformé pour guider la descente du gradient dans la «bonne direction». Cette méthode a été baptisée *tangent prop.*

Si x est la forme à classer et s un opérateur continu différentiable de paramètre α (par exemple une rotation d'angle α), on désire que la sortie $G(x)$ du classifieur varie le moins possible quand on applique s à x, avec α petit. Comme on souhaite être invariant *localement*, pour les petites valeurs de α, on va approximer $s(\alpha, x)$ au premier ordre

$$s(\alpha, x) - s(0, x) \approx \alpha \left(\frac{\partial s(\alpha, x)}{\partial \alpha} \right)_{\alpha=0} \tag{5.1}$$

(En général on ne désire pas d'invariance pour les transformations plus importantes : si l'on autorisait toutes les rotations, les lettres «u» et «n» seraient indistinguables ; mais l'approximation faite ici a surtout pour but d'arriver à un résultat implémentable.)

Pour incorporer ces invariances au classifieur, on ajoute un terme de *régularisation* à la fonction de coût optimisée durant l'apprentissage. La règle de calcul des poids

$$\Delta W = -\eta \frac{\partial E}{\partial w}$$

est alors remplacée par

$$\Delta W = -\eta \frac{\partial}{\partial w}(E + \mu E_r)$$

Le terme E_r va mesurer l'accroissement de la sortie G quand on transforme l'entrée x par s

$$E_r(x) = \left\| \left(\frac{\partial G(s(\alpha, x))}{\partial \alpha} \right)_{\alpha=0} \right\|^2 \tag{5.2}$$

Cette expression se calcule facilement, si on écrit

$$\left(\frac{\partial G(s(\alpha, x))}{\partial \alpha} \right)_{\alpha=0} = G'(x) \cdot \left(\frac{\partial s(\alpha, x)}{\partial \alpha} \right)_{\alpha=0}$$

Le terme $\partial s(\alpha, x)/\partial\alpha$ se calcule simplement grâce à l'equation (5.1) ; $G'(x)$ est le Jacobien de G par rapport à x. Simard et al. décrivent dans [193] un algorithme «connexionniste» (*réseau Jacobien*) pour calculer le Jacobien durant la rétro-propagation du gradient.

Cette méthode se généralise évidemment à des contraintes plus complexes : on peut imposer une valeur quelconque aux dérivées dans (5.2) ; on peut aussi introduire un nombre quelconque de transformations $s_i(\alpha, x)$. On veillera alors à choisir un générateur du groupe de transformations désiré ; par exemple si l'on vise une invariance aux translations de l'image x, il suffira d'utiliser les translations horizontales et verticales.

Le problème de cette méthode est le réglage du paramètre de régularisation μ, qui dose l'importance relative de l'erreur de classification et de l'invariance aux transformations. Il est toutefois plus simple d'agir sur ce paramètre que de générer une base d'exemples d'apprentissage représentative de toutes les transformations géométriques auquelles on veut rendre le classifieur insensible.

Simard et al. ont testé avec succès cette approche sur de petits problèmes de classification de chiffres, et montré que les performance en généralisation se trouvaient nettement améliorées par rapport à un apprentissage «classique», surtout lorsque peu d'exemples sont disponibles. Il n'est pas encore démontré que *tangent prop* arrive à améliorer les performances d'un réseau TDNN déjà très contraint comme LeNet sur d'importantes bases de données de chiffres.

5.3.4 Classifieur utilisant une *distance tangente*

Les classifieurs utilisant directement une distance entre les formes tels que kNN ou RBF reposent le plus souvent sur la distance euclidienne ou une autre distance simple. Hormis leurs agréables propriétés analytiques, ce type de distance n'est pas particulièrement adapté au problème à traiter. D'ailleurs, pour l'usage habituel, on n'a pas besoin des propriétés de *distance* rigoureuses : une simple *mesure de similarité* entre les formes suffit.

Une nouvelle mesure de similarité, nommée *distance tangente*, a très récemment été définie par Simard, Le Cun et Denker [192] et appliquée à la reconnaissance de chiffres. Cette mesure de similarité reprend les idées de *tangent prop* exposées plus haut (5.3.3) : il s'agit d'exprimer une invariance de la distance quand on applique une transformation infinitésimale à l'une des formes. Parmi les transformations pertinentes pour l'OCR manuscrit, on trouve les translations de la rétine, les rotations, les affinités, et aussi des transformations non linéaires comme la dilatation de l'épaisseur du trait de stylo, etc.

La forme bi-dimensionnelle x est vue comme un point dans l'espace de

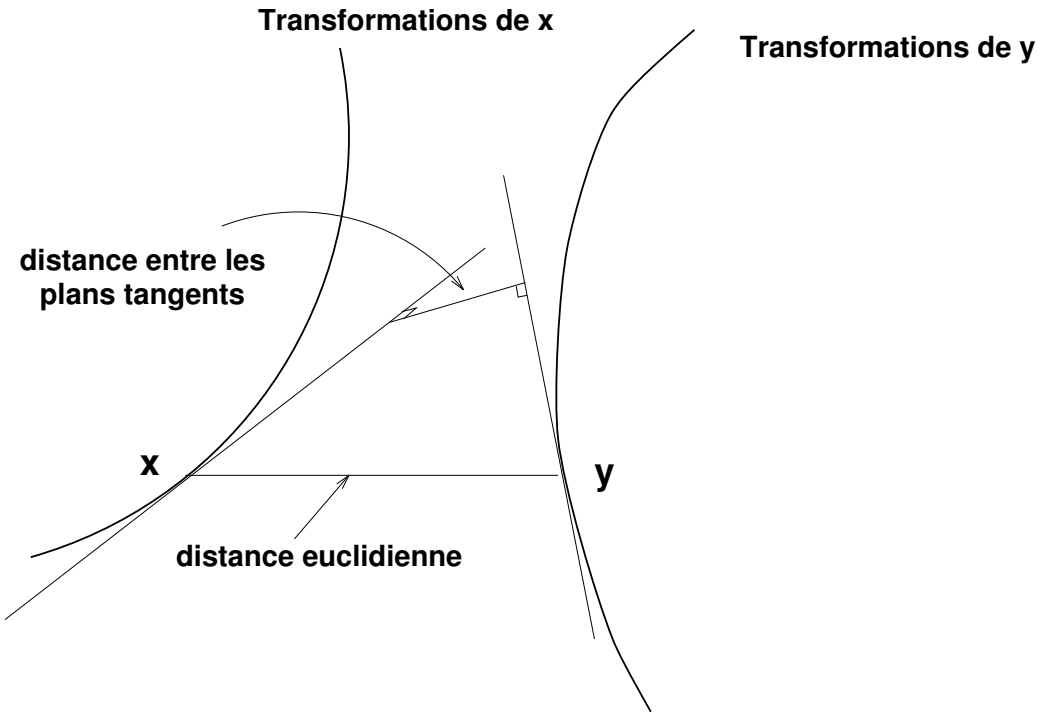


FIGURE 5.7 – Distance tangente entre les formes x et y.

configurations, typiquement de dimension 16x16. Si l'on applique une rotation d'angle α à l'image x, on génère une courbe dans l'espace de configurations, paramétrée par α. Plus généralement, la combinaison de m transformations va générer une *variété* de dimension m dans l'espace, que l'on notera S_x.

Si l'on désire s'affranchir complètement de l'influence des m transformations lors du calcul de la distance entre x et y, on peut calculer la distance minimale entre les variétés S_x et S_y : si la forme y s'obtient en transformant x, les deux variétés sont identiques. Toutefois, on a vu plus haut que l'on recherche généralement une invariance locale : les transformations plus importantes peuvent changer la classe de la forme traitée.

Dans ce cas, on peut approximer la variété S_x au premier ordre par son plan tangent en x. Le plan tangent est donné l'équation de paramètre $\vec{a}_x$

$$x'(\vec{a}_x) = x - L_x\,\vec{a}_x \tag{5.3}$$

ou' L_x est une matrice dont les colonnes sont les *vecteurs tangents* , $\partial s(\alpha, x)/\partial\alpha$, qui engendrent le plan tangent en x à la variété.

On définit alors la *distance tangente* entre x et y, $D(x, y)$, comme la

distance entre les deux plans tangents (voir figure 5.7)

$$D(x,y) = \min_{\vec{a}_x,\vec{a}_y} \|x'(\vec{a}_x) - y'(\vec{a}_y)\|^2 \tag{5.4}$$

Le calcul de $D(x,y)$ est ramené à un problème linéaire, qui se calcule rapidement. La partie la plus complexe est le calcul des vecteurs tangents ; pour les tranformations courantes, ce calcul peut s'implémenter simplement par des traitements d'image (e.g. convolutions) élémentaires.

On peut ensuite utiliser simplement la distance tangente dans un classifeur de type kNN. Le problème est alors le temps de calcul, que l'on réduit en utilisant un *clustering* ou une édition [59] de la base de données d'apprentissage, afin de réduire le nombre de prototypes.

Simard et al. ont obtenu par cette méthode une division de l'erreur par deux par rapport à LeNet (table 5.1), sur les bases de chiffres américains USPS (mentionnée plus haut) et NIST (voir plus loin). Le temps de calcul serait d'environ deux secondes par caractère sur une station de travail Sun (contre moins de 0.1 secondes pour le TDNN LeNet).

5.3.5 Le classifieur *Local* de Bottou et Vapnik

Bottou et Vapnik [17] se sont intéressés à la mise au point d'un nouveau type de classifieur, qualifié de *local* . Dans le cas des chiffres manuscrits, ce classifieur travaille sur les caractéristiques extraites par l'avant-dernière couche du TDNN LeNet.

L'idée est de contrôler localement la *capacité* [205] du classifieur, afin de tracer des frontières de décision plus ou moins précises suivant la densité d'exemples dans le voisinage. On pourra ainsi classer finement les formes ambiguës, habituellement dans des zones ou‘ de nombreux exemples sont disponibles, et rejeter les formes éloignées de tout.

La méthode proposée repose sur l'apprentissage d'un classifieur linéaire sur les exemples sélectionnés dans le voisinage de la forme à classer : c'est ainsi une variante des classifieurs kNN. L'algorithme «local» s'exprime donc comme suit, pour chaque nouvelle forme x à reconnaître :

1. rechercher k exemples de l'ensemble d'apprentissage dans le voisinage de x (par exemple les k plus proches voisins) ;
2. entraîner un réseau (à une couche, avec un fort *decay*) sur ces exemples ;
3. utiliser ce réseau pour classer la forme x.

Cet algorithme est présenté par Bottou et al. comme une illustration simple (et relativement inefficace) d'une démarche plus générale visant à

Méthode	Données	Erreur
Burel (§5.3.1)	1000 chiffres	6.4 %
LeNet (§5.3.2)	USPS (12 000 chiffres)	5.1 %
Dist. Tangente (§5.3.4)	idem	2.6 %
LeNet + Local (§5.3.5)	idem	3.3 %
Humain (américain moyen)	idem	≈ 2.5 %

TABLE 5.1 – Récapitulation des performances des systèmes de reconnaissance de chiffres isolés présentés dans ce chapitre. Les performances humaines ont été estimées par sondage à AT&T.

ajouter un paramètre de contrôle de la *localité* d'un classifieur (ici le nombre de voisins k), en plus du terme traditionnel de *decay*.

Les résultats présentés dans [17] sur les chiffres manuscrits de la base USPS indiquent une réduction très significative de l'erreur par rapport à LeNet. Les performances restent cependant inférieures de 0.7 % à celles du classifieur à «distance tangente» présenté plus haut (5.3.4). De plus, le temps de classification atteint 50 secondes par chiffre.

5.3.6 Etat de l'art : concours NIST

En 1992, l'Institut National des Standards et de la Technologie (NIST) américain a été chargé par l'office du recensement (Bureau of Census) d'évaluer la possibilité d'utiliser des systèmes de traitement automatique de documents.

L'office du recensement est chargé du recensement de la population aux Etats Unis. Tous les dix ans, 100 millions de formulaires sont dépouillés en quelques mois. L'office réalise aussi de nombreuses autres enquêtes sur divers secteurs de l'économie.

Pour évaluer les performances de la technologie actuelle, NIST a organisé un concours, ouvert à toutes les universités et entreprises désirant participer, visant à départager et comparer les meilleurs systèmes actuels de reconnaissance de chiffres manuscrits. Cette compétition a été clôturée par une conférence réunissant les participants.

Au préalable, NIST a constitué une base de données [77]. Cette base de données est distribuée sur disque optique (CD-ROM) et contient des images de formulaires (champs numériques et alphanumériques). Les images sont étiquetées, et fournies en version originale ou segmentée. Le premier concours utilisait uniquement les caractères segmentés.

Une deuxième base de données, contenant environ 100 000 chiffres, a été

Institution	Extracteur de caractéristiques	Classifieur	Erreur (chiffres)
OCR Systems	?	?	2.6
AT&T Bell Labs	aucun	kNN avec Distance Tangente	3.2
AEG Electrocom	prétraitements + ACP	classifieur polynomial	3.4
ELSAG-BAILEY	?	RN	3.4
IBM	morphologique, contours, ...	MLP	3.5
AT&T Bell Labs	TDNN (LeNet)	TDNN (LeNet)	3.7

TABLE 5.2 – Les meilleurs systèmes de reconnaissance de chiffres du concours NIST. La colonne de droite donne le pourcentage d'erreur sur les chiffres de la base de test. Les points d'interrogations indiquent que la méthode employée n'a pas été communiquée. Les systèmes d'ELSAG-BAILEY et IBM ressemblent probablement à celui que nous avons décrit en section 5.3.1 (RN désigne un algorithme «neuronal» non précisé).

utilisée par NIST pour mesurer les performances des systèmes participants. Cette base de test ne contient pas les mêmes scripteurs que la base d'apprentissage, et a été acquise dans des conditions très différentes.

Les résultats des meilleurs participants au concours sont résumés dans le tableau 5.2.

NIST a déclaré les performances présentées parfaitement suffisantes pour l'application de l'office du recensement. Il a été décidé de continuer l'étude sur les champs de chiffres non segmentés.

Conclusions Il est important de noter que les participants avaient la possibilité d'utiliser d'autres bases de chiffres que celle fournie par NIST pour entraîner leurs systèmes. Le meilleur participant, OCR Systems, a utilisé ses propres bases de données, contenant plus de 200 millions de caractères, et n'a pas donné d'indications sur la méthode utilisée. La taille des bases de données disponibles pour l'apprentissage est un facteur clé pour les performances de généralisation.

Les systèmes utilisant une extraction de traits adaptative (i.e. les réseaux de neurones) ont été pénalisés par les différences importantes entre les chiffres de la base d'apprentissage et ceux de la base de test.

La plupart des participants ont utilisés des pré-traitements non commu-

niqués (redressement des caractères, normalisation de l'épaisseur du trait, ...) qui influent apparemment beaucoup sur les performances.

On trouve en deuxième position le classifieur de Simard et al. présenté en section 5.3.4. Le réseau LeNet se place en cinquième position, et commet 0.5 % d'erreur de plus.

5.4 Segmentation des chiffres

Nous allons maintenant rapidement passer en revue quelques travaux récents dans le domaine de la segmentation de l'écriture manuscrite, et plus particulièrement des chiffres.

Il existe de nombreuses publications décrivant divers algorithmes de segmentation basés sur diverses considérations morphologiques : composantes connexes, projection des pixels dans des directions particulières, etc. (voir par exemple [115, 190, 24, 194, 81, 134, 135]).

Ces techniques ne cherchent pas à intégrer la segmentation avec la reconnaissance des chiffres ; elles utilisent des informations a priori. Nous nous intéresserons ici d'avantage aux travaux visant à l'intégration des deux tâches dans un seul système (ISR : *Integrated Segmentation Recognition*).

5.4.1 Système «COISR» de Martin et al.

Martin et al. [149] ont proposé un système de segmentation pour les champs de chiffres reposant sur le balayage de l'image par un réseau spécialisé dans la reconnaissance de chiffres isolés centrés (d'ou' l'appellation «COISR», pour *centered object integrated segmentation and recognition*). Ce système est assez proche de ceux que nous avons testés dans le chapitre 6 (section 6.7) et de l'un de nos systèmes de détection de visages (chapitre 8, section 8.4.1, page 239).

L'idée est simplement de balayer horizontalement le champ de chiffres à traiter. Le réseau, un TDNN simplifié, a une rétine de taille fixe de dimension suffisante pour contenir les plus grands chiffres rencontrés (36x20 pixels dans l'exemple présenté). Le réseau est doté de deux types de sorties : un groupe de dix cellules donne la classe reconnue, et une cellule supplémentaire indique la qualité du «centrage» de la rétine. Un seuillage de l'activité de cette cellule pendant le balayage permet de ne retenir que les positions représentant des chiffres. Pendant l'apprentissage, on montre au réseau des chiffres plus ou moins bien centrés, en donnant la classe à reconnaître et le centrage.

Afin de réduire le nombre de positions à explorer et donc le temps de calcul, Martin et al. ont ajouté une troisième cellule à la sortie du réseau.

Taille des champs	Erreur par champ	Rejet par champ
2 chiffres	1 %	4.8 %
3 chiffres	1 %	11.1 %
4 chiffres	1 %	19.1 %
5 chiffres	1 %	23.4 %
6 chiffres	1 %	35.7 %

TABLE 5.3 – Taux de rejet mesurés par Martin et al. sur les champs de chiffres de la base NIST, pour atteindre 1 % d'erreur par champ. Le système n'utilise pas d'information a priori sur la taille du champ à traiter, mais les performances se dégradent naturellement lorsque le nombre de chiffres augmente.

Cette cellule indique la distance probable (vers la droite) du centre du prochain caractère. On utilise cette information pour guider le déplacement de la rétine (*marche saccadique*).

Les performances rapportées dans [149] sont mesurées sur la base NIST (voir chapitre 6, section 6.2), et sont reproduites dans la table 5.3.

D'après les auteurs, le mode *saccadique* permet de diviser le nombre d'appels au classifieur par quatre, tout en ne dégradant pas sensiblement les performances.

5.4.2 Système ISR de Keeler et Rumelhart

Le système décrit par Keeler et Rumelhart [113, 114] permet d'intégrer la segmentation et la reconnaissance d'un groupe de plusieurs chiffres dans un seul réseau connexioniste, en une seule passe.

L'image présentée sur la rétine peut ainsi contenir un nombre quelconque de caractères. Les premières couches du réseau sont semblables à celles d'un TDNN ordinaires (masques de poids partagés) et permettent une extraction de caractéristiques indépendante de la position. Les couches suivantes calculent les probabilités d'appartenance à chacune des classes *en chaque position*. Cette information est ensuite regroupée sur une sortie unique (une cellule par classe), donnant ainsi directement les chiffres présents dans la rétine.

La particularité principale de ce système est que l'on n'utilise aucune procédure d'alignement des sorties (contrairement aux méthodes décrites plus loin) : la sortie donne les classes reconnues, mais pas leur positions.

Un avantage appréciable de cette méthode est que l'apprentissage uti-

lise directement des champs de chiffres, et non des chiffres préalablement segmentés.

Le système a été testé sur la base NIST. Les performances moyennées sur tous les types de champs sont comprises entre 15 et 20 % de rejet pour 1 % d'erreur, en utilisant deux réseaux différents (on accepte de classer quand les deux systèmes donnent la même réponse).

5.4.3 *Shortest Path Segmentation*

L'équipe de recherche en réseaux connexionnistes d'AT&T a mis au point un système de lecture automatique pour les codes postaux américains dont les performances semblent élevées. Ce système est décrit dans [150] et [31].

Ce système repose bien entendu sur l'utilisation d'un classifieur très précis, et surtout capable de fournir une estimation de la qualité de la reconnaissance (l'idéal étant une probabilité d'appartenance a-posteriori). Le réseau TDNN LeNet, développé par cette équipe, convient parfaitement et a été utilisé par les auteurs.

L'idée générale est d'utiliser le classifieur pour «noter» certaines segmentations effectuées à priori, puis d'appliquer un algorithme de recherche dynamique (type Viterbi [69]) pour trouver la segmentation la plus probable (d'ou‘ l'appellation *Shortest Path Segmentation*).

Le système opère en trois phases :

1. pré-traitements de l'image (redressements, normalisation, etc) ;
2. génération de segments candidats, à l'aide des techniques des composantes connexes [81], et des projections verticale de l'image ;
3. classification de chaque segment par le réseau de reconnaissance ;
4. contruction d'un graphe orienté combinant les segments ; on associe à chaque nœud du graphe une «longueur» proportionnelle à la probabilité donnée par le classifieur ;
5. recherche dynamique dans le graphe pour trouver le chemin le plus court, donc la segmentation la plus probable.

L'intérêt de ce procédé par rapport aux deux techniques présentées plus haut est de limiter le nombre d'appels au classifieur : on génère en moyenne deux fois plus d'hypothèses qu'il n'y a de chiffres présents dans l'image.

Burges et al. [31] donnent des performances (sans rejet) de 81 % mesurées sur les 2400 codes postaux (à 5 chiffres) de la base USPS des postes américaines.

Les performances de cette méthode sont donc très bonnes. Cependant, ce type d'approche possède plusieurs inconvénients :

— si la bonne segmentation ne figure pas dans les hypothèses générées lors de la phase 2, la reconnaissance échoue. On a intérêt à générer un nombre élevé d'hypothèses.
— la mise au point du système est compliquée, car de nombreux réglages sont nécéssaires, en particulier lors de la génération des hypothèses. L'algorithme repose sur de nombreuses heuristiques, utilisant divers seuils. Le réglage fin du système suppose l'optimisation combinée de tous ces seuils, et l'on ne dispose pas de méthode générale pour ce faire (il faut payer un ingénieur !).

5.4.4 *Space Displacement Neural Network*

L'utilisation d'un *Space Displacement Neural Network* (SDNN) a été proposée par la même équipe [151], et évite la génération de coupures à priori : le réseau balaye ici toute l'image.

Le volume de calcul est réduit car on exploite la structure du réseau LeNet (voir figures 5.6 et 5.8), qui permet une extraction de caractéristiques par convolution/sous-échantillonnages. La figure 5.8 montre l'architecture (en une dimension pour simplifier) du réseau LeNet : on voit qu'il est possible de passer les quatre premières couches du TDNN sur toute l'image en une seule opération ; on obtient alors une «carte de caractéristiques» de taille 16 fois plus faible (environ) que l'image originale. Chaque cellule sur cette carte représente un voisinage 4x4 sur l'image originale.

Dans le réseau LeNet «standard», tel qu'il est représenté en figure 5.6, les cellules de sortie utilisent 4 cellules de la couche précédente et voient ainsi 16 pixels de la rétine.

Lors de la segmentation d'un champs de chiffres, la taille des chiffres est variable, et il n'est pas possible de les normaliser correctement avant d'effectuer la segmentation. Aussi Matan et al. [151] proposent-ils d'utiliser plusieurs couches de sorties, connectées respectivement à 5, 4, 3 et 2 cellules (en largeur) sur la «carte de caractéristiques». Chaque sortie répond ainsi à une certaine largeur des chiffres.

Les temps de calcul mis en œuvre ne sont pas beaucoup plus importants que dans la démarche présentée dans la section précédente, ou‘ l'on ne tirait pas parti du recouvrement des segments à classer.

La décision finale est prise en utilisant un algorithme de Viterbi sur les sorties. Lors de l'apprentissage, on rétropropage dans tout le réseau, en utilisant les cellules de sortie choisies par l'algorithme de Vitterbi.

Avec ce système, Matan et al. sont arrivés à des performances légèrement inférieures à celles de la méthode précédente. Les problèmes recensés sont liés à la complexité de l'apprentissage (choix des cellules sur lesquelles rétropro-

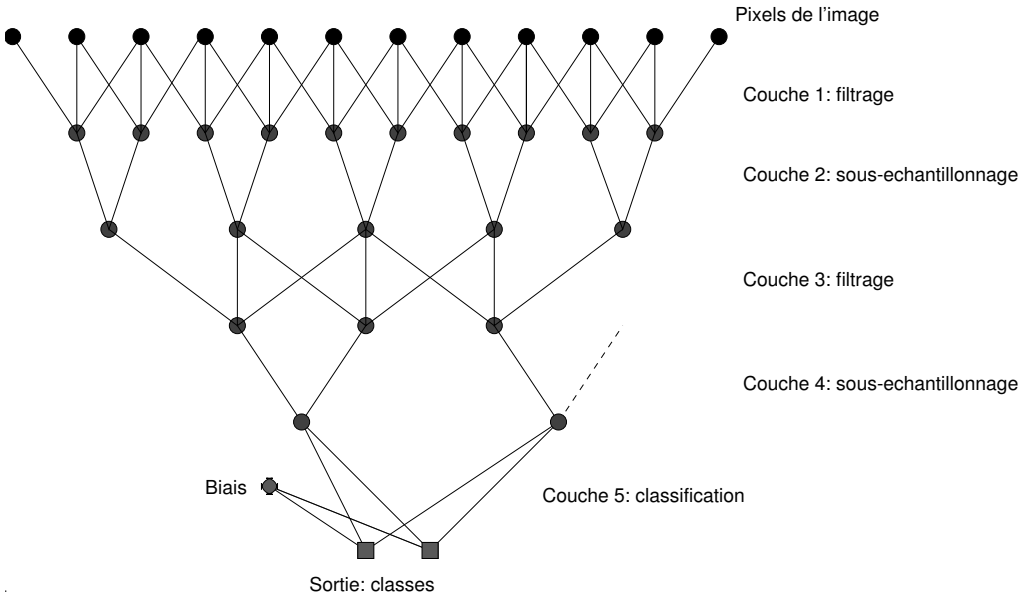


FIGURE 5.8 – Le réseau LeNet vu comme un filtrage de l'image. Nous avons ici représenté le traitement dans une seule dimension. En haut, les pixels de l'image à reconnaître. LeNet comporte quatre couches d'«extraction de caractéristiques», qui alternent filtrage (filtres 5x5, ici représentés par 3 branches) et moyennage/sous-échantillonnage (filtres 2x2 se décalant de 2). La dernière couche réalise la classification proprement dite (on a représenté deux cellules de sortie au lieu de 10). Tous les poids sont partagés, sauf sur la dernière couche. Chaque cellule de l'avant-dernière couche «voit» donc, si l'on néglige les effets de bords, 4x4 pixels de l'image originale.

pager) et aux variations de hauteur des caractères (comme on ne normalise pas, il faudrait également dupliquer les sorties dans le sens vertical).

Cependant, cette approche nous semble très prometteuse pour arriver à un système de lecture de chiffres robuste et d'implémentation facile (tous les traitements sont parallèlisables et aisément implementables sur une carte de traitement de signal ordinaire).

5.5 Conclusion

Les meilleures méthodes de reconnaissance de chiffres isolés présentées dans ce chapitre ont des performances égalant ou approchant celles d'un homme placé dans les mêmes conditions. Les temps de traitement restent souvent assez élevés, mais ce problème devrait bientôt disparaitre, par l'optimisation des algorithmes et l'utilisation de *hardware* spécialisé (voir par exemple [132, 84]). On peut donc considérer le problème de la reconnaissance de chiffres manuscrits isolés comme quasiment résolu.

La reconnaissance de champs de chiffres est un problème plus délicat, car se pose le problème de la segmentation. Les performances des techniques actuelles permettent sans aucun doute des gains de productivité important pour des applications comme le tri postal, ou‘ les erreurs ne sont pas très coûteuses. Elles semblent encore insuffisantes pour des applications plus sensibles comme le traitement automatique de chèques (à moins d'utiliser d'autres sources d'information, comme la somme en lettres).

Les techniques actuelles sont encore insuffisantes pour permettre la reconnaissance de l'écriture manuscrite cursive. Des solutions existent lorsque le vocabulaire est limité [134]. Dans ce domaine, le problème se pose de façon semblable à celui de la reconnaissance de la parole : on arrive à des systèmes performants si l'on limite le nombre de scripteurs, ou la taille du vocabulaire, mais la mise au point d'un système de lecture «universel» est encore loin.

Les problèmes liés à l'intégration de la segmentation et de la reconnaissance se retrouvent dans d'autres domaines, comme la reconnaissance de la parole continue : si l'on sait reconnaître individuellement les phonèmes, il est nettement plus difficile de transcrire une phrase. Ces difficultés sont liées à la prise en compte du contexte lors de la reconnaissance. La solution la plus en vogue actuellement est de coupler un classifieur balayant l'image et un alignement dynamique.

Chapitre 6

Simulations sur les chiffres manuscrits

6.1 Introduction

Nous décrivons dans ce chapitre les études que nous avons menées sur l'utilisation des algorithmes connexionnistes pour la reconnaissance d'images de chiffres manuscrits.

Nous nous sommes intéressé à cette application pour deux raisons. D'abord, ce problème est bien étudié : comme nous l'avons vu dans le chapitre 5, la littérature récente sur le sujet est très abondante. On a ainsi accès aux résultats de nombreuses équipes de recherche, travaillant souvent sur les mêmes données, ce qui permet de comparer aisément nos systèmes à ceux existant. La seconde raison est la disponibilité d'importantes bases de données *standards*, sur lesquelles de nombreux résultats sont disponibles. La taille de ces bases de données (couramment plusieurs dizaines de milliers d'images) permet de comparer avec une très grande précision les différentes méthodes de classification.

Ces conditions sont loin d'être réunies dans l'application à la reconnaissance des visages, pour laquelle nous avons dû utiliser nos propres données et ne disposions (surtout au début de cette thèse) que de très peu de publications.

6.1.1 Plan du chapitre

Ce chapitre commence par une description des données sur lesquelles nous avons travaillé, qui sont issues de la base de données publique NIST [77]. Nous précisons les ensembles d'apprentissage, évaluation et test utilisés, ainsi que

les pré-traitements appliqués et les intervalles de validité statistique obtenus lors des mesures de performances.

Nous décrivons ensuite plusieurs architectures de classifieurs TDNN [126]. Nous avons reproduit les résultats d'AT&T en utilisant le réseau LeNet [129], que nous comparons par la suite à diverses architectures mono et multi-modulaires originales.

Les premières architectures multi-modulaires présentées (section 6.4) utilisent un réseau TDNN simplifié pour l'extraction de caractéristiques. Nous avons mené une grande variété d'expériences avec ces architectures, ce qui nous a permis de comparer différents classifieurs (MLP, kNN, LVQ, RBF) sur les même données, de mesurer le gain apporté par la technique d'*apprentissage coopératif*, et enfin de comparer différents critères de rejet pour la détection des formes ambiguës ou éloignées (*outliers*).

Dans la section 6.5, nous proposons un nouveau type d'architectures multi-modulaires pour la classification, basées sur la combinaison d'un apprentissage auto-associatif pendant l'initialisation du système et d'un apprentissage MLP+ RBF coopératif. Nous montrons que ce système permet un gain de temps de calcul important (8 fois plus rapide pour des performances presque équivalentes à celles du réseau TDNN LeNet).

Enfin, nous illustrons en section 6.7 nos résultats par un petit exemple d'application à la segmentation de chiffres.

6.2 Description des données : base NIST

Nous décrivons dans cette section les données utilisées pour les études expérimentales décrites dans ce chapitre.

Comme nous l'avons vu au chapitre 5 (section 5.3.6), l'institut américain NIST [77] a réuni plusieurs bases de données de caractères manuscrits, qui sont accessibles aux chercheurs [1] sur disques optiques CD-ROM.

6.2.1 Bases de données NIST

La base «NIST Database 1» contient les images de 2100 formulaires scannés à 300 dpi. Les formulaires ont été écrits (à la main) par le personnel du bureau du recensement américain. La figure 5.1 (page 109) montre quelques champs de chiffres extraits de cette base.

La base «NIST Database 3» contient les caractères segmentés extraits des images de la base 1. Ces images sont toutes de taille 128x128 pixels, en noir

1. Voir l'adresse dans [77].

et blanc (1 bit/pixel). Elles contiennent en principe un seul chiffre, segmenté automatiquement par NIST à partir des images de formulaires pré-casés.

Les images sont toutes étiquetées.

Nous avons pu disposer d'une partie de la base NIST 3, nommée *hsf0*, et comportant 24 000 images de chiffres segmentés. Les chiffres ont été découpés dans des champs du types de ceux représentés en figure 5.1.

Notons qu'il s'agit de chiffres américains : le «7» est presque toujours écrit sans barre horizontale, le «1» est réduit à un trait vertical, et le «2» est plus «bouclé» qu'en France.

6.2.2 Ensembles d'apprentissage, évaluation et test

Nous avons découpé *hsf0* en trois sous-ensembles, utilisés pour les phases d'apprentissage, de validation et de test des différents systèmes.

La base n'a pas été mélangée avant ce découpage, ce qui implique que les trois parties contiennent des scripteurs différents.

Ensemble d'apprentissage : *hsf0_learn* est constitué des 15 000 premiers chiffres de *hsf0*. Ces images sont utilisées pendant l'apprentissage des systèmes, pour déterminer les valeurs optimales des paramètres (poids) des classifieurs (réseaux).

Ensemble d'évaluation : *hsf0_eval* est constitué des 4 000 chiffres suivants. Cet ensemble est utilisé pour mesurer les performances des classifieurs en généralisation pendant les apprentissages, et pour déterminer les valeurs des *seuils de rejet*.

Ensemble de test : *hsf0_test* comporte les 5 000 derniers chiffres de *hsf0*. Cet ensemble est utilisé pour mesurer les performances finales des systèmes de reconnaissance de chiffres.

Notons que durant la mise au point de chaque classifieur, on n'utilise que les ensembles d'apprentissages (pour l'optimisation des paramètres) et d'évaluation (pour décider d'arrêter l'apprentissage, puis fixer les seuils de rejet). L'ensemble de test n'est utilisé que pour comparer différents types de classifieurs.

Les trois ensembles que nous venons de définir sont constitués d'images de chiffres. Ils vont nous permettre de mesurer et comparer les erreurs de classification des différents systèmes. Ils vont aussi être utilisés pour mesurer la capacité de ces systèmes à rejeter les caractères ambigüs.

Nous souhaitons aussi comparer les propriétés de rejet de *distance* des différents algorithmes. Pour cela, il nous faut disposer d'ensembles définis d'images qui ne soient pas des chiffres, que l'on appellera *outliers*.

On a vu que l'on peut généralement découper les systèmes de reconnaissance de chiffres en deux étages : extraction de caractéristiques, puis classification. Ce découpage nous a conduit à définir deux catégories d'*outliers*.

La première catégorie est constituée d'images qui ne correspondent évidemment à aucun chiffre, mais possèdent le même type de «structure» que les images de chiffres. Sur de telles images, l'extraction de caractéristiques a encore un sens. Si l'on reprend l'exemple d'extraction de caractéristiques présenté plus haut (chapitre 5, section 5.3.1), on doit pouvoir énumérer les caractéristiques morphologiques, mesurer les profils, etc. , et obtenir des valeurs non aberrantes. Dans une application de segmentation de chiffres, les images de cette catégorie proviendront typiquement des chiffres collés (attachés) ou dont il manque une partie.

La seconde catégorie d'*outliers* est constituée d'images sans aucun rapport avec des images de chiffres. On s'attend dans ce cas à ce que l'extracteur de caractéristiques fournisse des valeurs aberrantes, en dehors de leur domaine habituel. En pratique, ce type d'images sera recolté par l'étage de segmentation dans les zones ne comportant pas d'écriture : fonds texturés ou images.

Nous avons constitué deux ensembles d'images, appartenant respectivement aux deux catégories que nous venons de définir.

Zarbi est un ensemble qui contient 765 images de lettres manuscrites, majuscules et minuscules, provenant d'une autre partie de la base NIST *hsf0*. Les images de lettres ressemblent en effet par leur structure aux images de chiffres. Un bon extracteur de caractéristiques sur les chiffres devrait être aussi bon sur les lettres.

On désire cependant que les images de *Zarbi* soient suffisamment éloignées des images de chiffres pour ne pas risquer d'être confondues : il est difficile de distinguer un zéro d'un O manuscrit, par exemple. Aussi avons-nous retenu les caractères suivants, dans des proportions égales, pour constituer la base *Zarbi* :

{ A, B, E, F, H, K, M, N, P, R, W, X, a, h, k, m, n, t, w }.

Les lettres { S, G, O } ont été exclues, car trop ressemblantes aux chiffres { 5, 6, 0 }. La figure 6.1 montre quelques images de *Zarbi*.

Random contient aussi 765 images, générées en utilisant un bruit blanc. Les pixels de ces images ne sont pas corrélés, et prennent des valeurs aléatoires uniformément dans $[-1, 1]$.

Nous aurions aussi bien pu utiliser des images extraites de scènes naturelles ou des textures standards [27]. Il semble, d'après quelques tests préliminaires, que les propriétés de rejet des *outliers* des réseaux TDNN ne soient pas très sensibles aux types d'images utilisées. Les

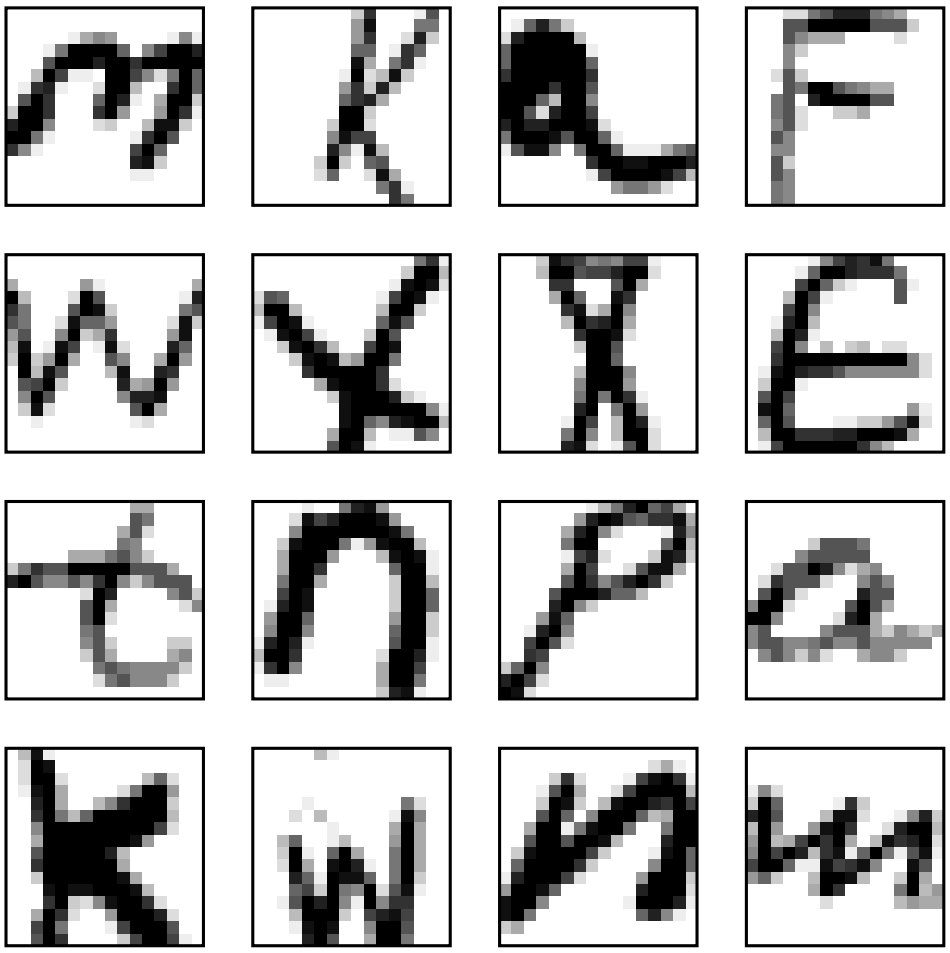


FIGURE 6.1 – Exemples de formes de l'ensemble *Zarbi*, ici normalisées en 16x16 pixels.

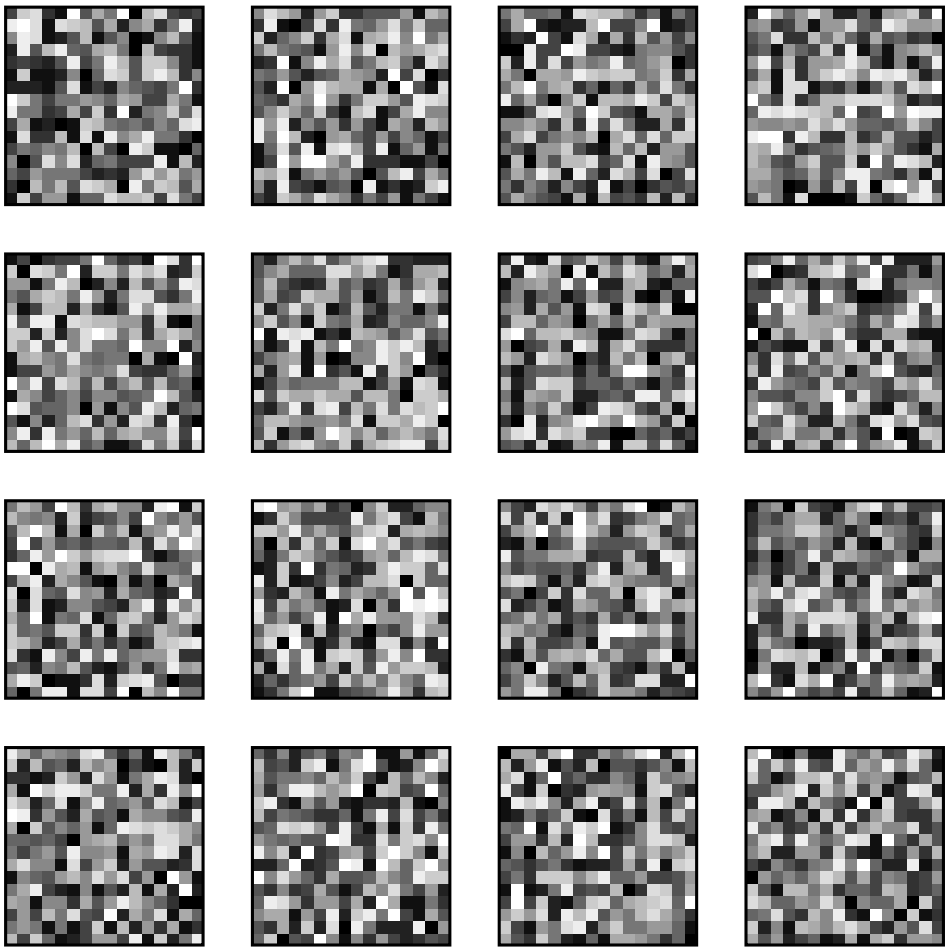


FIGURE 6.2 – Exemples de formes de l'ensemble *Random*, en 16x16 pixels.

Erreur (%)	Apprentissage ($m = 15\ 000$)	Eval. ($m = 4\ 000$)	Test ($m = 5\ 000$)
10	0.48 %	0.93 %	0.83 %
8	0.43 %	0.84 %	0.75 %
6	0.38 %	0.74 %	0.66 %
4	0.31 %	0.61 %	0.54 %
2	0.22 %	0.43 %	0.38 %
1	0.16 %	0.30 %	0.27 %

TABLE 6.1 – Intervalles de confiance à 95 % (t_{95}) pour différents taux d'erreurs sur les ensembles d'apprentissage, test et évaluation.

images de bruit blancs présentent l'avantage d'être faciles à définir et à synthétiser.

La figure 6.2 montre quelques images de la base *Random*.

6.2.3 Validité statistique

Les résultats obtenus sur les bases de données que nous venons de présenter sont significatifs dans un certain *intervalle de confiance*. Le principe utilisé pour déterminer cet intervalle de confiance est détaillé dans l'annexe A.

Le tableau 6.1 donne la demi-largeur (t_α) de l'intervalle de confiance, pour différents taux d'erreurs mesurés sur les bases d'apprentissage, évaluation et de test. On voit d'après le tableau que si l'on mesure 2 % d'erreur sur *hsf0_ eval*, l'erreur "réellé' (mesurée sur un ensemble de taille infinie) doit se situer dans l'intervalle $[1.6\%, 2.4\%]$.

6.2.4 Pré-traitements

Nous détaillons dans cette section les prétraitements mis en œuvre avant de présenter les images à l'entrée des réseaux de neurones.

Nous l'avons déjà dit (page 120), les pré-traitements sont souvent déterminants dans les performances des systèmes de reconnaissance de caractères. Les enjeux commerciaux dans ce domaine étant de taille, et l'intérêt scientifique de ces traitements étant souvent faible, les publications entrent rarement dans les détails. Notre travail n'étant pas centré sur la reconnaissance de caractères, nous n'avons pas beaucoup étudié et comparé les pré-traitements possibles. Aussi décrivons-nous simplement le traitement utilisé pour nos expériences.

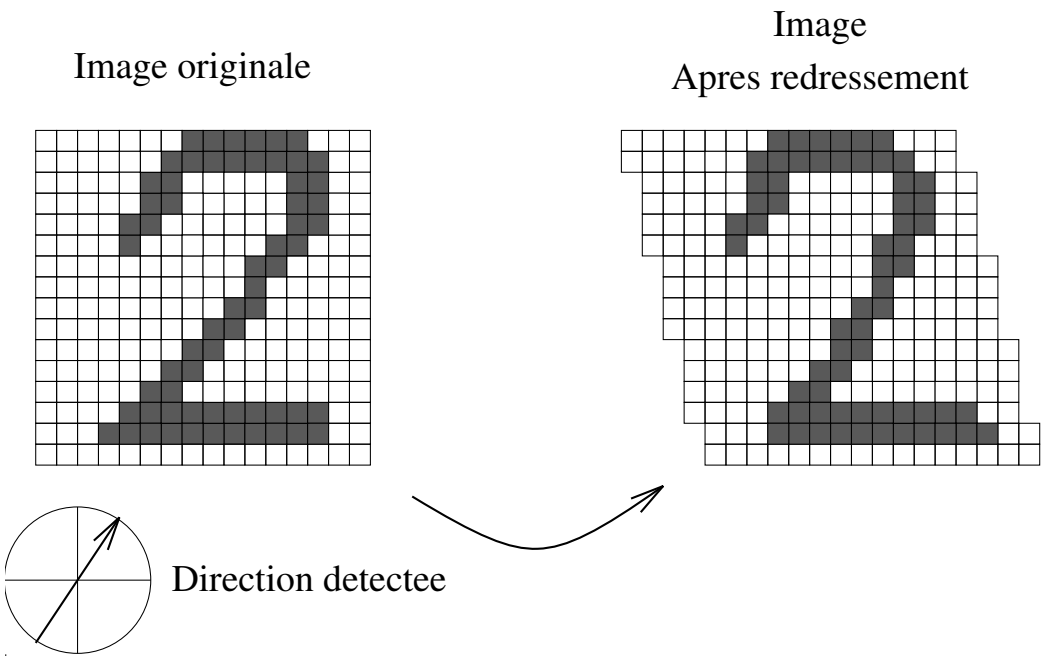


FIGURE 6.3 – Redressement vertical d'un caractère. Une fois la direction de redressement détectée, on translate simplement les lignes de l'image.

Les images de caractères de la base NIST sont des bitmaps carrées de côté 128 pixels, dans lesquelles le caractère occupe une taille variable.

Redressement

Les chiffres sont écrits avec diverses orientations, plus ou moins penchés à gauche ou à droite. La première étape du pré-traitement vise à réduire ces variations d'orientation, en *redressant* les caractères.

Pour cela, on va chercher autour de la verticale la direction selon laquelle le caractère parait le plus «droit». Puis on va déformer l'image pour faire coïncider la direction trouvée avec la verticale.

Soit $f(x, y)$ la fonction associant aux coordonnées (x, y) de l'image à traiter la luminosité (en niveaux de gris ou en noir et blanc, la valeur 0 représentant le blanc et la valeur 1 le noir). Soit D_{θ,x_0}, la droite passant par le point x_0, et d'angle θ par rapport à la verticale.

On peut calculer la projection de l'image sur l'axe horizontal, selon la direction θ

$$P(x_0, \theta) = \oint_{D_{\theta,x_0}} f(x, y)dl \tag{6.1}$$

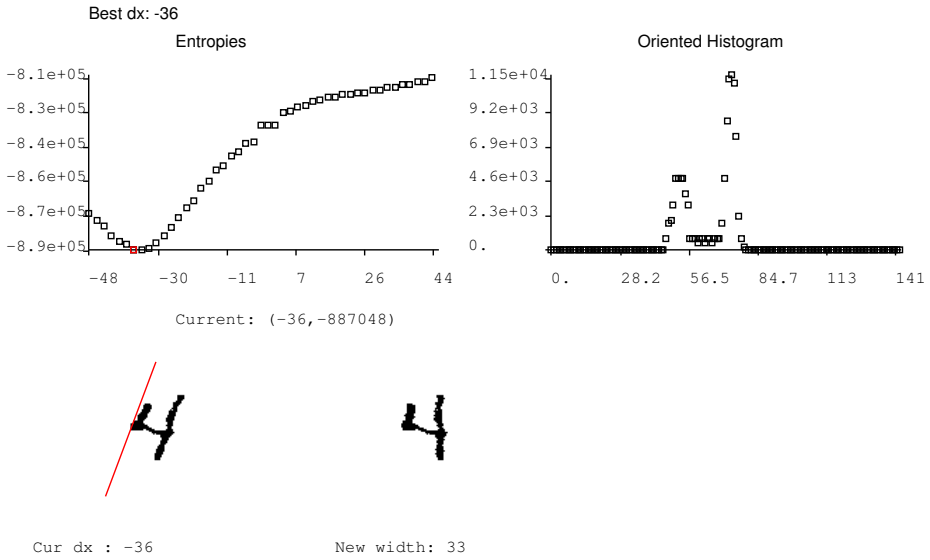


FIGURE 6.4 – Exemple de redressement vertical d'un chiffre «4». En haut à gauche, la fonction $H(\theta)$. En haut à droite, la projection de l'image dans la direction θ_0 du minimum. En dessous, le chiffre original, et redressé.

L'ensemble de ces projections est la *transformée de Radon* de l'image, très utilisée en tomographie [180].

On cherche la direction telle que cette projection soit la moins dispersée possible. Pour cela, on peut exprimer l'«entropie» de la projection, définie comme

$$H(\theta) = -\int_x P(x,\theta)\log P(x,\theta)dx \tag{6.2}$$

et chercher la direction θ_0 minimisant H.

Le redressement proprement dit de l'image est alors effectué en translatant horizontalement chaque ligne de l'image de la quantité ad-hoc (voir figures 6.3 et 6.4).

Ce processus introduit souvent quelques «bavures» sur les contours du chiffre, aussi est il souvent préférable de le faire suivre d'un lissage de l'image (convolution par un filtre gaussien par exemple).

Normalisation

Cette phase consiste à ré-échantillonner l'image noir et blanc du caractère afin d'obtenir une image en niveaux de gris (valeurs flottantes dans $[-1,\ 1]$) de la taille de la rétine du réseau de reconnaissance (en général 16x16 pour les systèmes décrits plus loin). Pour cela, on utilise une simple transformation linéaire. Cette transformation est équivalente à un filtrage passe-bas suivi d'un sous-échantillonnage de l'image, ce qui présente l'avantage d'atténuer les bruits (dus au scanner et aux artefacts introduits lors du redressement).

Le rapport hauteur/largeur du caractère est conservé en ajoutant une bordure symétrique.

La figure 6.5 montre quelques chiffres extraits de l'ensemble *hsf0_ test* après les pré-traitements.

6.3 Architectures TDNN testées

Nous allons décrire dans cette section les divers systèmes de reconnaissance de chiffres manuscrits que nous avons étudiés.

6.3.1 LeNet

Nous présentons d'abord les résultats du réseau LeNet sur les ensembles de caractères décrits dans la section précédente.

Nous avons décrit ce réseau dans la section 5.3.2.

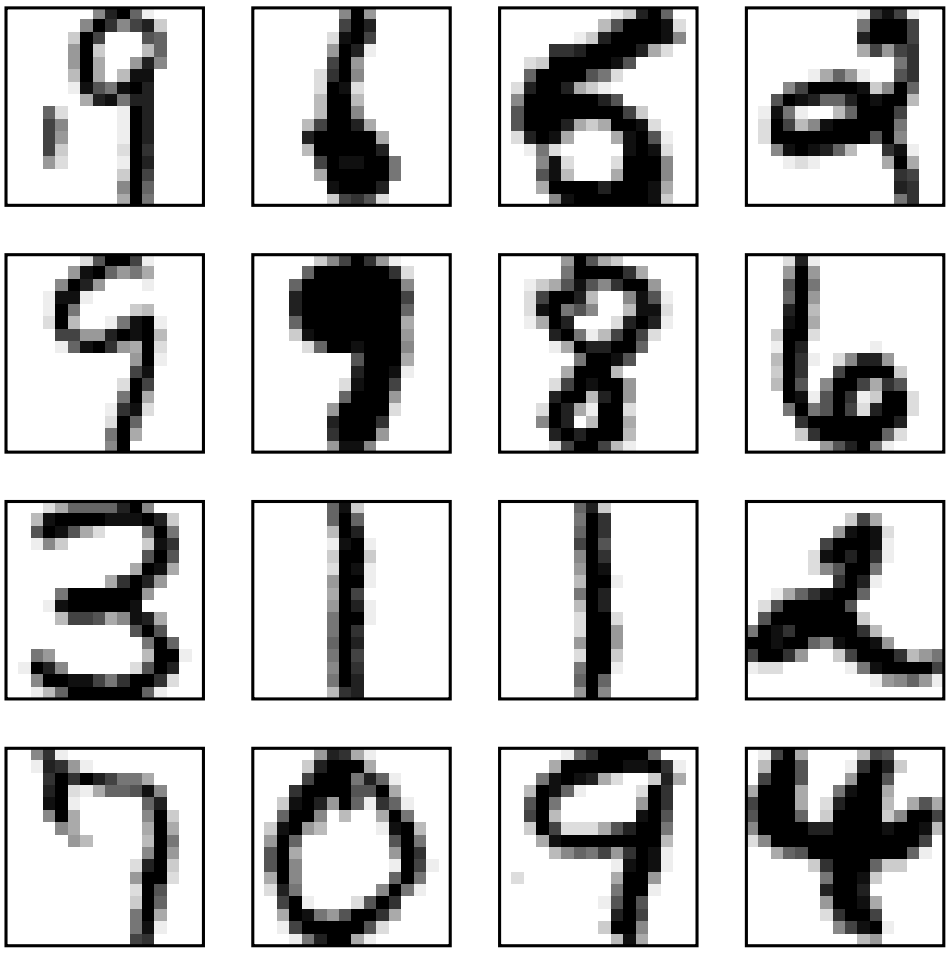


FIGURE 6.5 – Chiffres de l'ensemble *hsf0_ test*, après pré-traitements et normalisation en 16x16 pixels.

L'apprentissage a été effectué sur la base *hsf0_ learn* (15 000 caractères). Nous avons utilisé l'algorithme Levenberg-Marquardt [3, 29, 8] pour contrôler le gain durant l'apprentissage.

Ce réseau possède 98 442 connexions, ce qui rend l'apprentissage assez long sur une petite machine (plus de 80 heures sur un Sun Sparcstation 1, pour environ 80 passes de la base d'apprentissage de 15 000 exemples).

En fin d'apprentissage, les performances atteintes sont les suivantes :

hsf0_ learn	1.1 %
hsf0_ eval	1.4 %

Il est difficile de déterminer les performances optimales susceptibles d'être atteintes par des réseaux de ce type. En effet, l'erreur en généralisation continue de décroître, de plus en plus lentement, même après un grand nombre de passes. Les résultats indiqués plus haut on été obtenus en prolongeant l'apprentissage jusqu'a ce que l'erreur ne décroisse plus de façon observable.

La courbe donnant le rejet en fonction de l'erreur sera présentée en section 6.3.3.

6.3.2 TDNN Quick

Notre objectif n'est pas de proposer une nouvelle architecture de réseau multi-couches offrant des performances équivalentes à celles de LeNet. Nous l'avons déja dit, de nombreuses autres architectures ont été testées ailleurs, et ajouter la nôtre ne serait pas d'un grand intérêt.

Nous avons voulu mettre en œuvre une nouvelle approche, basée sur l'emploi d'architectures *multi-modulaires*, faisant coopérer différents algorithmes connexionnistes pour résoudre le problème.

On peut utiliser un réseau TDNN pour extraire des caractéristiques pertinentes de l'image à traiter. L'utilisation de couches de poids partagés permet, on l'a vu, d'obtenir une détection (filtrage) presque invariante par translation.

Le réseau LeNet est sans aucun doute un excellent extracteur de caractéristiques. Il possède cependant deux inconvénients :

1. le nombre de connexions est très élevé, aussi la reconnaissance d'un caractère demande un temps appréciables : 96 512 multiplications pour les quatre premières couches ;
2. la réduction de dimension de l'espace des données est relativement faible : on passe d'un caractère de taille 16x16 = 256 à un vecteur de caractéristiques de taille 192 (rapport 12/16 = 3/4). Dans un espace

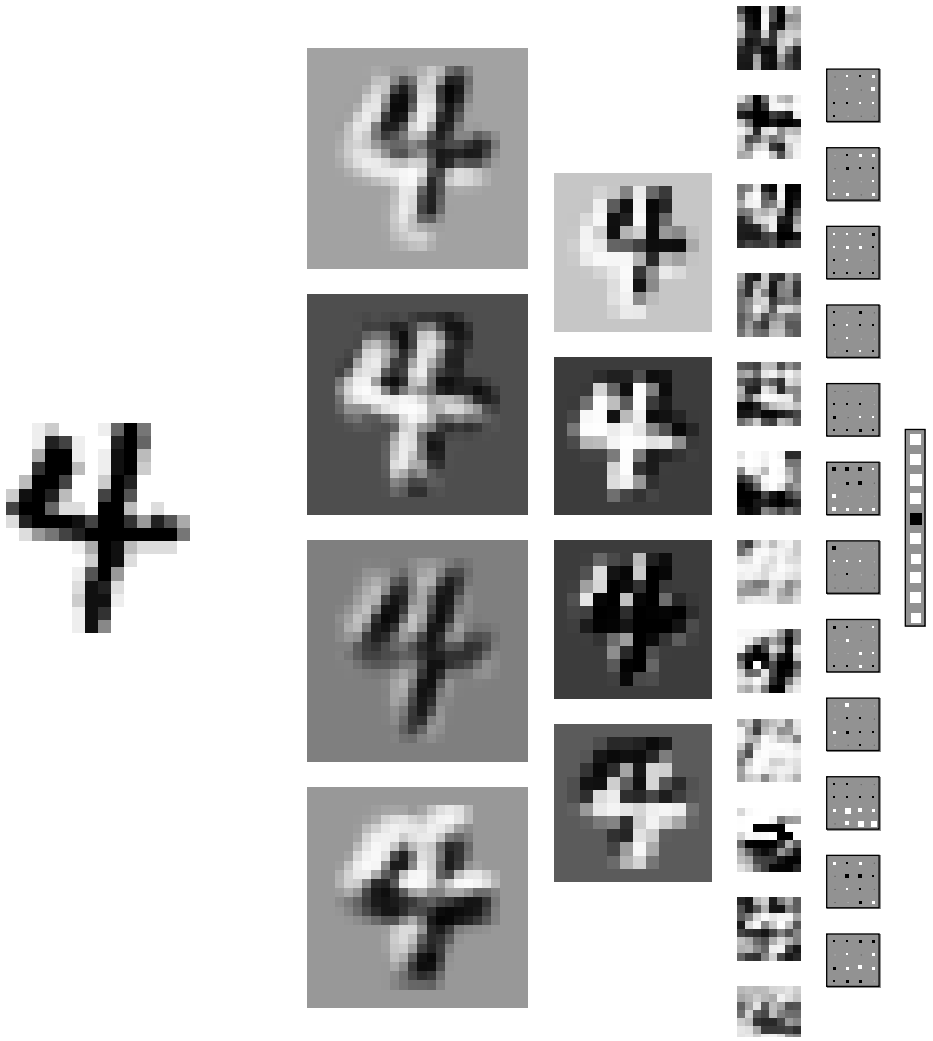


FIGURE 6.6 – Etats des cellules du réseau LeNet après apprentissage, lorsque l'on présente un "4" sur la rétine (à gauche). Le chiffre est ici correctement reconnu.

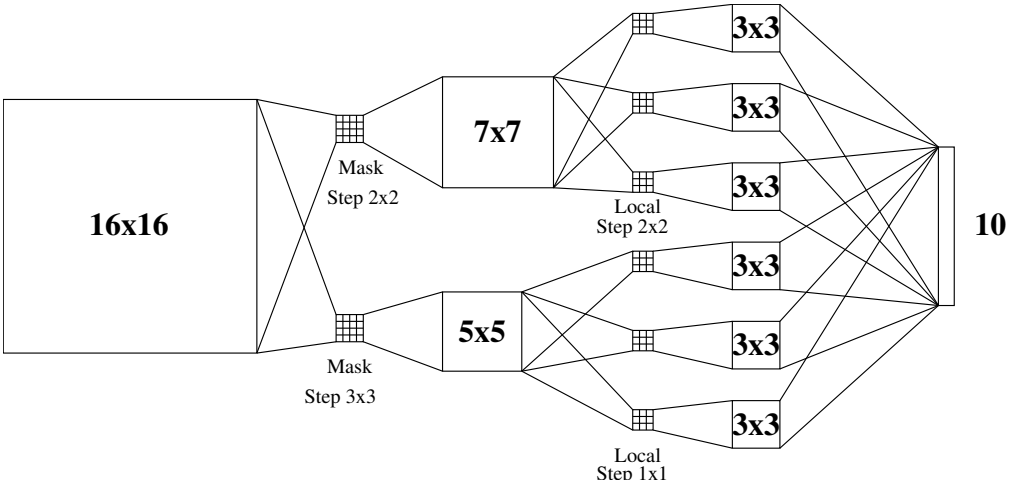


FIGURE 6.7 – Architecture du réseau *Quick*.

de cette dimension, il est délicat (sans parler des temps de calcul) d'utiliser un classifieur basé sur des calculs de distance [2].

Aussi avons-nous conçu un réseau ayant les propriétés suivantes : faible nombre de connexions, pour réduire les temps de calculs (ce qui nous a permis de mener une plus grande variété d'expériences), et faible dimension de l'avant dernière-couche (de l'ordre de 50), afin de pouvoir tester différents types de classifieurs.

L'architecture obtenue, nommée «*Quick*», est loin d'être optimale pour la reconnaissance de caractères. Elle va cependant nous permettre de mesurer les gains de performances obtenus par l'utilisation d'autres types de classifieurs (kNN, LVQ, RBF) remplaçant la dernière couche du TDNN.

Architecture du réseau Quick

L'architecture du réseau Quick [3] est représentée sur la figure 6.7. Ce réseau possède trois couches de connexions.

Le caractère à reconnaître est présenté sur la rétine, de taille 16x16 (sans ajout de bordure). La première couche cachée est divisée en deux groupes de cellules. Le premier, de taille 7x7, est relié à la rétine par un masque 4x4

2. Bottou et Vapnik [17] ont constaté que remplacer la dernière couche de LeNet par un classifieur kNN n'apportait aucune amélioration des performances (autre qu'une augmentation du temps de calcul...). Ce fait s'explique aisément par les considérations sur le nombre k^* optimal de voisin exposées en section 2.5.3 du chapitre 2.

3. Cette architecture a été proposée par B. Lamy, actuellement en cours de thèse à Parix VI, qui a étudié les performances de petits TDNN pour l'OCR manuscrit.

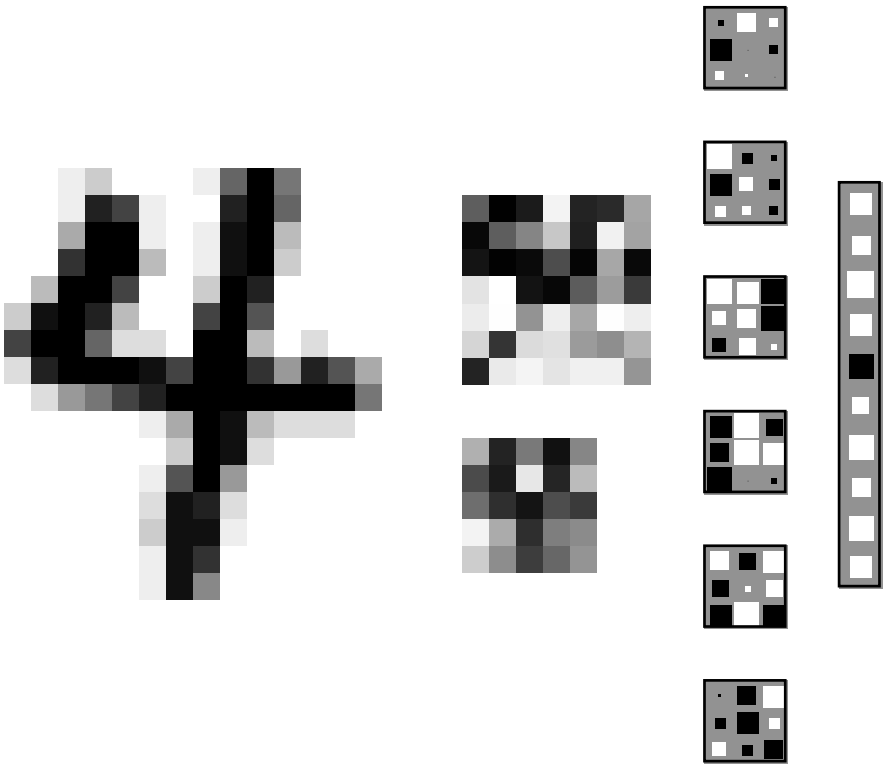


FIGURE 6.8 – Etats des cellules du réseau Quick après apprentissage, lorsque l'on présente un "4" sur la rétine (à gauche). Le chiffre est ici correctement reconnu.

de poids partagés se décalant par pas de 2 dans chaque direction. Le second groupe, de taille 5x5, utilise un autre masque 4x4, se décalant par pas de 3.

Les deux groupes de la première couche sont à leur tour chacun connectés à trois groupes de taille 3x3 par des connexions *locales*. L'avant dernière couche est ainsi de dimension 3x3x6 = 54 cellules.

Chaque cellule de sortie du réseau est associé à une classe (0-9), et connecté aux 54 cellules de la couche précédente. Cette couche réalise donc une classification quasi-linéaire.

Au total, Quick possède 394 cellules, 2348 connexions, et 1196 poids. Le temps de reconnaissance d'un caractère est ainsi divisé par 40 par rapport à LeNet.

Apprentissage

L'apprentissage du réseau Quick nécessite approximativement le même nombre de passes que LeNet (en utilisant l'algorithme de Levenberg-Marquardt), et est par conséquent très rapide (quelques heures).

Les performances atteintes en fin d'apprentissage sur *hsf0_ learn* sont les suivantes :

hsf0_ learn	3.0 %
hsf0_ eval	3.8 %

On voit que le réseau Quick commet presque trois fois plus d'erreurs que LeNet.

6.3.3 Comparaison des critères de rejet MLP

Dans cette section, nous allons comparer les performances des critères de rejets MLP présentés dans le chapitre 3.

Pour ces comparaisons, nous utiliserons le réseau Quick, entraîné sur *hsf0_ learn*, présenté dans la section précédente.

Vérification de la somme des cellules de sortie

On a vu dans le chapitre ? qu'après convergence, les cellules de sortie du réseau estimaient les probabilités bayésiennes a-posteriori d'appartenance de la forme présentée à chacune des classes. En particulier, il importe que les sorties somment à 1. C'est normalement le cas en pratique [177].

Nous avons vérifié ce point en calculant l'histogramme de répartition de la somme des sorties du réseau Quick sur la base d'évaluation *hsf0_ eval*. La courbe correspondante est tracée en figure 6.9. On voit que la somme des sorties est en moyenne légèrement inférieure à un.

Courbes de rejet sur *hsf0_ eval*

Nous présentons maintenant les courbes de rejet en fonction de l'erreur, obtenues sur les chiffres de *hsf0_ eval*.

Au cours du chapitre 3 (section 3.4.5), nous avons défini plusieurs critères de rejet appliquables aux classifieurs MLP. Nous allons ici les utiliser et les comparer.

Pour chacun des critères de rejet (MLP-1, MLP-2, MLP-3 et MLP-4), nous avons mesuré sur *hsf0_ eval* le taux d'erreur et le taux de rejet en fonction du seuil Θ. On a ensuite tracé la courbe donnant l'erreur en fonction du rejet en éliminant Θ.

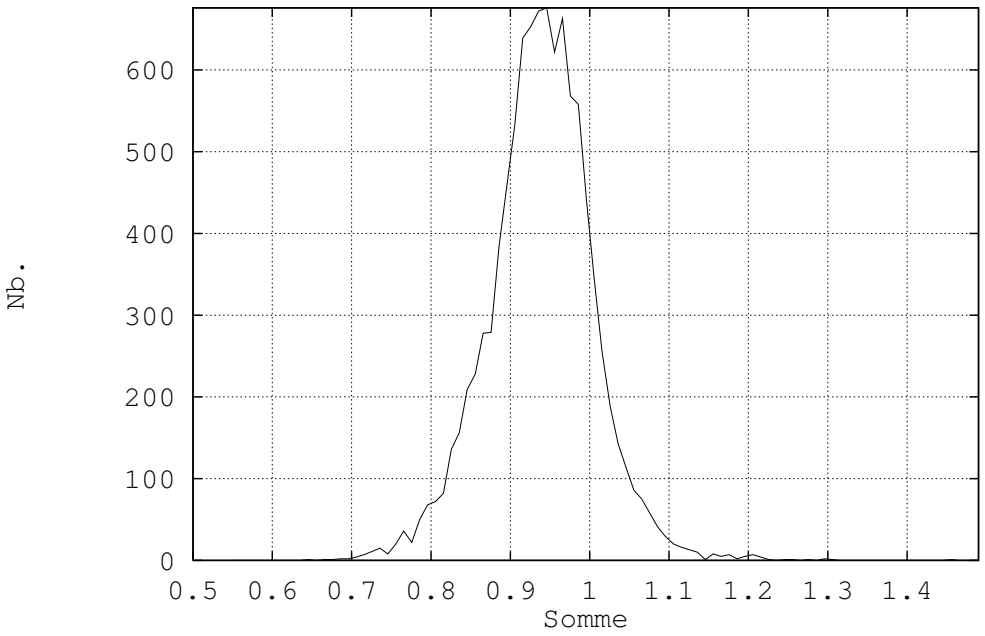


FIGURE 6.9 – Répartition de la somme des sorties du réseau Quick sur *hsf0_ eval*. La moyenne de la somme des sorties est 0.94, et la variance 0.06.

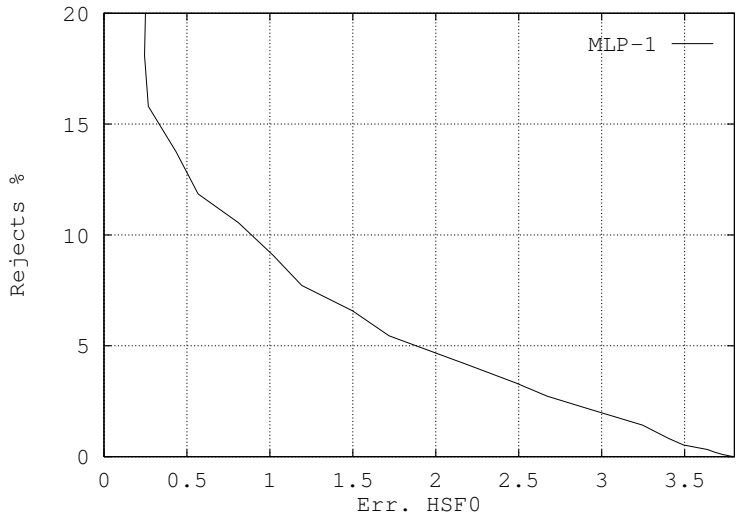


FIGURE 6.10 – Rejet sur *hsf0_ eval* en fonction de l'erreur pour les 4 critères de rejets MLP. Réseau Quick.

On voit sur la courbe 6.10 que les quatre critères sont totalement équivalents : les courbes sont presque confondues.

Ce résultat n'est pas surprenant. Les quatre critères, d'après leurs définitions, détectent de la même façon les images ambiguës de *hsf0_ eval*.

Courbes de rejet sur les outliers

Examinons maintenant les courbes de rejets sur les ensembles d'*outliers* : *Zarbi* et *Random* (voir figures 6.11 et 6.12).

Les outliers nous permettent ici de départager les critères de rejet. Sur *Zarbi*, le critère MLP-3 est nettement meilleur que les autres : il rejette en moyenne 10 % de plus que MLP-1. Sur *Random*, les critères MLP-2, MLP-3 et MLP-4 sont équivalents. Le critère MLP-1 est ici supérieur (d'environ 10 %).

Comme on pouvait s'y attendre, les images de bruit blanc (*Random*) sont toujours plus rejetées que les images de *Zarbi*.

Conclusion Pour le rejet des images ambiguës, les quatre critères de rejet sont équivalents, MLP-2 et MLP-4 étant légèrement supérieurs. C'est de cette façon que les critères sont habituellement comparés.

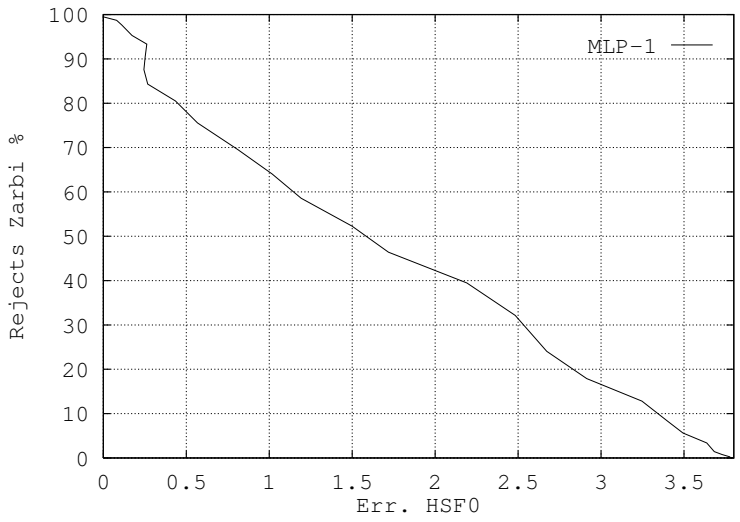


FIGURE 6.11 – Rejet sur *Zarbi* en fonction de l'erreur sur *hsf0_ eval* pour les 4 critères de rejets MLP. Réseau Quick.

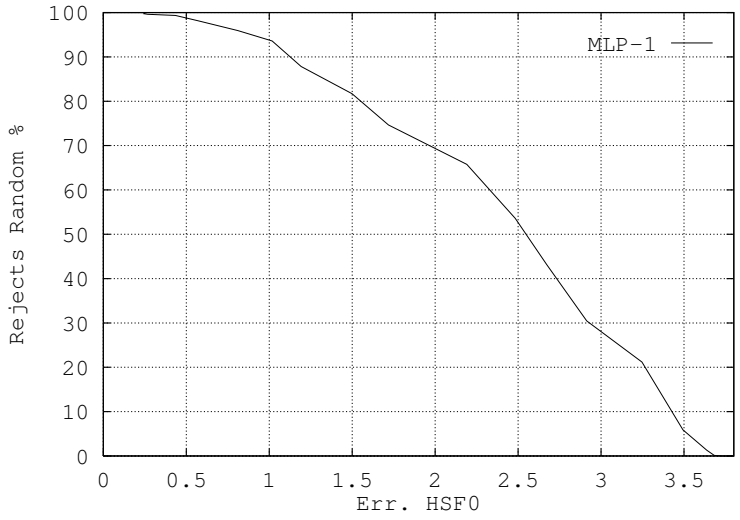


FIGURE 6.12 – Rejet sur *Random* en fonction de l'erreur sur *hsf0_ eval* pour les quatre critères de rejet MLP. Réseau Quick.

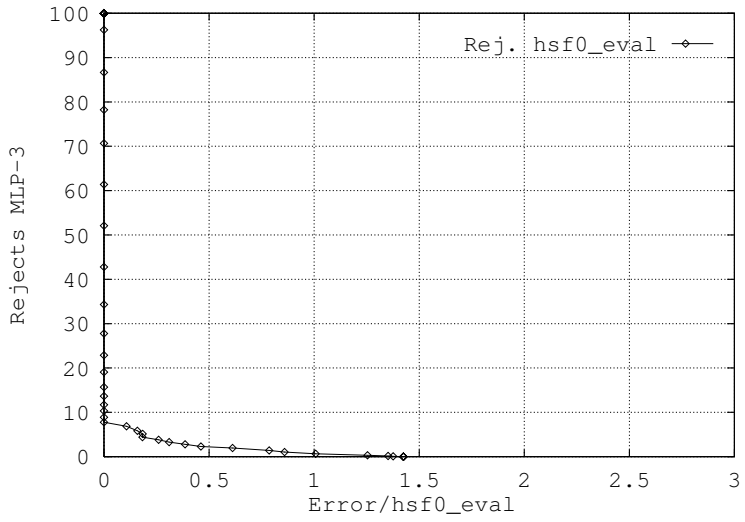


FIGURE 6.13 – Rejet sur *hsf0_ eval*, *Zarbi* et *Random* en fonction de l'erreur sur *hsf0_ eval*. Réseau LeNet, critère de rejet MLP-3.

L'usage d'ensembles d'images d'*outliers* permet de départager les critères. Dans une application ou‘ le classifieur va rencontrer une proportion non négligeable d'outliers de type *Zarbi* (erreurs de segmentation), le critère MLP-3 sera nettement préférable.

Dans la suite de ce chapitre, toutes les courbes de rejets présentées pour les réseaux MLP [4] seront calculées à l'aide du critère MLP-3.

Courbe de rejet du réseau LeNet

Voici maintenant la courbe de rejet du réseau LeNet (critère MLP-3), sur les trois ensembles d'évaluation *hsf0*, *Zarbi* et *Random* (figure 6.13). On voit que LeNet est, là aussi, bien supérieur à Quick : pour arriver à 1 % d'erreur, il lui faut rejeter 0.8 % de *hsf0_ eval*, contre 8.8 % pour Quick.

6.4 Architectures modulaires basées sur Quick

Dans cette section, nous allons montrer comment l'utilisation d'architectures modulaires basées sur le TDNN Quick permet d'augmenter les performances. Ces architectures sont basées sur un principe simple : la dernière

4. ce qui inclut les réseaux RBF.

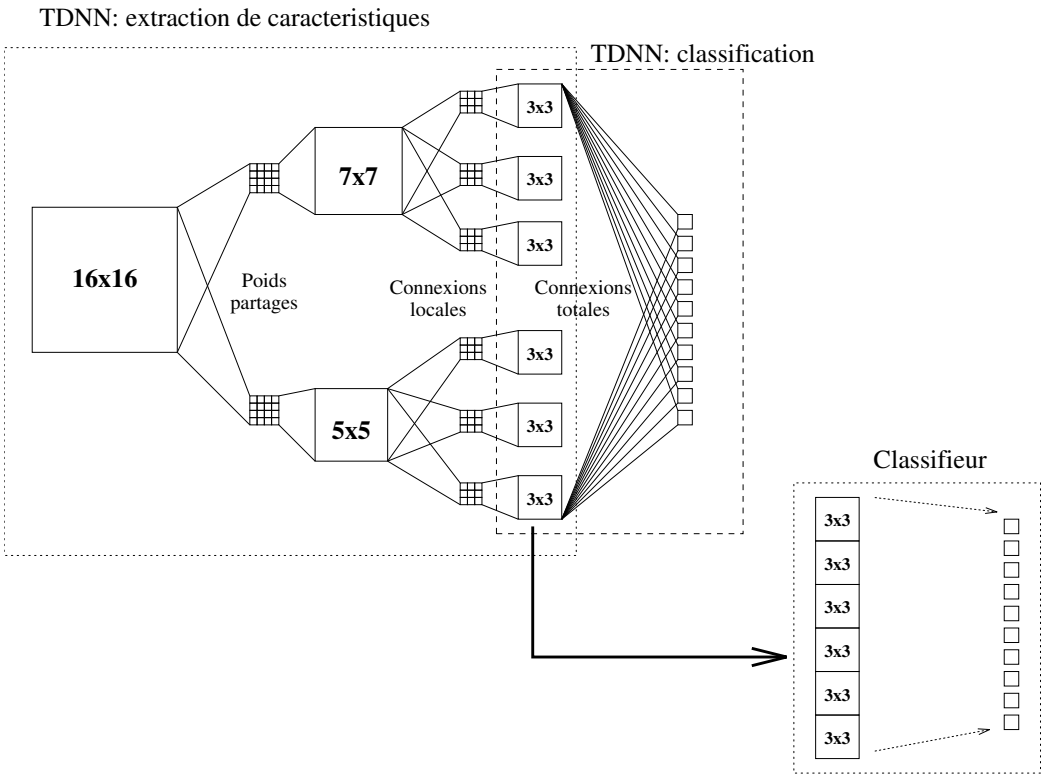


FIGURE 6.14 – Architectures modulaires basées sur le réseau TDNN Quick. Les deux premières couches de Quick constituent le premier module, extracteur de caractéristiques. La dernière couche du TDNN n'est plus utilisée que pour l'initialisation du système. Le deuxième module est un classifieur quelconque (kNN, LVQ, RBF) branché sur le vecteur de caractéristiques (dernière couche cachée de Quick).

couche du réseau Quick est un simple classifieur quasi-linéaire, optimisé selon le critère des moindres carrés. Remplacer cette couche par un classifieur plus adapté est susceptible d'améliorer le système (voir figure 6.14).

Nous allons dans un premier temps décrire plusieurs architectures sous-optimales, c'est à dire constituées de deux étages optimisés séquentiellement. On a vu dans le chapitre ? ? qu'il était possible d'entraîner simultanément (de façon coopérative) plusieurs modules connexionnistes par descente du gradient. Nous présenterons donc ensuite un tel système

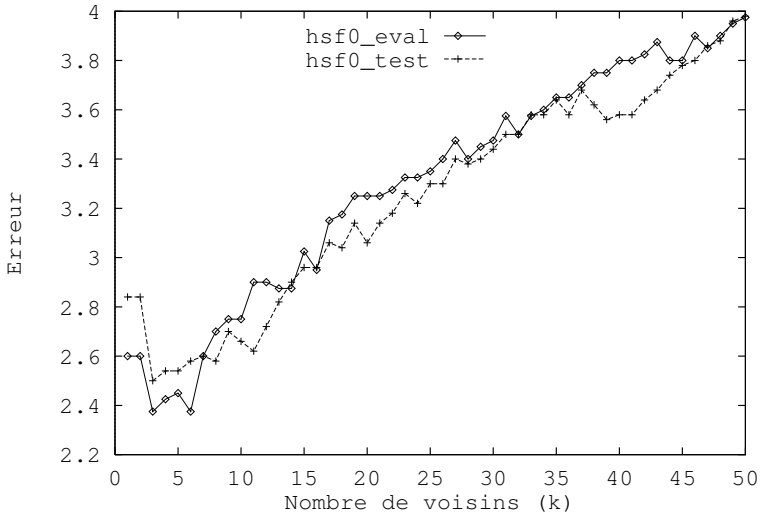


FIGURE 6.15 – Erreur du classifieur kNN sur *hsf0_ eval* et *hsf0_ test* en fonction du nombre k de voisins utilisés.

6.4.1 Quick + kNN

Nous avons d'abord testé un classifieur plus-proches voisins (kNN) [59, 72]. Ce classifieur bien connu n'est pas un réseau connexionniste, ne comporte pas d'apprentissage proprement dit (nous avons ici utilisé les 15 000 exemples de *hsf0_ learn* comme références), et est très lent. kNN sert cependant souvent de point de comparaison pour discuter les performances d'autres classifieurs.

Nous avons ici utilisé les caractéristiques calculées sur l'avant-dernière couche du TDNN Quick (dimension 54) en entrée du kNN. La version de kNN utilisée est la plus simple, avec vote (pluralité). En cas d'égalité, on a choisi la classe de la référence la plus proche, au sens de la distance euclidienne.

La courbe 6.15 donne l'erreur de classification sur *hsf0_ eval* et *hsf0_ test* en fonction du nombre de voisins utilisé pour le vote.

On voit que le nombre optimal de voisins est $k = 3$ (cf. la section 2.5.3).

L'erreur minimale est

hsf0_ eval	2.4 %
hsf0_ test	2.5 %

Notons que le faible nombre de voisins utilisé ne permet pas d'envisager une stratégie cohérente pour le rejet, ce qui est encore un inconvénient du classifieur kNN.

L'erreur a diminué de 1.4 % par rapport au réseau Quick seul. Ce résultat indique clairement que la dernière couche du réseau Quick, qui est totalement connectée, ne réalise pas une classification optimale des caractéristiques extraites. Une explication possible est que l'extracteur de caractéristique serait trop simple (2 couches de connexions) pour que les classes soient linéairement séparables dans l'espace de l'avant-dernière couche. Dans ce cas, une unique couche est insuffisante pour séparer sans erreur toutes les formes.

6.4.2 Quick + LVQ

Le classifieur kNN utilisé dans la section précédente améliore les performances de reconnaissance, au prix d'un volume de calcul prohibitif. Il est alors naturel d'utiliser un autre classifieur, basé lui aussi sur la distance entre les vecteurs de caractéristiques, mais plus efficace. Le classifieur LVQ est un excellent candidat pour cette tâche.

Nous avons détaillé le principe et le fonctionnement du classifieur LVQ dans le chapitre 3.

Algorithme utilisé

La variante de LVQ utilisée ici est LVQ2.1 (voir section 3.5.3).

Nous avons utilisé 20 vecteurs de référence par classe, ce qui semble d'après nos expériences le nombre optimal pour ce problème.

Pour l'initialisation des vecteurs de référence, nous avons employé l'algorithme k-means (voir section 2.7.2) à l'intérieur de chaque classe (20 références).

Performances

L'apprentissage LVQ est rapide : après quelques passes, la position des vecteurs de référence ne varie plus.

On arrive aux taux d'erreurs suivants :

hsf0_learn	1.3 %
hsf0_eval	2.7 %
hsf0_test	2.8 %

On voit que les performances sont dégradées de 0.3 % par rapport à kNN (on gagne un facteur 750 sur le volume de calculs).

L'amélioration par rapport au TDNN Quick seul est supérieure à 1 %, soit un quart d'erreurs en moins.

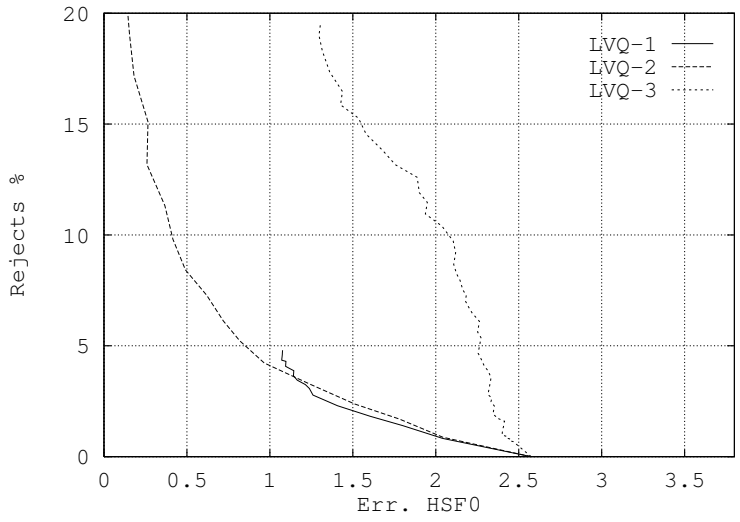


FIGURE 6.16 – Rejet en fonction de l'erreur, sur *hsf0_ eval*, pour les trois critères de rejet LVQ. Réseau Quick+LVQ.

Comparaison des critères de rejet LVQ

Dans le chapitre 3 (section 3.5.4), nous avons proposé plusieurs critères de rejet utilisables avec le classifieur LVQ. Ces critères sont appelés LVQ-1, LVQ-2 et LVQ-3.

Nous allons maintenant comparer les courbes de rejet obtenues par chacun de ces critères.

***Rejet sur* hsf0_eval** La figure 6.16 donne les courbes de rejet sur *hsf0_ eval* en fonction de l'erreur de classification, pour chacun des trois critères de rejet LVQ.

On voit que le critère de rejet LVQ-1 ne permet pas d'arriver à un taux d'erreur nul : les formes telles que au moins deux références d'une autre classe soient les plus proches ne sont jamais rejetées par ce critère.

Le critère LVQ-2, qui considère le rapport des distances aux *classes* les plus proches, prolonge naturellement le critère LVQ-1.

Le critère LVQ-3, qui ne considère que la distance à la référence la plus proche est très mauvais. Il est clair que l'on doit prendre en compte les distances à plusieurs classes pour rejeter efficacement les formes ambiguës.

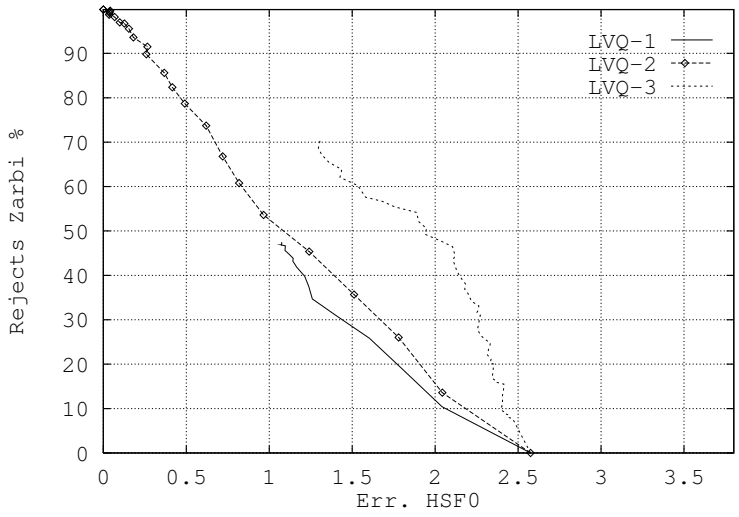


FIGURE 6.17 – Rejet sur *Zarbi* en fonction de l'erreur sur *hsf0_ eval*, pour les trois critères de rejet LVQ. Réseau Quick+LVQ.

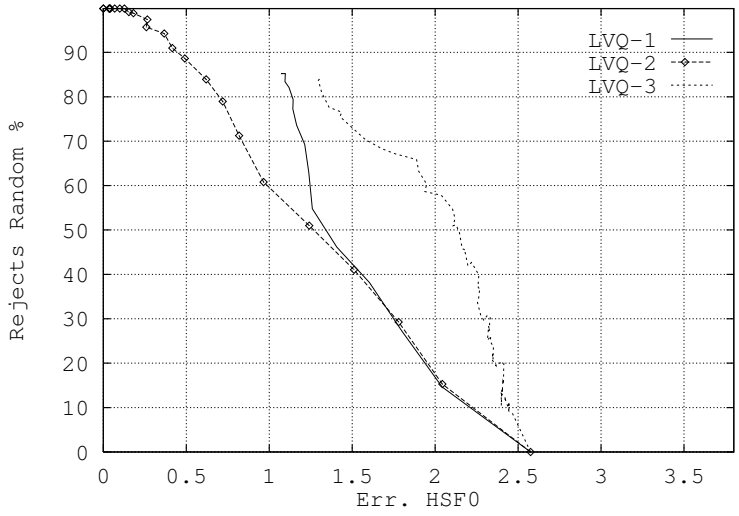


FIGURE 6.18 – Rejet sur *Random* en fonction de l'erreur sur *hsf0_ eval*, pour les trois critères de rejet LVQ. Réseau Quick+LVQ.

Rejet des outliers Les figures 6.17 et 6.18 donnent les taux de rejets sur *Zarbi* et *Random* en fonction de l'erreur de classification correspondante sur les chiffres de *hsf0_ eval*.

Ces résultats indiquent que les critères LVQ-1 et LVQ-2 sont équivalents ; nous préfererons LVQ-2 car il permet de rejeter toutes les formes, pour atteindre 0 % d'erreur.

Le critère LVQ-3 rejette plus efficacement les outliers. C'est le seul critère basé sur une distance absolue de la forme aux références, et non sur un rapport de ces distances ; c'est pourquoi il est plus efficace pour rejeter les formes très éloignées des références.

Notons qu'ici aussi, les formes de *Random* sont plus rejetées que celles de *Zarbi*.

Conclusion Les propriétés de rejet des réseaux LVQ sont globalement similaires à celles des réseaux MLP ; l'allure des courbes de rejet est semblable pour les deux classifieurs.

Nous retiendrons pour toutes nos expériences le critère de rejet LVQ-2, basé sur le rapport des distances de la forme présentée aux deux classes les plus proches. Ce critère est le plus efficace pour détecter les formes ambiguës, et est aussi capable de rejeter les *outliers*.

Conclusions

Le remplacement de la dernière couche du réseau TDNN Quick par un classifieur LVQ permet un gain important de performances (plus de 1 %).

LVQ commet à peine plus d'erreurs que kNN.

Notons aussi que le réseau LVQ nécessite (sans utiliser d'astuces spéciales) 10800 multiplications (i.e. 200 distances euclidiennes dans $\Re^{54}$), contre seulement 550 pour le TDNN (voir table 6.5).

6.4.3 Quick + RBF

Dans cette section, nous allons décrire les résultats obtenus en utilisant un classifieur RBF sur les caractéristiques lues sur l'avant-dernière couche du TDNN Quick.

Algorithme utilisé

Nous utilisons ici l'algorithme RBF décrit dans le chapitre 3 (section 3.8). Le module RBF est connecté à la sortie du module TDNN, mais il n'y a pas

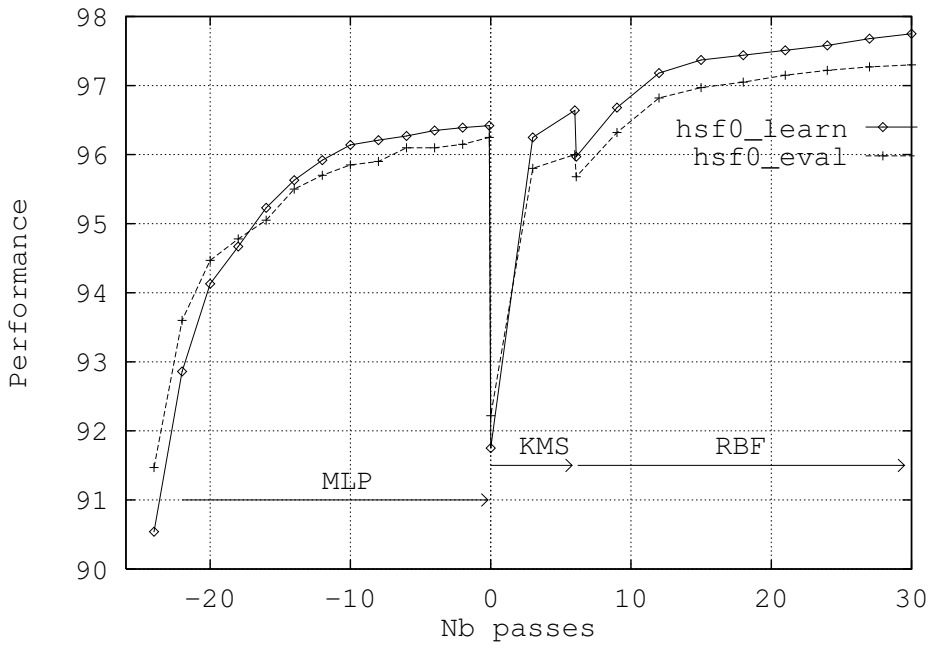


FIGURE 6.19 – Evolution des performances (pourcentage de chiffres bien reconnus) au cours des différentes phases de l'apprentissage du système Quick + RBF. Il s'agit ici d'un apprentissage séquentiel : les poids du TDNN restent fixes durant l'apprentissage du RBF. De gauche à droite, les trois phases : apprentissage du TDNN, k-means sur les caractéristiques, puis initialisation et apprentissage du réseau RBF. A chaque changement de phase, on note une diminution des performances qui est compensée après quelques passes.

modification de ce dernier durant l'apprentissage du RBF : nous sommes en mode *séquentiel*.

Comme pour LVQ, nous avons utilisé 20 centres RBF par classe.

La courbe 6.19 montre l'évolution des performances au cours de l'apprentissage. L'apprentissage RBF est nettement plus lent que l'apprentissage LVQ ; il s'apparente clairement à l'apprentissage d'un réseau MLP, et nécessite quelques dizaines de passes pour converger.

Performances

Le classifieur Quick+RBF arrive aux taux d'erreurs suivants :

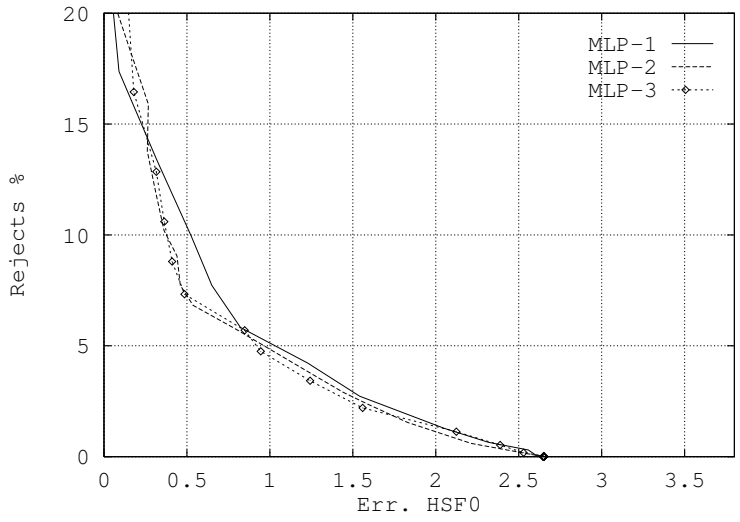


FIGURE 6.20 – Rejet en fonction de l'erreur, sur *hsf0_ eval*, pour les trois critères de rejet RBF.

hsf0_ learn	2.2 %
hsf0_ eval	2.7 %
hsf0_ test	3.0 %

Ces performances sont très proches de celles du réseau LVQ présentées dans la section 6.4.2.

Comparaison des critères de rejet RBF

Les critères de rejets utilisés avec le réseau RBF sont identiques à ceux utilisés avec les réseaux TDNNs : la dernière couche du RBF est en effet une couche de type «perceptron».

***Rejet sur* hsf0_eval** La figure 6.20 donne les courbes de rejet sur *hsf0_ eval* en fonction de l'erreur de classification, pour chacun des trois critères de rejet MLP, appliqués à la sortie du réseau Quick+RBF.

Comme pour le TDNN (voir section 6.3.3, figure 6.10), les trois critères sont équivalents.

D'autre part, les courbes de rejet du réseau LVQ (critère 2) et RBF (critère MLP-3) sont très proches. Il semble que les performances de rejet dépendent en fait peu de la nature du classifieur (et du critère utilisé), mais

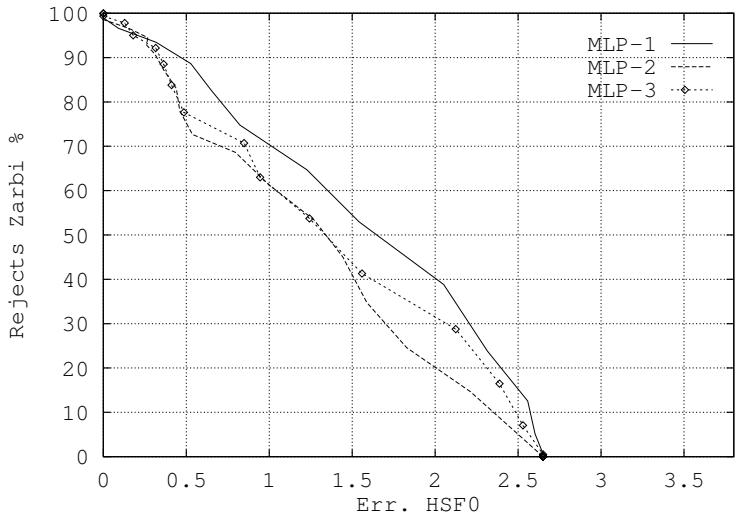


FIGURE 6.21 – Rejet sur *Zarbi* en fonction de l'erreur sur *hsf0_ eval*, pour les trois critères de rejet RBF. Réseau Quick+RBF

soient principalement déterminées par l'erreur de classification sans rejet.

Rejet des outliers Les figures 6.21 et 6.22 donnent les courbes de rejet sur les ensembles *Zarbi* et *Random*.

Les taux de rejets sont semblables pour tous les critères.

Le rejet des *outliers* par RBF est légèrement supérieur à celui du réseau LVQ (5 à 10 % sur de mieux *Zarbi*).

Conclusions

L'algorithme RBF que nous avons mis au point donne des performances du même ordre que celles du réseau LVQ.

Toutefois, l'implémentation d'un réseau RBF est plus complexe, et nécessite plus de calculs (voir table 6.5). L'apprentissage est aussi nettement plus lent.

La légère amélioration des performances de rejet des *outliers* ne justifie par conséquent pas l'emploi d'un classifieur RBF à la place d'un classifieur LVQ.

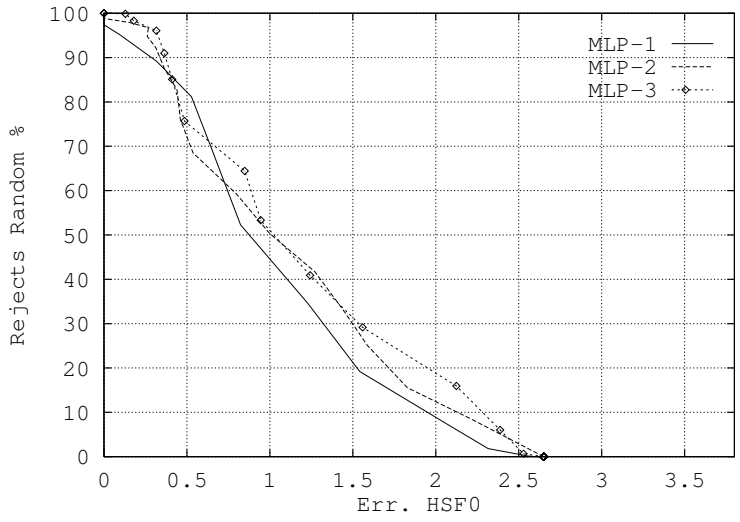


FIGURE 6.22 – Rejet sur *Random* en fonction de l'erreur sur *hsf0_ eval*, pour les trois critères de rejet RBF. Réseau Quick+RBF

6.4.4 Quick + RBF_coop

Nous avons vu dans le chapitre 3 qu'il était possible d'entraîner globalement une architecture multi-modulaire par descente du gradient. L'apprentissage global (ou *coopératif*) permet d'optimiser les paramètres de chaque module en fonction de la tâche réalisée par les suivants. Nous allons ici montrer que l'apprentissage coopératif de l'architecture Quick+RBF permet une nette amélioration des performances. Nous appellerons cette architecture Quick+RBF_coop. Elle ne diffère de la précédente que par le mode d'apprentissage : les poids du TDNN sont modifiés pendant l'apprentissage du RBF.

Algorithme utilisé

L'apprentissage se déroule comme pour le système Quick+RBF sous-optimal : on commence par entaîner le TDNN seul, puis on initialise le réseau RBF en utilisant un algorithme k-means. La seule différence est que, lors de l'apprentissage du réseau RBF, on rétro-propage le gradient de l'erreur à travers le RBF puis le TDNN, et que l'on modifie simultanément les poids des deux modules.

La lenteur relative (par rapport à l'algorithme LVQ) de convergence du

Nombre de centres	5	10	15	20	30
hsf0_learn	2.3	2.2	1.9	1.8	1.8
hsf0_eval	2.6	2.5	2.4	2.2	2.3

TABLE 6.2 – Taux d'erreur, sur l'ensemble d'apprentissage et d'évaluation, du système Quick+RBF_coop, avec différents nombres de centres RBF par classe.

réseau RBF permet au système de s'optimiser globalement ; après apprentissage, les paramètres du TDNN sont considérablement modifiés. La convergence du système Quick+RBF_coop nécessite le même nombre de passes de l'ensemble d'apprentissage que celle du système sous-optimal.

Performances

Nous rapportons ici les résultats du classifieur Quick+RBF_coop obtenus pour différents nombres de centres RBF par classe (en l'absence de critères simples guidant le choix, nous avons toujours employé un nombre fixe de centres par classe).

La table 6.2 donne les taux d'erreurs obtenus. Le nombre optimal de centres par classe est ici aussi de 20. Les performances semblent en tout état de cause peu sensibles au nombre exact de références.

L'erreur la plus faible sur *hsf0_eval* est 2.2 %, soit 0.4 % de moins que celle obtenue par le système Quick+RBF, et 0.1 % de moins que le classifieur Quick+kNN, qui restait jusqu'ici notre meilleur système.

Courbes de rejet

La figure 6.23 donne les courbes de rejet du système Quick+RBF_coop (20 centres/classe), obtenues en utilisant le critère MLP-3.

6.4.5 Conclusions

Dans cette section, nous avons montré qu'il était possible d'améliorer très sensiblement les performances de classification et rejet du TDNN Quick, en remplaçant la dernière couche de ce réseau par un classifieur plus adapté.

Nous avons vu que les classifieurs LVQ et RBF, aux performances similaires, approchent les performance de la méthode kNN.

La mise en œuvre d'un apprentissage coopératif permet de dépasser les performances de kNN, en adaptant l'étage d'extraction de caractéristiques au classifieur.

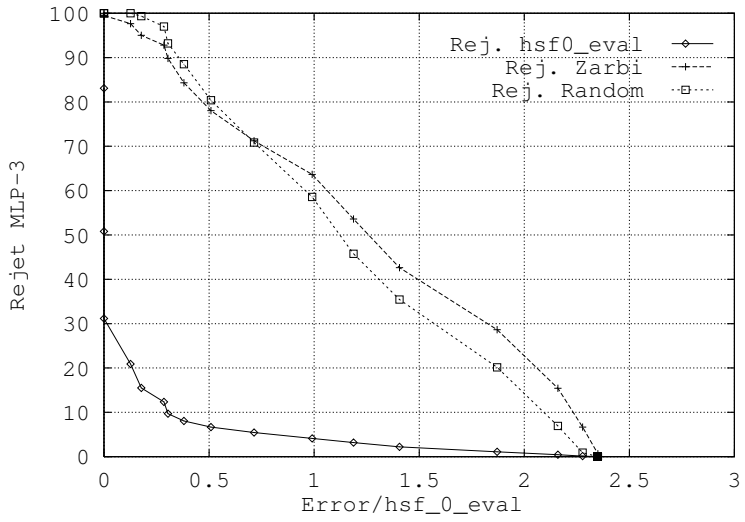


FIGURE 6.23 – Rejet sur *hsf0_ eval*, *Zarbi* et *Random* en fonction de l'erreur sur *hsf0_ eval*. Système Quick+RBF_coop, critère de rejet MLP-3.

Nous avons pu comparer les différentes méthodes de rejets pour chaque algorithme ; nous avons ainsi retenu le critère MLP-3 pour les réseaux TDNN et RBF, et le critère LVQ-2 pour LVQ. Les courbes de rejet, à erreur brute (sans rejet) égale, sont similaires pour tous les classifieurs, et dépendent relativement peu de la forme exacte du critère utilisé.

Si l'approche modulaire «TDNN + X» s'est révélé intéressante, elle ne nous a toutefois pas permis d'approcher les performances du système de référence (le TDNN LeNet). L'architecture Quick+RBF_coop donne un taux d'erreur supérieur de 1 % à celui du réseau LeNet (voir la table 6.6).

D'autre part, les performances de rejet sur les outliers sont décevantes et compromettent l'application de ces classifieurs à des tâches de segmentation. C'est pourquoi nous avons décidé de tester un nouveau type d'architectures, basées sur la mise en œuvre d'un réseau *auto-associatif*, ou *encodeur*. Nous allons détailler cette approche dans la section suivante.

6.5 Architectures modulaires basées sur un MLP encodeur

Jusqu'ici, nous avons utilisé un réseau TDNN comme extracteur de caractéristiques. Ce réseau était entraîné pour séparer les différentes classes de chiffres connues. Les caractéristiques ainsi obtenues sont donc particulièrement adaptées pour différencier les formes à reconnaître, mais ne sont pas conçues pour distinguer les formes inconnues (*outliers*). Ceci expliquerait les mauvaises performances de rejet sur *Zarbi* et *Random*.

Une approche différente consiste à entaîner un extracteur de caractéristiques sans utiliser d'information sur la classe des exemples présentés : il s'agit alors d'un apprentissage non supervisé. Après la mise au point d'un tel module, on utilise un deuxième étage indépendant pour la classification. Il est aussi possible d'affiner l'extracteur de caractéristiques en appliquant un apprentissage coopératif.

L'extracteur de caractéristiques adaptatif non-supervisé le plus simple est un réseau «diabolo». Ce type de réseaux possède une couche cachée et apprend à reconstruire en sortie la forme présentée en entrée. La couche cachée calcule donc un codage optimal des formes présentées durant l'apprentissage. Si le réseau est totalement connecté, et la fonction de transfert linéaire, ce codage est équivalent à une analyse en composantes principales (ACP) [21, 73]. Ce type d'analyse est depuis longtemps très utilisé en reconnaissance de formes (voir par exemple [43, 204]), pour réduire la dimension des données à traiter en minimisant la perte d'information.

Dans cette section, nous appellerons ces réseaux entraînés en auto-association «MLP encodeurs», pour les distinguer des MLP utilisés par ailleurs et utilisés en discrimination.

6.5.1 Réseaux encodeurs

Les réseaux diabolos habituels utilisés directement sur l'image à traiter ont un nombre de poids très élevé, du fait des connexions totales entre les couches de cellules. Ceci présente les inconvénients classiques : trop de paramètres libres (*overfitting*) et temps de calcul dissuasifs pour la plupart des applications [5]. C'est pourquoi de nombreux systèmes de compression d'images [158] ne travaillent pas directement sur les images à comprimer, mais les découpent préalablement en petites imagettes (de taille 4x4 par exemple), et utilisent un seul réseau diabolo (ou une transformation KL) pour compri-

5. Cottrell [45] propose pourtant l'utilisation de réseaux pour comprimer des images 64x64, soit 4096x80x4096.

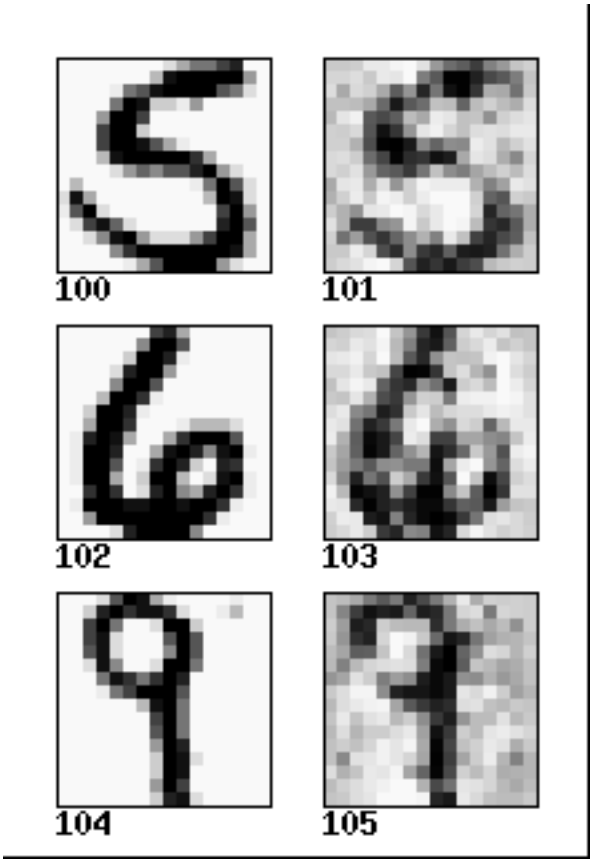


FIGURE 6.24 – Chiffres de *hsf0_ eval* avant et après compression par le diabolo totalement connecté.

mer les imagettes. Cette technique présente l'avantage de réduire la dimension de l'espace d'entrée, et de générer du même coup beaucoup d'exemples d'apprentissage (chaque image est découpée en une multitude d'imagettes).

Afin de tirer parti de cette idée tout en conservant une architecture simple pour extraire des caractéristiques, on peut utiliser un seul réseau diabolo dont les couches ne sont plus totalement connectées. Chaque cellule de la couche cachée «voit» alors une imagette sur la rétine, par des connexions locales. Si de plus les poids sont partagés, le réseau obtenu est équivalent aux systèmes de compression d'image habituels.

Diabolo totalement connecté

Nous avons commencé ces expériences en testant un réseau diabolo totalement connecté pour la compression de chiffres. Ce réseau possède 54 cellules

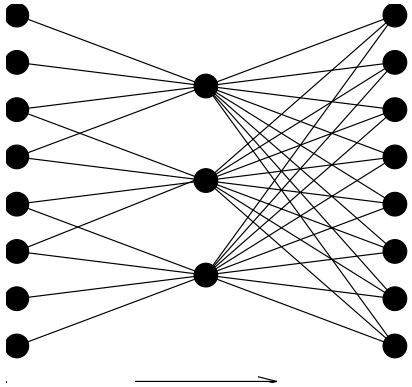


FIGURE 6.25 – Architecture du diabolo à connexions locales, dessinée en une dimension. Le chiffre est présenté sur la rétine (à gauche) et reconstruit sur la sortie (à droite). Les cellules de la couche cachée sont connectées à un voisinage 4x4 sur la rétine (*receiptive field* ou connexions locales), mais les poids ne sont pas partagés par les différentes cellules.

cachées (nous avons choisi ce nombre arbitrairement afin de garder la même dimension que dans la section précédente). Les couches d'entrée et de sortie possèdent chacune 256 cellules.

Ce réseau arrive à une erreur quadratique moyenne de 0.032 sur *hsf0_ eval* après quelques passes (l'erreur remonte si l'on poursuit l'apprentissage). La figure 6.24 montre quelques chiffres avant et après compression/décompression par ce réseau. La qualité de reconstitution est assez médiocre.

Diabolo à connexions locales

Nous présentons ici un réseau diabolo dont les couches ne sont plus totalement connectées : ce réseau utilise des connexions locales entre les deux premières couches. Les connexions locales sont 4x4, se recouvrant de 2 dans chaque direction. La deuxième couche utilise des connexions complètes (réseau nommé «256x49x256 local») ou locales à nouveau (on a alors une architecture symétrique) pour le réseau «256x49x256 local-local».

Le réseau «256x49x256 local» arrive à une erreur quadratique moyenne trois fois plus faible que le diabolo à connexions complètes (table 6.3). La figure 6.26 montre l'effet de ce réseau sur quelques chiffres : la qualité de restitution est nettement meilleure.

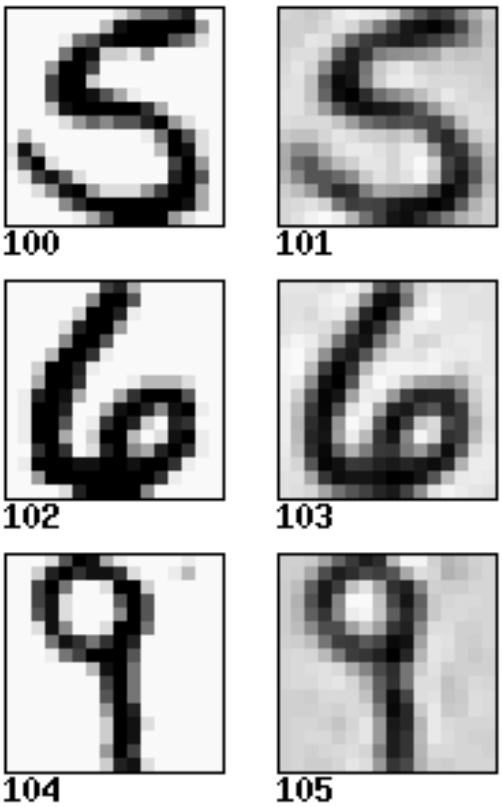


FIGURE 6.26 – Chiffres de *hsf0_ eval* avant et après compression par le diabolo à connexions locales.

Réseau	Erreur *hsf0_learn*	Erreur *hsf0_eval*
256x10x256	0.026	0.026
256x54x256	0.031	0.032
256x49x256 local	0.013	0.013
256x49x256 local partagé	0.014	0.014
256x49x256 local-local	0.016	0.016

TABLE 6.3 – Erreur quadratique moyenne de reconstruction pour différentes architectures diabolo. La notation «NxHxN» désigne comme d'habitude le nombre de cellules sur chaque couche. Le terme *local* qualifie les connexions locales décrites dans le texte, et *local partagé* l'utilisation de poids partagés. Enfin, l'architecture *local-local* utilise des connexions locales sur les deux couches.

Diabolo à connexions locales partagées

Nous avons aussi testé un diabolo avec la même architecture que le précédent (256x49x256 local) mais ou' l'on partage les poids des connexions locales, de façon à obtenir un masque de poids partagés. Le codage obtenu sur la couche cachée est alors le même pour toutes les cellules (traitement invariant par translation).

Ce système offre des performances très semblables au précédent (table 6.3).

Comparaison des architectures diabolos

La table 6.3 indique les erreurs quadratiques moyenne de reconstruction obtenues par différentes architectures diabolos que nous venons de présenter, toutes entraînées de la même façon.

On voit que la meilleure architecture est celle utilisant des connexions locales sur la première couche.

Nous avons représenté sur la figure 6.27 les poids du diabolo à connexions locales après apprentissage auto-associatif. Comme dans la plupart des cas, même pour cette architecture extrêmement simple, on ne peut pas interpréter visuellement le traitement effectué par cette couche de cellules.

6.5.2 MLP encodeur + MLP

Nous présentons rapidement ici une tentative de construction d'un classifieur MLP basé sur le réseau encodeur à connexions locales.

La façon la plus directe d'utiliser le codage extrait par la première couche du MLP encodeur pour construire un classifieur consiste à utiliser un réseau

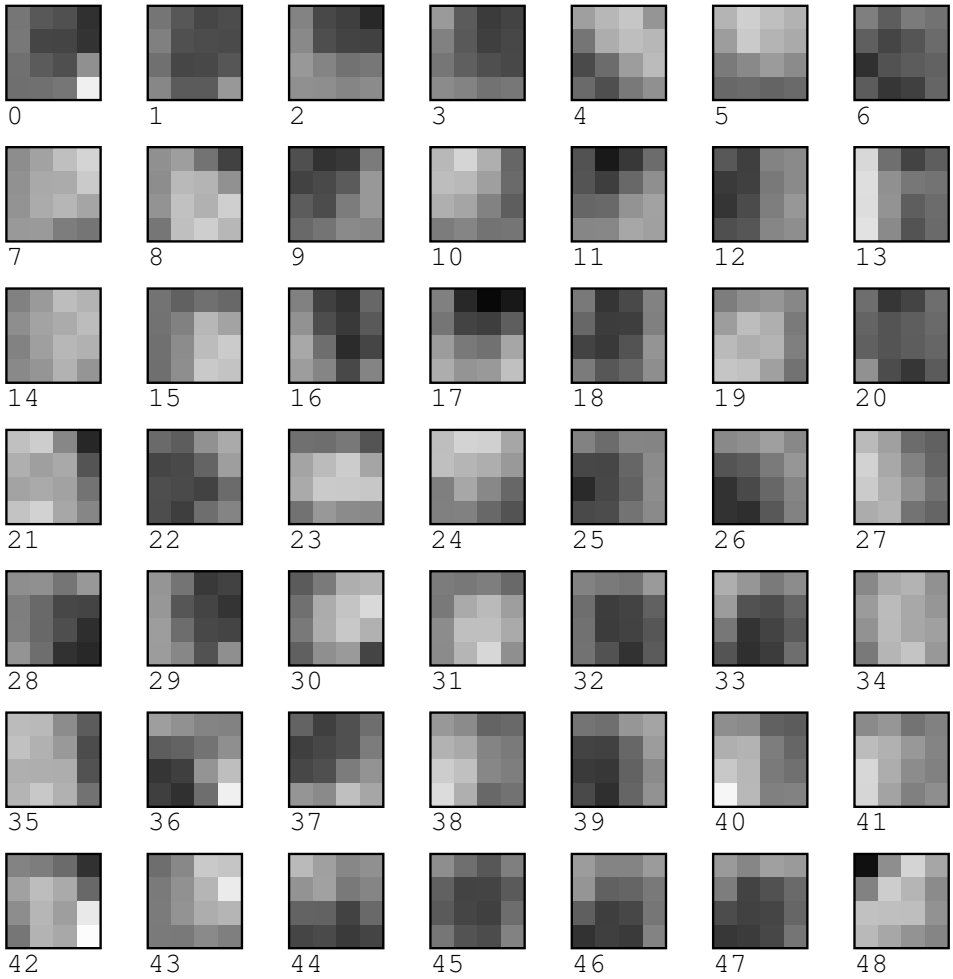


FIGURE 6.27 – Poids synaptiques associés aux connexions de la première couche du diabolo *local*, après l'apprentissage en auto-association. Les poids positifs sont en blancs, et les négatifs en noir. On a dessiné les 16 poids arrivant sur chacune des 49 cellules de la couche cachée. Les valeurs des biais ne sont pas représentées.

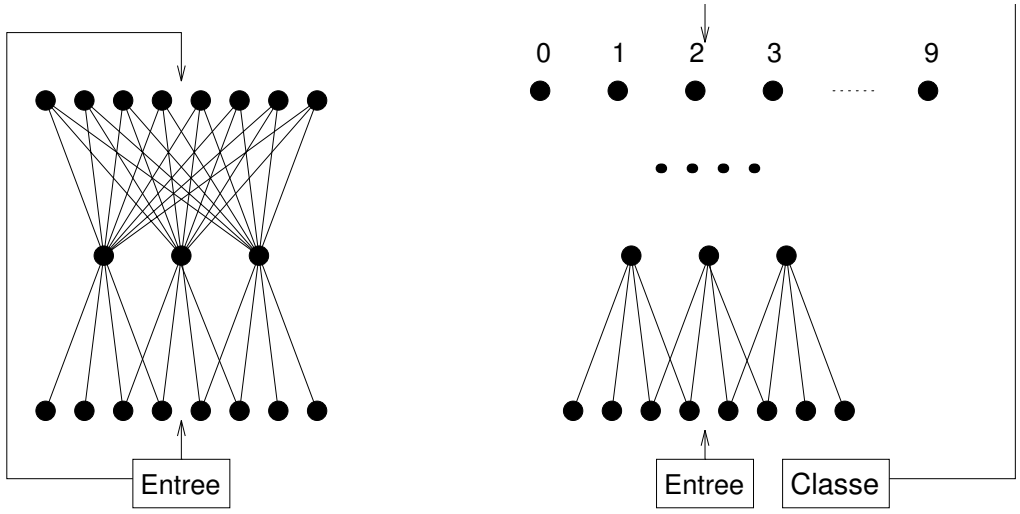


FIGURE 6.28 – Les deux phases de l'apprentissage du système «MLP encodeur + MLP». Lors de la phase 2 (à droite), on mène un apprentissage MLP classique, par rétropropagation de l'erreur de classification.

classifieur MLP connecté sur la couche interne de l'encodeur (voir figure 6.28).

Dans cette solution, l'apprentissage auto-associatif peut être vu comme un moyen d'initialiser les paramètres de la première couche du réseau final, afin de guider l'apprentissage de ce dernier.

Le deuxième MLP utilisé pour la classification peut utiliser à son tour plusieurs couches de cellules, avec des connexions locales ou non. L'apprentissage du système global (phase 2) porte sur toutes les couches de poids.

Il serait un peu surprenant que ce système arrive à des performances satisfaisantes car le réseau résultant reste nettement plus simple que l'architecture TDNN «Quick» présentée en section 6.3.2, dont les performances restaient faibles. Nous mentionnons ici cette architecture dans le but de comparer les performances de classification MLP et du RBF que nous allons présenter plus bas.

Après quelques essais d'architectures simples pour le deuxième MLP, nous avons constaté que la combinaison de 2 couches, comportant chacune 10 cellules, totalement connectées était optimale (le réseau complet comporte 4 couches de cellules, de tailles 256-49-10-10). Les performances mesurées sont les suivantes :

hsf0_learn	3.4	%
hsf0_eval	4.3	%

Réseau	Erreur *hsf0_learn*	Erreur *hsf0_eval*
256x10x256	10.0	10.1
256x54x256	1.6	3.9
256x49x256 local	0.8	2.4
256x49x256 local partagé	1.0	2.9
256x49x256 local-local	1.0	2.6

TABLE 6.4 – Erreurs de classification LVQ sur la couche interne des diabolos.

soit environ 0.5 % d'erreur de plus que le TDNN Quick.

En conclusion, ce mode d'apprentissage n'améliore pas les performances des classifieurs MLP.

6.5.3 MLP encodeur + LVQ

Nous allons maintenant utiliser un classifieur LVQ pour classer les caractéristiques extraites par la couche cachée du diabolo auto-associatif. On utilise ici le même algorithme LVQ qu'en section 6.4.2, avec 20 vecteurs de référence par classe (en dimension 49).

La table 6.4 donne les erreurs de classification mesurées sur *hsf0_learn* et *hsf0_eval* pour les différentes combinaisons «diabolo + LVQ».

Le meilleur système est ici aussi le diabolo à connexions locales. Ce résultat est parfaitement cohérent avec celui obtenu plus haut (table 6.3) : cette architecture réalise le meilleur codage pour la reconstruction et la classification.

Il est intéressant de noter que l'utilisation de LVQ permet de diviser l'erreur par un facteur deux environ par rapport au MLP présenté en 6.5.2.

Les performances du système «diabolo-local + LVQ [6]» sont légèrement inférieures aux performances du classifieur Quick+RBF_coop (voir section 6.4.4).

La figure 6.29 donne les courbes de rejet obtenues avec ce système. Contrairement à nos attentes, le rejet des *outliers* est inférieur à celui du système Quick+RBF_coop (figure 6.23).

6.5.4 MLP encodeur + RBF_coop

Nous n'avons pas testé le classifieur MLP+RBF séquentiel. Nous avons en effet remarqué lors des études sur Quick+RBF que les performances des classifieurs RBF et LVQ étaient très proches.

6. Pour alléger les notations, on appelera dorénavant les architectures «diabolo-local + X» MLP +X.

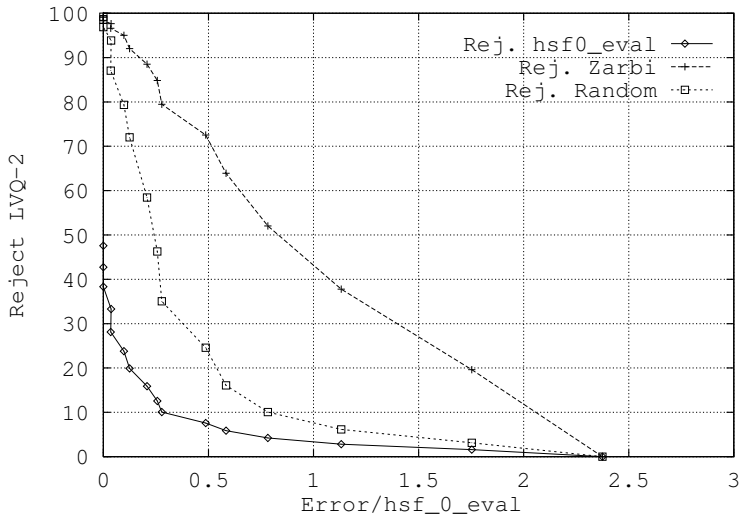


FIGURE 6.29 – Rejet sur *hsf0_ eval*, *Zarbi* et *Random* en fonction de l'erreur sur *hsf0_ eval*, par le classifieur MLP+LVQ (critère LVQ-2).

Par contre, le système MLP+RBF_coop, entraîné coopérativement, s'est révélé très performant.

Mode d'apprentissage

Le classifieur MLP+RBF_coop s'utilise exactement comme Quick+RBF_coop (voir section 6.4.4) : on entraine d'abord le réseau diabolo en auto-association, puis l'on applique la procédure habituelle d'initialisation et d'apprentissage du réseau RBF sur la couche interne.

On obtient avec ce classifieur les taux d'erreur suivants :

hsf0_ learn	1.3 %
hsf0_ eval	1.9 %
hsf0_ test	2.0 %

Du fait de la simplicité du module «MLP» (1 couche, 49 cellules et 833 poids), l'apprentissage coopératif du réseau RBF modifie profondément les paramètres de ce module, comme le montre la figure 6.30. La phase d'apprentissage en auto-association est simplement utilisée pour atteindre des valeurs de poids sur la première couche permettant une bonne initialisation du système MLP+RBF_coop. Cette initialisation est indispensable pour permettre de placer les références RBF et leur assigner une *largeur* σ correcte. Si l'on

FIGURE 6.30 – Poids synaptiques associés aux connexions de la première couche du diabolo *local*, après apprentissage coopératif MLP+RBF_coop. La comparaison avec la figure 6.27 montre que l'apprentissage coopératif a grandement modifié cette couche de poids. Ce fait est confirmé par le mesure de l'erreur de reconstruction du diabolo : pour cela, on a reconnecté la deuxième couche de ce dernier et calculé l'erreur sur *hsf0_eval*. L'erreur moyenne atteint maintenant 0.026, contre 0.013 précédement (table 6.3) ; le diabolo n'est plus utilisable pour comprimer les images.

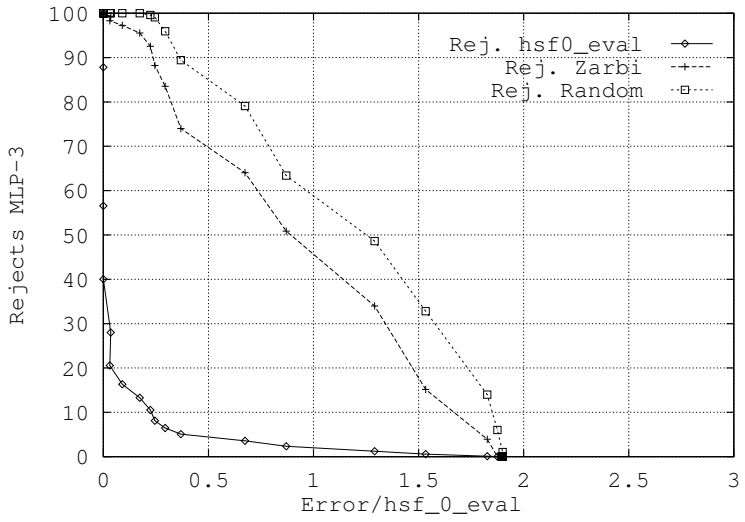


FIGURE 6.31 – Rejet sur *hsf0_ eval*, *Zarbi* et *Random* en fonction de l'erreur sur *hsf0_ eval*. Système MLP+RBF_coop, critère de rejet MLP-3.

tente d'apprendre directement le système MLP+RBF_coop en initialisant les poids du MLP de la façon habituelle (valeurs aléatoires suivant une loi uniforme), l'apprentissage reste en général bloqué dans un minimum local (les centres RBF sont trop mal positionnés) et les performances de classification sont très mauvaises.

Rejet

Les courbes de rejet sont tracées sur la figure 6.31.

Ces résultats montrent que le système MLP+RBF_coop est indiscutablement la meilleure architecture multi-modulaire que nous ayons mis au point ; les erreurs de classification et le rejet des *outliers* sont supérieurs à celles obtenues par les architectures Quick+RBF_coop ou MLP+LVQ.

6.6 Récapitulation des résultats

Nous résumons dans cette section les résultats présentés dans ce chapitre.

La table 6.5 indique la complexité de chaque module, exprimée en nombre de neurones (cellules), de poids (paramètres libres du module) et de connexions. Le nombre de multiplications nécessaire au calcul de la sortie d'un module

Architectures mono-modulaires (Module 1)		MLP-encodeur	TDNN LeNet	TDNN Quick
	neurones	305	4624	384
	connexions	833	96512	1798
	poids	65	648	646
Module classifieur (Module 2)				
aucun	neurones	256 [a]	10	10
	connexions	12800	1930	550
	poids	12800	1930	550
kNN				15000 R^{54} dist.
LVQ	connexions	9800	(38400)	10800
RBF	connexions	(12010)	(40610)	13010
RBF_coop	connexions	12010	(40610)	13010

a. Ce système ne réalise pas de classification.

TABLE 6.5 – Complexités des différentes architectures mono et multi-modulaires présentées dans ce chapitre. Cette table indique le nombre de cellules (neurones) de l'extracteur de caractéristiques (module 1), ainsi que les nombres de poids et de connexions pour chaque module. Notons que pour les classifieurs LVQ et RBF, le nombre de poids et de connexions sont égaux (pas de poids partagés). Le nombre de multiplications nécessaire est toujours donné par le nombre de connexions.

est égal au nombre de connexions de ce module. On voit que le classifieur MLP+RBF_coop est 8 fois moins coûteux en calculs que le TDNN LeNet.

La table 6.6 résume les taux d'erreurs mesurés sans rejet sur les trois ensembles (apprentissage, évaluation et tests) de chiffres extraits de la base NIST. Le meilleur classifieur est le TDNN LeNet. Notre système multi-modulaire MLP + RBF_coop commet 0.5 % d'erreurs en plus. Ce dernier système est par contre supérieur à tous les autres «petits» systèmes présentés.

Enfin, la table 6.7 donne les taux d'erreur et de rejet sur les différents ensembles, mesurés lorsque l'on ajuste le seuil de rejet de chaque système de façon à obtenir 1 % d'erreur sur *hsf0_ eval*. On constate là aussi que LeNet est supérieur pour le rejet des formes ambiguës : pour atteindre 1 %, il lui faut rejeter 0.8 % des chiffres de *hsf0_ test*, contre 1.8 % pour MLP+RBF_coop.

Par contre, le rejet des *outliers* (*Zarbi* et *Random*) est bien plus faible (27 % pour LeNet, contre 48 % pour MLP+RBF_coop). Cette comparaison n'est pas très significative, dans la mesure ou' LeNet rejette très peu pour atteindre 1 % d'erreur (le seuil de rejet est très haut). Il est plus correct de

Données / Architecture	*hsf0_learn*	*hsf0_eval*	*hsf0_test*
Mono-modulaire			
MLP-encodeur [a]	0.014	0.014	0.014
LeNet	1.1	1.4	1.4
Quick	3	3.8	4.1
Multi-modulaire			
MLP+MLP	3.4	4.3	nd [b]
MLP+LVQ	1.1	2.7	3.1
MLP+RBF_coop	1.3	1.9	2.0
Quick+kNN	0	2.4	2.5
Quick+LVQ	1.3	2.7	2.8
Quick+RBF	2.2	2.7	3.0
Quick+RBF_coop	1.8	2.3	2.5

a. Erreur Quadratique moyenne de reconstruction
b. Non mesurée

TABLE 6.6 – Erreur de classification sans rejet sur les chiffres de la base NIST.

comparer le rejet des *outliers* à taux de rejet d'ambiguïté fixé, et non pas à erreur de classification fixée. Nous reviendrons sur ce point dans la section suivante.

6.6.1 Critère de mesure des performances pour la segmentation

Nous nous intéressons dans cette section à la façon de définir les performances d'un classifieur en vue d'une application de segmentation, afin de pouvoir comparer objectivement dans ce cas les propriétés des systèmes présentés dans les sections précédentes.

Courbe de détection en segmentation

Dans une tâche de segmentation, on doit contrôler deux taux :

1. taux de non détection («faux négatifs») : il s'agit dans notre cas des chiffres de *hsf0_eval* qui sont rejetés ;
2. taux de fausse détection («faux positifs» ou fausses alarmes) : ici, les formes de *Zarbi* et *Random* qui ne sont pas rejetées.

Données	*hsf0_ eval*		*hsf0_ test*		*Zarbi*	*Random*
Taux	rejet	erreur	rejet	erreur	rejet	rejet
Architecture						
Mono-modulaire						
LeNet	0.8	1	0.8	1	27	12
Quick	8.8	1	8.5	1	75	87
Multi-modular						
MLP+LVQ	4.9	1	5.7	1	43	8
MLP+RBF_coop	1.8	1	1.8	1.2	48	59
Quick+LVQ	3.9	1	4.1	1	53	61
Quick+RBF	4.5	1	5.2	0.8	62	53
Quick+RBF_coop	3.9	1	4.4	0.75	64	59

TABLE 6.7 – Taux de rejet obtenus par les systèmes calibrés pour donner 1 % d'erreur sur *hsf0_ eval*. (Quick+kNN ne figure pas car il ne permet pas d'effectuer de rejet.)

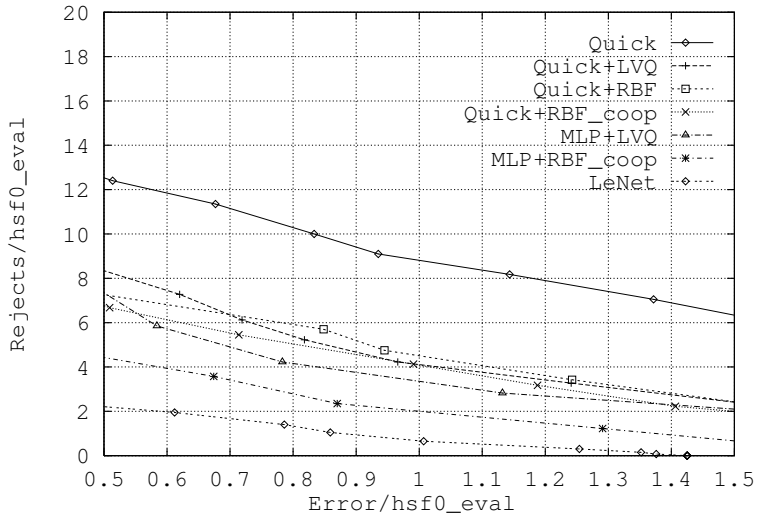


FIGURE 6.32 – Rejet sur *hsf0_ eval* en fonction de l'erreur sur *hsf0_ eval* pour les différents systèmes.

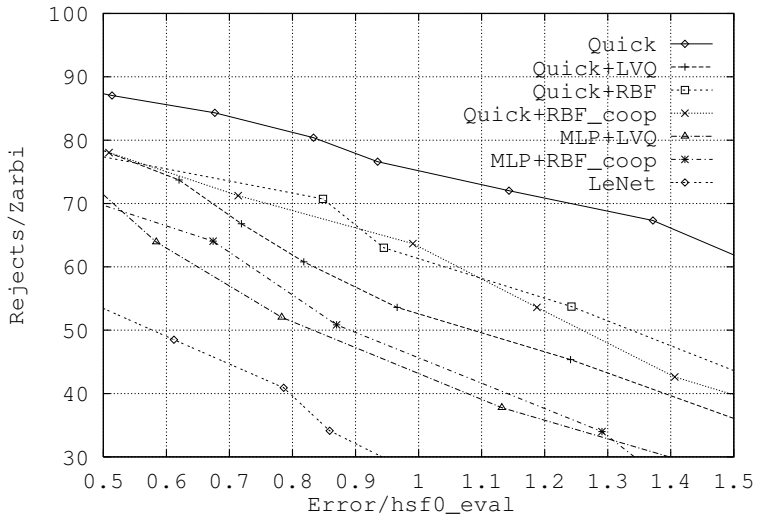


FIGURE 6.33 – Rejet sur *Zarbi* en fonction de l'erreur sur *hsf0_ eval* pour les différents systèmes.

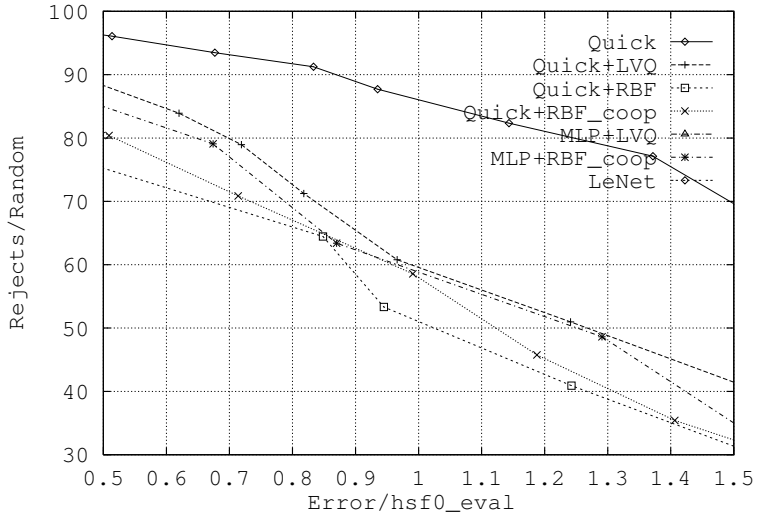


FIGURE 6.34 – Rejet sur *Random* en fonction de l'erreur sur *hsf0_ eval* pour les différents systèmes (certaines courbes sortent de l'échelle car les taux de rejet sont trop faibles).

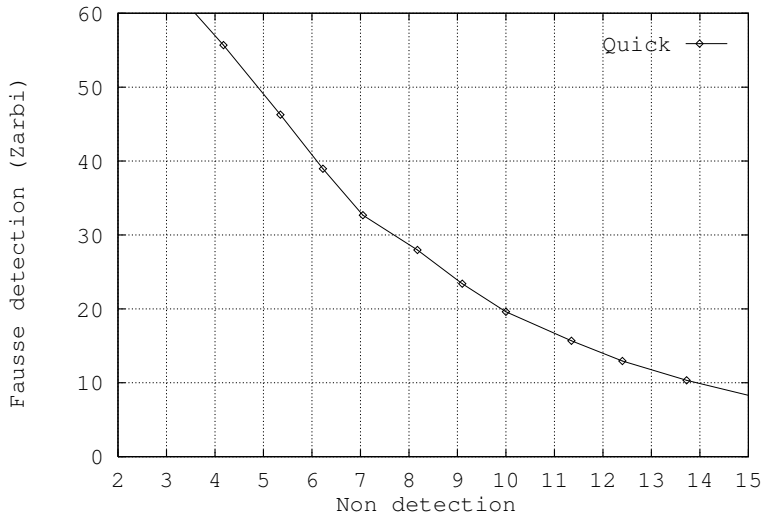


FIGURE 6.35 – Taux de fausse détection (acceptation mesurée sur *Zarbi*) en fonction du taux de non détection (rejet sur *hsf0_ eval*).

Le meilleur système est celui qui offre le plus faible taux de fausse détection pour un niveau de non détection donné. Typiquement, la question posée est du type : «si j'accepte de ne pas détecter X % des objets dans l'image, combien de fausses alarmes vais-je obtenir ?».

Nous avons tracé sur les figures 6.35 et 6.36 les courbes donnant le taux de fausse détection en fonction du taux de non détection, mesurées sur *Zarbi* et *Random*.

Sur les *outliers* de *Zarbi* (figure 6.35), ressemblant à des chiffres, on voit que les performances en détection de LeNet et de MLP+RBF_coop sont extrêmement proches. LeNet est très légèrement meilleur, quoique de façon non significative statistiquement.

Sur les *outliers* de *Random* (figure 6.36), le système MLP+RBF_coop est franchement meilleur : il accepte 20 % des formes de *Random* en moins. (Curieusement, le plus mauvais système est ici MLP+LVQ ; les autres systèmes sont en gros équivalents).

Taux d'erreur global en segmentation + reconnaissance

Si les processus de segmentation et de reconnaissance sont couplés et traités par le même classifieur, on peut définir un *taux d'erreur global*.

Le classifieur va rencontrer des formes «correctes» (dans notre cas les

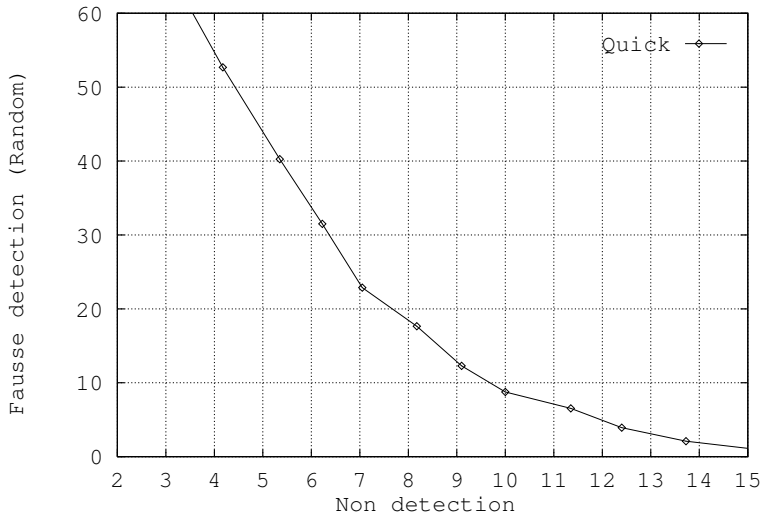


FIGURE 6.36 – Taux de fausse détection (acceptation mesurée sur *Random*) en fonction du taux de non détection (rejet sur *hsf0_ eval*).

chiffres de *hsf0*), et des formes «incorrectes» (*outliers* de *Zarbi* et *Random*). Notons p la proportion des formes correctes. Si l'on utilise en amont un segmenteur précis, p sera proche de un.

Le classifieur prend la décision de rejeter ou de classer chaque nouvelle forme présentée. Soit e_g le taux d'erreur commis sur les formes correctes (e.g. le taux d'erreur sur *hsf0*), τ_g et τ_b les taux de rejet sur les formes correctes et incorrectes (rejet sur *hsf0* et *Zarbi* ou *Random*).

L'erreur globale τ_e provient des deux sources suivantes : erreur de classification sur les formes correctes (c'est la seule quantité que l'on mesure habituellement) et décision de classer des formes incorrectes. Ce qui s'écrit simplement :

$$\tau_e = p\,(1-\tau_g)e_g + (1-p)(1-\tau_b) \tag{6.3}$$

Dans cette équation, les quantités τ_g, e_g and τ_b sont toutes fonction d'un seuil de rejet θ. Comme leur variation est monotone, on peut éliminer θ pour exprimer e_g et τ_b en fonction de τ_g.

Pour notre application, on peut constater que l'erreur globale τ_e est toujours inférieure avec LeNet qu'avec les autres architectures. En effet, pour toutes les valeurs de τ_g :

— e_g est inférieure pour LeNet, comme le montre la figure 6.32 ;

— le taux de fausse détection $1-\tau_b$ est plus petit pour LeNet que pour

les autres architectures (figure 6.35), si l'on utilise les performances sur *Zarbi*.

Nous avons tracé sur la figure 6.37 le taux de non détection en fonction du taux d'erreur global, pour différentes composition du flux de formes à l'entrée du classifieur. On conclut que :

— LeNet est comme prévu toujours meilleur ;
— pour $p \approx 1$ (fig. 6.37a), les courbes sont séparées par environ 2 % ;
— pour $p = 0.9$, les courbes sont pratiquement superposées.

Conclusion

En conclusion, pour une application de segmentation-reconnaissance, on doit sans aucun doute utiliser LeNet si le segmenteur amont est parfait. Par contre, si le segmenteur laisse passer une proportion importante de formes ne correspondant pas à des chiffres, on a intérêt à utiliser le classifieur MLP+RBF_coop, qui, à taux d'erreur équivalent, nécessite 8 fois moins de calculs.

Dans le système *Shortest Path Segmentation* proposé par Burges et al. [31], qui est une des meilleures méthodes de lecture de codes postaux actuelles, (voir chapitre 5, section 5.4.3) on génère dans une première phase un certain nombre d'hypothèses de segmentation. Si ce nombre est trop faible, on court le risque de rater la solution. On est donc contraint de générer de nombreuses hypothèses. Ces hypothèses sont présentées à un classifieur, qui se trouve ainsi en présence d'un proportion importante ($\approx$ 10 %) d'*outliers*. Dans ce type de situations, à moins de disposer de *hardware* spécialisé éliminant le problème du temps de traitement, on a tout intérêt à utiliser un classifieur rapide comme MLP+RBF_coop.

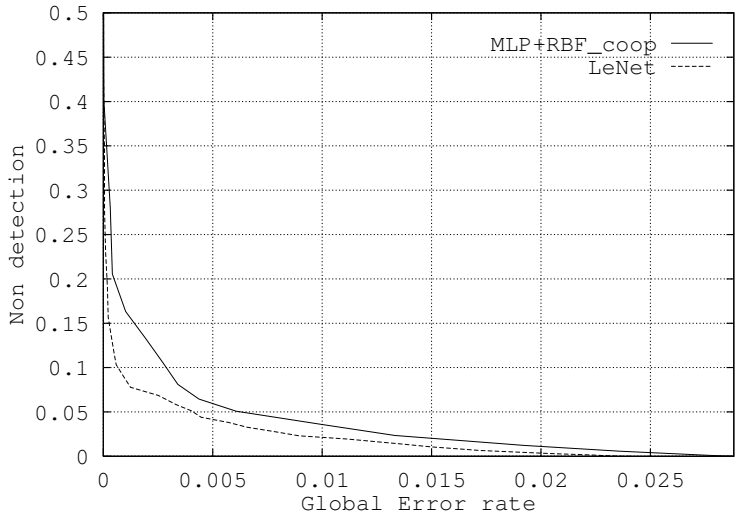
6.7 Un exemple d'application : segmentation de chiffres

Avant de conclure ce chapitre, nous allons décrire une application très simple de lecture de champs de chiffres basées sur les classifieurs décrits plus haut.

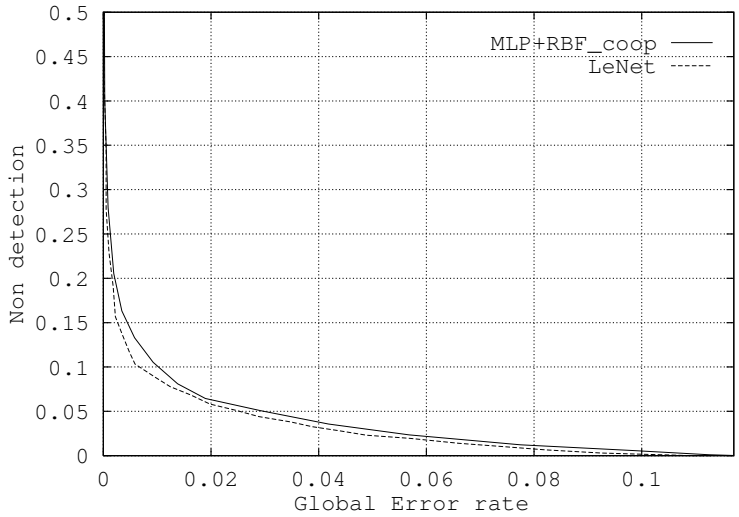
Nous n'avons pas développé durant cette thèse un système opérationnel complet de lecture de champs de chiffres, tâche qui occuperait facilement plusieurs années et ne serait pas forcément d'un grand intérêt scientifique.

Par contre, il nous a semblé intéressant d'appliquer les idées présentées plus haut dans un cas simple, afin d'illustrer notre propos.

La méthode retenue repose sur les principes suivants :



(a) $p = 0.99$



(b) $p = 0.90$

FIGURE 6.37 – Taux de non détection τ_g en fonction du taux d'erreur global τ_e, pour deux types de situations : en haut, on présente au classifieur 99 % de formes correctes ($p = 0.99$) ; en bas, on présente 10 % d'*outliers* (de *Zarbi*). On a tracé les courbes pour les deux meilleurs systèmes, LeNet et MLP+RBF_coop.

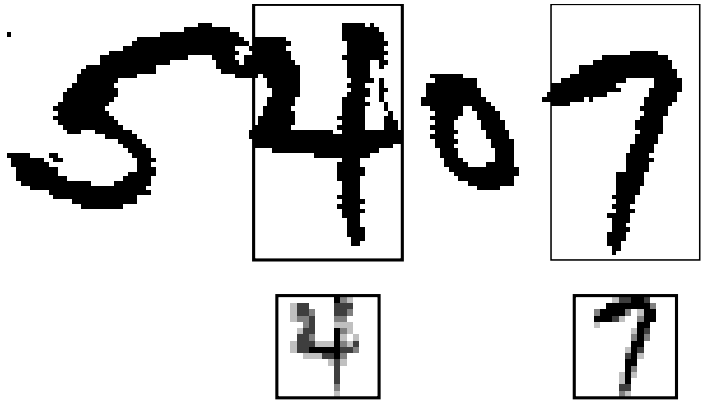


FIGURE 6.38 – Cette figure montre un champ de chiffres (extrait de la base NIST) typique, sur lequel on a dessiné deux positions de la fenêtre utilisée lors du balayage. En chaque position de la fenêtre, on passe au réseau classifieur l'image obtenue en resserrant un rectangle (*lasso*) tangent à l'écriture, puis on normalisant (ré-échantillonnage) l'image pour la mettre à la dimension de la rétine. Nous avons dessiné ici les deux rétines correspondantes (de taille 16x16).

1. appliquer le minimum de pré-traitements spécifiques ;
2. n'utiliser aucune hypothèse de segmentation : on se place délibérément dans le cas ou' le classifieur va rencontrer plus d'*outliers* que de chiffres corrects ;
3. utiliser un classifieur entraîné uniquement sur des chiffres centrés, afin de rester dans la problématique de ce chapitre.

On va donc balayer l'ensemble du champ à lire pour détecter les positions correspondant à des chiffres. Notez que cette approche est proche de celle que nous allons appliquer pour la localisation de visages (voir chapitre 8), cas dans lequel il n'existe pas de méthode adéquate permettant de générer des hypothèses de segmentation.

6.7.1 Description du système

Le but est de lire un champ de chiffres manuscrits comme ceux présentés sur la figure 5.1 (page 109).

Le seul pré-traitement utilisé est un redressement de l'image (global sur le champ), similaire à celui présenté en section 6.2.4.

Pour cela, on se donne une fenêtre de largeur fixe w_0. On va passer cette fenêtre successivement sur toutes les positions horizontales (voir figure 6.38).

En chaque position, on calcule la sortie du classifieur, ainsi que la valeur du critère de rejet. On peut ensuite tracer ces valeurs sur un axe horizontal, comme sur les figures 6.39 et 6.40.

La lecture de ces courbes permet ensuite de reconnaître le texte. La méthode de lecture la plus adaptée doit être un algorithme d'alignement type Viterbi [69]. Nous n'avons pas implémenté ce module de post-traitement.

6.7.2 Exemples

Nous n'avons pas mesuré précisément les performances du système, car nous ne disposions pas de base de données de champs de chiffres importante et n'avons pas implémenté de post-traitement.

Aussi présentons-nous simplement deux exemples sur les figures 6.39 et 6.40. Dans les deux cas, on peut obtenir la réponse correcte par simple seuillage de la courbe de rejet.

6.8 Conclusion

Pour conclure ce chapitre, récapitulons les principaux résultats auquels nous sommes arrivés.

6.8.1 Comparaison des classifieurs LVQ et RBF

Nous avons comparé le classifieur LVQ2, bien étudié dans la littérature [122, 152, 56, 57, 123] avec le classifieur RBF que nous avons proposé, basé sur l'optimisation conjointe de l'ensemble des paramètres par descente du gradient. Nous avons mené cette comparaison (sections 6.4.2 et 6.4.3) en utilisant les caractéristiques extraites par un réseau TDNN simple, sans réapprentissage (coopératif) de ce dernier.

Nous avons constaté que ces deux classifieurs offraient des performances de classification équivalentes. Cependant, le classifieur RBF rejette mieux les *outliers*.

L'apprentissage LVQ est beaucoup plus rapide que celui du RBF (d'un facteur 5 à 10). Ceci peut être gênant pour certaines application avec apprentissage en temps réel, mais ne pose pas de problème dans la plupart des cas. De plus cette propriété facilite l'apprentissage *coopératif* dans les systèmes multi-modulaires.

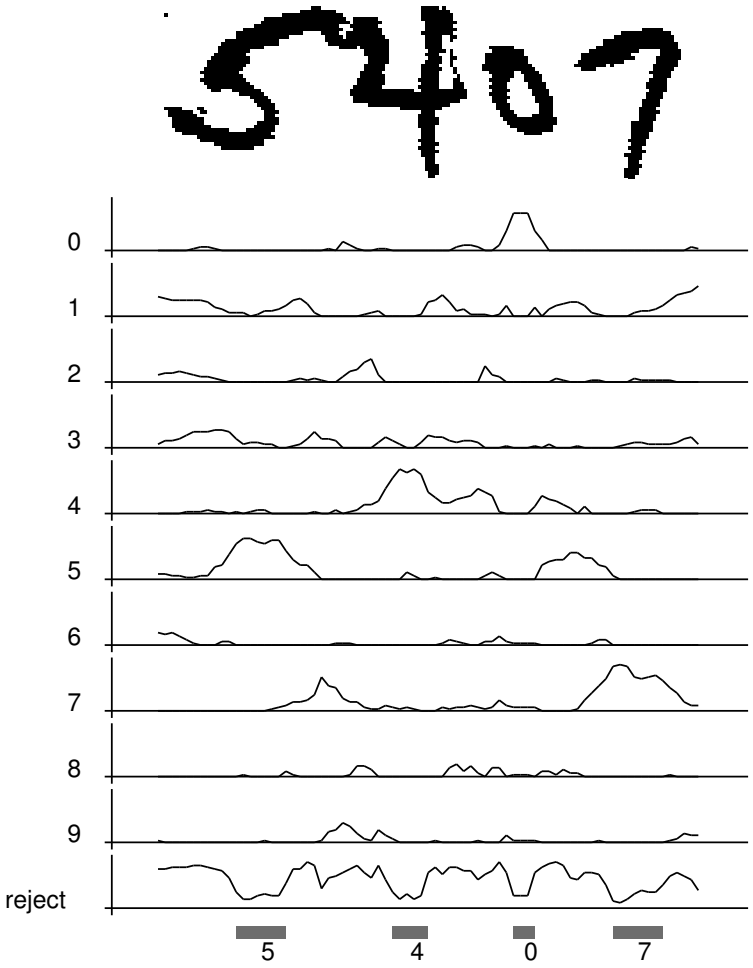


FIGURE 6.39 – Un exemple d'application simple du réseau classifieur MLP+RBF_coop pour la lecture d'un champ de chiffre. Ce champ contient deux chiffres attachés (5 et 4), aussi une méthode simple de segmentation par composantes connexes ou recherche de coupure verticale aurait échoué. Les courbes représentent les valeurs de sortie du réseau pour chaque classe. La valeur correspondante du critère de rejet MLP-3 est représentée sur la courbe du bas. Les rectangles gris indiquent les zones dans lesquelles le critère de rejet est inférieur à un seuil θ, choisi tel que l'on commette 1 % d'erreur sur les chiffres de *hsf0_ eval*. On constate que les chiffres sont tous reconnus, même lorsqu'ils sont attachés et déformés.

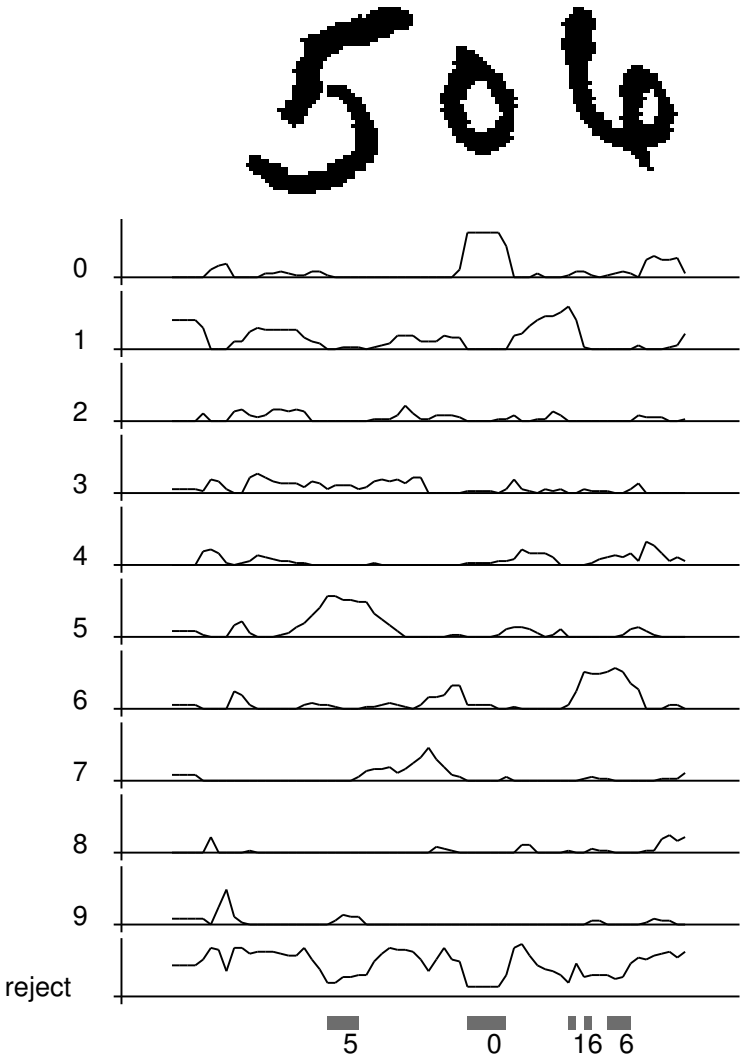


FIGURE 6.40 – Un autre exemple. Remarquez que l'on détecte un «1» juste avant le «6» (les «1» américains n'ont généralement pas de barre en haut). Cette fausse détection pourrait facilement être éliminée en considérant les distances inter-caractères, ou en appliquant une méthode plus sophistiquée d'alignement.

6.8.2 Comparaison des critères de rejet

D'autre part, nous avons comparé expérimentalement différents critères de rejet adaptés à chaque classifieur. Nous avons constaté que ces critères sont à peu près équivalents pour le rejet des formes ambiguës, mais n'ont pas tous la même aptitude à rejeter les formes éloignées (*outliers*) (voir section 6.3.3).

Nous avons ainsi retenu le critère MLP-3 pour les réseaux MLP (TDNN inclus) et les réseaux RBF. Rappellons toutefois (voir chap. ? ?) que ce critère n'est guère adapté lorsque le nombre de classes est élevé (plusieurs dizaines). Dans ce cas, le critère MLP-2 habituel sera sans doute préférable.

Pour les réseaux LVQ, le critère LVQ-2, qui considère le rapport des distances aux deux classes les plus proches, est le plus performant.

6.8.3 Systèmes modulaires

Ce chapitre nous a permis de montrer que l'approche multi-modulaire permet de bâtir des systèmes de classification très performants à partir de modules connexionnistes très simples (MLP à connexions locales, classifieur RBF), entraînés *coopérativement*.

Le problème habituel posé par les architecture multi-modulaires est l'initialisation judicieuse des paramètres de chaque module. Un des avantages appréciables des réseaux MLP est qu'il suffit d'initialiser aléatoirement les poids avant l'apprentissage. Nous avons proposé une méthode originale d'initialisation pour les architectures «MLP + X», basée sur un rapide apprentissage auto-associatif du MLP permettant d'obtenir un extracteur de caractéristiques peu performant pour la classification, mais suffisant pour initialiser les centres des neurones de type «distance».

Nous avons constaté (section 6.4.4) que l'apprentissage coopératif TDNN+RBF_coop permettait de dépasser les performances du classifieur de référence TDNN+kNN.

L'avantage majeur de l'approche multi-modulaire est le faible nombre de connexions du système global obtenu, d'ou‘ une grande vitesse de calcul : notre système MLP+RBF_coop est 8 fois plus rapide que le TDNN LeNet [129], pour des performances quasi-équivalentes dans les applications de segmentation de chiffres. Nous avons présenté un exemple très simple de système intégrant la segmentation et la reconnaissance de chiffres, dans lequel le gain de temps de calcul dans le classifieur est utilisé pour explorer toutes les positions possibles, ce qui évite de générer a-priori des hypothèses de segmentation.

Dans la suite de cette thèse, nous allons appliquer ces résultats au traitement de visages (détection et identification). Les conclusions des études

menées sur le problème de reconnaissance de chiffres nous ont permis de développer rapidement et avec succès des systèmes très performants de localisation et identification de visages, problème difficile sur lequel peu de données et de publications sont disponibles.

Chapitre 7

Application à l'identification de visages

7.1 Introduction

Nous abordons dans ce chapitre l'application des réseaux de neurones à la reconnaissance de visages.

La reconnaissance de visages s'insère dans la famille des techniques d' *identification*, qui tentent de déterminer l'identité d'une personne à partir d'un ensemble de mesures : image, son, empreintes digitales, odeur... Dans la vie courante, l'homme se base surtout sur la silhouette du visage et sur le son de la voix pour reconnaître ses voisins. L'être humain est capable de localiser et identifier les visages très rapidement, dans des situations très variées et avec une très grande fiabilité. Cette faculté est très robuste : elle résiste à d'importants changements de l'image dus à l'angle de vue, à l'éclairage, aux changements d'expressions, au vieillissement, au port de lunettes noires, aux passages chez le coiffeur, etc. C'est pourquoi le traitement visuel des visages par l'homme a fasciné des scientifiques tels qu'Aristote ou Darwin depuis des siècles.

Les systèmes de reconnaissance de visages sont non seulement intéressants par les connaissances théoriques qu'ils pourraient nous apporter sur le système visuel humain, mais possèdent aussi de nombreuses applications pratiques, que nous passons brièvement en revue dans la section suivante. Nous reviendrons ensuite, à titre de curiosité, sur les performances du système visuel humain.

7.1.1 Applications possibles

Un système automatique capable de localiser et identifier des visages humains serait utile dans maintes applications. Les plus évidentes concernent le contrôle d'accès (banques, salles machines, ...) et l'identification de criminels (de longue date, la police utilise des fichiers de portraits robots). En fait, toutes les situations dans lesquelles l'identification d'une personne est nécessaire sont potentiellement concernées.

D'autres applications envisagées concernent en général les interfaces homme-machine, et la communication d'images (videophones).

Les applications suivantes sont le plus souvent citées dans la littérature :

— **Interfaces Hommes-Machines :** il s'agit d'adapter un système informatique à l'opérateur, de façon à améliorer l'ergonomie de la machine. On peut imaginer une grande variété d'applications. Il a par exemple été proposé d'utiliser la reconnaissance du visage et le mouvement des lèvres pour améliorer la compréhension de la parole [167, 189].

— **Contrôle d'accès :** le contrôle d'accès consiste à authentifier l'appartenance d'un candidat à une liste de personnes connues. Le contrôle d'accès concerne surtout la sécurité militaire et la protection industrielle. Il demande une très grande fiabilité : le coût d'une acceptation erronée est très élevé. Le candidat peut être mal-intentionné et chercher à tromper le système.

La plupart des systèmes de contrôle d'accès actuels utilisent des cartes, des mots de passes ou des codes, qui les rendent contraignants pour l'usager (perte ou vol des cartes, oubli du code,...). De plus, un intrus peut toujours se procurer illicitement la clef.

Les systèmes basés sur la mesure de caractéristiques physiques supposées infalsifiables : empreintes digitales [168], image de la rétine, son de la voix [1] suscitent actuellement un grand intérêt de la part des chercheurs et des industriels.

Il a été proposé d'utiliser l'image du visage pour le contrôle d'accès [200, 199]. De tels systèmes présenteraient l'avantage d'être très simples et agréables d'emploi : une caméra placée près d'une porte et reliée à un petit calculateur ferait l'affaire, sans demander de manipulations spéciales au candidat.

Toutefois, il nous semble que les caractéristiques du visage sont a priori beaucoup plus facilement falsifiables que la voix ou les empreintes digitales : il est relativement aisé à un professionnel de se déguiser de façon très convaincante en une autre personne. Le développement de

1. voir par exemple [4]

ces méthodes pour des applications nécessitant une grande fiabilité ne semble par conséquent pas très réaliste.

— **Criminologie :** le cataloguage des criminels est historiquement la première motivation des systèmes anthropométriques. La police est toujours confrontée à d'importants fichiers, comportant en général une description et/ou une photo par individu. Il s'agit ensuite, étant donné une nouvelle photographie, de rechercher les personnes ressemblantes dans les fichiers. Pour cela, on a cherché de longue date [75, 76] à obtenir une paramétrisation concise des visages : c'est la notion de portrait-robot.

— **Autres :** Pour des applications concernant la transmission d'images à l'aide de lignes à faible débit, comme les videophones, il est utile de connaître la position du visage dans l'image afin d'appliquer un algorithme de compression d'image sélectif (l'identification n'est pas ici nécessaire).

Il est important de distinguer différents modes d'utilisation des systèmes de reconnaissance de visages :

— **Authentification** [2] **:** Il s'agit ici d'accepter ou refuser une identité proclamée par le candidat. C'est la situation correspondant aux applications de contrôle d'accès. Le système possède une référence (jeu de caractéristiques) pour chaque identité. Il extrait de l'image présentée un jeu de caractéristiques et compare à un seuil la distance entre ces dernières et la référence (voir figure 7.1). Les caractéristiques de référence peuvent être stockées dans le système ou fournies en même temps que l'image, en utilisant par exemple une carte à puce individuelle pour le stockage.

— **Recherche** (ou classification) d'un visage parmi un grand nombre de visages connus (quelques centaines ou milliers). C'est typiquement le cas des application policières. En général, on ne dispose dans ce cas que d'un faible nombre d'images (voire une seule photo) pour chaque personne. On ne peut donc espérer une grande robustesse de l'identification. Ce type de problème se prête assez mal à une solution basée sur un apprentissage, car on a peu d'exemples. C'est par contre le domaine de prédilection des système basés sur la mesure (manuelle ou automatisée) de caractéristiques supposées a priori discriminantes, telles que largeur du visage, distance séparant les yeux, etc.

— **Reconnaissance** d'un visage dans une famille de visages connus de faible taille. Dans ce type d'applications, on désire en général une grande robustesse, qui peut être obtenue par l'utilisation d'un nombre

2. On parle aussi de vérification.

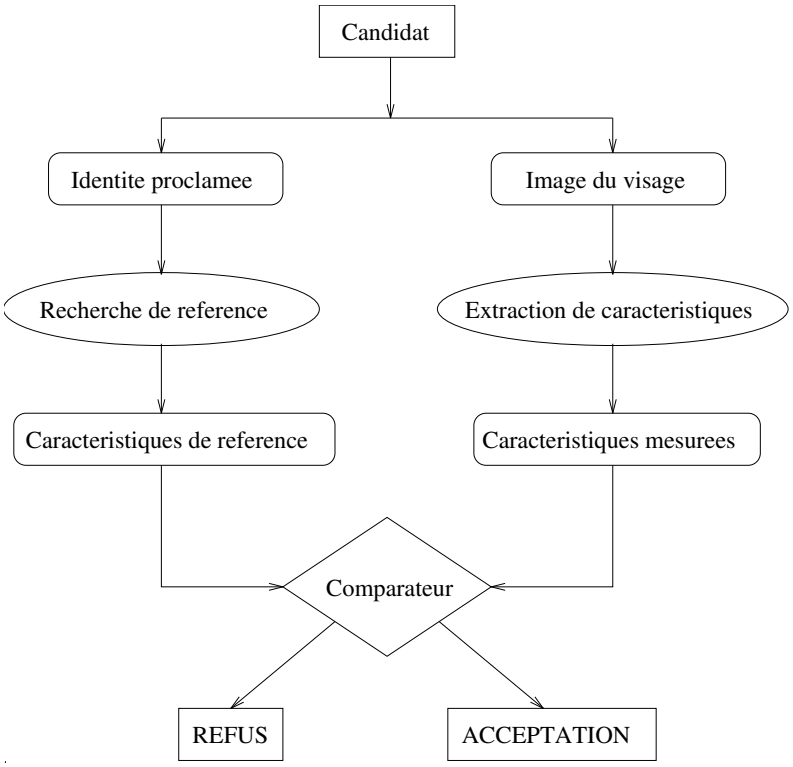


FIGURE 7.1 – Architecture d'un système d'authentification

élevé d'images exemples de chaque personne connue. Il est aussi souhaitable que le système détecte les personnes inconnues, de façon à ne pas leur attribuer une identité erronée.
Comme on dispose de nombreux exemples, il est possible d'utiliser un apprentissage pour déterminer l'extracteur de caractéristiques, ce qui permet d'atteindre des performances élevées avec un apport minimal de connaissances a priori sur le problème.

Compte tenu des données à notre disposition durant cette thèse, nous nous sommes toujours placé dans ce dernier cas. Nous avons travaillé avec des «familles» comprenant de 2 à 15 personnes. Notons qu'il s'agit d'un cas intéressant du point de vue de la classification : on dispose de grands nombres d'exemples, ce qui permet de comparer très finement les classifieurs utilisés.

7.1.2 Performances de l'être humain

Dans cette section, nous rappelons brièvement les performances du système visuel humain. C'est après tout le système de reconnaissance de visage que nous connaissons le mieux, et on peut le considérer comme un bon point de référence. Sans chercher à copier le système humain, on peut toujours s'en servir pour spécifier a-priori les performances d'une chaîne de traitement.

Codage

Le premier stade dans la perception d'un visage est un codage des stimuli visuels. On sait actuellement très peu de choses sur la nature de ces représentations.

Notons simplement, par curiosité, qu'il semblerait que la représentation interne des visages familiers soit différente de celles des visages inconnus [60].

Détection

Les humains détectent les visages dans une scène très rapidement, après une inspection grossière, même si le visage n'est pas familier. Les visages sont perçs dans leur ensemble (Gestalt), et non comme une addition de caractéristiques (nez, bouches, yeux, ...). En particulier, on reconnaît sans difficulté les visages partiellement cachés : tout se passe comme si le système visuel remplissait les parties manquantes.

Les visages sont détectés dans une grande variété de conditions : mauvais éclairage, grandes distances, occlusions. On peut aussi constater un biais perceptuel tendant à confondre diverses formes d'objet avec des visages. On imagine ainsi souvent des visages dans les nuages, les flammes, etc.

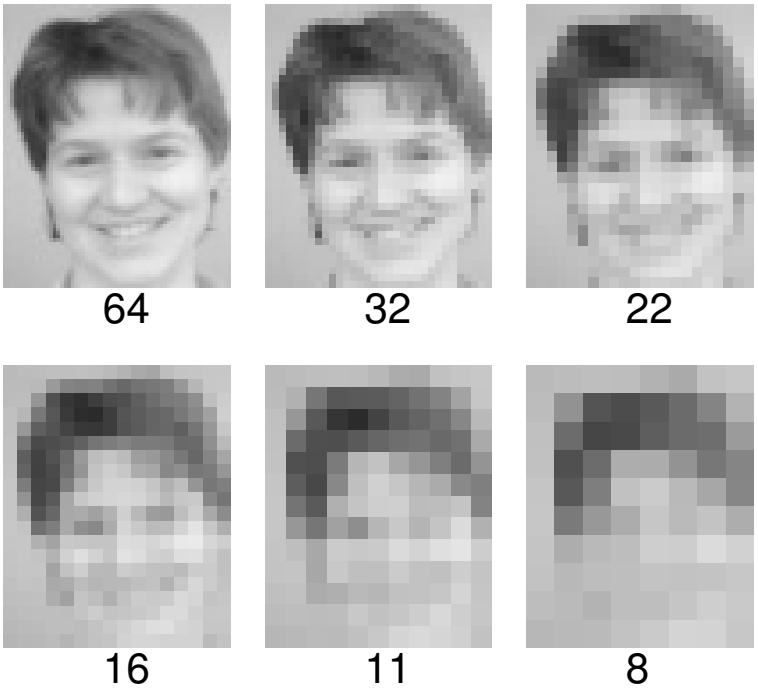


FIGURE 7.2 – Le même visage vu à différentes résolutions. De gauche à droite et de haut en bas : 64, 32, 22, 16, 11 et 8 pixels de largeur (8 bits/pixel).

La détection ne semble pas dépendre de caractéristiques détaillées comme les cils ou les mèches de cheveux. La forme générale, seule visible de loin, suffit ; le système visuel fonde la détection d'un visage sur des caractéristiques générales (forme arrondie, bouche, yeux, ...) et leur positions relatives.

Pour tenter de quantifier les limites du système de détection, on peut mesurer la résolution minimale nécessaire à l'homme pour reconnaître un visage. La figure 7.2 montre des images du même visage à différentes résolutions spatiales. Sur l'image de taille 32x32 pixels, on distingue clairement un visage. C'est beaucoup moins évident sur l'image 16x16. On s'accorde généralement [91] sur le fait qu'une résolution minimale comprise entre 20x20 et 32x32 pixels est nécessaire pour la détection et l'identification de visages.

Un nombre assez faible de niveaux de gris semble suffisant pour détecter un visage. La figure 7.3 montre le même visage codé sur 8, 4, 3, 2 et 1 bits/pixel. La qualité visuelle de l'image se dégrade sérieusement en dessous de 4 bits/pixels.

Samal [186] suggère qu'un codage sur 4 bits/pixels associé à une résolution

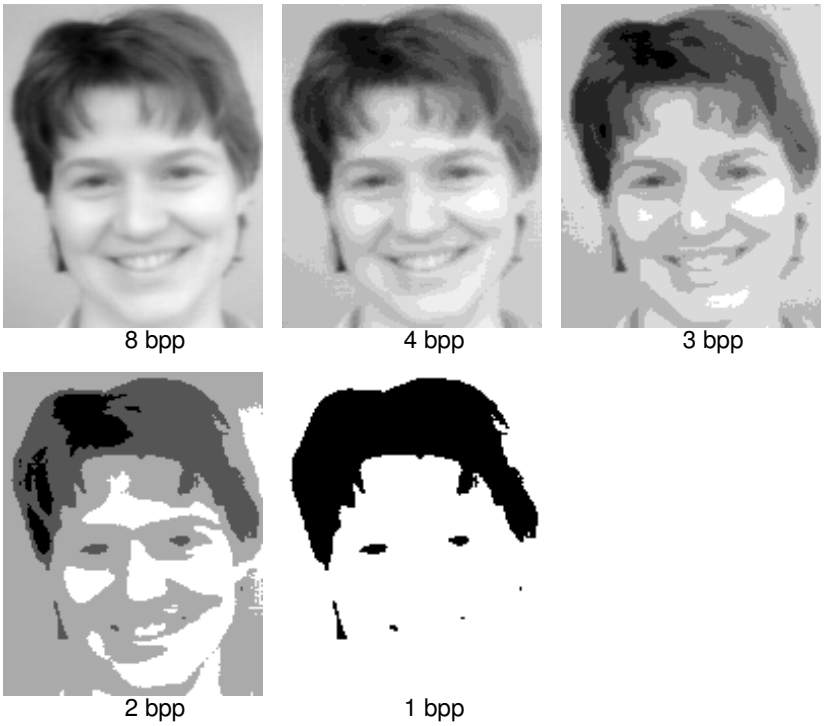


FIGURE 7.3 – Le même visage, codé sur différents nombres de bits par pixel. De gauche à droite et de haut en bas : 8, 4, 3, 2 et 1bits/pixel. (On pourrait bien sûr améliorer la qualité des images en utilisant un *dithering*, c'est à dire des pointillés noirs et blancs pour compenser le manque de niveaux de gris au prix d'une petite diminution de la résolution spatiale.)

de 32x32 est suffisant pour la détection des visages.

Identification

Une fois le visage détecté, l'étape suivante consiste à l'identifier. Le système de reconnaissance de visages humain est très fiable et précis : il est difficile de tromper un humain par un déguisement. C'est d'ailleurs pourquoi les timbres postaux et pièces de monnaie utilisent souvent des dessins de visages célèbres, sensés être plus difficiles à imiter et rapidement identifiables.

On estime [60] qu'un homme est en moyenne capable d'identifier sans hésitation environ 700 visages de personnes différentes [3].

L'identification d'un visage dans des conditions normales est quasi-instantanée. Le temps de réponse augmente lorsque l'image est inhabituelle (par exemple tournée).

Comme on l'a déjà noté, le système d'identification humain est extrêmement robuste aux variations de l'image. Cette robustesse est d'ailleurs d'autant plus grande que le visage est familier.

En dépit de notre faculté à reconnaître les visages sans effort, il nous est très difficile de décrire un visage. La méthode la plus commune consiste à énumérer différentes caractéristiques : couleur des cheveux et des yeux, forme de la bouche, ... Il semble pourtant que la reconnaissance n'utilise pas ce genre de caractéristiques de «haut niveau», mais se base plutôt sur un ensemble de caractéristiques de «bas niveau». Des études [35] sur les stratégies employées par le cerveau pour identifier les visages indiquent que les caractéristiques de «haut niveau» et leurs relations géométriques ne constituent pas une représentation suffisamment riche pour rendre compte des performances d'un adulte humain.

7.2 Historique en Traitement Automatique des Visages

Dans cette section, on effectue un rapide historique des travaux menés en traitement automatique des visages. Nous abordons d'abord les travaux «classiques», basés sur l'extraction, manuelle ou automatique, de caractéristiques, puis passons en revue les développements récents basés sur l'utilisation d'algorithmes connexionnistes.

3. On estime à quelques centaines de milliers le nombre de visages rencontrés pendant une vie.

7.2.1 Systèmes classiques

Les premiers travaux en reconnaissance automatique de visage ont été basés sur la détection d'objets morphologiques de haut niveau tels que les yeux, la bouche, ou le profil. On mesure ensuite diverses relations géométriques entre ces objets (par exemple distance entre les yeux, surface du nez, intensité des cheveux, ...), qui sont ensuite regroupées dans un vecteur numérique de faible dimension, que l'on classifie à l'aide de diverses techniques statistiques.

Ces approches manquent de robustesse : les mesures effectuées sur une nouvelle image diffèrent souvent de celles obtenues sur le visage de référence à cause de changements d'expression ou de position du sujet, ou de l'éclairage. De plus, la moindre occultation du visage est fatale. On impose donc des conditions d'observation très strictes, qui rendent le système inutilisable en pratique.

Anthropométrie : Bertillon et Galton

L'anthropométriste Bertillon (1853-1914) fut sans doute le premier à proposer une approche formelle pour l'identification de visages. Il a inventé un système basé sur une description concise du visage (en language naturel) pouvant être associé sans ambiguïté à chaque personne. Ce système à été utilisé par les polices françaises et d'autres pays, jusqu'à l'avènement des empreintes digitales.

A la fin du dix-neuvième siècle, Galton [75, 76] a mis au point un système permettant de réduire un portrait de visage à une formule numérique mêlant diverses mesures géométriques (de façon analogue aux «jauges» utilisées à l'époque par les anglais pour comparer leurs voiliers de course).

L'idée est de représenter divers profils soigneusement choisis (par exemple le profil du nez) par un codage numérique adéquat, basé sur le repérage de points caractéristiques. On montre ces points sur la figure 7.4. Ce codage permet bien entendu de reconstruire chaque profil pour obtenir une sorte de portrait.

Premiers travaux : Bledsoe, Harmon, Goldstein

Les premiers travaux modernes en reconnaissance de visage ont cherché à utiliser le profil. En effet, le profil peut se coder facilement comme une courbe et est relativement aisé à extraire automatiquement (détection du contour sur une image vue de profil).

Les travaux de Bledsoe [10, 9] semblent être les premiers visant à obtenir un système de reconnaissance de visages informatisé. Il propose un système

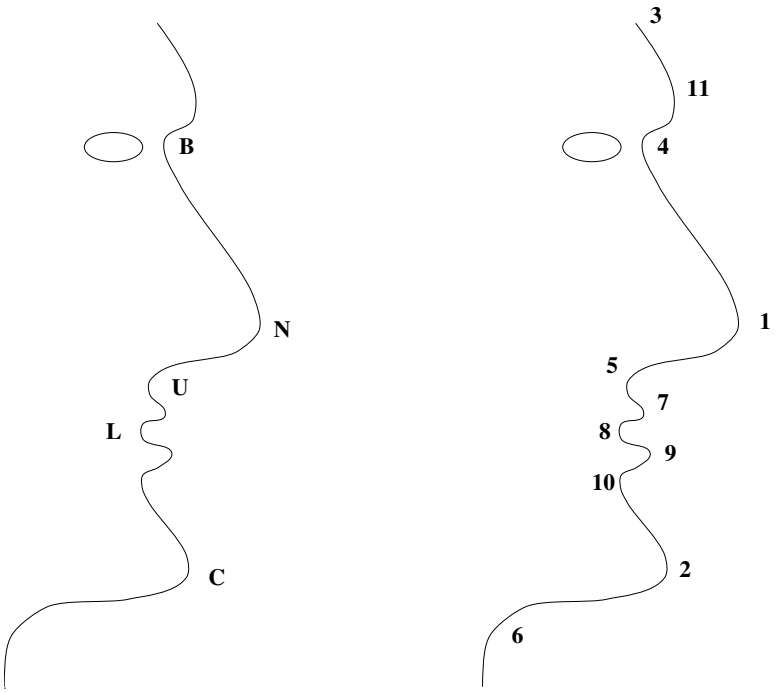


FIGURE 7.4 – Points caractéristiques du profil utilisés par Galton (à gauche) et Harmon (à droite)

hybride : un opérateur humain saisit manuellement des coordonnées de points caractéristiques (coins des yeux et de la bouche, centre du nez etc) sur les images à traiter, puis l'ordinateur se charge de la classification. Le choix de ces points est très proche de celui de Galton (voir figure 7.4). Les paramètres utilisés pour la classification sont des rapports de distances entre les points caractéristiques sur le profil du visage.

Ces travaux ont été prolongés par Goldstein [79] puis Harmon [92, 94], qui ont cherché à optimiser le nombre de caractéristiques à choisir sur les contours. Enfin, Harmon a proposé un système automatique [93], qui fonctionne à partir d'images prises de profil sur un fond très contrasté, de façon à pouvoir obtenir le contour du visage.

L'utilisation du profil pour la reconnaissance, malgré ses inconvénients évidents pour de nombreuses applications, est toujours étudiée [221]. Son principal attrait est la simplicité et la rapidité du traitement. Ces études sont surtout destinées à la criminologie, application ou‘ l'on peut imposer précisément les conditions de prise de vue.

Systèmes basés sur la recherche de caractéristiques

Une approche plus ambitieuse consiste à utiliser pour la reconnaissance une image du visage vu de face, et à rechercher dans cette image les constituants du visage (yeux, bouche, nez, oreilles). Une fois ces objets localisés, on mesure diverses distances (convenablement normalisées) et on applique une méthode de classification classique sur le vecteur obtenu. On travaille ici directement sur l'image, d'ou‘ des coûts de calculs plus importants.

Les premiers systèmes utilisant cette approche ont été proposés par Fischler et Elshlager [66]. L'idée générale est d'utiliser un «template matching» local. On décrit la forme («template») à localiser par une courbe paramétrique que l'on cherche à placer au mieux sur l'image (en minimisant un coût par une méthode variationnelle).

Les travaux récents les plus importants dans ce domaine sont ceux de Craw et Ellis [46], Yuille et al. à Harvard [222], Shackleton, Waite et Welsh à British Telecom [191, 214].

Il faut noter que ces systèmes sont non seulement utilisables pour donner l'identité du visage présenté, mais peuvent aussi fournir la position des constituants du visage. En particulier, il est intéressant pour certaines applications de connaître les coordonnées de la bouche [167, 189] (compréhension de la parole) ou des yeux [140] (compression sélective d'images).

La recherche des caractéristiques dans l'image de visage peut aussi s'effectuer à l'aide de techniques connexionnistes. Plusieurs équipes des laboratoires de British Telecom se sont penchées sur cette question [140] [210]. Les per-

formances de leurs réseaux pour la détection des yeux et de la bouche sont assez comparables à celles des systèmes basés sur les templates élastiques cités ci-dessus. Ces systèmes rencontrent les même problèmes : si un œil est caché (port de lunettes par ex.), la méthode échoue.

7.2.2 Systèmes connexionnistes

Les systèmes connexionnistes de reconnaissance de visages permettent une approche globale : on cherche à classer l'image d'un coup, sans découper explicitement la tâche en plusieurs étapes (recherches de caractéristiques, mesures, classification). Cette approche est possible car les réseaux de neurones sont capables de traiter de très importants volumes de données (par exemple une image 512x512), et d'en extraire l'information pertinente pour la tâche de classification.

Kohonen [121] est le premier à appliquer un système auto-associatif simple au débruitage et à la classification d'images de visages. Il utilise cette application comme illustration de son concept de mémoire associative, et ne l'étudie pas spécifiquement en détail.

Utilisation de transformation KL : réduction de dimension

Kirby et Sirovich [197, 116] ont proposé d'utiliser la tranformation de Karhunen-Loeve (KL) pour coder les images de visage.

La transformation KL est une technique bien connue pour réduire la dimension de données à classer ou à transmettre (voir la section 2.8). La même technique est appelée Analyse en Composantes Principales en analyse de données. Le but de la transformation KL est de représenter une image en utilisant un système de coordonnées *optimal*, ce qui permet de réduire la dimension de l'espace en perdant le minimum d'information. Les vecteurs de base formant le système de coordonnées sont appelés *eigenpictures* («images propres»). Appliqué au traitement de visages, Turk [203] (voir section suivante) nomme ces vecteurs *eigenfaces* («visages propres»).

Kirby et Sirovich aménagent la transformation KL pour tirer parti des propriétés spécifiques de symétrie des visages.

Travaux de Turk et Pentland

Turk et Pentland [204] ont d'abord utilisé explicitement la transformation de Karhunen-Loeve pour calculer un codage des visages à reconnaître, et appliqué sur ce codage un classifieur type plus-proche voisins pour l'identification. Ils ont ensuite implémenté le même principe avec un réseau de

	Variation d'éclairage	Variation d'orientation	Variation de tailles
Sans rejet	4 %	15 %	36 %
20 % de rejet	0 %	6%	26 %

TABLE 7.1 – Taux d'erreur mesuré par Turk et al. sur trois ensembles d'images.

neurones totalement connecté [203].

Pour leurs expériences, 2500 images de taille 128x128 ont été utilisées, représentant 16 personnes. Les résultats présentés sont des performances moyennes sur trois ensembles d'images, au sein desquels on a fait varier l'éclairage, l'orientation ou la taille des visages. Les taux d'erreurs mesurés sont récapitulés dans le tableau 7.1.

Travaux de G. Cottrell

Cottrell [44, 45, 67] a ensuite repris ce principe en utilisant des réseaux MLP auto-associatifs, entraînés à l'aide de la rétro-propagation du gradient. Un réseau auto-associatif (voir figure 7.5) est entraîné à reconstruire les images de visages qui lui sont présentées. Une fois cet apprentissage effectué (c'est à dire lorsque l'erreur de reconstruction atteind un minimum), on branche sur la couche cachée un deuxième réseau, possédant une seule couche de poids, qui est à son tour entraîné à classer les images. Sur les sorties de ce réseau, on code l'identité (une cellule par classe), le sexe (2 cellules), et on ajoute une cellule utilisée pour tester si l'image représente un visage ou autre chose.

Cottrell utilise des images de taille 64x64 et 80 cellules cachées (soit de l'ordre de 650000 connexions pour le MLP auto-associatif).

Notons que ce dispositif est très semblable à celui que nous avons testé pour la reconnaissance de chiffres manuscrits (chapitre 6, section 6.5.2). Nous avons d'ailleurs constaté que le remplacement du classifieur MLP par un réseau RBF améliorait considérablement les performances.

Les images utilisées pour les expériences décrites sont de bonnes qualité. En particulier, la position des yeux dans les images, ainsi que la luminosité, sont maintenues constantes. 204 images de visages (plus 27 images d'autres objets) sont utilisées. L'apprentissage est effectué avec 77 images.

Les performances rapportées sont excellentes. Toutefois, l'ensemble de test est de trop petite taille, et les conditions trop restrictives (qualité des

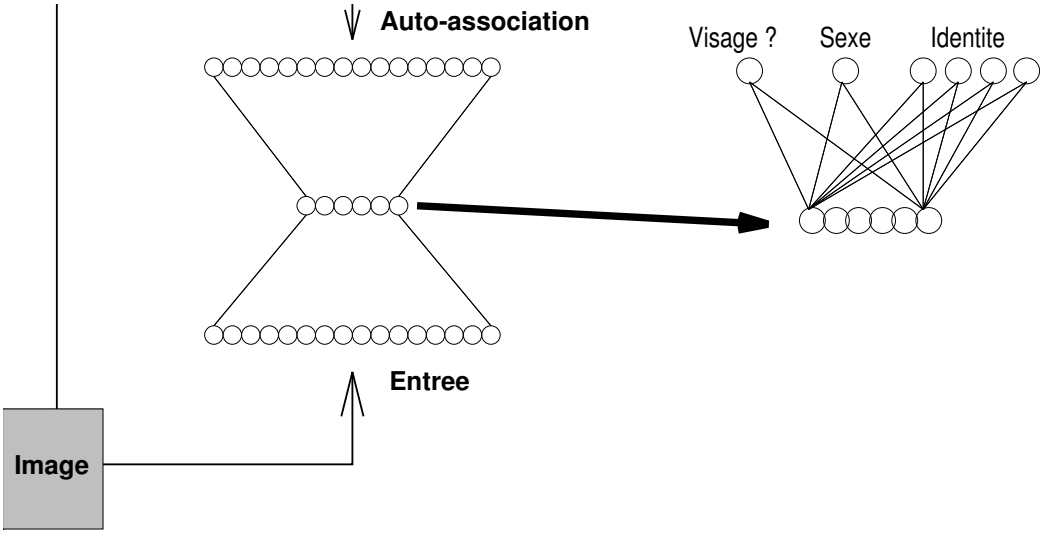


FIGURE 7.5 – Architecture utilisée par G. Cottrell. L'image, de taille 64x64, est présentée sur la rétine du réseau de compression (à gauche). Le codage lu sur la couche cachée est présenté à un réseau de classification à une seule couche (à droite).

images) pour tirer de ces expériences des conclusions générales.

Citons aussi Golomb et al. [80] (*Sexnet*), qui ont utilisé le même type d'architecture pour déterminer le sexe d'une personne à partir d'une image de visage.

Travaux de P. Frasconi

L'équipe de P. Frasconi [71] a mené récemment des expériences et s'est intéressé au problème du rejet des visages inconnus, souvent passé sous silence dans les travaux précédents. Les auteurs ont renoncé à utiliser les valeurs de sortie du classifieur pour détecter les images inconnues, et proposent un système mixte, comportant deux types de sorties : d'une part une image reconstruite, d'autre part une cellule par individu à reconnaître (voir figure 7.6). Pendant l'apprentissage, on présente l'image à reconnaître sur la rétine d'entrée et sur la couche de sortie (auto-association), et l'on fixe les autres cellules de sortie pour coder l'identité de la façon habituelle.

Pour leurs expériences, ils ont utilisé une base de données contenant 8 personnes, avec environ 400 exemples par personne. Les images originales, de taille 512×512, sont normalisées à la taille 14×13 (centre du visage, sans les cheveux ni le menton, codage nous semblant difficile à automatiser).

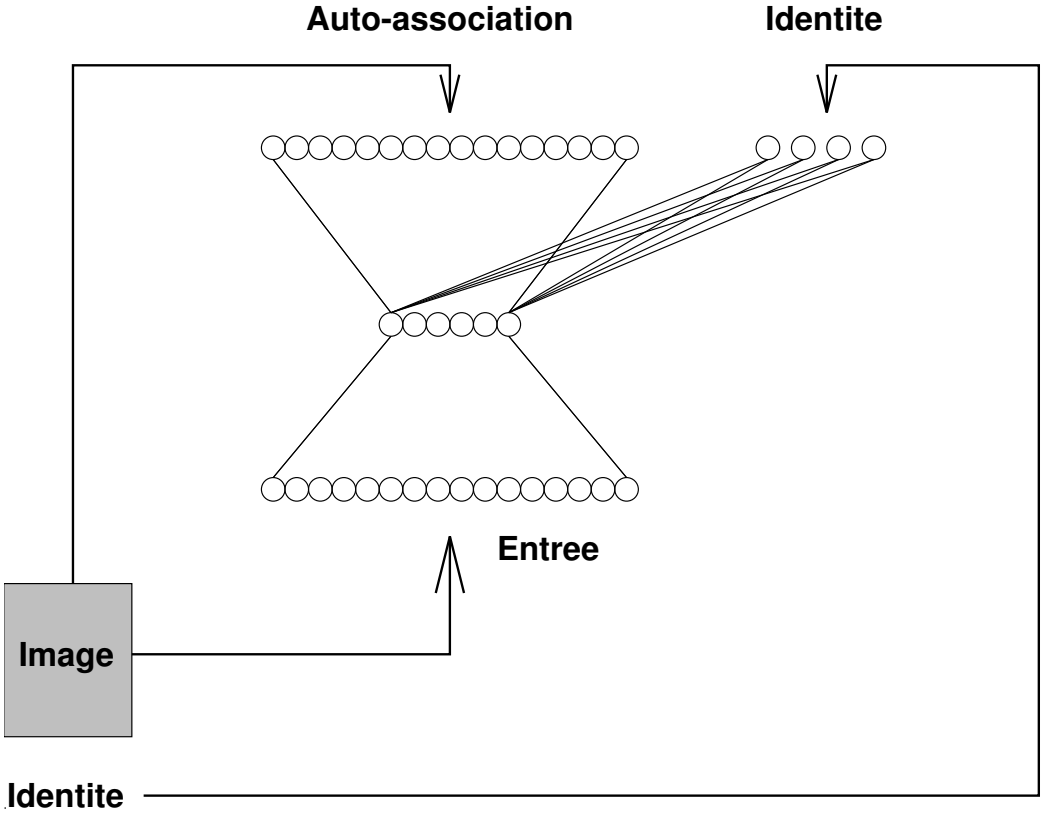


FIGURE 7.6 – Architecture proposée par Frasconi et al. L'image est ici de taille 14x13. L'erreur de reconstruction est utilisée pour détecter les visages inconnus.

Le réseau, totalement connecté, est de taille $182 \times 8 \times 190$. Les résultats reportés indiquent un taux d'erreur de 2 % pour 27 % de rejet. Toutefois, ils ne semblent pas distinguer le rejet d'ambiguïté du rejet des invités (ou imposteurs).

Système de vérification de Los Alamos

Des expériences importantes ont été menées récemment aux laboratoires de Los Alamos dans le but de bâtir un système de vérification d'identité [200, 199]. L'application visée est le contrôle d'accès.

Le système utilisé est un simple réseau MLP à connexions totales, utilisant une rétine 35x40 (dimension 1400), une couche cachée de 20 cellules, et une valeur de sortie. Ce classifieur doit en principe distinguer un visage connu de tous les autres visages possibles.

La base de données, constituée pour l'occasion, rassemble 511 personnes, avec environ 20 images par personne. Solheim et al. [199] proposent une méthode visant à sélectionner pour chaque personne un ensemble de personnes «ressemblantes», afin de mener l'apprentissage.

D'après les études comparatives des auteurs, ce système permet des performances d'authentification meilleures que les systèmes commercialisés utilisant la voix ou les empreintes digitales.

Autres travaux récents utilisant des réseaux MLP

L'utilisation de réseaux à connexions totales a été testée récemment par de nombreuses équipes. Voir par exemple [166] et [125]. Malheureusement, les bases de données d'images utilisées diffèrent d'une équipe à l'autre. De plus, le nombre d'images utilisées est souvent beaucoup trop faible pour être significatif.

Conclusion

Tous les systèmes connexionistes présentés ci-dessus traitent l'image à reconnaître comme un vecteur mono-dimensionnel : les connexions totales entre couches ne permettent pas de tirer parti de la structure bi-dimensionnelle de l'image.

On est ainsi passé directement de systèmes apportant des connaissances trop détaillées sur le problème (recherche des yeux à tel endroit, ou points caractéristiques sur le profil par exemple), à des systèmes n'utilisant aucune connaissance sur le problème. Les premiers systèmes manquent de robustesse (si une partie du visage est cachée, on ne peut rien faire), les seconds

possèdent trop de paramètres ajustables (trop de poids synaptiques) et nécessiteraient par conséquent des nombres d'exemples très grands pour obtenir une généralisation correcte.

On a vu dans les chapitres précédents que les réseaux TDNN (ou SDNN) introduits en reconnaissance de parole puis de caractères, permettaient d'apporter des connaissances générales sur le problème à résoudre : les masques de poids partagés expriment par exemple une certaine invariance de la reconnaissance aux translations de l'image. Cet apport de connaissance se traduit par une réduction du nombre de paramètres libres du système.

Nous allons montrer dans la suite que l'on peut appliquer avec succès ces réseaux à la reconnaissance de visage.

D'autre part, très peu d'équipes se sont penchées sur le moyen de détecter les visages inconnus. C'est pourtant un problème important que l'on ne peut se permettre de négliger dans une application pratique. Dans la plupart des travaux décrits ci-dessus, on suppose que toutes les images vues par le système de classification appartiennent aux classes pré-définies. Cette supposition est rarement vérifiée dans la pratique : on va rencontrer des images ne représentant pas des visages (venant par exemple d'erreurs du module de segmentation), et des images de visages inconnus. Le taux d'erreur final du système de classification doit donc être mesuré sur des données rendant compte de cette situation.

7.3 Description des données

Nous décrivons dans cette partie les données utilisées pour nos études sur les systèmes de reconnaissance de visage. Malheureusement, il n'existe pas de bases de données publiques d'images de visages, comme c'est le cas en parole (base TIMIT [65] par exemple) ou en reconnaissance optique de caractères (base NIST [77] par exemple). Cette situation empêche la comparaison précise des résultats obtenus par différentes équipes de recherches. La plupart des publications de résultats (voir section 7.2) portent ainsi sur des petites bases de données d'images constituées pour l'occasion.

Les données sur lesquelles nous avons travaillé nous ont été prêtées par la société Mimétics, qui développe des applications utilisant des systèmes de reconnaissance et localisation de visages.

Nous avons ainsi disposé successivement de deux bases de données d'images de visages, que nous désignerons dans toute la suite sous les noms B1 et B2. Dans les sections suivantes, nous décrivons en détail la nature de ces bases de données.

7.3.1 Base B1

La base B1 est la première dont nous ayons disposé. Cette base est constituée de 643 imagettes déjà segmentées de taille 40x50 pixels, codés chacun sur 8 bits.

Composition de la base B1

Dix personnes différentes sont également représentées, soit environ 65 imagettes par personne. La composition de la base B1 est la suivante :

— 4 femmes.
— 2 enfants (frère et sœur jumeaux, environ 6 ans).
— 4 hommes.

Conditions d'acquisition

Les images ont été acquises à l'aide d'une caméra vidéo ordinaire reliée à un magnétoscope.

On a utilisé successivement trois éclairages différents pour chaque personne :

1. éclairage diffus (lumière ambiante naturelle).
2. éclairage de face (projecteur hallogène).
3. éclairage de profil (projecteur hallogène).

Afin de varier l'entourage des visages, les visages des acteurs ont chacun été placés devant deux types de fonds :

1. fond blanc uni.
2. fond avec motifs irréguliers (type papier peint à fleurs).

Dans chacune des six configurations obtenues par la combinaison des conditions que l'on vient d'énumérer, il a été demandé à l'acteur de répéter approximativement la même séquence de mouvements de la tête : rotation à droite, à gauche, lever la tête, baisser la tête. On a ensuite digitalisé une image toute les deux secondes sur le film vidéo obtenu. Il faut noter qu'on trouve de nombreuses *occultations* dans cette base, surtout dues à des mains inopportunément placées devant les visages. Les acteurs ont aussi été encouragés à grimacer, sourire, bailler etc.

Pré-traitements

La segmentation (découpage d'un rectangle englobant tangent au visage) a été effectuée manuellement assez approximativement.

FIGURE 7.7 – Quelques imagettes extraites de la base de visages B1 (taille 40x50, 8 bits/pixel). Notez la ressemblance entre les images 3 et 4, qui correspondent à deux personnes différentes. Ces images font partie de l'ensemble de test, et seront toutes reconnues sans erreur.

Erreur	Apprentissage ($m = 515$)	Test ($m = 128$)
25.0	3.1	6.3
20.0	2.9	5.8
15.0	2.6	5.2
10.0	2.2	4.4
5.0	1.6	3.2
1.0	0.7	1.5

TABLE 7.2 – Cette table donne l'intervalle de confiance à 95 % t_{95} sur les ensembles d'apprentissage et de test de la base B1, pour différents taux d'erreurs.

La sous-image correspondant au visage a ensuite été réduite à une taille de 40x50 pixels, en utilisant un algorithme d'interpolation classique.

Les imagettes obtenues n'ont pas subi de pré-traitements particuliers : les valeurs des pixels ont simplement été normalisées entre 0 et 255 ($= 2^8 - 1$).

La figure 7.7 montre quelques visages représentatifs de la base de données B1.

Division de la base B1

Afin de tester les algorithmes d'apprentissage, on a divisé une fois pour toutes la base B1 en un ensemble d'*apprentissage* et un ensemble de *test*. Pour cela, on a d'abord mélangé aléatoirement les 643 imagettes, puis gardé les 515 premières (4/5) pour l'apprentissage, et placé les 128 restantes dans l'ensemble de test.

Du fait de la petite taille de la base B1, les mesures de taux d'erreurs sur cette base sont entachées d'une grande incertitude statistique, comme l'indique la table 7.2 (obtenue selon la méthode rappelée en annexe A). De ce fait, il est souvent impossible de comparer valablement les performances des différents systèmes sur ces données.

7.3.2 Base B2

Nous avons pu disposer de la base B2 plus récemment (été 1992). Cette base de données, de taille nettement plus importante que la précédente, nous a permis de tester très précisément les limites des systèmes de reconnaissance, ainsi que leur capacité à détecter les visages inconnus.

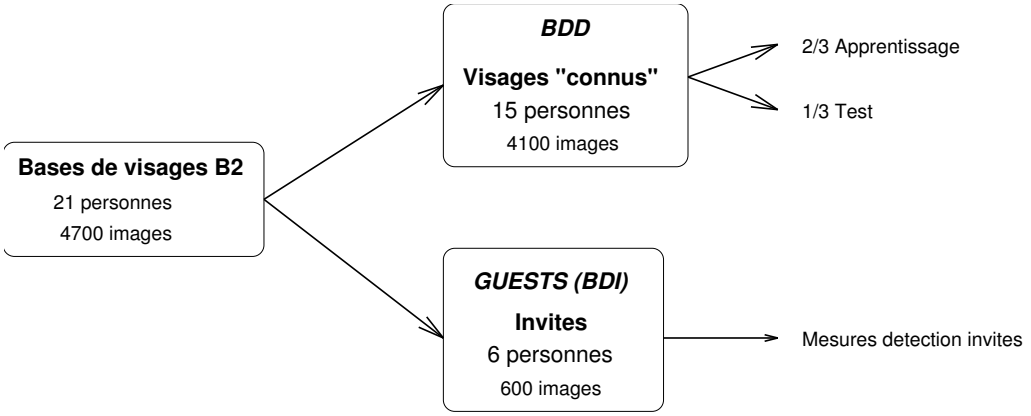


FIGURE 7.8 – Structuration de la base de visages B2

Contrairement à la base B1, nous avons pu disposer ici des images originales avant découpage et pré-traitements, ce qui nous a laissé plus de liberté dans le choix de ces derniers. En particulier, nous avons développé à cet occasion un algorithme de découpage automatique des visages, remplaçant la fastidieuse segmentation manuelle qui avait été utilisée pour les images de B1.

Composition de la base B2

La base B2 comporte au total 21 personnes. Elle se divise (voir figure 7.8) en deux parties, appelées respectivement *BDD* et *BDI*.

BDD contient 15 personnes, avec environ 270 images de chacune. On considèrera ces personnes comme «connues» : les images seront utilisées pour entraîner les systèmes d'identification. Parmi ces personnes, on trouve :
— 5 femmes adultes, dont deux de peau noire ;
— 4 enfants ;
— 6 hommes adultes, dont un barbu.
La figure 7.9 montre quelques visages extraits de *BDD*.

BDI contient 6 autres personnes, et une centaine d'images de chacune. On réserve cette base pour tester la capacité des systèmes à détecter les invités. Les 6 personnes se répartissent comme suit :
— 2 femmes ;
— 4 hommes.
La figure 7.10 montre quelques visages extraits de BDI.

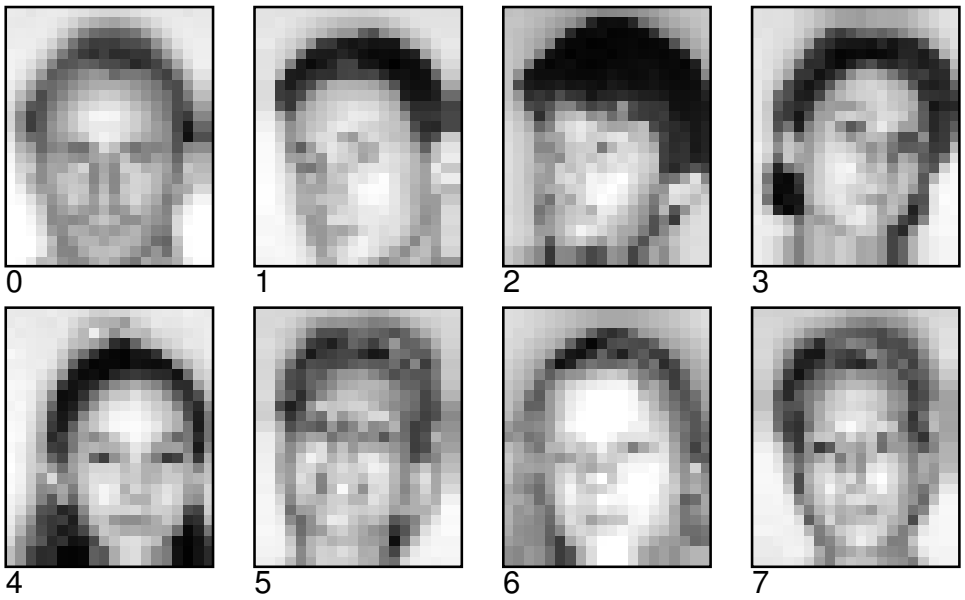


FIGURE 7.9 – Quelques imagettes extraites de la base de visages *BDD*. (Reproduites ici à partir d'images 20x25 normalisées.)

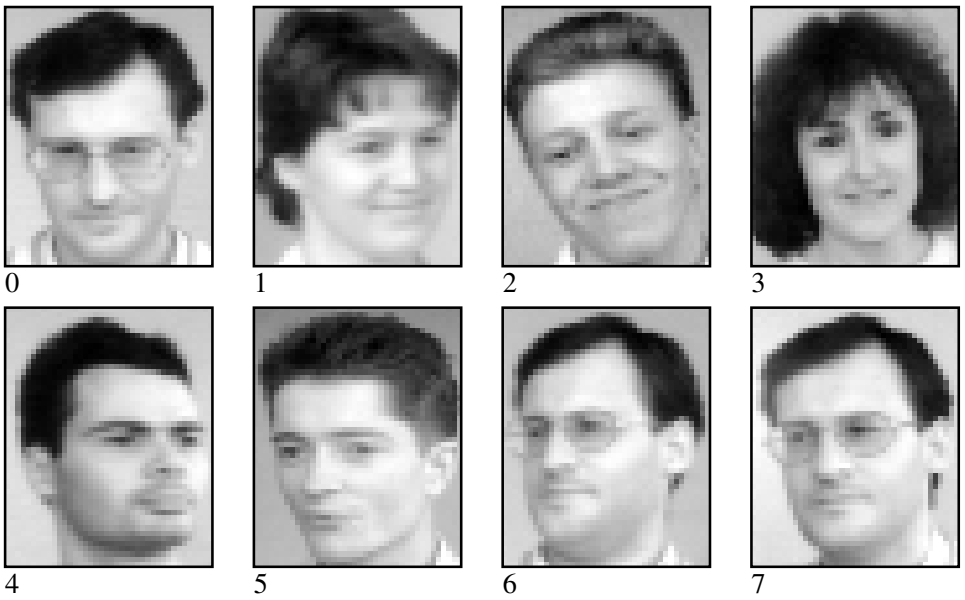


FIGURE 7.10 – Quelques imagettes extraites de la base de visages BDI. (Reproduites ici à partir d'images 40x50.)

Erreur	Apprentissage ($m = 1650$)	Test ($m = 1320$)
10.0	1.4	1.6
5.0	1.0	1.2
4.0	0.9	1.0
3.0	0.8	0.9
2.0	0.6	0.7
1.0	0.5	0.5

TABLE 7.3 – Cette table donne l'intervalle de confiance à 95 % t_{95} sur les ensembles d'apprentissage et de test de la base *BDD*, pour differents taux d'erreurs.

Intervalles de confiance La table 7.3 donne les intervalles de confiance pour les taux d'erreur observés sur *BDD*, quand on utilise 11 personnes (cas le plus fréquent pour les expériences décrites dans ce chapitre). L'ensemble d'apprentissage comporte alors 11 x 150 = 1650 images, et l'ensemble de test 11 x 120 = 1320 images.

Conditions d'acquisition

Les images de B2 ont été acquises à l'aide d'un capteur CCD, donnant une résolution de 765x580 pixels. Le capteur utilisé était sensible dans le proche infra-rouge, et disposait d'un flash incorporé éclairant dans cette gamme de fréquences. Ceci a peu d'importance pour notre application, les images obtenues par ce capteur étant très proches de celles obtenues par une caméra vidéo standard dans le visible [4]. Ce capteur présente l'avantage de permettre la numérisation en temps réel de séquences d'images (à une cadence maximum de 4 images/secondes), ce qui nous a permis d'acquérir en peu de temps une quantité inhabituellement importante d'images.

L'éclairage des visages est toujours dirigé de face (flash).

Les acteurs ont toujours été placés devant un fond blanc uni.

Durant l'acquisition des images, il a été demandé aux acteurs d'éviter de trop gesticuler. En effet, nous nous étions rendu compte que les visages de la base B1 étaient souvent trop déformés (fortes rotations, occultations, etc), et donc non représentatifs d'une situation réelle, ou‘ l'acteur est supposé collaborer avec le système de reconnaissance (ou du moins ne pas chercher à

4. Par contre, l'utilisation du flash entraîne des conséquences importantes sur les images de scènes utilisées par le module de segmentation, qui seront discutées dans le chapitre 8

se dissimuler !). Cette précaution explique en grande partie les différences de résultats constatées entre B1 et B2.

Découpage et pré-traitements

Le nombre d'images obtenues (4700) excluait cette fois de découper les visages manuellement. Par contre, les conditions d'acquisition étaient suffisamment «simples» pour permettre la mise au point d'un algorithme de découpage des visages automatique : éclairage de face, fond uni, pas d'occultations. Cet algorithme, basé sur la recherche du contour du visage par un filtre gradient, est détaillé en annexe B.

La sortie de cet algorithme est un rectangle de rapport hauteur/largeur égal à 25/20. En effet, tous les réseaux que nous avons testés ont une rétine possédant ce rapport hauteur/largeur, qui correspond au rapport moyen des dimensions des visages.

7.4 Description de l'application

Nous décrivons et discutons dans cette section les résultats des systèmes d'identification de visages que nous avons mis au point et testés sur les différentes bases de données décrites dans la section 7.3.

Les systèmes décrits ici visent à atteindre la plus grande précision possible sur un faible nombre de personnes différentes (classes). On cherche à obtenir un système *robuste*, c'est à dire capable de bonnes performances dans des conditions variables, et pas forcément bien représentées dans l'ensembles des images d'apprentissage : variation d'éclairage, d'attitudes, occultations partielles, faible résolution, changements de coiffures etc.

En particulier, il est très important que le module d'identification soit le moins sensible possible au cadrage exact des visages. En effet, dans l'application finale, le cadrage va être effectué par le module de segmentation (décrit dans le chapitre 8), et sera susceptible d'imprécisions.

Une autre propriété désirable est la capacité de détecter les visages inconnus (qualifiés d'invités).

Nous avons utilisé la base de données B1 (voir section 7.3.1) pour valider l'approche connexionniste employée (architectures, apprentissage).

Devant les résultats encourageants obtenus, nous avons ensuite mené des études détaillées sur la base B2.

Cellules cachées	Nombre de poids	Erreur Test
5	10 065	28 ± 7 %
6	12 076	25 ± 6 %
7	14 087	19 ± 6 %
8	16 098	20 ± 6 %
9	18 109	20 ± 6 %
10	20 120	22 ± 6 %
20	40 230	23 ± 6 %
100	201 110	24 ± 6 %

TABLE 7.4 – Erreur de classification sur B1. Réseaux à connexion totale. Les erreurs présentées sont des moyennes observées sur 10 apprentissages utilisant les mêmes paramètres. La barre d'erreur indiquée correspond à l'intervalle de confiance t_{95}.

7.5 Résultats préliminaires sur la base B1

Les expériences présentées dans cette section ont pour but de valider l'utilisation de réseaux TDNN pour la reconnaissance de visages.

Nous avons utilisé la base B1 présentée plus haut : 10 personnes, imagettes 40x50 ; 515 images sont utilisées pour l'apprentissage, et 128 pour mesurer les performances en généralisation.

7.5.1 Réseaux à connexions totales

Afin de reproduire les résultats présentés dans la littérature (voir section 7.2.2), nous avons d'abord mesuré les performances atteintes sur la base B1 par un réseau classifieur à connexions totales.

Les réseaux de ce type possèdent deux couches de poids. Ils sont complètement caractérisés par le nombre de cellules dans chaque couche, ici $(40 \times 50, N, 10)$, le nombre de classes étant égal à dix pour les expériences sur B1. La valeur optimale du nombre de cellules cachées N dépend de la complexité du problème et du nombre d'exemples disponibles. Il est usuellement déterminé par essai/erreur. Cette procédure est envisageable car ces réseaux apprennent vite (leur implémentation, en terme de deux produits de matrice, est habituellement très efficace), et la taille de la base B1 est petite.

Comme indiqué dans la section 7.3.1 (table 7.2), la validité statistique de ces résultats est faible : lorsque l'on mesure une erreur de 15 % sur les images de test, l'erreur est probablement dans l'intervalle [10 %, 20 %]. Ceci explique en grande partie les fluctuations de performances que nous avons

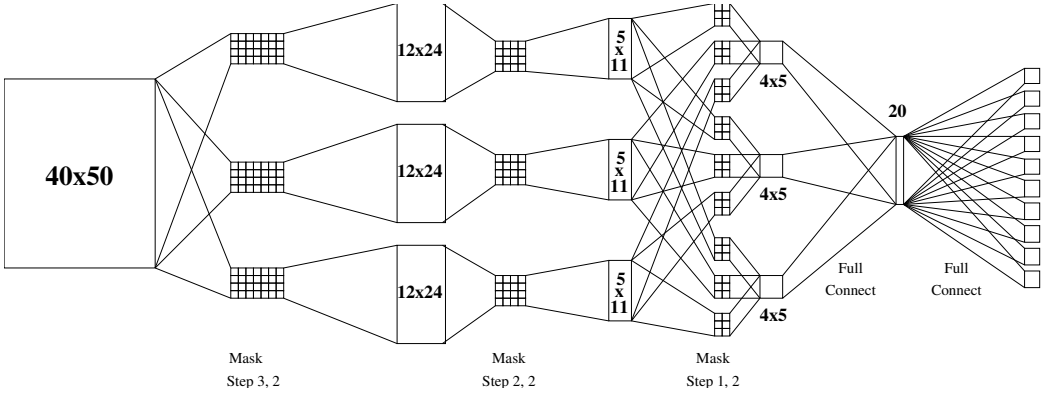


FIGURE 7.11 – Architecture du réseau TDNN «faci40x50» utilisé pour la reconnaissance des visages de la base B1. La rétine comporte 40x50 cellules. La taille de la couche de sortie peut varier entre 2 et 16 suivant le nombre de classes à reconnaître (fixé à 10 pour la base B1).

observées d'un apprentissage à l'autre. Afin de gommer ces fluctuations, nous présentons dans la table 7.4 les erreurs moyennes (sur 10 apprentissages utilisant les mêmes paramètres et différentes initialisations aléatoires des poids) mesurées sur les 128 images de l'ensemble de test.

On voit que le réseau «optimal» comporte 7 cellules cachées, et arrive à 19 % d'erreur.

Il est clair que tous ces réseaux comportent un nombre bien trop élevé de paramètres libres. On a toujours observé une chute des performances sur l'ensemble de test après l'atteinte de l'optimum, tandis que l'erreur d'apprentissage tendait vers 0. Une solution à ce problème pourrait être de *régulariser* le réseau, en ajoutant un terme de «decay» au coût global[143, 87, 162]. Nous avons préféré contraindre l'architecture même du réseau, pour y incorporer des connaissances sur le problème (structure 2D, invariance par translations) et réduire du même coup le nombre de paramètres libres.

7.5.2 Réseau TDNN : faci40x50

Description de l'architecture

Nous avons ensuite dessiné un réseau utilisant des masques de poids partagés. On a nommé ce réseau «**faci40x50**», il est représenté sur la figure 7.11.

Nous sommes arrivés à ce réseau après en avoir essayé quelques autres du même type, que nous avons raffinés peu à peu (en jouant sur le nombre et la taille des différents masques de poids) avant d'obtenir des performances

Taux de rejet	Erreur
0 %	10 %
10 %	8 %
20 %	4 %
30 %	2 %
50 %	1 %

TABLE 7.5 – Erreur sur l'ensemble de test de la base B1 (10 personnes), pour différents taux de rejet.

optimales.

C'est une architecture de type TDNN, qui comprend au total 6 couches de cellules (4 couches cachées). La couche de sortie comporte 10 cellules, associées à chacune des classes à reconnaître. On a un total de 3119 cellules, 30431 connexions et 1844 poids (paramètres libres), contre plus de 100 000 pour les réseaux à connexions complètes. L'utilisation de nombreux masques de poids partagés permet de maintenir un nombre de paramètres libres raisonnable, malgré un nombre de connexions élevé.

Résultats

La table 7.5 donne les taux d'erreurs mesurés sur l'ensemble de test (qui comprend 128 images) en fonction du taux de rejet. Le rejet est ici effectué en utilisant le critère *MLP-3*, présenté dans la section 3.4.5.

La figure 7.12 montre les premiers visages de l'ensemble de test rejetés, et la figure 7.7 (page 205) quelques visages correctement reconnus. Il s'agit ici bien entendu de rejet d'ambiguïté : le classifieur rejette car la forme présentée est proche de plusieurs classes.

7.5.3 Discussion

Les résultats obtenus sur la base B1 montrent que les réseaux TDNN permettent d'obtenir des performances très intéressantes en identification de visage. En particulier, on a diminué le taux d'erreur de 9 % (division par 2) grâce à l'utilisation d'un réseau TDNN [19, 20]

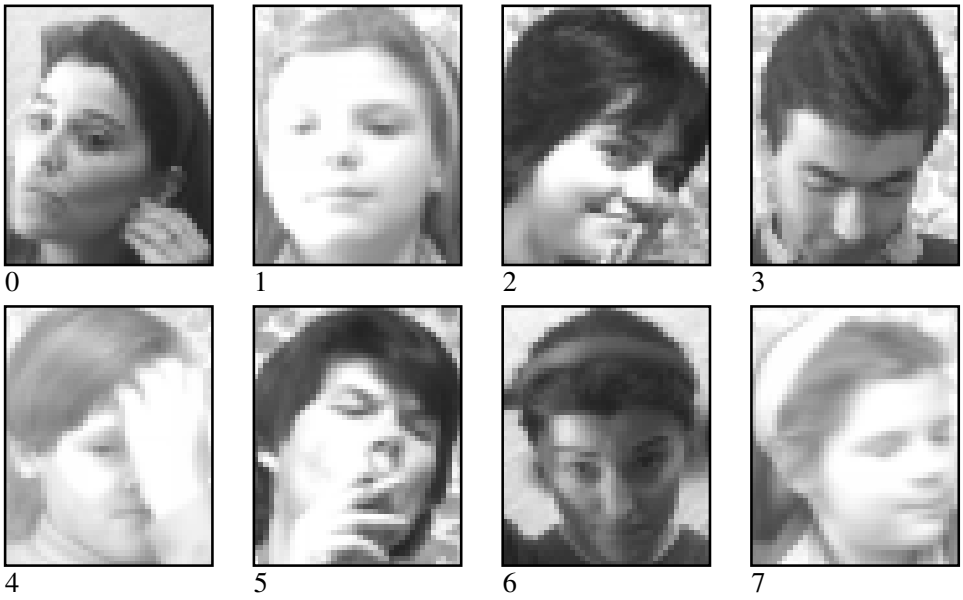


FIGURE 7.12 – Visages rejetés dans l'ensemble de test de B1.

7.6 Résultats sur la base B2

Après le succès des expériences préliminaires décrites plus haut, nous avons voulu étudier plus finement les performances du système d'identification de visages. Pour cela, nous avons utilisé la base B2.

Nous allons détailler les expériences menées sur cette base. Nous nous sommes intéressé aux points suivants :

1. résolution minimale nécessaire ;
2. effet de la quantification ;
3. nombre d'exemples nécessaires ;
4. variation des performances avec le nombre de classes ;
5. tolérance aux translations et changements de tailles des images ;
6. rejet des images ambiguës, et détection des inconnus.

Nous allons passer en revue ces différentes questions.

7.6.1 Variation de la résolution

Il est communément admis (voir section 7.1.2) que la résolution minimale nécessaire à la caractérisation des visages est comprise entre 20x20 et 32x32

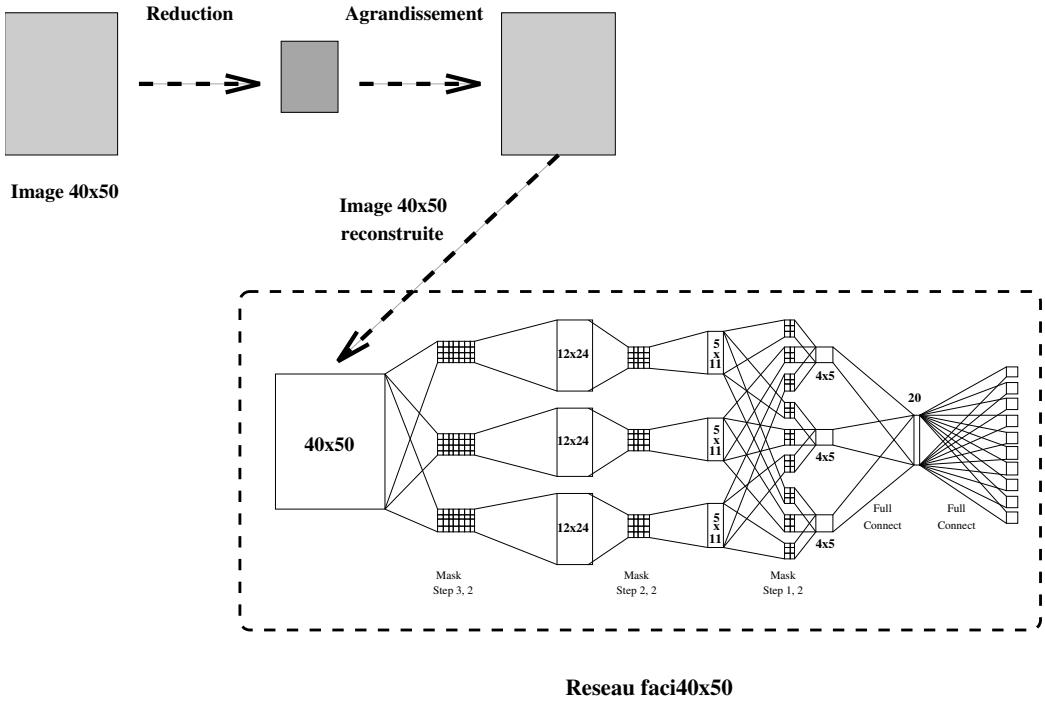


FIGURE 7.13 – Architecture utilisée pour mesurer l'effet des changements de résolution.

pixels. Cependant, dans les expériences préliminaires décrites ci-dessus, on a utilisé une résolution de 40x50. Il semble donc que l'on puisse sans risque travailler avec des images plus petites.

Afin de comparer aisément les performances d'identification (taux d'erreurs), pour différentes résolutions, nous avons choisi d'utiliser le même réseau, «faci40x50», et de faire varier la *résolution effective* des images. Pour cela, nous avons réduit la taille des images initiales (40x50) jusqu'a une taille intermédiaire (w, h) (en utilisant une interpolation suivie d'un sous-échantillonnage), puis ré-agrandit l'image à sa taille initiale. Cette image est ensuite présentée au réseau de classification (voir figure 7.13). Cette technique revient simplement à effectuer un filtrage passe-bas de l'image (*smoothing*).

Le taux d'erreur a été mesuré sur l'ensemble de test de la base B2, avec une famille de 11 personnes. 150 exemples par personne ont été utilisés pour l'apprentissage.

Les résultats sont présentés dans la table 7.6. Ces résultats appellent une remarque préliminaire : les taux d'erreurs sont ici bien plus faibles que ceux obtenus sur la base B1. Ceci s'explique, nous l'avions déjà annoncé, par la meilleure qualité de la base (peu d'occultations, pas de fortes rotations et pro-

Taille effective	Erreur test
40 x 50	2.0
25 x 31	2.0
20 x 25	2.0
15 x 18	3.4
10 x 12	5.8
5 x 6	15.3

TABLE 7.6 – Erreur d'identification (sans rejet) en fonction de la résolution. 11 personnes, 150 exemples/personne, réseau **faci40x50** (voir figure 7.13).

fils), par une segmentation plus précise (car automatisée), ainsi que par des pré-traitements plus judicieux (voir annexe B). Notons au passage que cette variation des résultats confirme le fait qu'il faut comparer les performances annoncées dans la littérature avec précaution.

D'autre part, ces résultats indiquent clairement que les performances ne se dégradent pas tant que la résolution reste supérieure à 20x25 pixels. Le filtrage passe-bas de l'image ne dégrade pas les performances de façon sensible. Ceci s'explique par le fait que le filtrage, dans un premier temps, diminue le bruit sur les images, sans nuire à l'information utile pour la classification. Cette technique figure d'ailleurs parmi les pré-traitements standards en OCR (voir par exemple [87]). Le filtrage passe-bas permet de réduire la dimension de l'espace d'entrée.

Nous avons tiré parti de ces résultats en concevant un réseau de taille plus réduite, doté d'une rétine 20x25.

7.6.2 Réseau 20x25 : faci20x25

Le réseau **faci20x25** est représenté sur la figure 7.14. Son architecture a été calquée sur celle du réseau **faci40x50**. La réduction de la taille de la rétine a conduit à supprimer une couche cachée.

Ce réseau est beaucoup plus petit que le précédent. Pour une couche de sortie comportant 10 cellules, on a un total de 821 cellules, 5861 connexions et 1877 poids. En fait, la capacité de ce réseau est du même ordre que celle du précédent, mais le nombre de connexions est 6 fois plus faible, ce qui permet de réduire le temps de traitement.

Nous avons utilisé ce réseau pour toutes les expériences suivantes.

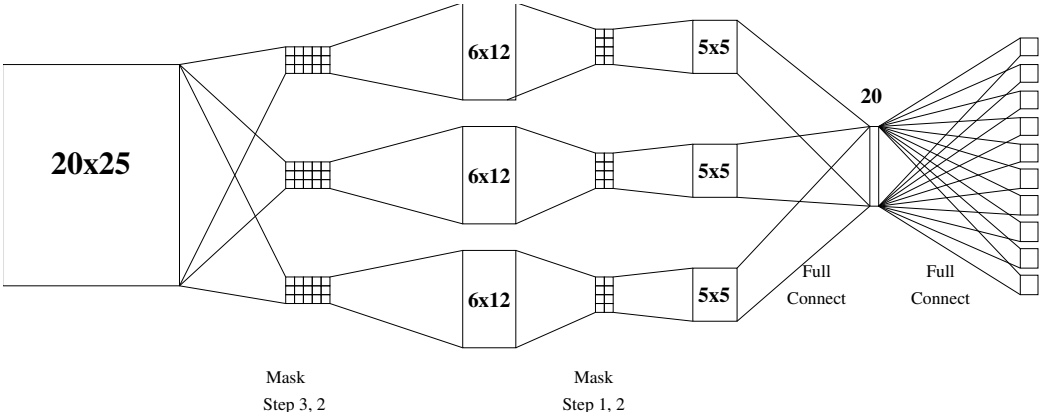


FIGURE 7.14 – Architecture du réseau TDNN «faci20x25». Ce réseau est similaire à «faci40x50» présenté plus haut, mais est doté d'une rétine 20x25. Il comporte une couche de cellules cachées en moins. Ici aussi, la taille de la couche de sortie peut varier suivant le nombre de classes à reconnaître.

7.6.3 Variation de la quantification

Nous avons évoqué plus haut (section 7.1.2) l'importance de la quantification des images pour la détection et la reconnaissance de visages. Les résultats présentés ici permettent de chiffrer les variations de performances du module de reconnaissance lorsque le nombre de bits/pixel varie.

Le tableau 7.7 donne l'erreur (mesurée sur l'ensemble de test) pour différentes quantifications. L'apprentissage est ici effectué en utilisant les images avec la même quantification. Le réseau est **faci20x25**, avec les mêmes ensembles d'apprentissage et de test qu'en section 7.6.1.

Bits/pixel	Erreur test
8	1.9
4	2.1
3	4.1
2	7.4

TABLE 7.7 – Erreur d'identification (sans rejet) en fonction du nombre de bits/pixels. Réseau **faci20x25**, 11 personnes, 150 exemples/personne.

L'erreur de classification augmente d'abord lentement, puis double si l'on utilise moins de 4 bits/pixel. En pratique 6 bits/pixel seraient souvent suffisant (en particulier pour une implémentation sur hardware dédié), mais l'on

Nombre d'images apprentissage	Erreur apprentissage	Nombre d'images test	Erreur test
200	1.0	100	1.6
100	0.7	200	2.0
60	0.5	240	4.0
30	0.2	270	10.0
15	0.0	285	12.0

TABLE 7.8 – Erreur en fonction du nombre d'exemples par personne. 11 classes.

utilise couramment 8 bits (1 octet) sur les calculateurs ordinaires.

7.6.4 Variation du nombre d'exemples

Nous nous intéressons ici aux variations de performances lorsque le nombre d'exemples utilisés pour l'apprentissage varie. La famille utilisée est la même que précédemment (11 personnes). On fait varier de 15 à 200 le nombre d'images de chaque personne utilisées pour l'apprentissage. Dans chaque cas, on a mesuré l'erreur sur un ensemble de test composé de toutes les images restantes (la taille de l'ensemble de test est donc ici variable).

Les résultats sont donnés dans la table 7.8. On observe ici le comportement attendu : plus le nombre d'exemples utilisé est faible, plus l'erreur sur l'ensemble de test augmente. Le réseau apprend alors «par cœur» la base d'apprentissage, et est incapable de généraliser correctement.

Pour atteindre une erreur inférieure à 2 %, il est nécessaire d'utiliser plus de 100 images différentes par personne. On peut aisément atteindre ce nombre à l'aide d'une camera vidéo reliée directement à un calculateur.

7.6.5 Variation du nombre de classes

La figure 7.15 donne le taux d'erreur mesuré en fonction du nombre de personnes dans la famille apprise. On utilise toujours le même réseau (faci20x25), dont on adapte simplement le nombre de cellules de la dernière couche à la taille de la famille.

Pour chaque taille N de famille, on a sélectionné aléatoirement une dizaine (au plus) de familles distinctes parmi les 15 personnes de la base B2. On a ensuite effectué l'apprentissage du réseau sur chacune des familles obtenues (en utilisant toujours 150 exemples par personne). Pour chaque taille, on a relevé le meilleur et le pire taux d'erreur observé (voir courbe 7.15). L'erreur

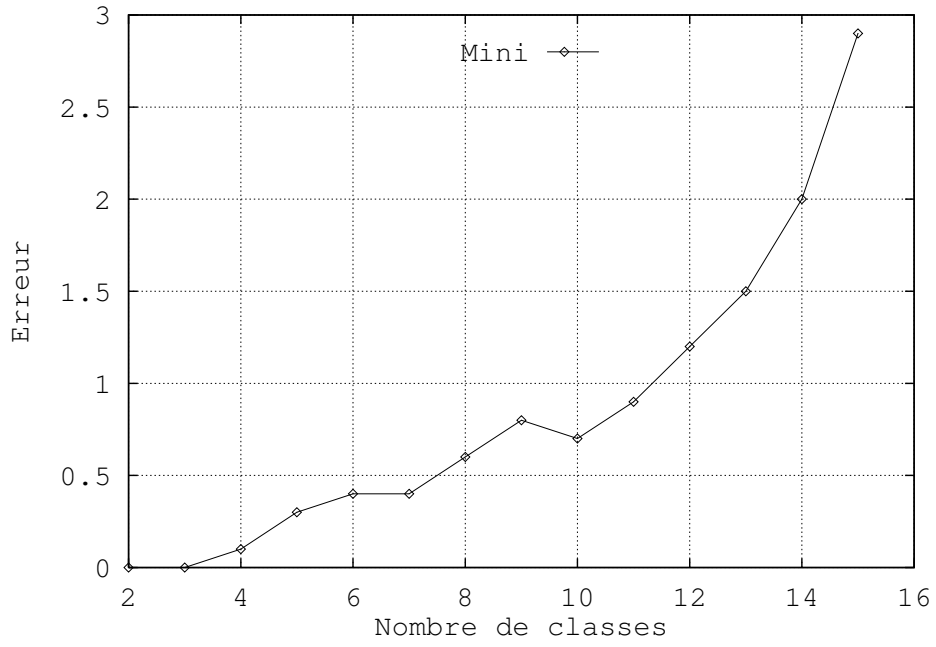


FIGURE 7.15 – Erreur en fonction du nombre de personnes. 150 exemples par personne. On a tracé les meilleures et pires performances obtenues sur des familles (composées aléatoirement) de chaque taille.

la plus faible est observée dans les familles ou‘ il est très facile de distinguer les différents membres. L'erreur augmente quand plusieurs personnes ressemblantes se retrouvent au sein de la même famille.

On constate que les performances se dégradent lentement lorsque l'on augmente le nombre de classes. Le taux d'erreur reste à un niveau très satisfaisant pour une famille de 15 personnes.

Nous n'avons pu, faute de données, tester le système sur des familles de plus grandes tailles. Si l'on extrapole ces résultats, on peut constater que le réseau utilisé va assez rapidement se trouver insuffisant, pour des familles de tailles supérieures à 20 personnes environ. Pour traiter ce problème, il est envisageable d'adopter une stratégie permettant de découper la tâche de classification en plusieurs sous-tâches. Y. Bennani [5] a récemment proposé une architecture hybride, basée sur l'emploi d'un réseau aiguilleur, pour les problèmes d'identification du locuteur, ou‘ le nombre de classes dépasse couramment la centaine (voir figure 7.16).

7.6.6 Tolérance aux translations et changements de tailles

La tolérance du classifieur aux transformations géométriques de l'image présentée est une propriété très importante en pratique. En effet, le module d'identification va être précédé par un module de segmentation chargé du cadrage des visages. Toute imprécision de ce dernier va se reporter sur le module d'identification. Il est de toute façon illusoire de compter sur un cadrage au pixel près.

Compte tenu de la nature de l'algorithme de segmentation que nous allons utiliser (voir chapitre 8), deux types de transformations sont à prendre en compte prioritairement : les translations (horizontales et verticales) de l'image, et les dilatations/réductions.

Translations des visages

Afin de mesurer la tolérance du réseau d'identification aux petites translations, nous avons procédé de la façon suivante : nous sommes parti du réseau faci20x25 entraîné sur une famille de 11 personnes, comme précédemment (150 exemples/personne). Nous avons ensuite translaté la rétine du réseau horizontalement ou verticalement. Pour chaque valeur de translation (nombre de pixels), nous avons mesuré le taux d'erreur sur l'ensemble de test (150 images/personne).

Les résultats de cette expérience sont donnés dans les tableaux 7.9 et 7.10
.

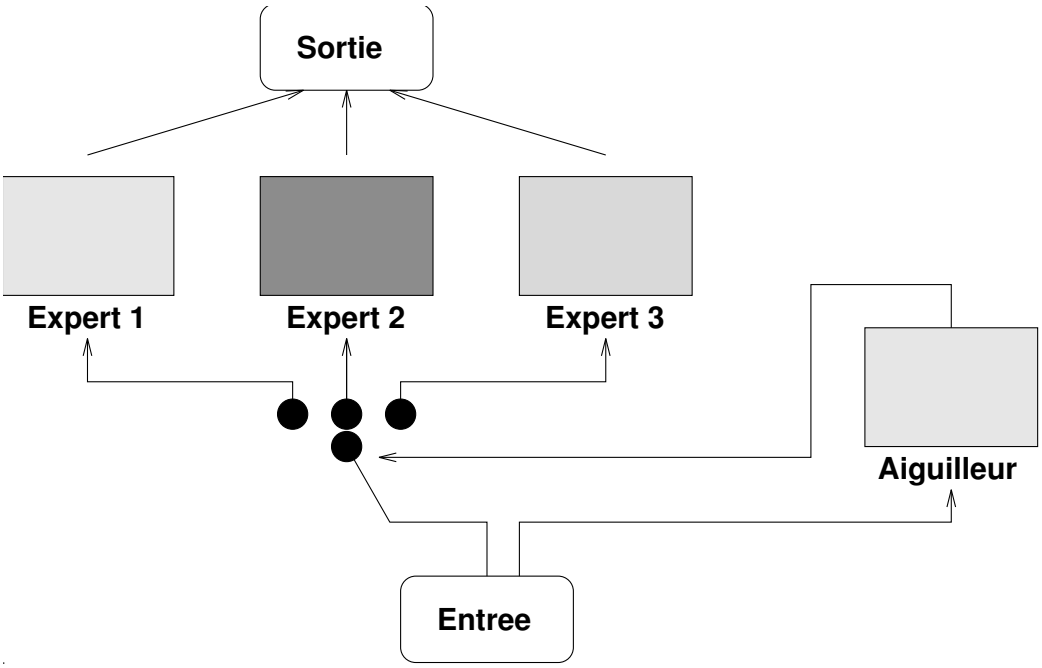


FIGURE 7.16 – Architecture multi-modulaire avec aiguilleur pour traiter les problèmes de classification à nombre élevé de classes. Chaque réseau expert est spécialisé dans la reconnaissance d'un *type* donné de formes. Par exemple, en reconnaissance de locuteur, on peut aisément séparer les voix masculines et féminines. Pour l'identification de visages, on pourrait envisager de séparer les blonds des bruns, etc...

Translation (dx)	Erreur
0	1.9
1	2.1
2	3.5
3	65.
4	91.
5	92.

TABLE 7.9 – Erreur en fonction du nombre de pixels de translation horizontale (rétine 20x25).

Translation (dy)	Erreur
0	1.9
1	2.7
2	4.3
3	29.
4	44.
5	53.

TABLE 7.10 – Erreur en fonction du nombre de pixels de translation verticale (rétine 20x25).

On voit qu'au delà de deux pixels de décalage, les performances chutent brutalement. Ceci correspond à un décalage d'environ 10% de la largeur du visage (20x25). On remarque aussi que, pour les petites valeurs de translation, la classification est plus sensible aux décalages verticaux qu'horizontaux. Ceci s'explique par la marge laissée à gauche et à droite des visages (voir figure 7.9), qui est plus importante (et variable dans l'ensemble d'apprentissage) que la marge supérieure.

Dilatation/réduction des visages

On a procédé de la même façon pour mesurer la sensibilité aux dilatations. Dans ce cas, on a (au moment du découpage du visage, avant pré-traitements) dilaté ou réduit le rectangle englobant le visage. On a ensuite mesuré les performances d'identification sur les visages obtenus, pour chaque taux de dilatation.

Les résultats sont donnés dans le tableau 7.11.

On voit que les performances se dégradent très rapidement lorsque le

Dilatation	Erreur
0.7	13.3
0.8	3.4
0.9	2.2
1.0	1.9
1.1	8.2
1.2	13.4
1.3	32.0

TABLE 7.11 – Erreur en fonction de la dilatation du visage. Le taux de dilatation indiqué mesure la taille du visage exprimé en fonction de la taille de la rétine.

visage est grossi. Dans ce cas, en effet, le contour du visage sort de la rétine et n'est plus disponible pour l'identification. Par contre, il n'est pas très gênant de réduire la taille du visage, jusqu'a 20 %.

7.6.7 Rejet et Détection des inconnus

Position du problème

Idéalement, le module d'identification de visage devrait être susceptible de fournir l'une des réponses suivantes lorsqu'on lui présente une image :

— identité du visage ;
— visage ambigü, refus de classer ;
— visage inconnu (invité) ;
— non-visage : image représentant un objet quelconque.

Cette situation est résumée par la figure 7.17. En fait, le classifieur que nous employons nous donne une estimation des probabilités a-posteriori d'appartenance de l'image aux classes de visages connus [177]. Nous avons vu dans le chapitre 3 que l'on pouvait utiliser cette information pour décider de rejeter ou de classer la forme présentée.

Pour cela, on utilise un *critère de rejet*. D'après les résultats des comparaisons menées sur les chiffres manuscrits (voir chapitre 6, section 6.3.3), le critère *MLP-3* est le plus performant, en particulier en présence d'*outliers*.

Si l'on mène une analogie avec le problème de reconnaissance et rejet des chiffres manuscrits (chapitre 6, page 132), on peut identifier les cas précédents aux ensembles de chiffres :

— visage connu (reconnu ou ambigü) : base *hsf0* ;
— visage inconnu (invité) : base *Zarbi* (lettres) ;

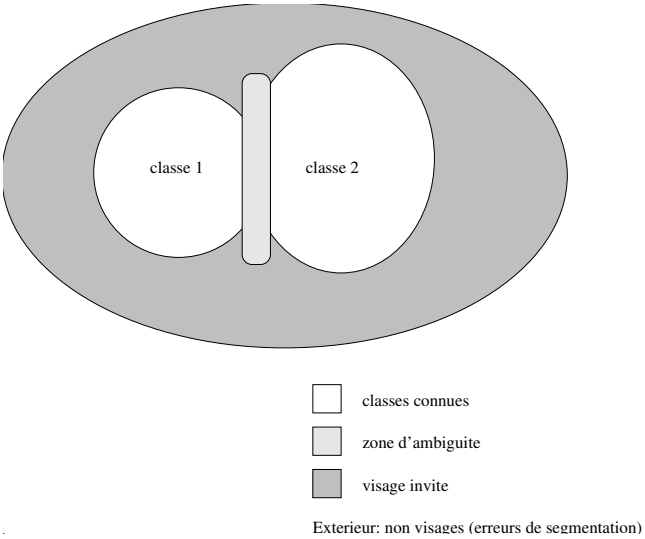


FIGURE 7.17 – Les quatres régions à distinguer dans un système d'identification de visages.

— non-visage : base *Random* (bruits blancs).

En effet, les visage inconnus sont proches des visages connus (mêmes caractéristiques bas niveau), comme les lettres manuscrites le sont des chiffres. On a vu que lors des expériences sur les chiffres, les images de la base *Random* étaient rejetées de la même façon que les images de *Zarbi*. Ce résultat semble indiquer qu'il va être difficile d'employer un classifieur (entraîné sur les visages connus) pour séparer les invités des autres images.

L'algorithme de rejet des invités peut commettre deux types d'erreurs : prendre une personne connue pour un invité, ou identifier un invité comme une personne connue. Plus on veut être sévère, c'est à dire détecter une forte proportion d'invités, et plus on risque de rejeter des images de personnes connues. Il convient donc de choisir judicieusement les paramètres de l'algorithme de détection (*seuils de rejet*) pour réaliser le meilleur compromis.

Mesure des taux de rejet

Afin de mesurer la capacité du réseau à rejeter les invités et les images inconnues, nous avons utilisé deux bases de données d'images supplémentaires :

1. la base *BDI*, décrite en section 7.3.2, contient 600 images, de 6 personnes ;
2. la base *Random*, contenant 765 images de bruits blancs, générées de la même façon que dans le chapitre 6, cette fois en taille 20x25 pixels.

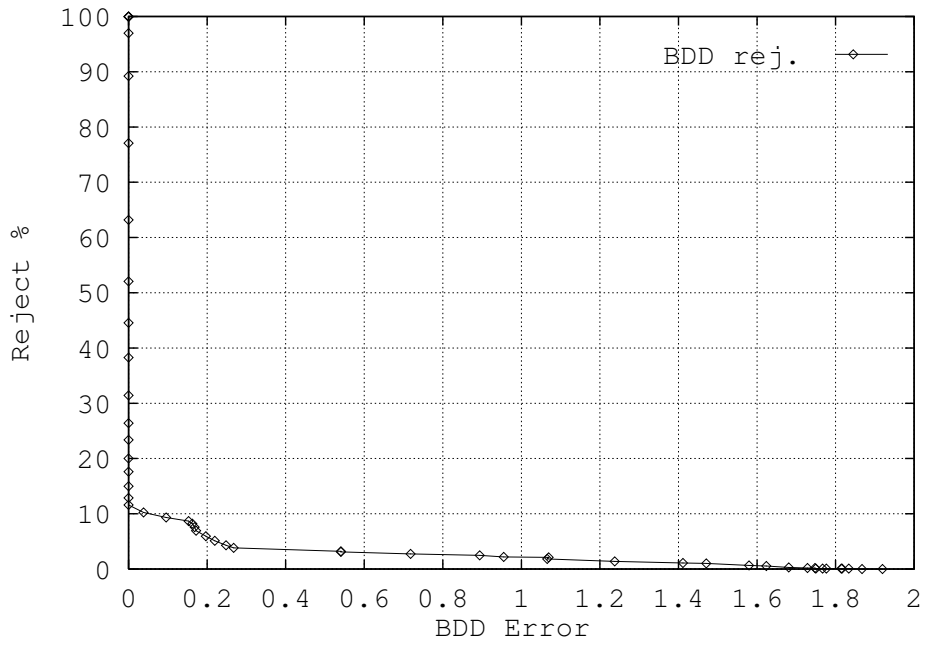


FIGURE 7.18 – Taux de rejet en fonction de l'erreur de reconnaissance, sur trois ensembles d'images : *BDD* (visages connus), *BDI* (invités) et *Random* (bruits aléatoires).

La courbe 7.18 donne le taux de rejet sur *BDD* (ensemble de test, 11 personnes connues), *BDI* et **Random**, en fonction de l'erreur de classification sur *BDD*. Cette courbe a été obtenue en utilisant le critère de rejet *MLP-3* (voir section 3.4.5), et en éliminant le seuil Θ. Le réseau utilisé est faci20x25, entraîné sur 11 personnes de *BDD*.

Comme prévu, le classifieur n'est pas capable de distinguer entre les visages inconnus et les autres images : à 1 % d'erreur, 60 % des invités et 73 % des images de *Random* sont rejetés. Cette limitation est principalement due au fait que l'on n'utilise que des visages «connus» pour l'apprentissage.

La détection des images inconnues (qu'elles soient des visages ou non) est toutefois satisfaisante : à 5 % de rejet sur les visages connus, on commet 0.2 % d'erreur de classification, et l'on rejette plus de 80 % des visages inconnus.

Ces résultats montrent que dans une chaîne automatique d'identification de visages, ou‘ le module de segmentation est peu fiable, il faudra utiliser un premier étage de classification pour séparer les images de visages (connus ou

non) des autres. Nous reviendrons sur ce point dans le chapitre 8.

7.7 Conclusion

Les expériences décrites dans ce chapitre nous permettent de tirer les conclusions suivantes, aux conséquences importantes pour la mise au point d'une chaîne automatique d'identification de visages :

- **Type d'images à employer** : des imagettes de 20x25 pixels, quantifiées sur 4 à 8 bits sont suffisantes pour l'identification au sein de familles comprenant de 2 à 15 personnes ;
- **Nombre d'exemples** : la robustesse du classifieur croît évidemment avec le nombre d'exemples d'apprentissage employé. Il est nécessaire d'utiliser au moins une centaine d'images par personne pour obtenir une erreur (en généralisation) inférieure à 2 %.
- **Classifieur** : l'utilisation d'un réseau TDNN ad-hoc nous a permis de diviser l'erreur de classification par deux, par rapport aux réseaux à connexions complètes (qui sont les seuls présentés dans la littérature récente en traitement de visages).
- **Détection des inconnus** : l'utilisation d'un critère de rejet simple sur les sorties du TDNN permet de détecter 80 % des visages inconnus en rejetant 5 % des visages connus. Il n'est pas possible d'utiliser le même classifieur pour séparer les visages inconnus des images d'autres types (bruits).

Chapitre 8

Système de segmentation de visages

8.1 Introduction

Ce chapitre décrit les expériences que nous avons menées pour mettre au point un algorithme de localisation (segmentation) de visages dans des images de scènes. Cet algorithme constitue le premier étage d'une chaîne automatique de détection/identification de visage.

On travaille ici sur des images représentant des scènes naturelles (la plupart du temps en intérieur), pouvant contenir un nombre quelconque de personnages, dont le visage est plus ou moins visible. On n'impose aucune contrainte sur les décors.

Avant de continuer, il convient de spécifier plus précisément les performances attendues de notre dispositif. Les spécifications que nous nous sommes données sont les suivantes :

— Position : on doit détecter les visages tournés vers l'objectif de la caméra. On autorise (comme en identification) des rotations latérales de la tête de $\pm 30°$, ainsi que de légères ($\sim 10°$) rotations verticales. On doit aussi détecter les visages partiellement occultés (par une main, un journal, ...).

— Distance : la distance réelle du visage dépendant de l'optique utilisée, on parlera plutôt de *taille apparente*, exprimée en pixels sur l'image traitée. On vu dans le chapitre 7 que la taille minimale nécessaire à l'identification d'un visage était d'environ 20x25 pixels. Il est raisonnable de penser que cette taille est aussi suffisante pour détecter le visage dans une scène.

— Eclairage : il est difficile de spécifier quantitativement la qualité de

FIGURE 8.1 – Une scène typique, dans laquelle on désire localiser les visages.

l'éclairage. Disons qu'il doit permettre de distinguer les détails des visages (yeux, nez). Le contraste entre le visage à détecter et le fond avoisinant est tout aussi important, mais ne dépend pas du seul éclairage : on ne distinguera jamais de contour entre des cheveux noirs et un fond noir...

Une approche «*haut-niveau*» de la segmentation Nous avons choisi de traiter la détection de visages comme un problème de segmentation *haut-niveau*. En effet, il est vain d'essayer de qualifier un visage, pouvant être tourné ou partiellement caché, par une texture ou un type de contour spécifique. La seule information exploitable est que l'on cherche une forme de type «visage» ! Nous allons donc rassembler des exemples d'images de visages, et tenter de construire un dispositif capable de distinguer entre une image de visage et n'importe quelle autre type d'image. Il s'agit là d'un projet ambitieux ; d'une part, l'ensemble de toutes les images de visages possibles est assez mal défini : quantité de facteurs (attitudes, couleur de peau, éclairage, coiffure,...) contribuent à une très grande variabilité de ces images. L'ensemble des autres images est quant à lui désespérément grand : il parait insensé de caractériser «toutes les images qui ne sont pas des visages».

Ce problème est intéressant du point de vue de la classification et de l'apprentissage : il s'agit de construire un classifieur distinguant une classe (les visages) du reste du monde (les autres images). On voit qu'il s'agit d'un problème apparenté aux techniques d'*authentification*. En contrôle d'accès basé sur la parole, par exemple, on cherche un codage de la voix de la personne à authentifier qui permette de la distinguer de tous les autres type de signaux

sonores possibles.

Dans ce chapitre, nous allons explorer plusieurs codages possibles pour les visages, et montrer comment l'utilisation de plusieurs classifieurs discriminants en cascade permet de résoudre le problème.

Une approche *multirésolution* Il ne suffit pas de disposer d'un classifieur distinguant des images de visages (par exemple de taille 20x25) des autres. Les visages présents dans les scènes ont des tailles variables, données par leur distance à l'objectif. Dans notre cas, la largeur des visages varie de 20 à 150 pixels environ. Pour nous affranchir de cette difficulté, nous allons utiliser une *analyse multirésolution*, décomposant la scène en une série de vues à différentes échelles. Au lieu de traiter une seule image de taille 512 par exemple, nous aurons à traiter une suite d'images de tailles $\{512, 362, 256, 181, 128, 90, 64, \ldots\}$. Chacune de ces images représente la scène entière, avec plus ou moins de finesse dans le détail. Le même visage apparaîtra donc dans chacune de ces images avec une taille différente.

Nous pouvons maintenant utiliser notre classifieur, doté d'une rétine de taille fixe (par ex. 20x25), pour balayer toutes les positions de chacune des images issues de l'analyse multirésolution.

Une telle approche paraît de prime abord irréaliste par le volume de calculs qu'elle engendre : plus de 100 000 appels au classifieur «visage/non-visage» pour traiter une image de scène. C'est le prix à payer pour bâtir un système de segmentation n'utilisant aucune hypothèse spéciale sur la position des objets à rechercher. De telles hypothèses pourrait être du type : les visages intéressants sont situés entre 1.4 et 2.1 mètres du sol, ou encore utiliser d'autres sources d'information pour contraindre les zones de recherche (suivi dynamique dans le cas de séquences d'images par exemple).

Toutefois, nous verrons qu'il est possible (et nécessaire) d'utiliser un classifieur de très petite taille, donc peu consommateur de calculs, apparenté à un filtrage non linéaire de l'image. Le système final que nous avons développé utilise moins de 50 secondes de calculs par image sur une station de travail Sun Sparc 1.

Plan du chapitre Nous commencerons par une brève revue bibliographique des travaux publiés en segmentation de visages.

Nous décrirons ensuite (section 8.3) les données à notre disposition pour cette étude, et établirons différents critères de mesure des performances.

La section 8.4 décrit deux approches non supervisées [1] du problème de

1. L'emploi du terme «non-supervisé» est ici impropre. On désigne dans ce chapitre par «non-supervisées» les techniques n'utilisant pas d'exemples de la classe «non-visage»

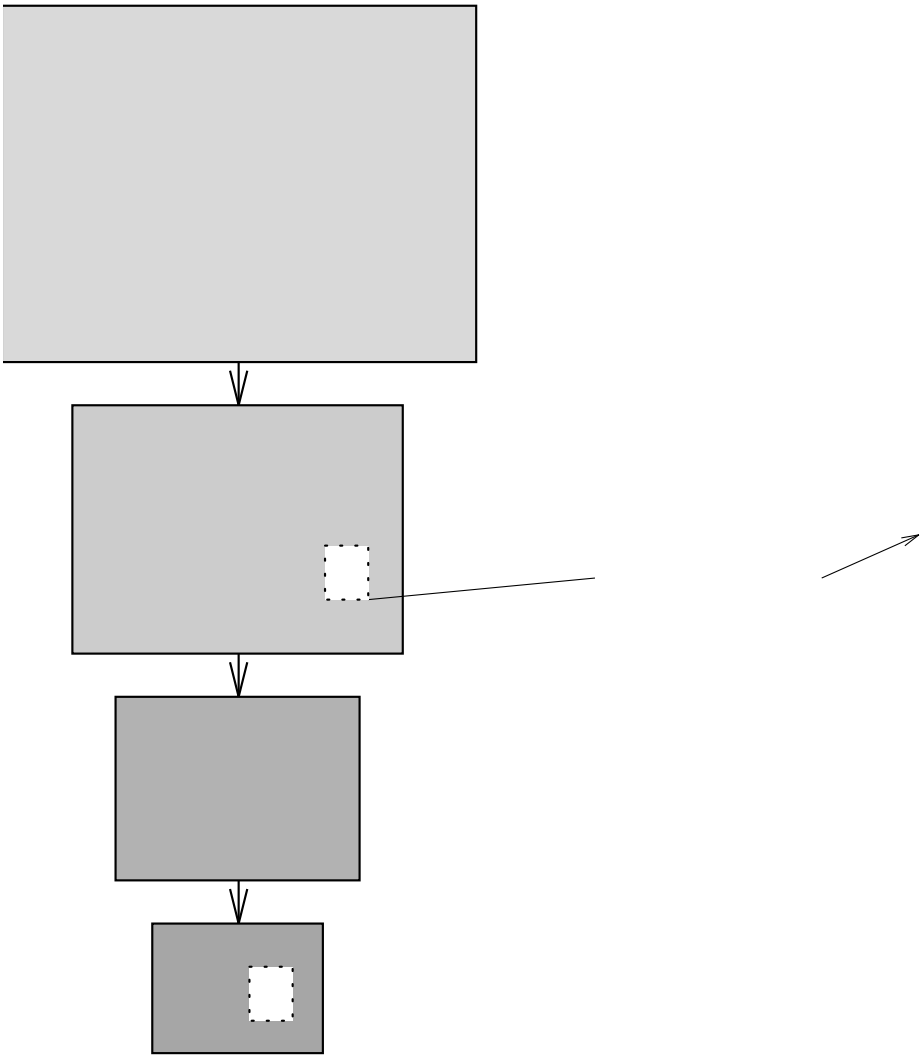


FIGURE 8.2 – Principe du système de détection de visage. L'image à analyser est décomposée en plusieurs échelles (à gauche). Chaque échelle est balayée par la rétine d'un classifieur, qui décide en chaque position s'il s'agit ou non d'un visage. Le problème principal, auquel nous consacrons l'essentiel de ce chapitre, est de bâtir un tel classifieur.

discrimination visage/fond [2] : la première est basée sur l'emploi d'un critère de rejet sur un classifieur de visages, la seconde sur un réseau auto-associatif. Nous montrerons que ces approches sont très insuffisantes.

La section 8.5 détaille la mise au point d'un classifieur *supervisé*, utilisant des exemples de fonds et de visages pendant l'apprentissage. Nous discutons du choix des données, des pré-traitements à appliquer, et donnons finalement les performances obtenues par ce système, qui restent médiocres.

Dans la section 8.7, nous abordons la conception d'un système à deux étages, permettant une sélection «intelligente» des exemples de fonds pendant l'apprentissage, et la réduction du taux de fausses alarmes sur les scènes. Ce système permet la mise au point d'un module de localisation opérationnel, dont nous détaillons le fonctionnement dans la section 8.8.

8.2 Bibliographie

Cette section résume brièvement quelques travaux publiés par d'autres équipes concernant la localisation de visages.

8.2.1 Approches «*template matching*»

Nous avons déja évoqué au début du chapitre sur la reconnaissance des visages diverses méthodes de détection d'objets (visages ou parties de visages) basées sur l'ajustement de courbes analytiques sur les contours de l'image [222, 83, 214, 46]. La forme à détecter est décrite par une courbe (par exemple une ellipse pour un visage) dont on ajuste les paramètres sur l'image.

Ces méthodes donnent de bons résultats dans des conditions bien spécifiées : on doit pouvoir décrire mathématiquement à l'avance la forme à détecter (par exemple un œil dans un visage vu de face [222]). Dans des cas plus complexes, comme celui qui nous concerne, l'application de ces méthodes semble hypothétique : la forme d'un visage dans une scène peut complètement changer lorqu'il est tourné ou partiellement voilé par un autre objet.

Govindaraju et al. [82, 83] ont tenté de mettre au point un système de détection de visages permettant de localiser des visages dans des photographies de journaux, en utilisant une modellisation des contours. L'algorithme proposé semblent donner de bon résultats si les visages sont de face et très contastés (fonds avoisinant blanc). De plus, on confond de nombreux objets avec des visages, du fait de la simplicité du modèle.

pour l'apprentissage.

2. On appellera «fond» dans ce chapitre toute image autre qu'un visage.

8.2.2 Utilisation de l'Analyse en Composantes Principales

Turk et Pentland[203] (voir aussi section 7.2.2) ont mis au point un système de localisation de visages dans des scènes. Leur système travaille sur des séquences d'images, et utilise l'information de changement d'une image à l'autre pour restreindre la zone de recherche de visages. Cette zone est balayée par une rétine, et l'on calcule pour chaque position la distance de l'imagette au sous-espace des composantes principales (*face space*) (voir la section 2.8 pour des détails sur l'ACP).

Ce système est assez proche de celui que nous proposons dans ce chapitre. Cependant, il n'intègre pas de recherche multi-échelle (la taille de la rétine utilisée est déduite de la taille du visage détecté dans l'image précédente de la séquence). D'autre part, les temps de calculs sont importants, malgré quelques astuces dans le calcul de l'ACP.

Nous avons testé ce type d'approche (section 8.4.2 de ce chapitre), et obtenu des résultats bien inférieurs à ceux des systèmes à base de classifieurs TDNN discriminants (fonds/visages) que nous proposons.

8.2.3 Système de détection des yeux de British Telecom

Les laboratoires de British Telecom ont mené une série de recherches visant à mettre au point et à comparer des algorithmes de détection de la bouche et des yeux dans des images de visages [140]. L'application en vue est la compression sélective d'images pour la communication par vidéophone : les yeux et la bouche doivent être transmis avec le maximum de fidélité, tandis que le reste de l'image peut être dégradé sans diminution du confort visuel.

Des approches «classiques» (par exemple [214], cité ci-dessus) ont été comparées à différents algorithmes connexionnistes [140].

Parmi les systèmes proposés figure un module de localisation des yeux, basé sur une analyse multirésolution [32] de l'image suivie par un balayage de chaque niveau par un réseau MLP [210, 211, 89]. Ce système est très proche de celui que nous avons proposé citeviennet :1992.1,viennet :1992.2. Le réseau est entraîné à distinguer des imagettes d'yeux de fonds. Les exemples de fonds sont sélectionnés manuellement dans des zones ressemblant (subjectivement) à des yeux.

La conclusion des auteurs est mitigée : le système semble rater une proportion importante d'yeux. Ceci est certainement imputable au faible nombre d'images utilisé pour l'apprentissage et peut être à la simplicité des MLP utilisés pour la classification. Néanmoins, ces travaux proposent d'intéressantes voies de recherches.

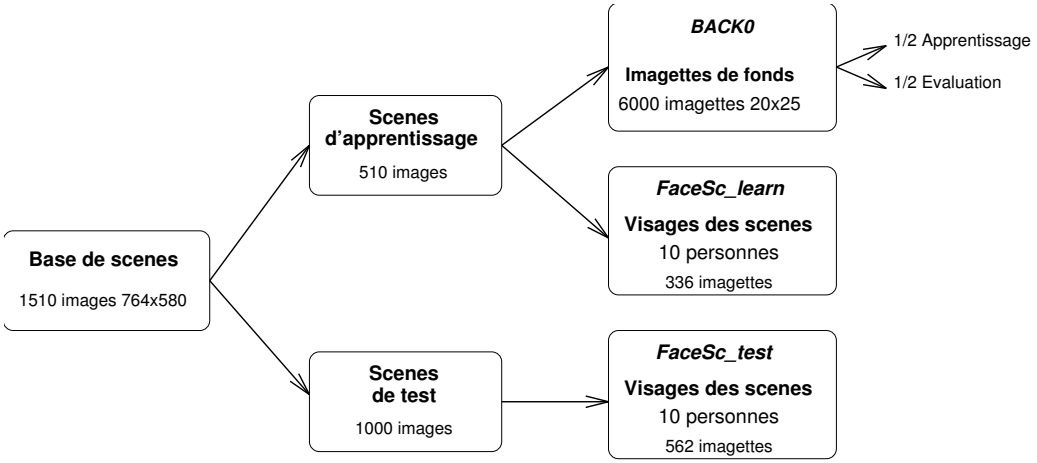


FIGURE 8.3 – Structuration des données utilisées pour la mise au point de l'algorithme de détection de visages (voir texte).

8.3 Description des données

Nous décrivons dans cette section les données sur lesquelles nous avons mené les expériences de détection de visages décrites dans ce chapitre.

8.3.1 Base de scènes

Nous avons pu disposer [3] d'un ensemble de 1510 images de taille 764x580 pixels, en 256 niveaux de gris (8 bits/pixels).

Les scènes contiennent entre zéro (c'est malheureusement fréquent !) et six visages. Ce sont des images prises en intérieur, dans des décors type mobilier de bureau ou intérieur d'appartement.

Nous avons repéré manuellement dans ces scènes les positions des rectangles encadrant chaque visage respectant les conditions énumérées dans l'introduction : visage vu approximativement de face, de largeur au moins égale à 20 pixels.

Les images ont ensuite été réparties (aléatoirement) en deux ensembles (voir figure 8.3) :

Scènes d'apprentissage : Cet ensemble contient 510 images, que l'on peut utiliser comme bon nous semble pour entraîner le module de détection de visages. De ces 510 images, on extrait 336 imagettes de

3. Cette base de données nous a été prêtée pour cette étude par la société Mimétics. L'ensemble des images de scènes représente plus de 600 mega-octets de données.

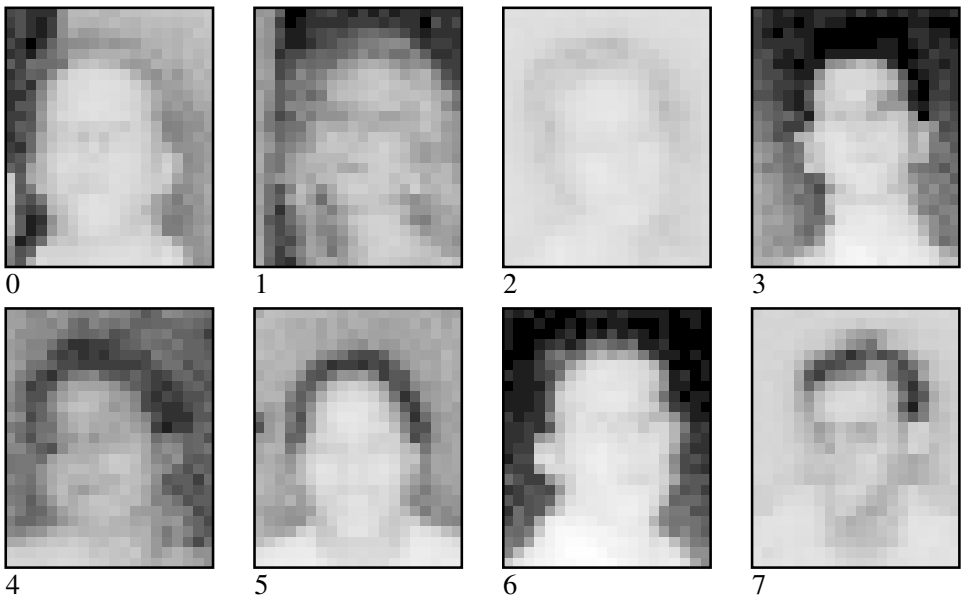


FIGURE 8.4 – Quelques imagettes extraites de l'ensemble *FaceSc*. Ces visages sont moins bien contrastés que ceux de la base B2 (figure 7.9), car l'éclairage des scènes est souvent médiocre (sur ou sous-exposition), tandis que les images de B2 sont acquises en éclairage artificiel de face.

visages, appartenant à 10 personnes différentes, qui constituent l'ensemble nommé *FaceSc_learn*.

Scènes de test : On réserve cet ensemble de 1000 images pour mesurer les performances du système après sa mise au point. Il contient 562 imagettes de visages, formant l'ensemble nommé *FaceSc_test*.

8.3.2 Visages extraits des scènes : base *FaceSc*

Les images de visages de la base *FaceSc* sont d'une qualité bien inférieure à celle des images des base B1 et B2 que nous avons utilisées pour l'identification de visage (chapitre 7). La figure 8.4 montre quelques unes de ces images (normalisées en 20x25 pixels).

8.3.3 Visages de B2

Nous disposons aussi de l'ensemble d'images de visages B2. Nous avons utilisé dans certaines expériences de ce chapitre la partie *BDD* de B2, qui

contient 4100 images de 15 personnes (voir chapitre 7, section 7.3.2, page 207).

8.3.4 Exemples de fonds

On appelle *BACK0* un ensemble de 6000 imagettes extraites des scènes d'apprentissage. Ces imagettes ont été obtenues en sélectionnant *aléatoirement* dans les 510 images de scènes des rectangles de taille quelconque (mais de rapport hauteur/largeur égal à 25/20) ne recouvrant aucun visage.

Il est important de noter que cet ensemble contient par construction des imagettes représentatives de ce que l'on rencontre quand on balaye les images de scènes à différentes résolutions. L'ensemble *BACK0* est très redondant : une forte proportion d'imagettes sont uniformément grises. Cet ensemble n'est donc pas spécialement conçu pour servir d'exemple d'images à distinguer des visages. (Pour la détection des yeux sur des images de visages, Hand et al. [89] construisent manuellement un ensemble de fonds ad hoc pour l'apprentissage de leur système. Cette solution nous parait très fastidieuse et irréaliste si l'on veut employer beaucoup d'exemples.)

Dans la section 8.7, nous montrerons comment constituer un ensemble d'imagettes de fonds contenant surtout des cas ambigüs, proches de visages pour le classifieur.

8.3.5 Critères de mesure des performances

Les performances d'un système de détection de visages se mesurent à l'aide de deux taux, qu'il convient de réduire au minimum :

1. Nombre moyen de fausses alarmes par image ;
2. Proportion de visages non détectés parmi les visages présents dans les images.

A ces deux taux, on pourrait ajouter une mesure exprimant la qualité du cadrage obtenu, qui est un paramètre important pour une chaîne de localisation/identification de visages.

Nous qualifierons ces résultats de *performances globales* du système. En effet, ces performances dépendent du réglage de toutes les parties de l'algorithme : pré-traitements, classifieur (paramètres et seuils de rejet), post-traitement. La mesure de ces taux est coûteuse (avec le matériel à notre disposition durant cette thèse) : elle nécessite l'application de l'algorithme de segmentation sur les 1000 images de l'ensemble de test. Nous indiquerons par conséquent les performances globales une fois le système figé (voir sections 8.6 et 8.8.5).

Pour la mise au point du classifieur visage/non-visage, nous utiliserons les *courbes de détection* pour comparer les performances. Ce type de courbes, déjà utilisé dans le chapitre 6 pour comparer les classifieurs de chiffres, donne le taux de fausses alarmes (i.e. acceptation erronée d'images de fonds) en fonction du taux de non détection de visages. On calcule ces courbes de différentes façons suivant le type de classifieur utilisé. Par exemple, si l'on utilise un critère de rejet, on obtient aisément la courbe de détection en éliminant le seuil de rejet θ des fonctions paramétrées (taux_alarme(θ), non_detection(θ)).

Les courbes de détection sont calculées sur des ensembles d'imagettes (visages et fonds). Les taux obtenus permettent de comparer avec fiabilité les différents systèmes, mais on ne peut pas en déduire directement les performances globales, qui sont calculées sur des images de scène. Pour passer des courbes de détection à ces performances, il faut prendre en compte le nombre d'imagettes à traiter dans une scène, qui dépend à son tour du nombre de niveaux de décomposition de l'analyse multirésolution.

Hormis dans la section 8.7, nous mesurerons le taux de fausses alarmes sur le dernier tiers (évaluation) de l'ensemble *BACK0* (2000 imagettes). Le taux de non détection sera en général mesuré sur l'ensemble *FaceSc_test* (562 imagettes de visages).

8.4 Approches "non supervisées"

Nous abordons dans cette section une première tentative pour réaliser un classifieur visages/fonds.

Les deux approches employées ici tentent d'extraire un codage des images de visages qui permettent de les distinguer des autres images, en utilisant uniquement des images de visages pour l'apprentissage. Cette démarche est séduisante, car elle évite de définir une classe de «fonds» lors de la conception du classifieur. Le caractère «infini» de cette classe, et le problème de constituer un ensemble d'exemples représentatifs de tous les fonds est ainsi évité.

Le classifieur ainsi entraîné devra d'une part *généraliser* son codage aux nouveaux visages qu'il rencontrera, et d'autre part *rejeter* toute forme autre qu'un visage.

Nous avons envisagé l'utilisation de deux types de classifieurs. La première idée est d'appliquer un critère de rejet sur un classifieur entraîné à identifier des visages. La seconde est d'employer un réseau auto-associatif pour extraire un codage des visages.

8.4.1 Utilisation d'un réseau d'identification

Nous avons étudié dans le chapitre 7 le problème de la détection des invités. Nous avons alors proposé d'utiliser un critère de rejet sur la sortie du classifieur pour rejeter les visages inconnus.

Nous allons ici appliquer le même critère pour séparer les imagettes de fond de celles de visages.

Pour cela, on reprend le réseau d'identification présenté au chapitre 7, entraîné sur 15 personnes (150 exemples/personnes) de la base *BDD*. On calcule ensuite pour chaque imagette la valeur du critère de rejet *MLP-3*.

On a tracé sur la figure 8.5a la répartition des ces valeurs sur les ensembles d'images *BACK0*, *BDD* (partie de test), et *FaceSc*. On constate que, si les visages connus sont bien séparés des fonds, ce n'est pas le cas des visages extraits des scènes. Les valeurs du critère de rejet pour ces visages se répartissent sur une large plage de valeurs, qui recouvre en partie les valeurs prises sur les imagettes de fonds.

La courbe de détection correspondante est représentée en figure 8.5b. On a ajouté sur cette courbe les visages «inconnus» de B2 (ensemble *BDI*). On voit que ces derniers sont traités de la même façon que les visages extraits des scènes, c'est à dire fortement rejetés.

La conclusion de cette expérience rejoint celle à laquelle nous étions arrivés lors de l'étude sur la détection des invités (chap 7, page 224) : le réseau d'identification rejette indifféremment dans une forte proportion toutes les images de visages inconnus et de fonds.

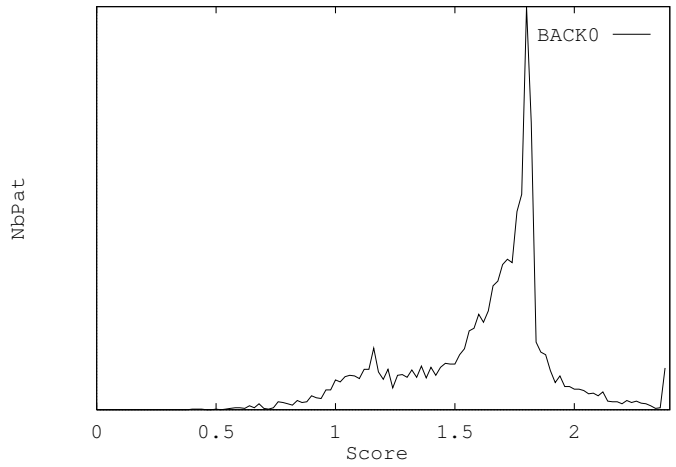
Nous avons vérifié que cette caractéristique restait vraie pour les réseaux d'identification à faible nombre de classes (2 à 7).

8.4.2 Utilisation d'un réseau auto-associatif

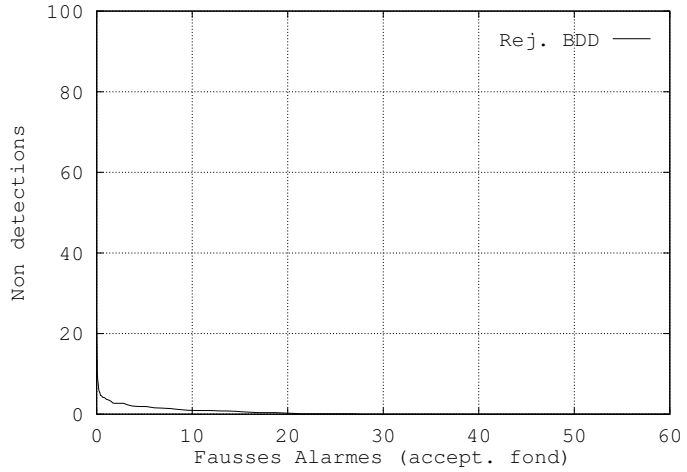
Nous avons cité dans le chapitre 7 plusieurs travaux utilisant des réseaux auto-associatifs pour calculer un codage des imagettes de visages (voir [116, 44, 203, 71]). Nous avons aussi utilisé ce type de réseaux pour la reconnaissance de chiffres (section 6.5). On a vu que ce type de réseaux permettait d'effectuer une variante d'analyse en composantes principales (ACP) (voir section 2.8).

L'architecture de ces réseaux, que l'on appelle souvent *réseaux diabolos*, est très simple : les cellules de la rétine sont connectées à toutes les cellules de la couche cachée (de taille N), elles-même connectées à la couche de sortie, de même taille que la rétine (voir figure 8.6).

L'apprentissage d'un réseau diabolo consiste à minimiser l'erreur quadratique de reconstruction sur un ensemble d'images d'apprentissage.



(a) Répartition des valeurs du critère de rejet ($\|S^k - D^k\|$).



(b) Courbes de détection. Les taux de fausses alarmes (en abscisse) sont mesurés sur l'ensemble d'imagettes *BACK0*.

FIGURE 8.5 – Résultats obtenus en en employant le critère de rejet *MLP-3* sur les sorties du réseau *faci20x25*, entraîné sur les 15 personnes de *BDD*. On a mesuré les taux sur trois ensembles de visages : *BDD* (visages connus, imagettes de test), *BDI* (visages inconnus de B2), et *FaceSc* (visages extraits des scènes d'apprentissage).

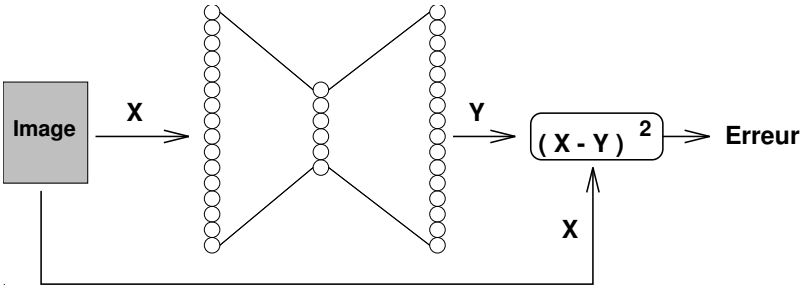


FIGURE 8.6 – Architecture d'un réseau auto-associatif diabolo. On présente une image X. Le réseau encode cette image dans la couche cachée, et reconstruit l'image Y sur la couche de sortie.

Description de l'expérience

Nous avons ici employé les visages de *BDD* (15 personnes, 150 images/personne) pour entraîner notre réseau, qui comporte 5 cellules cachées et une rétine 10x25, (soit 5505 paramètres libres). Nous sommes clairement dans une situation ou‘ le nombre de paramètres libres excède le nombre d'exemples disponibles. Augmenter le nombre de cellules cachées diminue d'ailleurs rapidement les performances de généralisation. Il n'est pas possible de réduire d'avantage la taille du réseau, qui devient alors trop petit pour apprendre la tâche. Une solution serait d'employer un réseau, éventuellement plus gros, avec une fonction de coût pénalisant la complexité de la solution (terme de régularisation). Nous ne l'avons pas mis en œuvre ici.

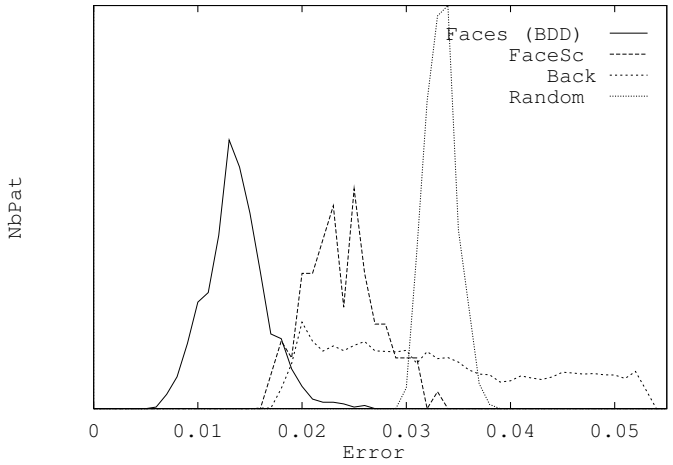
Une fois le réseau entraîné (i.e. l'erreur de reconstruction sur un ensemble d'évaluation ne décroît plus), on a mesuré les erreurs sur plusieurs ensembles d'imagettes (voir figure 8.7) : les visages connus *BDD* (test), les inconnus *BDI*, les visages des scènes *FaceSc*, les imagettes de fonds *BACK0*, et aussi les imagettes de bruit blancs *Random*.

Un simple seuillage de l'erreur de reconstruction est alors utilisé pour classer l'image comme visage ou fond. Les courbes de détection correspondantes sont tracées sur la figure 8.7b.

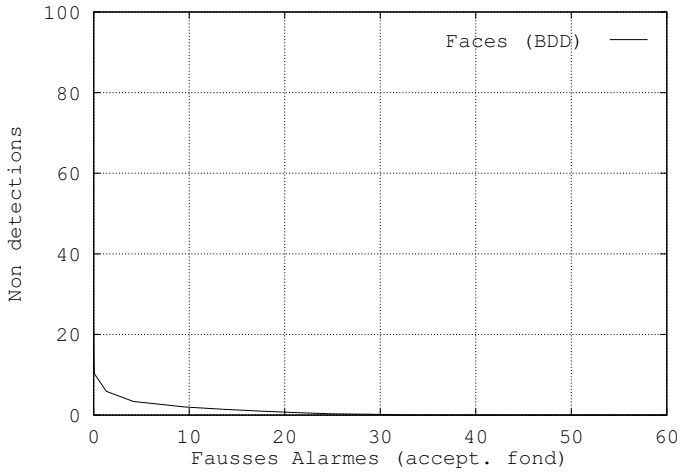
Résultats

Les résultats de détection obtenus par ce système sont plus mauvais que ceux du réseau de reconnaissance testé dans la section précédente. Ici aussi, les fonds sont bien séparés des visages connus. Les visages extraits des scènes sont par contre confondus avec les fonds.

On voit sur la courbe 8.7b que les erreurs de reconstruction se répartissent, pour les deux ensembles visages et l'ensemble *Random*, selon trois distribu-



(a) Répartition des valeurs de l'erreur de reconstruction.



(b) Courbes de détection.

FIGURE 8.7 – Erreurs de reconstruction (a) et courbes de détection obtenues en utilisant l'erreur de reconstruction d'un réseau auto-associatif (5 cellules cachées) entraîné sur les visages de *BDD*.

tions d'aspect gaussien et assez bien localisées. Par contre, les erreurs sur les imagettes de fonds se distribuent presque uniformément dans une gamme très large, qui recouvre les trois ensembles de visages. Dans l'approximation linéaire, la première couche du réseau diabolo effectue une projection sur le sous-espace des composantes principales. La courbe 8.7b indique que certaines imagettes de fonds se projettent sans perte significative d'information sur cet espace.

Les visages inconnus *BDI* sont ici moins rejetés que les visages (aussi inconnus d'ailleurs) extraits des scènes. Ce phénomène est dû à l'importante différence de qualité entre les images de B2 (*BDD* et *BDI*) et les images de visages extraits des scènes (voir figure 8.4). Nous reviendrons sur ce point par la suite.

Nous avons répété cette expérience en variant le nombre de cellules cachées du diabolo et les paramètres d'apprentissage, sans modification notable des résultats.

8.4.3 Conclusion

Les deux approches présentées dans cette section ont échoué. Ces deux approches sont fondées sur des principes différents : la première utilise un réseau de reconnaissance pour estimer la densité de probabilité des images de visages, en espèrant généraliser cette estimation aux visages inconnus. On a vu (courbe 8.5a) que ce n'est pas le cas : le classifieur rejette indifféremment toutes les images inconnues.

La seconde approche repose sur la construction d'un *encodeur*, supposé ne savoir coder que les images de visages. La courbe 8.7a montre que les nouveaux visages sont bien codés, mais qu'une proportion importante d'imagettes de fonds sont aussi bien reconstruites. Le codage en composantes principales n'est pas utilisable pour séparer les visages des fonds.

Il semble par conséquent difficile de se passer d'exemples de fonds pour la mise au point d'un classifieur objet/fond. Dans les sections suivantes, nous indiquons comment utiliser des exemples de fonds pendant l'apprentissage.

8.5 Approches "supervisées"

Nous allons ici construire un classifieur visages/fonds supervisé, c'est à dire entraîné en utilisant des exemples étiquetés de ces deux classes. Cette démarche pose immédiatement deux problèmes : (i) quels exemples de visages employer ? (ii) quels exemples de fonds ?

Le choix des exemples de visages est le plus simple : on dispose de deux ensembles d'images candidats, *BDD*, qui contient 4100 exemples d'images de bonne qualité, et *FaceSc_learn*, contenant 336 imagettes extraites des scènes, de qualité très médiocre. Utiliser *BDD* présente l'avantage du nombre d'exemples, avec l'inconvénient que ces images sont très différentes de celles utilisées en test (scènes).

Pour les exemples de fonds, le problème est très ouvert. On pourrait par exemple envisager l'emploi d'images synthétiques (formes géométriques, textures, bruits, ...) couvrant systématiquement une partie de l'ensemble de toute les imagettes possible (en fait $[0, 256]^{500}$). Nous manquons ici de critère théorique pour guider le choix.

L'ensemble d'exemples de fonds devrait contenir deux types d'exemples : des imagettes représentatives de tout ce que l'on pourra un jour rencontrer dans des scènes naturelles, et d'autre part une bonne proportion d'exemples «proches» des visages et susceptibles d'être confondus, de façon à obtenir une frontière de décision précise dans ces régions. Le problème est que l'on ne peut définir ce critère de proximité avant de disposer du classifieur...

Plutôt que de chercher à constituer un ensemble d'exemples de fonds en utilisant des heuristiques plus ou moins justifiées (voir par exemple [89]), nous avons décidé de commencer par utiliser notre ensemble de 6000 imagettes *BACK0*, représentatif des fonds rencontrés dans les scènes d'apprentissage (voir section 8.3).

8.5.1 Apprentissage sur les visages de *BDD*

Détaillons maintenant les expériences utilisant un classifieur entraîné sur les visages de *BDD* et les fonds de *BACK0*.

Pour l'apprentissage, on utilisera 2250 imagettes de visages (15 personnes), et 3000 (la première moitié) imagettes de fonds de *BACK0*.

Nous mesurerons toujours les performances sur les jeux d'imagettes de visages habituels, et réserverons la seconde moitié de *BACK0* pour mesurer les courbes de détection.

Réseau utilisé

Nous avons déjà dessiné un réseau TDNN pour l'identification d'imagettes de visages de taille 20x25. Nous rappellons l'architecture de ce réseau, nommé **faci20x25** en figure 8.8.

On peut considèrer ce TDNN comme un extracteur de caractéristiques, bien adapté au traitement des images de visages. Nous allons par conséquent

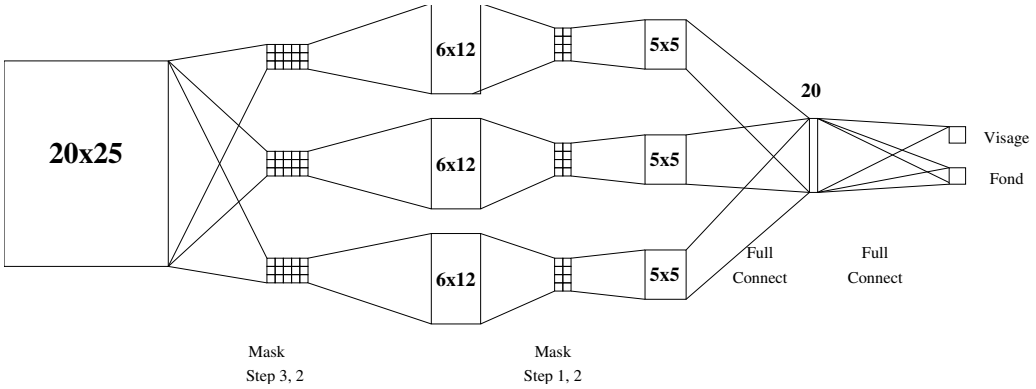


FIGURE 8.8 – Architecture du réseau TDNN "faci20x25".

employer ce même réseau, muni d'une couche de sortie de deux cellules, que l'on associera respectivement à la classe «visages» et la classe «fonds».

Première expérience

Pour cette expérience, nous avons mené l'apprentissage du réseau TDNN normalement (avec la fonction de coût habituelle).

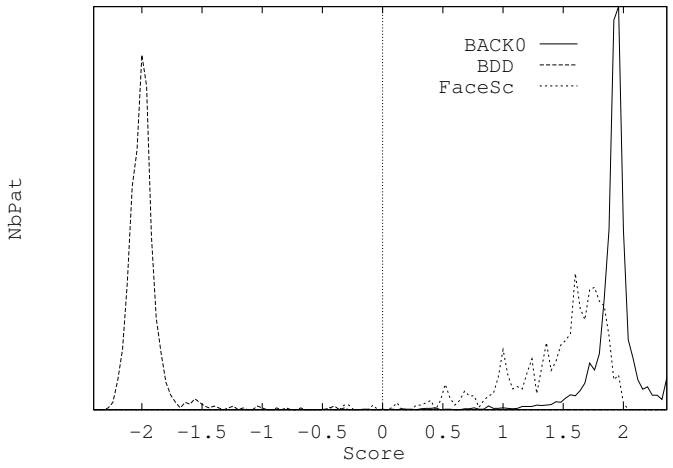
Les performances de classification atteintes sont excellentes : il n'est pas difficile de séparer les imagettes de visages «propres» des 3000 imagettes de *BACK0*.

Après apprentissage, on calcule la différence des deux cellules de sorties, et l'on utilise cette valeur pour déterminer la classe (fond ou visage) des imagettes présentées. Les résultats sont présentés figure 8.9.

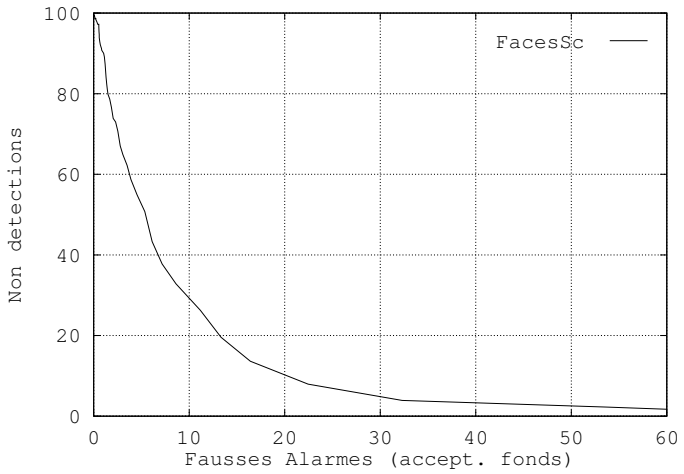
On voit que les performances de détection obtenues ici sont très supérieures à celles présentées dans la section précédente. Cette voie est donc prometteuse. Toutefois, les taux restent éloignés des valeurs nécessaires à la construction d'un dispositif de segmentation utilisable : si l'on accepte 10 % de fausses alarmes *par imagette* (voir courbe 8.9b), on rate plus de 30 % des visages. Si l'on estime à 100 000 le nombre d'imagettes de fonds à classer pour balayer une image de scène, un tel taux se traduit par environ 10000 fausses alarmes par scène, tout en ratant un visage sur 10 ! Il nous faut par conséquent améliorer sérieusement les performances de détection.

Comparaison de différents pré-traitements

Dans la section précédente, nous n'utilisions aucun pré-traitement pour les images de scènes : les imagettes de fonds, comme celles de visages extraits des scènes, étaient juste découpées dans l'image source, sans modification des



(a) Répartition des valeurs de sortie $S_0 - S_1$.



(b) Courbes de détection.

FIGURE 8.9 – Résultats obtenus par le classifieur **faci20x25**. Le critère de classement ici employé est la différence des deux sorties du TDNN.

valeurs des pixels.

La courbe 8.9b montre une importante différence de performances entre les visages de *BDI* et ceux de *FaceSc*. Ceci s'explique en partie par la différence de qualité entre les visages de B2 et ceux de scènes, que nous avons déjà évoquée. Pour tenter d'atténuer cette différence, nous allons tester plusieurs *pré-traitements* appliqués à toutes les imagettes (pendant l'apprentissage et le test).

Normalisation des valeurs Ce traitement est assez violent. Il consiste à appliquer aux pixels des imagettes une transformation linéaire choisie pour ramener le pixel maximum de l'imagette à la valeur +1 (blanc), et le pixel minimum à −1 (noir). Pour les visages de bonne qualité, cette opération est peu sensible : le point le plus foncé de l'image est proche du noir (cheveux ou œil le plus souvent), et de même pour le plus clair. Le contraste des visages de qualité moyenne s'en trouve amélioré.

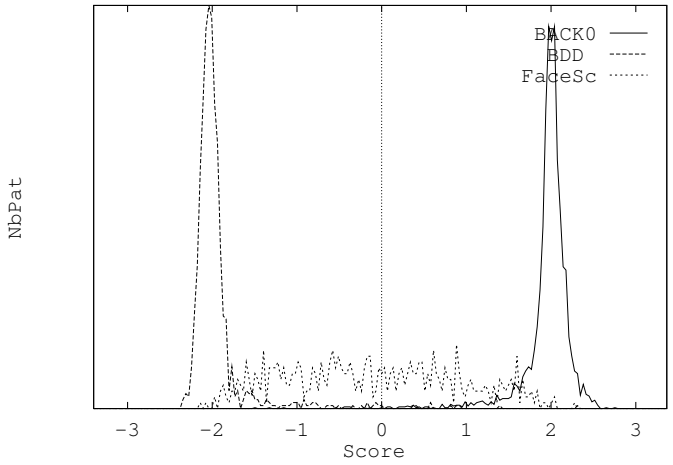
L'opération est par contre très sensible pour les images uniformes (comme de nombreux fonds) : une imagette grise, dont les valeurs varient dans $[-0.1, 0.1]$ va trouver ses valeurs dilatées dans $[-1, 1]$. On a ainsi une forte amplification du bruit. Au total, l'effet de la normalisation est très négatif.

Gradient L'idée de travailler sur une image gradient est séduisante : on s'affranchit de cette façon des variations de luminosité globales de l'image, pour se concentrer sur les changements locaux.

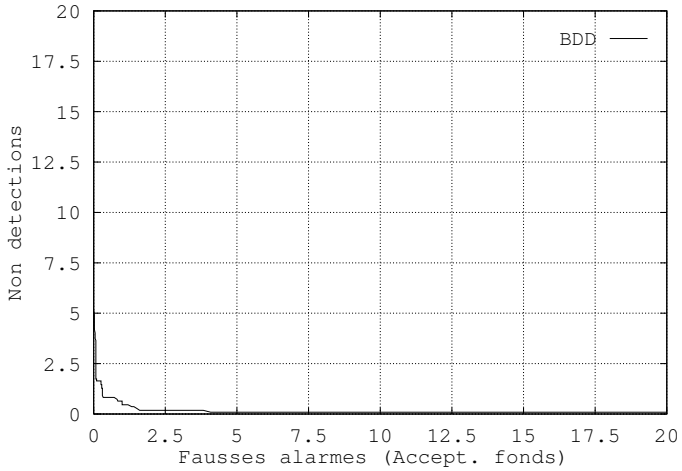
Nous avons testé cette approche en utilisant la norme du gradient à l'entrée du classifieur. Ce prétraitement entraine une perte d'information : on n'est plus capable de distinguer les transitions noir-blanc (de gauche à droite par exemple) des transitions blanc-noir.

Les résultats de ces expériences, que nous ne détaillons pas ici, se sont révélés décevants : les performances de classification ont certes augmenté par rapport à l'expérience préliminaire sans pré-traitements (on a réduit en moyenne de 10 % le taux de non-détection), mais restent faibles. Nous attribuons ceci à deux facteurs : (i) la perte d'information évoquée plus haut (ii) le réseau TDNN employé est normalement capable d'extraire des caractéristiques de type "gradient" dans ses premières couches (masques de poids partagés). Le pré-traitement n'apporte ainsi pas d'information nouvelle.

Modification de l'histogramme Lorsqu'on visualise une image, on emploit souvent une *palette logarithmique* (ou LUT pour *LookUp Table*). L'action d'une telle palette est d'appliquer aux valeurs u des pixels de l'image à



(a) Répartition des valeurs de sortie $S_0 - S_1$.



(b) Courbes de détection.

FIGURE 8.10 – Résultats obtenus par le classifieur **faci20x25** et le prétraitement «logarithmique». Le critère de classement ici employé est la différence des deux sorties du TDNN.

afficher une tranformation du type :

$$f(u) = \log(1 + u), \qquad u \geq 0$$

Cette transformation augmente le contraste dans les parties sombres de l'image. Les visages des scènes étant souvent sombres et mal contrastés, nous avons décider d'appliquer ce pré-traitement simple aux imagettes.

Les résultats sont donnés figure 8.10. Etonnament, ce pré-traitement permet de diviser le nombre de non-détection par dix, à taux de fausses alarmes constant.

Dans toute la suite du chapitre, nous conserverons ce pré-traitement.

Simplification du classifieur

Jusqu'a présent, nous avons utilisé comme classifieur le TDNN **faci20x25**. Avec deux cellules de sortie, ce réseau comporte 5693 connexions, et 1709 poids.

Nous allons maintenant tenter de réduire la taille de ce réseau, avec deux objectifs : (i) améliorer, si faire ce peut, les performances de détection ; (ii) réduire le nombre de multiplications nécessaires à la classification d'une imagette, afin d'arriver à un système utilisable pour balayer les images de scènes.

Après quelques essais, nous sommes arrivé à l'architecture TDNN **quickseg**, représenté sur la figure 8.11.

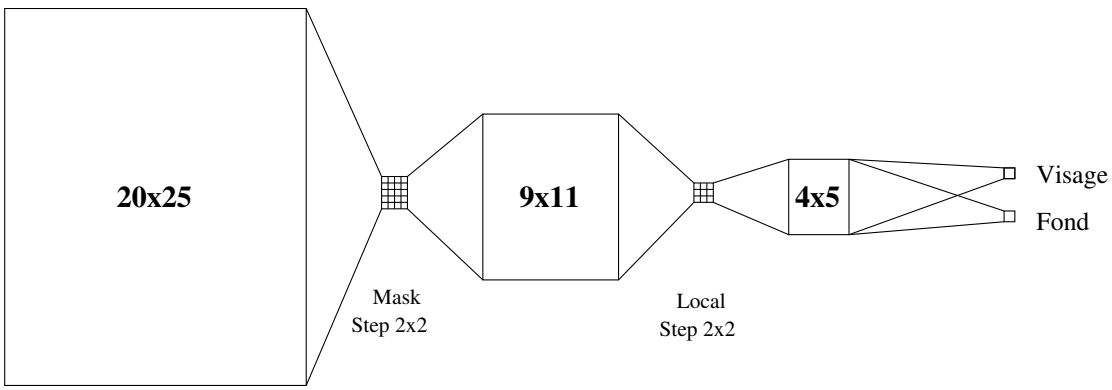


FIGURE 8.11 – Architecture du réseau **quickseg**. La rétine est connectée à la première couche par un seul masque de poids partagés 4x5 se décalant de 2 en 2. La première couche cachée a une taille de 9x11. Elle est reliée à une deuxième couche, de taille 4x5, par des connexions locales 3x3. Les deux cellules de sortie, associée chacune à une classe, sont connectées aux 20 cellules de la dernière couche cachée.

Le réseau **quickseg** comporte 2321 connexions, et seulement 263 paramètres libres.

Les résultats obtenus en utilisant **quickseg** (avec le pré-traitement «logarithmique») sont donnés en figure 8.12. On voit que les performances sont équivalentes à celles du réseau **faci20x25**. La réduction de capacité du classifieur a permis de conserver les propriétés de généralisation.

Simplification extrême du classifieur

On peut chercher à simplifier encore d'avantage le classifieur.

Un simplification intéressante consiste à supprimer la première couche du réseau **quickseg** : on remplace cette couche par un simple ré-échantillonnage de l'image, pour réduire les imagettes à la taille 9x11 pixels. Cette simplification permet un gain très important de vitesse. En effet, non seulement le classifieur est plus rapide, mais la rétine du réseau est plus petite. Or, comme on le verra plus loin, le nombre d'appels au classifieur pour analyser une scène est proportionnel au côté de la rétine utilisée pour le balayage.

L'architecture du MLP (on ne peut plus parler de TDNN) correspondant est représentée sur la figure 8.13.

Malheureusement, cette simplification ne va pas sans dégrader les performances. Les résultats obtenus par ce réseau sont donnés figure 8.14.

Clairement, les imagettes de taille 9x11 sont insuffisantes pour distinguer sans erreur les visages des fonds. A cette résolution, les détails tels que les yeux sont rarement distinguables. Seule la forme générale est utilisable.

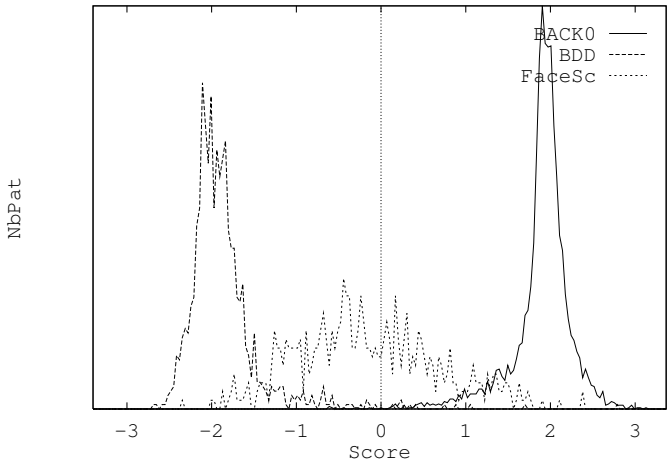
Cependant, un tel classifieur peut se révéler utile, pour réaliser une première analyse de la scène, dans le cadre d'une approche «*coarse to fine*». Nous reviendrons sur l'intérêt de cette approche dans la section 8.7.

8.5.2 Apprentissage sur les visages des scènes

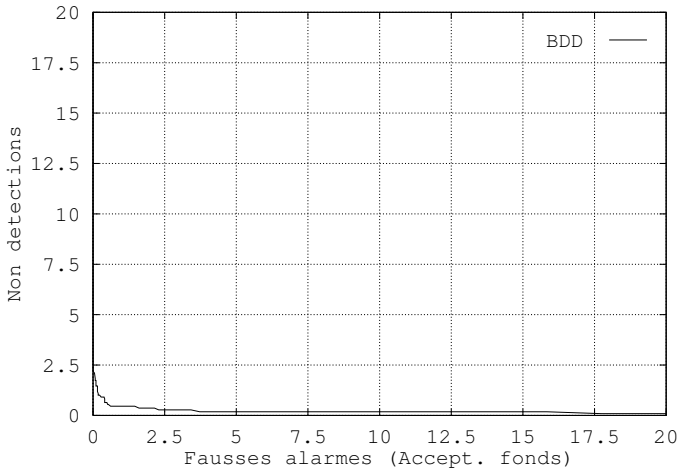
Une autre voie à explorer pour l'apprentissage du classifieur est d'utiliser comme exemples les visages des scènes d'apprentissage, et non les visages de la base *BDD*. Ces images de visages sont en effet assez différentes pour plusieurs raisons : capteur utilisé, résolution (taille du visage dans l'image initiale), type d'éclairage...

Le problème est ici que nous ne disposons que d'environ 300 exemples de visages.

Plusieurs auteurs ont proposé de modifier l'algorithme d'apprentissage stochastique dans les situations ou' les effectifs des classes à apprendre sont disproportionnés. Par exemple J. Vincent [210] propose une technique qu'il appelle *selective training*. La recette consiste à ne rétropropager l'erreur que



(a) Répartition des valeurs de sortie $S_0 - S_1$.



(b) Courbes de détection.

FIGURE 8.12 – Résultats obtenus par le classifieur **quickseg**.

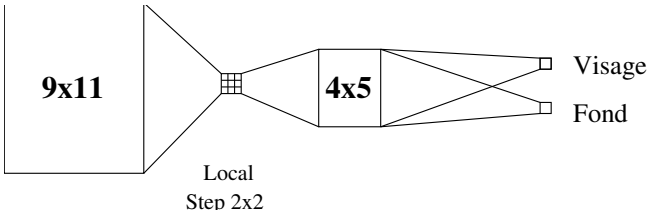


FIGURE 8.13 – Architecture du réseau **quickseg9x11**. Il s'agit du réseau **quickseg** auquel on a enlevé la première couche. Ce réseau possède 242 connexions (et ici par conséquent 242 paramètres libres).

lorsqu'elle dépasse un certain seuil, et à modifier l'ordre de présentation des exemples de façon à éviter que le réseau «oublie» la classe la moins représentée.

Une autre solution, plus simple à mettre en œuvre, consiste à modifier la fonction de coût. Le coût instantané habituel, pour une forme de la classe ω_i, s'exprime comme :

$$E_i = \frac{1}{2}\|o - t\|^2$$

o étant la sortie du réseau, et t le vecteur d'état désiré. Si la fréquence des formes de la classe ω_i est $f(\omega_i)$, on peut définir le nouveau coût :

$$E'_i = \frac{1}{2}\frac{\|o - t\|^2}{f(\omega_i)}$$

Ce coût égalise les contributions globales des deux classes au signal d'erreur.

Apprentissage standard

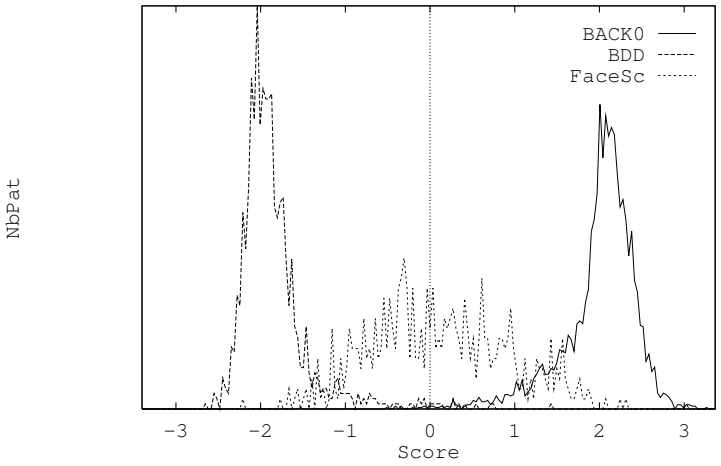
Nous avons commencé par entraîner le réseau **quickseg9x11**, à l'aide de 250 exemples de visages (ensemble *FaceSc_learn*), et des 3000 imagettes de fonds (*BACK0_learn*). Nous avons gardé la fonction de coût habituelle.

La figure 8.15 indique les résultats de cet apprentissage. Ce système donne des courbes de détection très proches pour les visages de *BDD* et ceux de *FaceSc_test*.

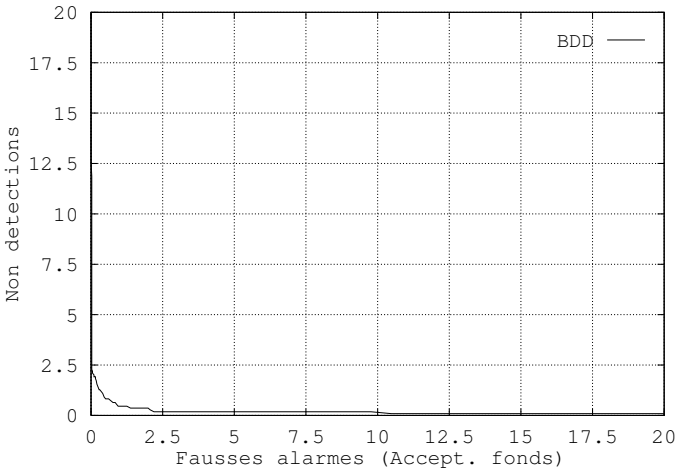
Les performances sont meilleures que pour les systèmes utilisant uniquement *BDD* présentés dans la section précédente.

Modification du coût

Nous reprenons ici le même système, en modifiant la fonction de coût optimisée durant l'apprentissage, comme indiqué plus haut.

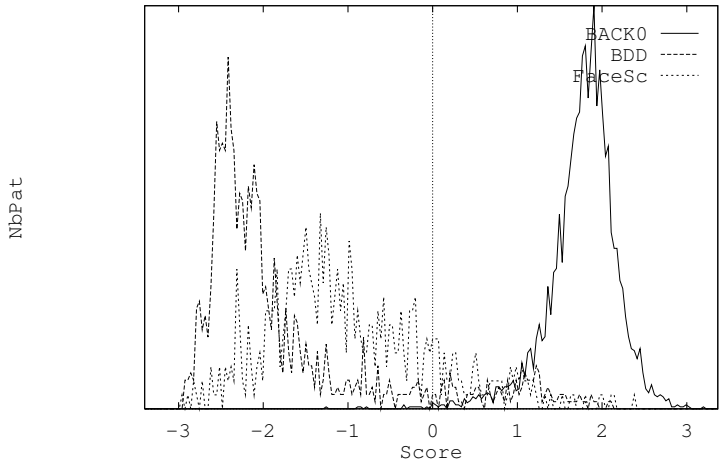


(a) Répartition des valeurs de sortie $S_0 - S_1$.

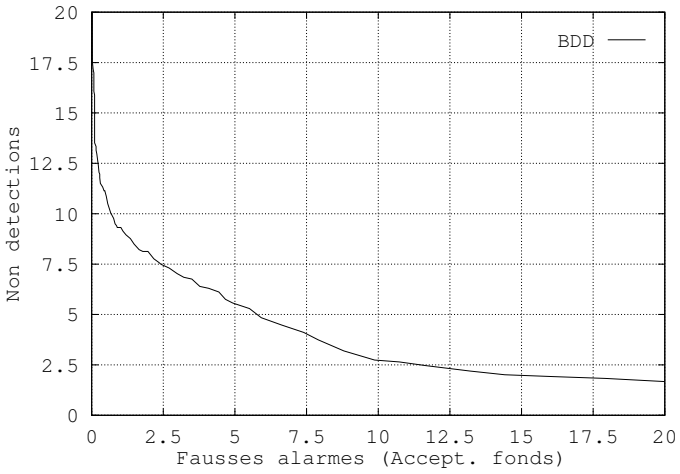


(b) Courbes de détection.

FIGURE 8.14 – Résultats obtenus par le classifieur **quickseg9x11**.

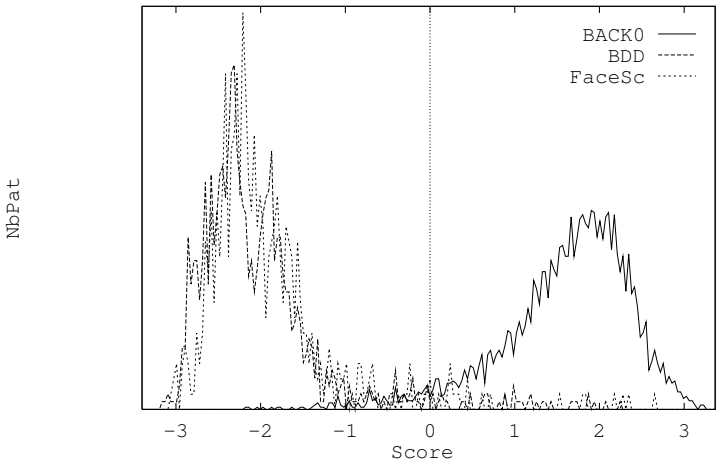


(a) Répartition des valeurs de sortie $S_0 - S_1$.

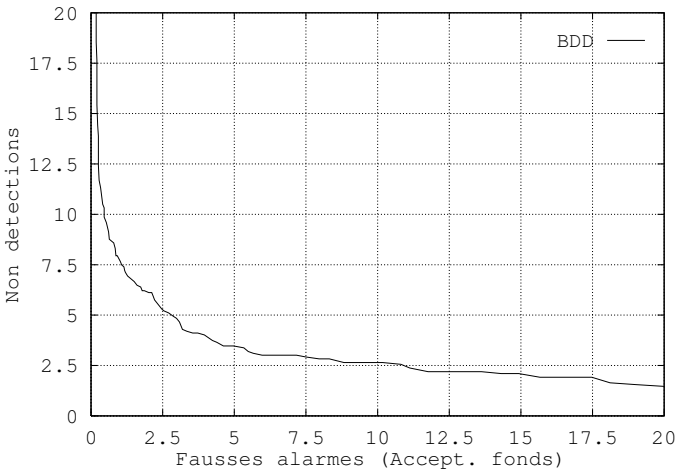


(b) Courbes de détection.

FIGURE 8.15 – Résultats obtenus par le classifieur **quickseg9x11**, entraîné sur les 250 visages de *FaceSc_learn*. (Fonction de coût standard).



(a) Répartition des valeurs de sortie $S_0 - S_1$.



(b) Courbes de détection.

FIGURE 8.16 – Résultats obtenus par le classifieur **quickseg9x11**, entraîné sur les 250 visages de *FaceSc_learn*, avec modification de la fonction de coût.

Les résultats (figure 8.16) indiquent que la modification du coût a permis d'améliorer sensiblement l'apprentissage : le taux de non-détections est divisé par deux.

Sur la courbe donnant la répartition des sorties du classifieur (8.16a), on remarque que ce classifieur est moins «net» : les valeurs se répartissent sur des plages assez larges, pour tous les groupes d'imagettes, mais les visages sont bien séparés des fonds. Les visages de *BDD* et ceux de *FaceSc* se répartissent strictement sur le même intervalle, ce qui est très bon signe concernant les propriétés de généralisation du classifieur.

Réseau 20x25

Avant de terminer cette section, nous donnons les performances du réseau **quickseg** (rétine 20x25), entraîné sur les visages de *FaceSc_learn* avec une fonction de coût modifiée.

Les courbes 8.17 indiquent que ce classifieur est le meilleur que nous ayons développé : on observe environ un taux de non détection de 2.5 %, pour un taux de fausses alarmes égal lui aussi à 2.5 %.

8.5.3 Tolérance aux translations/dilatations

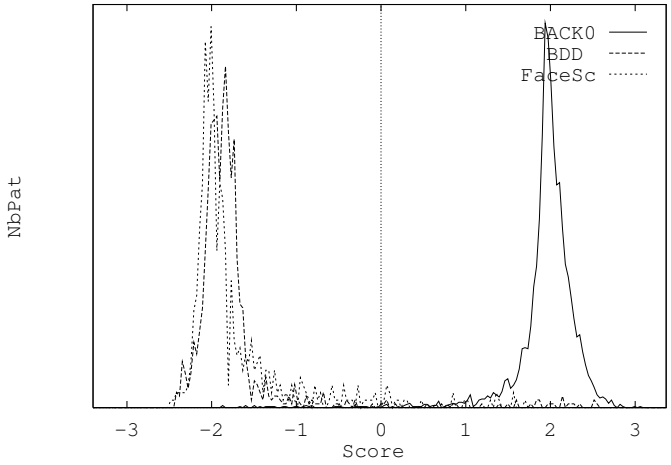
Les propriétés de tolérance du classifieur vis à vis des petites translations et dilatations de l'image sont fondamentales pour notre application : elles vont permettre de définir l'algorithme de balayage des images de scènes. Par exemple, si les visages sont détectés de la même façon après une translation de cinq pixels, on va pouvoir se contenter d'extraire une imagette tout les cinq pixels.

Nous avons mesuré les performance du classifieur **quickseg**, entraîné sur les visages centrés de *BDD* (comme en section 8.5.1). (Le comportement décrit ici est sensiblement identique pour tous les classifieurs 20x25, à l'exception des réseaux à connexions totales (diabolo) que nous n'avons pas testés ici).

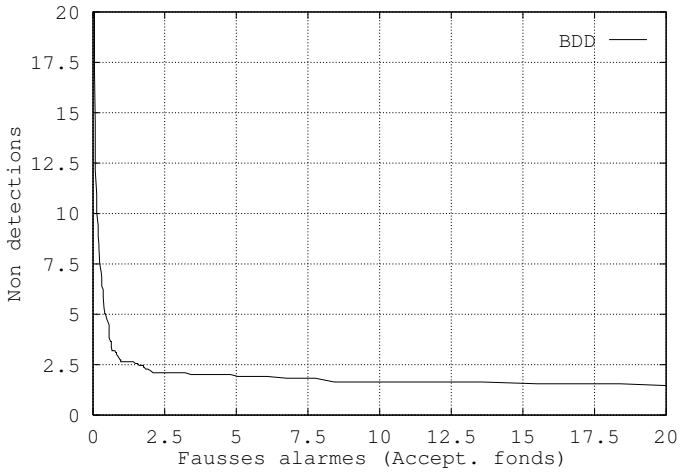
Pour le test, nous utilisons d'une part les imagettes de fonds de *BACK0*, d'autre part les visages de *FaceSc_test* ayant subies diverses transformations :

1. décalage latéral de 1 à 9 pixels de la rétine (20x25) ;
2. décalage vertical de 1 à 9 pixels ;
3. dilatation de la rétine de -40 % à +40 %.

Les performances relevées sont résumées dans la figure 8.18. Elles indiquent que le classifieur tolère des décalages de ±3 pixels, horizontalement

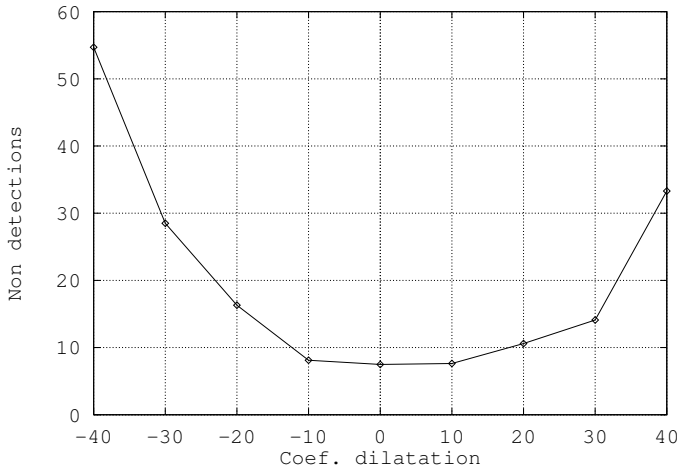


(a) Répartition des valeurs de sortie $S_0 - S_1$.

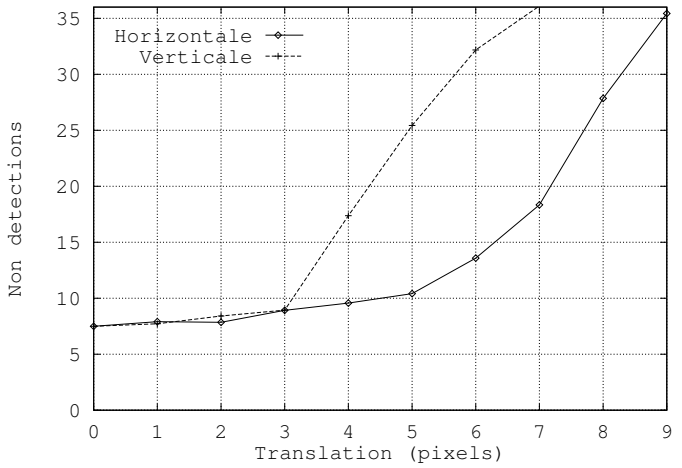


(b) Courbes de détection.

FIGURE 8.17 – Résultats obtenus par le classifieur **quickseg** (rétine 20x25), entraîné sur les 250 visages de *FaceSc_learn*, avec modification de la fonction de coût.



(a) Dilatations/réductions de la rétine.



(b) Translations de la rétine.

FIGURE 8.18 – Taux de non détection des visages de *FaceSc_test*, en fixant le taux de fausses alarmes à 5 % (sur *BACK0*). Réseau **quickseg**, entraîné sur *BDD*. Sur la courbe (a), on fait varier la taille du visage dans la rétine 20x25. Sur la courbe (b), on translate horizontalement et verticalement le visage (on a ici moyenné les résultats des décalages vers le haut et vers le bas).

et verticalement, sans grande dégradation des performances. Ce comportement était attendu, car il correspond approximativement à la précision de l'étiquetage (segmentation manuelle ou automatique, voir chapitre 7) des visages. Notons aussi qu'un œil a en moyenne 3 pixels de large dans une rétine 20x25.

La variation des performances lorsque l'on change la taille du visage dans la rétine (courbe 8.18b) est plus sensible. On voit que la diminution de la taille de la rétine entraîne une chute assez rapide des performances : le contour du visage sort de la rétine.

Impact sur l'analyse multirésolution

Les résultats précédents conditionne la conception de l'analyse de la scène.

La tolérance aux translations indiquée autorise à n'utiliser qu'une position sur deux (dans chaque direction) lors du balayage des images issues de la décomposition multirésolution.

Le choix du *facteur de résolution* de l'analyse est guidé par la courbe 8.18b. On voit que les performances de détection sont acceptables pour des tailles de rétine w comprises entre 0.8 et 1.25 fois la taille du visage.

Si l'on utilise une succession de rétines de tailles [4]

$$w_j = w_0 \, \alpha^j,$$

$j \geq 0, \alpha > 1.0$, pour détecter un visage de largeur w_f, on le détectera sûrement dans au moins l'une des rétines si :

$$\exists j \ / \ \ 1.2 \, w_j \geq w_f \ \ \text{ou} \ \ 0.8 \, w_{j+1} \leq w_f$$

On en déduit immédiatement $\alpha \leq 1.2/0.8 = 1.5$.

Un facteur de résolution $\alpha = \sqrt{2}$ est ainsi satisfaisant.

8.6 Premier système de localisation de visages

On a vu que les performances de détection des classifieurs obtenus plus haut ne laissaient a-priori guère d'espoir de bâtir un système de localisation utilisable : le meilleur système (section 8.5.2) rate environ un visage sur dix à 1 % de fausses alarmes. Si l'on extrapole rapidement ces taux à une scène, on arrive à 1000 fausses alarmes par image de scène.

Deux approches sont possibles pour réduire ce nombre :

4. En réalité, on gardera la rétine de taille fixe et fera varier la résolution de l'image, ce qui revient au même.

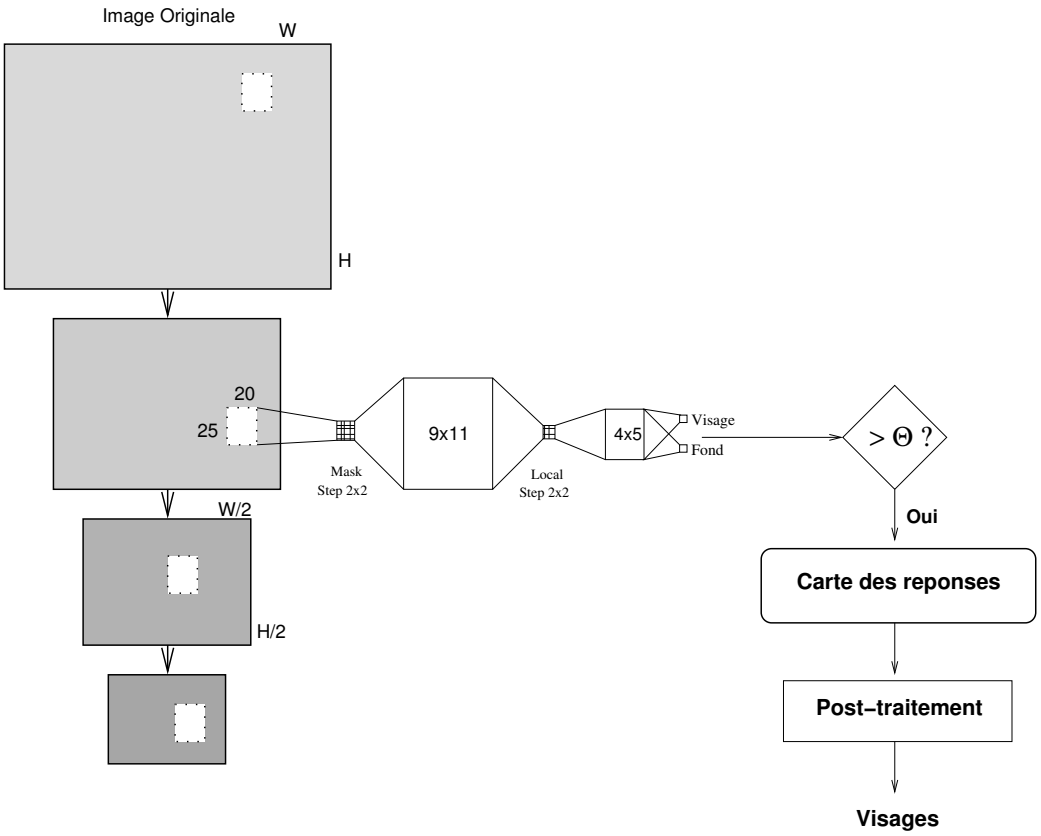


FIGURE 8.19 – Architecture schématique du système de localisation de visages à un étage.

1. Construire un classifieur plus performant, un utilisant un deuxième étage : c'est l'approche *coarse to fine* que l'on abordera dans la section 8.7 ;
2. Mettre au point un post-traitement utilisant les corrélations entre les alarmes pour en éliminer la plus grande partie.

Nous allons maintenant décrire rapidement cette deuxième approche, qui à fait l'objet de deux publications dans des conférences internationales [207, 208]. On le qualifie de «système à un étage» car il n'utilise qu'un niveau de réseau de neurones.

8.6.1 Architecture générale

L'architecture globale est schématisée sur la figure 8.19.

On utilise ici le classifieur *quickseg* (voir section 8.5.2), doté d'une rétine 20x25.

8.6.2 Analyse Multirésolution

La première étape du traitement consiste à décomposer l'image pour obtenir une suite d'images de tailles décroissantes. Pour cela, on utilise une *analyse multirésolution par ondelettes avec un facteur* $\sqrt{2}$.

Le principe de cette analyse est détaillé dans le chapitre 4. Nous utilisons ici uniquement le canal «passe-bas», qui donne une série de vues de la même image à différentes échelles. Ce traitement peut se voir comme une simple interpolation des valeurs de l'image suivie d'un ré-échantillonnage.

On a vu dans la section 8.5.3 qu'un facteur de résolution de $\sqrt{2}$ était suffisant pour espérer détecter chaque visage dans au moins un niveau de l'analyse.

Le nombre de niveaux (appelé *profondeur* de l'analyse) dépend de la taille de l'image initiale et de la gamme de tailles de visages (i.e. distances) que l'on souhaite pouvoir détecter.

Un rapide examen de la base de scènes nous a montré que les visages à détecter se répartissaient entre 32 pixels de largeur et 170 pixels, soit environ un facteur 5 entre le plus grand et le plus petit. Afin de détecter ces visages, nous avons choisi la plus petite taille de rétine égale à 32. La rétine va prendre les largeurs apparentes (i.e. ramenées à l'image initiale) suivantes :

$$R_{w_i} = 32 \times 2^{i/2} \qquad \rightarrow \{32, 45, 64, 90, 128, 181\}$$

Les images de scène ont une taille égale à :

$$W^0 = 764,\ H^0 = 580 \ \text{ pixels.}$$

La largeur des images décomposées est maintenant donnée par

$$W_i = W^0 \frac{20}{R_{w_i}}$$

ou‘ 20 est la largeur de la rétine du classifieur.

On remarque ici que la taille des images à analyser est proportionnelle à la taille de la rétine du réseau. Lorque l'on passe du réseau 20x25 au réseau 9x11, on gagne environ un facteur 4 sur le volume de calcul (indépendamment de la complexité du classifieur).

Nous travaillerons sur des images de tailles :

$$(W_i, H_i) \in \{(478, 362), (340, 258), (238, 181), (170, 128), (119, 90), (84, 64)\}$$

Soit un total d'environ 340 000 pixels.

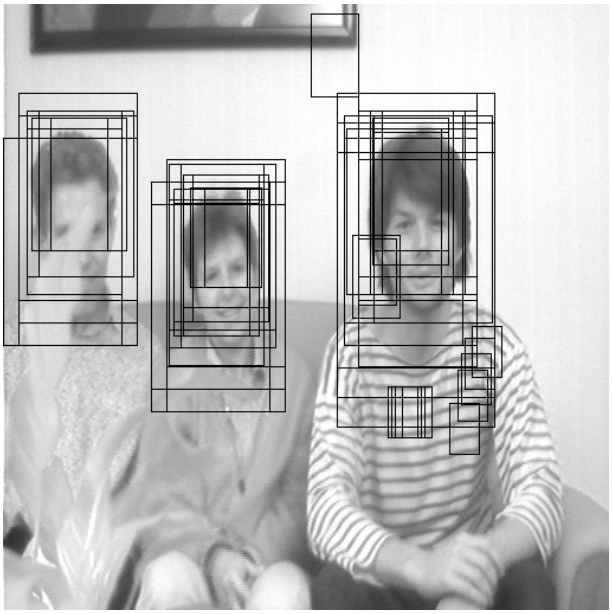


FIGURE 8.20 – Scène typique : on a tracé tous les rectangles détectés après balayage de la scène, avant sélection et regroupement par l'algorithme de post-traitement.

8.6.3 Balayage des images

Après l'analyse multirésolution, on balaye chaque image obtenue avec la rétine du réseau de neurones : on estime ainsi en chaque position (rectangle) la probabilité d'avoir affaire à un visage centré dans la rétine.

Les résultats de la section 8.5.3 permettent de n'explorer qu'une position sur 4 ($dx = dy = 2$), ce qui réduit le nombre d'appels au classifieur à environ 85 000 pour une scène.

On ajuste un seuil θ sur la différence des deux cellules de sortie du classifieur, de façon à se placer à 10 % de non détections sur la courbe 8.17b, soit 0.3 % d'acceptation des imagettes de fonds.

Lors du balayage, on récolte les positions ou‘ la réponse du classifieur dépasse le seuil θ, et l'on dresse ainsi une *carte des réponses*.

Après cette opération, la carte de réponse contient en moyenne 280 *alarmes*, dont 250 ne correspondent pas à un visage.

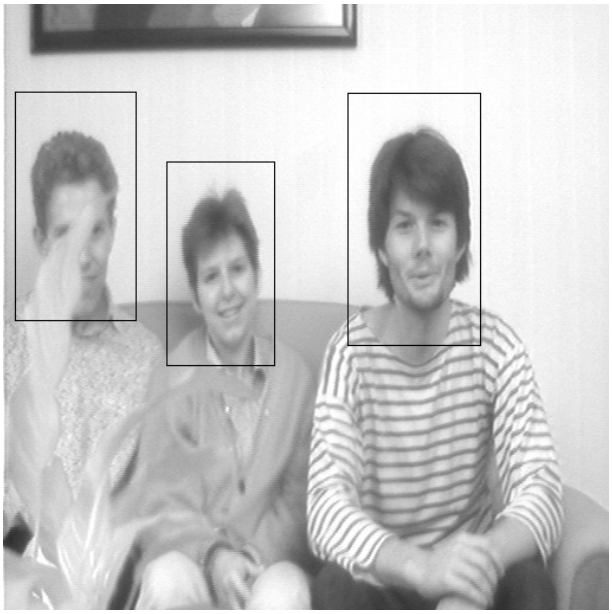


FIGURE 8.21 – Rectangles calculés par le post-traitement.

8.6.4 Post-traitement

La remarque suivante permet d'éliminer une partie importante des fausses alarmes. On voit sur la figure 8.20 que chaque visage est entouré par un nuage important de rectangles détectés à différentes échelles, tandis que de nombreuses fausses alarmes sont isolées. Cette redondance de la détection de visages s'explique par le fait que la tolérance du classifieur aux petites translations et dilatations a été volontairement sous-estimée dans la conception du détecteur.

Nous avons développé un algorithme de post-traitement tirant parti de cette remarque. Le fonctionnement de l'algorithme est grossièrement le suivant :

1. agglomération des alarmes selon un critère de distance géométrique ;
2. élimination des groupes comportant peu de rectangles, et des groupes dans lesquels on ne trouve pas une série de rectangles contigus (en espace ou en échelle) ;
3. calcul d'un rectangle englobant chaque groupe restant.

Le bon fonctionnement de ce type d'algorithme dépend du réglage (manuel) d'un bon nombre de seuils (tailles minimales, etc) qui en rend la mise au point pénible (et les performances un peu aléatoires, car on suppose certaines conditions réunies...).

Néanmoins, on arrive ainsi à réduire le nombre de fausses alarmes à un niveau acceptable. La figure 8.21 montre le résultat du post-traitement sur une scène typique.

8.6.5 Performances

Les performances finales de ce système devraient être mesurées sur l'ensemble des scènes de test (1000 images).

Le temps de calcul à mettre en œuvre a rendu cette mesure impossible : il faut environ 30 minutes de calcul par image sur une petite station de travail Sun (Sparcstation 1). Aussi avons-nous évalué ce système sur un sous-ensemble des scènes de test, comportant 100 images.

On arrive aux taux suivants :

Non détection :	16 %
Fausses alarmes :	2.4 par image

Le taux de non détections est plus fort que prévu, en partie parce que le post-traitement élimine malencontreusement certains visages.

8.7 Système à deux étages

La section précédente a montré que les performances obtenues par un système utilisant un seul classifieur étaient insuffisantes, tant par les taux de détection atteints que par les volumes de calculs générés.

A l'examen des résultats, on s'aperçoit que nombre de fausses alarmes arrivent sur des type d'imagettes peu ou non représentés dans l'ensemble d'apprentissage des fonds. On revient alors au problème de la sélection des exemples de fonds, évoqué dans les sections 8.5 et 8.3.

Un début de solution serait de collecter ces erreurs lors du balayage des scènes d'apprentissage, puis de les ré-injecter dans l'ensemble d'apprentissage *BACK0* et de reprendre toute la procédure. On observe effectivement ainsi une amélioration des performances globales. Cependant, de nouvelles fausses alarmes peuvent apparaître. De plus, cette idée ne permet pas de réduire les temps de calcul.

Une solution plus efficace consiste à utiliser deux classifieurs en série, pour l'apprentissage et pour l'analyse des scènes.

Le premier de ces classifieurs va recevoir toutes les imagettes lors du balayage ; il doit être de petite taille pour deux raisons :

1. le but est de réaliser une première classification grossière des imagettes, basée sur leur forme générale : la capacité du classifieur doit être faible ;

2. le nombre d'appels à ce classifieur va être très élevé (comme dans le premier système proposé en section 8.6) : il doit calculer rapidement, et nécessiter peu de multiplications.

Nous utiliserons le réseau **quickseg9x11** pour cette tâche. Nous avons vu (voir section 8.5.1) que ce réseau est très rapide : 242 multiplications seulement pour calculer l'état de la sortie. Les performances de ce réseau sont certes moins bonnes que celles obtenues en utilisant une rétine 20x25 (voir figure 8.16). Mais nous compenserons cette faiblesse par l'emploi d'un second classifieur.

8.7.1 Intérêt d'utiliser deux réseaux

Pour un problème de détection, l'emploi de plusieurs classifieurs en cascade est naturelle. Le premier étage de classification est utilisé pour dégrossir le problème, c'est à dire rejeter le minimum de visages, quitte à accepter un grand nombre de fonds. Le second niveau reçoit ainsi un flux d'imagettes dans lequel la proportion de visages est plus forte que dans l'image initiale.

La façon la plus simple d'exploiter cette idée est d'utiliser un petit réseau pour le premier étage, et un classifieur plus complexe pour le second. On obtient ainsi un gain élevé sur les temps de calcul, sans diminuation importante des performances.

Mais l'intéret essentiel de l'approche en plusieurs étages est la possibilité de mener l'apprentissage du second étage en utilisant uniquement les données sélectionnées par le premier. On règle ainsi élégamment le problème de la sélection des exemples de fonds. Chaque étage de classification est alors parfaitement adapté à la distribution d'imagettes à laquelle il a affaire. Un exemple de premier étage très simple serait un filtre rejetant toutes les imagettes constantes (uniformément grises), qui encombrent la base *BACK0*. L'apprentissage d'un second niveau en serait facilité. Bien entendu, pour ne pas compliquer la mise en œuvre de cette approche, on a intérêt à ne pas multiplier inutilement le nombre d'étages de classification. Nous nous sommes ici limités à deux étages, qui sont tous deux des réseaux TDNN testés plus haut dans ce chapitre.

8.7.2 Constitution de la base «BACK1»

La base *BACK1* sera utilisé pour l'apprentissage du deuxième étage de classification.

On commence par constituer un ensemble d'imagettes découpées à des positions aléatoires (mais hors des visages) dans les scènes d'apprentissage :

c'est la base *BACK0* (voir section 8.3). Cet ensemble d'imagettes est utilisé pour l'apprentissage du premier étage, qui apprend ainsi à distinguer grossièrement les visages du reste (réseau **quickseg9x11**).

Une fois ce classifieur au point (voir figure 8.16), on l'utilise pour balayer les scènes d'apprentissage et classer toutes les imagettes de ces scènes. On se fixe un seuil de détection Θ et l'on garde toutes les imagettes détectées au dessus de ce seuil qui ne correspondent pas à un visage (voir la figure 8.22). Lors des expériences décrites ici, nous avons ajusté le seuil Θ de façon à récolter un ensemble de 10 000 imagettes lors du balayage des scènes d'apprentissage [5].

La base *BACK1* ainsi obtenue est bien plus riche que *BACK0* en situations ambiguës. La figure 8.23 montre quelques imagettes représentatives de *BACK1*.

8.7.3 Apprentissage et résultat sur «BACK1»

Le deuxième étage de classification est entraîné sur les exemples de fonds de *BACK1* d'une part (5000 imagettes), et d'autre part sur les visages extraits des scènes *FaceSc*.

Cet étage utilise une résolution 20x25, afin de resoudre la majorité des situations ambiguës à la résolution 9x11 utilisée par le premier classifieur. Pour cette tâche, on peut utiliser le réseau **faci20x25** ou le réseau **quickseg**, aux performances équivalentes (voir 8.5.1). Nous avons choisit **quickseg**, qui possède trois fois moins de connexions.

Lors de l'apprentissage, nous avons gardé la fonction de coût modifiée (il y a ici 50 fois plus d'exemples de fonds que de visages) et le même prétraitement que dans les sections précédentes.

La figure 8.24 donne les résultats de cet apprentissage. Le taux de fausses alarmes est ici bien entendu mesuré sur les imagettes de *BACK1* (ensemble de test, 5000 imagettes), et non plus *BACK0*.

La courbe de détection (8.24b) indique que si l'on rejette 2.5 % des visages, le taux de fausses alarmes est de l'ordre de 0.05 %.

Sur la courbe 8.24a, on voit que les ensembles d'imagettes sont bien séparés. Contrairement à ce qui était observé au niveau du premier étage, les imagettes de bruits blancs se répartissent maintenant sur une plage de valeurs bien plus large que celles de *BACK1*. Ceci indique sans doute une plus grande «structure» dans la base *BACK1*, dont les imagettes, après la sélection du premier étage, ne s'apparentent plus à un bruit blanc.

5. Lors de ce balayage, on a aussi veillé à ne pas récolter systématiquement les alarmes adjacentes, afin d'obtenir la plus grande variété de fonds possible dans la base *BACK1*.

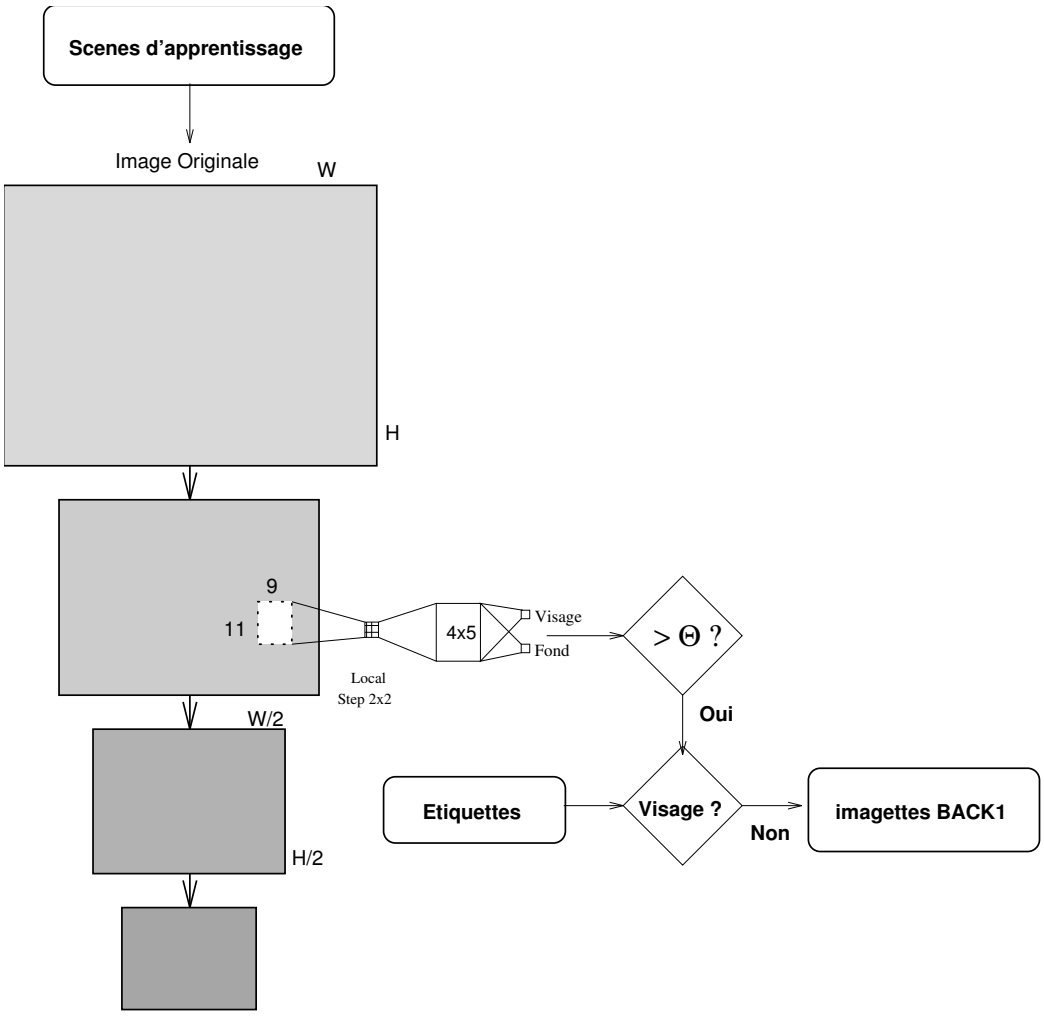


FIGURE 8.22 – Constitution de la base de données *BACK1*. Le réseau **quick-seg9x11** balaye les différents niveau de décomposition de l'image. On compare les positions des imagettes détectées avec les étiquettes donnant les coordonnées des visages dans la scène et l'on conserve dans *BACK1* les imagettes qui correspondent à des fausses alarmes.

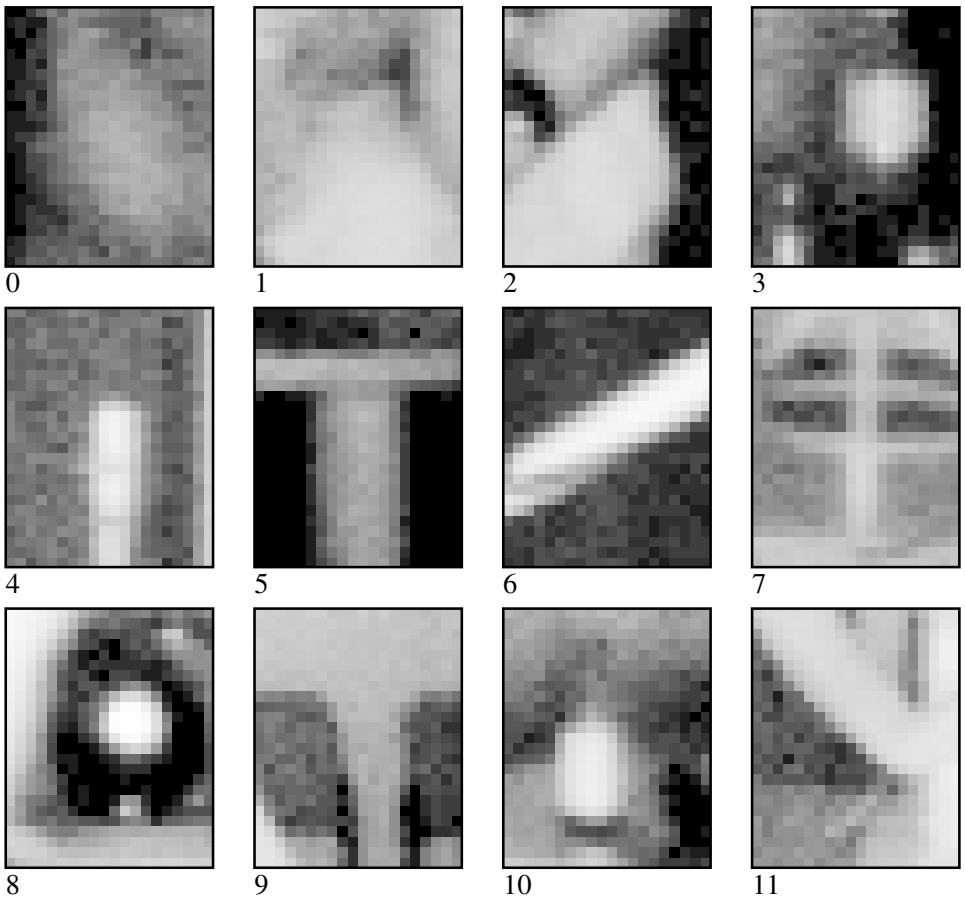
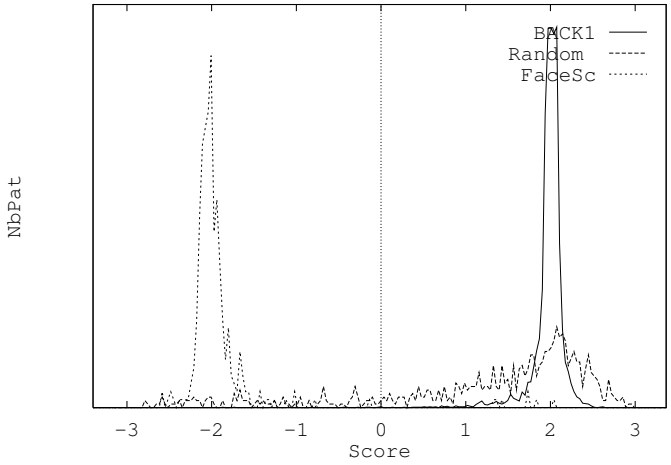
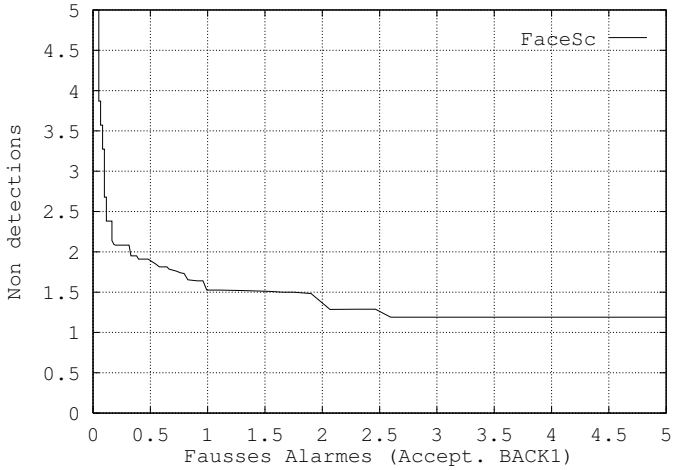


FIGURE 8.23 – Quelques imagettes extraites de la base *BACK1*. Ces imagettes ont été prises pour des visages par le premier étage, et sont utilisées pour l'apprentissage du deuxième étage de classification.



(a) Répartition des valeurs de sortie $S_0 - S_1$.



(b) Courbes de détection.

FIGURE 8.24 – Résultats obtenus par le classifieur **quickseg** (rétine 20x25), entraîné sur les 250 visages de *FaceSc_learn* et 3000 fonds de *BACK1*, avec modification de la fonction de coût.

8.8 Présentation du système complet de localisation de visages

Dans cette dernière section, on décrit la chaîne complète de localisation de visages et indiquons les performances globales mesurées sur l'ensemble des scènes de tests.

8.8.1 Architecture globale du système

L'architecture du système est en resumé la suivante :

1. pré-traitement de l'image (correction de l'histogramme, voir section 8.5.1) ;
2. décomposition multirésolution (voir section 8.6.2) ;
3. balayage des images issues de la décomposition par le réseau **quickseg9x11** (200 multiplications par point) ;
4. les imagettes pour lesquelles la réponse du réseau dépasse un seuil sont présentées à un deuxième réseau (**quickseg**). On enregistre les coordonnées des imagettes classées comme «visages» par ce deuxième réseau.

8.8.2 Réglages des seuils

On peut agir sur les performances du système par le réglage de deux seuils, correspondant aux seuils Θ_1 et Θ_2 de détection de chaque étage de classification.

On a vu dans la section 8.6.3 que le nombre d'imagettes extraites lors du balayage d'une scène était de l'ordre de 85 000 [6]. Parmi ces imagettes, quelques-unes (≈ 10) représentent des visages.

Les performances visées sont de descendre en dessous d'une fausse alarme par image, tout en ratant moins d'un visage sur dix.

On peut atteindre ces taux en choisissant Θ_1 (voir courbe 8.16) tel que la proportion de visages non détectés soit de 5 %. On récupère alors 1.6 % des imagettes de fonds, soit environ 1400 imagettes par scène.

Le deuxième seuil Θ_2 est alors choisi (voir courbe 8.24) pour rejeter 3 % des visages, soit 0.1 % de fausses alarmes. Il reste alors théoriquement 1.4 fausse alarme par scène, et l'on a rejeté au total environ 8 % des visages.

Ces estimations de performances sont nécessairement approximatives. Il aurait fallu tracer la courbe de détection du deuxième étage (8.24b) en utilisant uniquement les visages non rejetés par le premier niveau. D'autre part, la

6. En réalité, le nombre d'imagettes est ici égal à 69 000 environ du fait du changement de la taille de la rétine.

base *BACK1* n'est pas exactement représentative des imagettes reçues par le second classifieur lors du balayage. On a vu (8.7.2) que l'on avait selectionné ces imagettes pour obtenir une variété maximale, et donc un apprentissage plus robuste. Les performances calculées dans cette section sont ainsi normalement *sous-estimées*.

8.8.3 Post-traitements

Contrairement au système présenté plus haut (section 8.6), les post-traitements jouent ici un rôle mineur.

Le post-traitement se borne en effet à regrouper les alarmes adjacentes ou se recouvrant largement, afin d'éviter de compter deux fois le même visage.

8.8.4 Complexité algorithmique

Evaluons maintenant la complexité de notre système, en termes du nombre de multiplications nécessaire pour traiter une image de taille 764 sur 580 pixels.

On peut négliger les opérations de pré-traitements (analyse multirésolution, une dizaine de multiplications par pixel) et de post-traitements (regroupement des rectangles, pas de multiplications) devant l'opération de balayage des images par le réseau.

Le premier réseau est doté d'une rétine 9x11 et de 242 connexions. Le nombre de positions possibles, calculé comme en section 8.6.2, est ici approximativement égal à 69 000 (si l'on décale la rétine par pas de 1). L'analyse des 6 niveaux de décomposition nécessite donc 16 millions de multiplications.

On appelle en moyenne le deuxième réseau classifieur 1100 fois par image (voir section 8.8.2). Le second réseau (**quickseg**) possède 2321 connexions. Le second étage nécessite par conséquent de l'ordre de 2.5 millions de multiplications.

L'analyse complète d'une image 764x580 demande au total 8.5 millions de multiplications. Notons cependant que l'algorithme, par construction, est massivement parallélisable. La première phase de l'analyse s'apparente à un filtrage (sophistiqué) de l'image, opération pour laquelle on peut développer des circuits dédiés très performants.

8.8.5 Performances obtenues

Nous donnons dans cette section les performances obtenues par le système. Nous avons pu mesurer ces performances sur la base de scènes de test

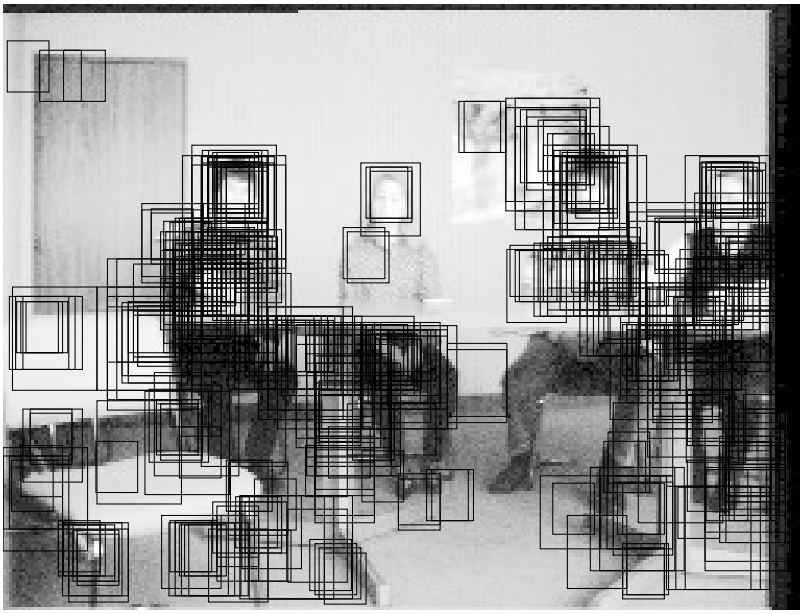


FIGURE 8.25 – Rectangles détectés par le premier réseau classifieur (**quick-seg9x11**).

complète (1000 images). En effet, grâce à la réduction de la taille du réseau, les temps de calculs sont descendus à 45 secondes par image sur Sun SparcStation1 [7].

Les performances globales mesurées sont les suivantes :

Taux de non détection	9.5 %
Fausses alarmes/scène	0.32

Ces résultats sont cohérents avec les estimations faites en section 8.8.2. En effet, d'après l'étalonnage des seuils de rejet sur les ensembles de validation, on s'attendait à rejeter environ 8 % des visages. Le taux de non détection finalement mesuré est un peu supérieur, ce qui est une conséquence de l'analyse multi-résolution : les visages dont la taille est située loin des deux niveaux de décomposition les plus proches sont plus difficile à détecter.

Le taux de fausses alarmes est par contre plus bas que prévu, ce qui était attendu (voir la fin de la section 8.8.2).

7. Seul le premier étage de classification à été optimisé lors du codage, le deuxième étage utilisant les fonctions génériques du simulateur de réseaux connexionnistes. La vitesse du système pourrait être encore amélioré. Une mesure sur PC 486 50MHz a donné moins de 25 secondes par image.

FIGURE 8.26 – Rectangles classés comme visages par le deuxième classifieur (**quickseg**), après avoir été détecté par le premier.

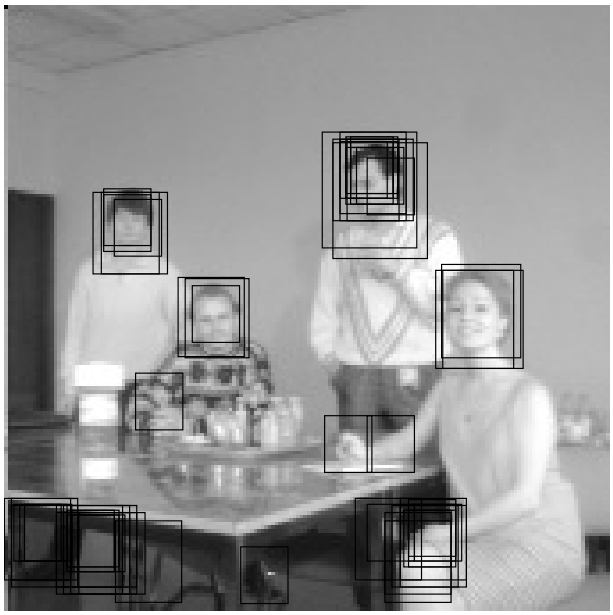


FIGURE 8.27 – Un autre exemple : rectangles détectés par le premier étage.

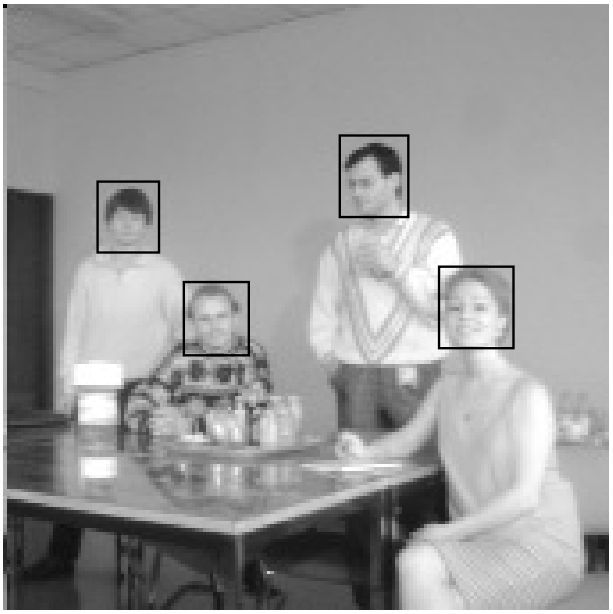


FIGURE 8.28 – Un autre exemple : visages localisés par le système.

8.9 Conclusion

Dans ce chapitre, nous avons présenté la démarche expérimentale qui nous a permis de construire un système de détection de visages performant.

Nous avons d'abord constaté que les approches «non supervisées», basées soit sur l'utilisation d'un réseau d'identification, soit sur un réseau auto-associatif réalisant une analyse en composantes principales des imagettes, étaient très insuffisantes pour résoudre le problème : de nombreux visages sont non détectés, et de nombreux objets sont confondus avec des visages. Rappelons que le système de localisation proposé par Turk et Pentland [203] repose sur une approche très similaire à celles ci.

Nous nous sommes ensuite intéressés à la mise au point d'un classifieur «visages/fonds» entrainé à l'aide d'imagettes sélectionnées dans un ensemble de scènes d'apprentissage. Nous avons dû comparer plusieurs prétraitement des images, différentes architectures TDNN multi-couches et différentes méthodes d'apprentissage (modification du coût) avant de retenir un classifieur aux performances suffisantes pour nos besoins.

La mesure précise des performances de ce classifieur (en particulier de sa tolérance aux translations et dilatations de l'image) nous a permis de proposer un premier système de localisation de visages, utilisant un étage unique de décision. Les performances de ce système, prometteuses mais insuffisantes

pour une exploitation réelle, nous ont conduit à définir une seconde architecture, basée sur une approche «*coarse to fine*» et utilisant deux classifieurs en cascade.

Cette architecture, inédite, permet de mieux tirer parti de l'analyse multirésolution de l'image : on utilise d'abord les échelles de faible résolution pour une détection grossière (rétine 9x11), puis l'on exploite sélectivement les niveaux de résolution plus fine dans le deuxième étage (rétine 20x25).

Ce système règle aussi le problème de la sélection des exemples d'imagettes de fonds pour l'apprentissage du réseau de segmentation.

Enfin, notre système à deux étages permet une réduction des temps de calcul par un facteur quarante, par rapport au premier système.

Les performance de ce module de détection sont très satisfaisantes. Insistons sur le fait que les images utilisées pour les tests représentent des situations réelles : intérieurs de bureaux ou d'appartement, personnes en mouvement ou dans une grande variété de positions, éclairage variable de qualité souvent assez mauvaise. Ces conditions sont assez différentes de celles utilisées dans la littérature (fonds blancs, personnes de face).

Pour montrer la faisabilité d'un système complet de localisation et identification de visages, il nous faut encore présenter les résultats du chaînage de nos deux modules. C'est l'objet du chapitre suivant.

Chapitre 9

Enchaînement localisation-identification

9.1 Introduction

Nous avons exposé dans les deux chapitres précédents la mise au point et les performances des modules de localisation et d'identification de visages. Les performances de ces modules ont été mesurées indépendamment, sur des ensembles d'images constitués à cet effet.

Nous évaluons dans ce chapitre une chaîne de traitement de visage complète, qui prend une image de scène en entrée, et fournie la liste des identités des personnes détectées en sortie. Il s'agit ici d'intégrer la localisation et la reconnaissance d'objets. Cette intégration fournit l'occasion de tester en vraie grandeur la qualité de nos algorithmes ; les défauts de chacun vont interagir et risquent de dégrader les performances globales du système.

Nous exposons d'abord l'architecture de la chaîne de traitement, qui consiste à appliquer le module d'identification de visages à la sortie du module de localisation, et expliquons quels critères utiliser pour évaluer les performances. Nous décrivons ensuite les données utilisées, qui sont une partie des ensembles d'images utilisés dans les chapitres précédents. Enfin, nous détaillons les résultats obtenus.

9.2 Principe et mesure des performances

Le système présenté dans ce chapitre est une illustration de l'utilisation de nos algorithmes, qui ne correspond pas forcément à une application très réaliste.

On suppose que le système «connait» une famille de personnes (de taille

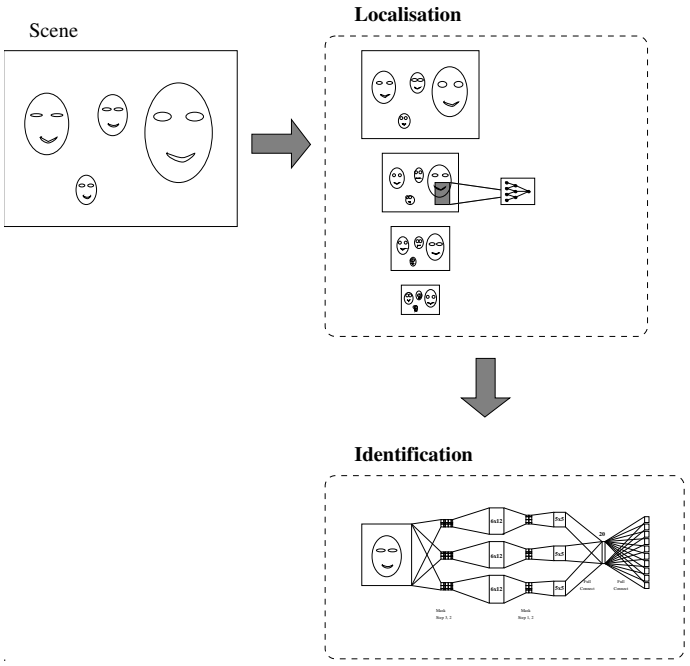


FIGURE 9.1 – Architecture de la chaîne de traitement.

comprise entre 1 et 15), dont on a utilisé des images de visages pour l'apprentissage du module d'identification. On fournit au système des images de scènes, pouvant contenir un nombre quelconque de personnes appartenant ou non à la famille. Le système doit compter et localiser les personnes présentes tournées vers l'objectif de la camera, et les identifier si elles appartiennent à la famille.

Il s'agit donc d'une application de surveillance un peu particulière, à laquelle il faudrait imaginer une utilité... Notons que notre système peut travailler sur une image unique : on n'utilise jamais d'informations *dynamiques* sur le déplacement des personnes dans la scène, technique souvent utilisée pour faciliter la détection [204, 203, par exemple].

9.2.1 Architecture générale

L'architecture que nous proposons ici est simple. Il s'agit en effet d'enchaîner, sans modifications, les modules de localisation et d'identification mis au point précédemment.

9.2.2 Mesure des performances

Nous avons déjà évoqué (section 6.6.1), à propos des systèmes de segmentation de chiffres, divers critères permettant de mesurer les performances d'un module intégrant localisation et identification. On a alors définit un *taux d'erreur global*, mesurant la capacité du classifieur à rejeter les formes inconnues et à identifier sans erreur les formes connues.

Dans le cas présent, il convient d'être un plus précis. En effet, on désire exprimer les performances par *scène*, et non plus par *forme*, ou imagette. Se rajoute aussi le problème des visages inconnus, que l'on souhaite localiser mais pas identifier.

Soit une image de scène, contenant N_f^0 visages, dont N_c^0 de personnes connues et N_i^0 inconnus (visages invités) : $N_f^0 = N_c^0 + N_i^0$.

Le module de localisation va détecter N rectangles, répartis entre N_f^1 visages, N_c^1 connus et N_i^1 inconnus, et N_b fausses détections. Dans le chapitre 8, on a indiqué :

— Taux de non détection $= < (N_f^0 - N_c^1 - N_i^1)/N_f^0 >$
— Fausses alarmes/scène $= < N_b >$

Le module d'identification, prend la décision d'identifier ou de rejeter (classer comme invité) chaque rectangle détecté.

FIGURE 9.2 – Rôle du module d'identification

Il peut commettre les erreurs suivantes :
— Si la forme est identifiée :
  — c'est une forme connue, mais elle est mal identifiée ;
  — la forme est un visage inconnu ou un fond.
— Si la forme est classée comme invité :
  — il s'agit en réalité d'un visage connu ;
  — il s'agit d'un fond. Cette erreur est imputable au module de localisation.

Les types d'erreurs sont donc variés. En l'abscence d'application précise, il est difficile de définir un critère unique de mesure des performances. Il faudrait en effet affecter à chaque type d'erreur un coût spécifique, suivant l'importance qu'on lui accorde. Par exemple, on pourra mettre l'accent sur

le comptage (exactitude du nombre de visages connus détectés), ou sur la qualité de l'identification.

Nous nous contenterons des mesures suivantes :

— erreur de comptage, $E_c = < \mid N_f^1 + N_b - N_f^0 \mid >$, dans laquelle on accorde la même importance aux non détections et aux fausses alarmes ;

— erreur d'identification, ou' l'on compte uniquement les erreurs parmi les formes identifiées.

9.3 Données

Les données utilisées ici sont extraites de l'ensemble d'images de scènes décrit en section 8.3 du chapitre précédent.

Afin de tester le système dans les conditions définies plus haut, nous avons selectionné les images de scènes contenant des personnes *connues*, c'est à dire pour lesquelles nous disposions d'images de visages dans la base *BDD*, décrite en section 7.3.2.

Parmi les quinze personnes de *BDD*, cinq figurent dans les images de scènes. Nous avons décidé de mesurer les performances sur les 420 images, extraites de l'ensemble des scènes de test (voir 8.3), dans lesquelles elles ces personnes sont présentes. Dans ces scènes figurent sept personnes, dont deux «invités». On a en moyenne trois visages environ par image.

9.4 Résultats

9.4.1 Localisation

Nous avons utilisé le module de localisation avec les même réglages qu'en section 8.8.

Les performances mesurées sur les 420 images selectionnées sont les suivantes :

Taux de non détection	9.2 %
Fausses alarmes/scène	0.34

soit des résultats très proches de ceux mesurés sur l'ensemble de toutes les scènes de test.

9.4.2 Identification

Sur les 420 scènes, le module d'identification reçoit 1173 imagettes :

— 1030 visages, dont 178 invités ;

— 143 fausses alarmes.

Le seuil de rejet du module d'identification a été réglé pour mesurer 1 % d'erreur sur l'ensemble de test de *BDD* (voir figure 7.18, page 226). On rejette alors 2 % des visages connus de *BDD*.

Le module accepte d'identifier 70.6 % des formes reçues. Sur ces formes, il commet 8.2 % d'erreur. Il est intéressant de constater que les trois sources d'erreurs contribuent à part presques égales : on observe en effet, pour 100 formes identifiées, 2.5 fonds, 3 invités, et 2.6 erreurs d'identification (confusion entre deux personnes connues).

86 % des invités sont détectés comme tels, et 71 % des fausses alarmes sont classées comme invités. Ce taux de rejet plus faible pour les fausses alarmes, s'explique en partie par le fait qu'un objet (sorte de sucrier arrondi) présent dans plusieurs scènes a été systématiquement confondu avec une personne connue, contribuant à diminuer le taux de rejet. N'oublions pas que des taux de cet ordre, mesurés sur un millier d'observations sont évidemment soumis à une importante incertitude statistique.

Le taux de rejet de visages connus est de 8.2 %, ce qui est nettement supérieur à la valeur mesuré sur la base *BDD* (2 %). Ceci s'explique par la différence de qualité et de cadrage entre les visages de *BDD* et les visages détectés dans les scènes et cadrés par le module de localisation. Notons que le taux d'erreur est passé de 1 % à 2.6 %, augmentation plus faible que le taux de rejet : le système à tendance à rejetter les visages de mauvaise qualité, plutôt qu'a les confondre avec d'autres.

9.4.3 Performances Globales

Récapitulons les performances mesurées, selon les critères énoncés plus haut (section 9.2.2) :

— erreur de comptage moyenne par image = 0.58 (0.24 non detections et 0.34 fausses détections par image) ;
— erreur d'identification = 8.2 %.

9.5 Conclusion

Nous avons montré dans ce chapitre qu'il était possible d'enchaîner notre module de localisation de visages et notre module d'identification. Ce résultat est très satisfaisant, car il prouve que la qualité du cadrage des visages détecté par notre système est suffisante pour permettre une identification par le réseau TDNN entrainé sur des visages de bonne qualité parfaitement

cadrés. Pour cela, il était indispensable que ce réseau soit tolèrant aux dilatations, réductions et translations du visage présenté, ce que nous avions vérifié en section 7.6.6.

Le taux d'erreur d'identification est passé à 8 %, avec un taux de rejet des visages connus de 8.2 %. Insistons sur le fait que ce taux d'erreur, qui peut sembler élevé, est dû aux deux tiers à des acceptations erronées d'images de fonds ou de visages inconnus. Une amélioration des performances de rejet de ce module permettrait par conséquent d'améliorer significativement ce résultat. Pour cela, on pourrait mettre en œuvre une approche multi-modulaire (TDNN+RBF_coop par exemple), ce que nous n'avons malheureusement pas encore pu tester. Rappelons que cette approche (voir chapitre 6) permet, dans le cas des chiffres manuscrits, d'améliorer les performances de classification et surtout de rejet.

A notre connaissance, ce système de détection/identification de visages offre des performances inédites. Aucun des systèmes décrits à jour dans la littérature ne permet de détecter des visages dans des scènes quelconques, sans aucune information a priori sur la distance, l'attitude ou la position de la personne, ni sur l'éclairage ou le décors environnant.

Les volumes de calculs, de l'ordre de 8.5 millions de multiplications par image, sont encore trop grands pour un implémentation temps réel (voir section 8.8.4). Néanmoins, l'algorithme de localisation est facilement parallélisable dans une machine dédiée. Notons à ce propos que l'identification des visages, qui nécessite 5500 multiplications, est extrèmement rapide.

Chapitre 10

Conclusion

Lorsque nous avons commencé ce travail en 1990, les modèles connexionnistes commençaient à peine à sortir des quelques laboratoires universitaires les ayant développés, et l'on envisageait timidement des applications sur des problèmes réels. Une partie de notre travail a été de montrer que ces techniques étaient applicables à des problèmes de traitement d'image de grande taille, et comment les incorporer dans une chaîne de traitement complète.

Dans la thèse, nous avons exploré les possibilités d'utilisation des réseaux connexionnistes pour la segmentation et la reconnaissance d'images. Les études expérimentales sur le traitement des chiffres manuscrits nous ont permis de valider certaines architectures pour la segmentation d'image, que nous avons ensuite appliqué pour la mise au point de notre chaîne de localisation/identification de visages.

Nous avons constamment privilégié la recherche d'architectures de petite tailles, permettant une mise au point (apprentissage) plus facile et offrant des performances (temps de calculs) élevées. Pour cela, nous avons mis en œuvre des approches modulaires, permettant de bâtir des systèmes complexes à partir de modules simples, dont il est plus facile de choisir l'architecture et de contrôler la capacité.

Pour le traitement des chiffres manuscrits, nous avons pu montrer que l'approche multi-modulaire combinant un pré-traitement adaptatif des données par un réseau MLP et une classification par un classifieur RBF permettait un gain de complexité très important pour des performances compétitives avec les meilleurs systèmes de reconnaissance actuels.

Au début de ce travail, l'identification automatique de visages était un problème ouvert ; plusieurs types de solutions avaient été proposés, mais aucune n'avait pu être testée sur une application «grandeur nature» avec des résultats satisfaisants. L'intérêt pour ce problème a considérablement augmenté ces dernières années, avec l'arrivée de machines bon marché dotées

de fortes puissances de calcul et la banalisation des moyens d'acquisition d'images.

Nous avons étudié en détail les conditions nécessaires à l'identification d'images de visages vus de face, et avons montré que l'on pouvait obtenir d'excellentes performances à l'aide de réseaux MLP de petites tailles.

Une fois ce problème résolu, nous nous sommes attaqués à la localisation des visages dans des images de scènes. Cette application est extrèmement difficile car les visages peuvent prendre des aspects très variés suivant leur position, leur distance à l'objectif et l'éclairage ambiant. De plus, la taille des images de scènes entraine des volumes de calculs importants qu'il convient de maitriser.

L'approche que nous avons proposé repose sur une analyse multirésolution de la scène et l'emploi d'un module de classification à deux étages. A l'occasion de la mise au point de ce dernier, nous avons comparé expérimentalement différentes approches, supervisées ou non, pour la classification formes/bruits.

La méthode retenue est très générale, et peut s'appliquer à d'autres problèmes de détection de formes quelconques, aussi bien dans des signaux (e.g. séries temporelles) que des images. En effet, nous n'avons jamais introduit d'hypothèses fortes sur le type d'objets traités.

Les performances des modules de localisation et d'identification de visages sont suffisantes pour permettre leur enchaînement : nous avons discuté les performances du système global obtenu, qui permet la localisation de toutes les personnes présentes dans une scène, et l'identification de celles connues du système.

Les études menées au cours de cette thèse soulèvent un certain nombres de problèmes théoriques difficiles. A plusieurs reprises (et en particulier pour les classifieurs fonds/visages), nous nous sommes trouvés confontés au problème de la sélection d'un ensemble d'apprentissage pertinent. Cette question se pose aussi en reconnaissance de caractères, ou‘ l'on utilise couramment des ensembles d'apprentissages comprenant plus de 100 000 exemples. Une méthode permettant de réduire la taille de ces ensembles en sélectionnant les éléments pertinents pour la classification serait très utile. De nouvelles approches théoriques semblent prometteuses, commes celles issues des considérations de Vapnik [13] ou des approches Bayesiennes [144].

L'incorporation d'invariances géométriques lors de conception d'un classifieur est un moyen très efficace d'augmenter la qualité de la généralisation tout en limitant le nombre d'exemples nécessaires. Nous avons à plusieurs reprises utilisé des contraintes simples sur les connexions (poids partagés) de nos réseaux pour imposer des traitements invariants par translation. Au cours du chapitre 5 nous avons évoqué une autre voie, basée sur l'utilisation

d'une nouvelle mesure de similarité (distance tangente) [193, 192]. Il serait intéressant d'appliquer ces considérations à la détection multirésolution, afin d'augmenter la robustesse aux changements de tailles. Il reste de nombreuses directions à explorer dans ce domaine.

Annexe A

Validité Statistique

A.0.1 Introduction

De nombreuses recherches ont été (et sont toujours) menées pour mesurer l'impact du nombre d'exemples utilisé dans les ensembles d'apprentissage et de test sur la qualité de l'estimation des paramètres d'un classifieur et des taux d'erreurs attendus [175, 205, 25, 72].

Dans cette annexe, nous décrivons une méthode simple permettant d'estimer l'intervalle de validité statistique de résultats de classification mesurés sur un ensemble de test de taille N [4]. Cet intervalle permet de s'assurer de la pertinence des comparaisons de résultats obtenus par plusieurs classifieurs.

D'autres méthodes d'estimation, plus précises, existent (voir par exemple [17]). La méthode que nous avons utilisé offre l'avantage de la simplicité et suffit amplement à nos besoins.

A.0.2 Intervalle de Confiance à α %

Etant donné une variable aléatoire X dont on connait la distribution dans une population E, déterminer l'intervalle de confiance revient à construire un intervalle $[b_{\text{inf}}, b_{\text{sup}}]$ qui contienne toute valeur X avec la probabilité $(1-\alpha)$, α étant le risque d'erreur.

Ceci revient à déterminer la quantité positive Z_α telle que

$$P(X \in [-Z_\alpha, Z_\alpha]) = 1 - \alpha \tag{A.1}$$

Z_α se déduit simplement de la table de la distribution de X.

Lors de la mesure d'un taux de reconnaissance, un test élémentaire consiste à savoir si une forme est bien reconnue ou non. La probabilité d'observer k succès sur n formes indépendantes suit une loi binomiale.

Soit p la probabilité de succès (i.e. reconnaissance correcte). On mesure $X = X_1 + X_2 + \ldots + X_n$. Si on suppose les formes choisies aléatoirement et indépendemment les unes des autres, on a

$$E(X) = n.p \qquad \text{et } \mathrm{Var}(X) = n.p(1-p) \tag{A.2}$$

i.e. X suit une loi binomiale $B(np,\, np\,(1-p))$.

Notons $T = X/N$ le taux de succès mesuré. Alors

$$T \approx B(p, \frac{p\,(1-p)}{n}) \tag{A.3}$$

Sin n est assez grand (i.e. $n = N \gtrsim 50$), on peut approximer la loi binomiale par une loi normale :

$$T \approx \mathcal{N}(\, p, \frac{p\,(1-p)}{N}) \tag{A.4}$$

Il est plus pratique de considérer la loi normale centrée et réduite de T,

$$R = \frac{T - E(T)}{\sqrt{\mathrm{Var}(T)}} = \frac{T - p}{\sqrt{\frac{p\,(1-p)}{N}}} \tag{A.5}$$

Définir l'intervalle de confiance à $\alpha\%$ revient donc à chercher Z_α tel que :

$$P(|R| < Z_\alpha) = 1 - \alpha \tag{A.6}$$

Or

$$|R| < Z_\alpha \;\Leftrightarrow\; |T - p|^2 < Z_\alpha^2 \,\frac{p\,(1-p)}{N} \tag{A.7}$$

D'ou'

$$p^2\,(1 + \frac{Z_\alpha^2}{N}) - 2p\,(T + \frac{Z_\alpha^2}{2N}) + T^2 \;<\; 0 \tag{A.8}$$

Cette inéquation est satisfaite si p est compris entre les racines du trinôme :

$$p_1 = \frac{T + \frac{Z_\alpha^2}{2N} - Z_\alpha\sqrt{\frac{T\,(1-T)}{N} + \frac{Z_\alpha^2}{4N^2}}}{1 + \frac{Z_\alpha^2}{N}} \tag{A.9}$$

$$p_2 = \frac{T + \frac{Z_\alpha^2}{2N} + Z_\alpha\sqrt{\frac{T\,(1-T)}{N} + \frac{Z_\alpha^2}{4N^2}}}{1 + \frac{Z_\alpha^2}{N}} \tag{A.10}$$

L'intervalle de confiance à $\alpha\%$ est $[p_1, p_2]$. Notons que si N est très grand, l'expression ci-dessus devient :

$$p_i = T \,\pm\, Z_\alpha\,\sqrt{\frac{T\,(1-T)}{N}} \tag{A.11}$$

Pour les intervalles de confiances calculés dans les chapitres 6, 7 et 8 on a utilisé les équations A.9 et A.10, avec en général $\alpha = 0.95$.

Annexe B

Algorithme de découpage rapide des visages

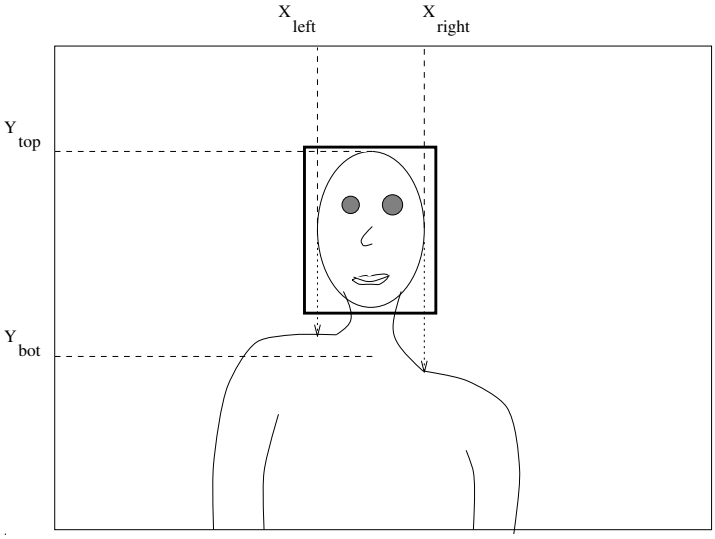


FIGURE B.1 – Points utilisées pour définir le rectangle entourant le visage.

Nous décrivons dans cette annexe l'algorithme de découpage automatique des visages utilisé pour traiter les images de la base B2 (voir section 7.3.2).

Cet algorithme travaille sur une image contenant une personne en son centre, sur fond blanc uni, à peu près de face (voir figure B.1). Il fournit en sortie les coordonnées d'un rectangle tangent au contour du visage, et de rapport largeur sur hauteur égal à 20/25.

On procède comme suit :

1. Calcul des images de gradient horizontal et vertical. Pour cela, on a utilisé le filtre récursif uni-dimensionnel de Deriche, que nous avons présenté en section 4.2.2.
2. Recherche du contour supérieur du crâne. Cette recherche s'effectue sur chaque colonne en partant du haut de l'image, et en descendant tant que le gradient vertical reste inférieur à un seuil donné. Le point supérieur de ce contour, noté y_{top}, sera pris comme la limite supérieure du visage (voir figure B.1).
3. De la même façon, on obtient les limites gauche et droite en utilisant l'image de gradient horizontal et en partant successivement de l'extrême gauche et l'extrême droite.
4. Détermination du bas du visage. Pour celà, on recherche l'intersection de la droite verticale partant de la gauche du visage avec le contour de l'épaule (obtenu lors de l'étape 2). On procède de même à droite. Le bas du visage est défini comme la moyenne des coordonnées trouvées.

5. Mise au bon rapport hauteur/largeur. Le rectangle obtenu précédemment est légèrement dilaté pour garantir un rapport hauteur/largeur égal à 25/20. A ce stade, on procède aussi à diverses vérifications de cohérence de coordonnées afin d'éliminer les images aberrantes.

Bibliographie

[1] T. Artieres, Y. Bennani, Gallinari P., and Montacié C. Connectionist and conventional models for free-text talker identification tasks. In EC2, editor, *Proceedings of Neuro-Nîmes'91*, 1991.

[2] P. Baldi and K. Hornik. Neural networks and principal component analysis : learning from examples without local minima. *Neural Networks*, 2 :53–58, 1989.

[3] S. Becker and Y. Le Cun. Improving the convergence of back-propagation learning with second order methods. In D.S. Touretzky and G.E. Hinton T.J. Sejnowsky, editors, *Proc. of Connectionist Models Summer School*, pages 29–37, San Mateo, CA., 1988. Morgan Kaufman.

[4] Younès Bennani. *Approches Connexionnistes pour la Reconnaissance Automatique du Locuteur : Modélisation et Identification.* PhD thesis, Université Paris XI, 1992.

[5] Younès Bennani. Text-independent talker identification system combining connectionist and conventional models. In *Proceedings of the IEEE-SP 1992 Workshop*, Neural Networks For Signal Processing, pages 131–138, Denmark, 1992.

[6] Chris Bishop. Curvature-driven smoothing in backpropagation neural networks. In *Proceedings of INNC Paris*, volume 2, page 749, 1990.

[7] Chris Bishop. Improving the generalization properties of Radial Basis Function neural networks. *Neural Computation*, 3 :579–588, 1991.

[8] Chris Bishop. Exact calculation of the hessian matrix for the multilayer perceptron. *Neural Computation*, 4 :494–501, 1992.

[9] W. W. Bledsoe. Man-machine facial recognition. Technical Report 22, Panoramic Research Inc., Palo Alto, CA, August 1966.

[10] W. W. Bledsoe. The model method in facial recognition. Technical Report 15, Panoramic Research Inc., Palo Alto, CA, August 1966.

[11] Michel De Bollivier. *Incorporation de la modularité dans les réseaux connexionnistes par décomposition de tâche.* PhD thesis, Université Paris XI, 1992.

[12] Michel De Bollivier, Patrick Gallinari, and Sylvie Thiria. Cooperation of neural nets for robust classification. In *Proceedings of IJCNN Washington*, volume 1, pages 113–120, 1990.

[13] Bernhard Boser, Isabelle Guyon, and Vladimir Vapnik. A training algorithm for optimal margin classifiers. *Submitted to ACM Workshop on Computational Learning Theory*, 1992.

[14] Léon Bottou and Patrick Gallinari. A formalism for neural net algorithms. In EC2, editor, *Proceedings of Neuro-Nîmes'91*, 1991.

[15] Léon Bottou and Patrick Gallinari. A framework for the cooperation of learning algorithms. In R.P. Lippmann, J.E. Moody, and D.S. Touretzky, editors, *Advances in Neural Information Processing Systems*, 3, pages 781–788, 1991.

[16] Léon Bottou and Patrick Gallinari. A unified formalism for neural net training algorithms. In *Proceedings of IJCNN Baltimore*, volume 4, pages 7–12, 1992.

[17] Léon Bottou and Vladimir Vapnik. Local learning algorithms. *Neural Computation*, 4 :888–900, 1992.

[18] Léon Y. Bottou. *Une approche théorique de l'apprentissage connexionniste ; applications à la reconnaissance de la parole.* PhD thesis, Université Paris XI, 1991.

[19] Hazem Bouattour, Francoise Fogelman Soulié, and Emmanuel Viennet. Neural nets for human face recognition. In *Proceedings of IJCNN Baltimore*, volume 3, pages 700–704, 1992.

[20] Hazem Bouattour, Francoise Fogelman Soulié, and Emmanuel Viennet. Solving the human face recognition task using neural nets. In I. Aleksander and J. Taylor, editors, *Proceedings of ICANN'92 Brighton*, volume 2 of *Artificial Neural Networks*, pages 1595–1599. Elsevier, 1992.

[21] H. Bourlard and Y. Kamp. Auto-association by multilayer perceptrons and singular value decomposition. *Biological Cybernetics*, pages 291–294, 1988.

[22] H. Bourlard and N. Morgan. A continuous speech recognition system embedding MLP into HMM. In Touretzky, editor, *Advances in Neural Information Processing Systems*, 2, pages 186–193, 1990.

[23] H. Bourlard and C.J. Wellekens. Links between Markov models and multilayer perceptrons. In D.Z. Anderson, editor, *Advances in Neural Information Processing Systems*, 1, pages 502–510, 1989.

[24] R.M. Božinović and S.N. Srihari. Off-line cursive word recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(1) :68–83, January 1989.

[25] L. Breiman, J. Friedman, R.A. Olsen, and C.J. Stone. *Classification and Regression Trees*. Wadsworth, Belmont, CA, 1984.

[26] John S. Bridle. Probabilistic interpretation of feedforward classification network outputs, with relationship to statistical pattern recognition. In F. Fogelman Soulié and J. Hérault, editors, *Neurocomputing, algorithms, architectures and applications.*, volume 68 of *NATO ASI Series in Computer and Systems Sciences*, pages 227–236. Springer Verlag, 1990.

[27] P. Brodatz. *Textures : A Photographic Album for Artists and Designers*. Dover Publishing Co., Toronto, 1966.

[28] D.S. Broomhead and D. Lowe. Multivariate functional interpolation and adaptative networks. *Complex Systems*, 2, 1988.

[29] Wray L. Buntine and Andreas S. Weigend. Computing second derivatives in feed-forward networks : a review. *IEEE Trans. on Neural Networks*, 1993. To appear (also available on neuroprose ftp archive).

[30] Gilles Burel, Isabelle Pottier, and Jean Yves Catros. Recognition of handwritten digits by image processing and neural networks. In *Proceedings of IJCNN Baltimore*, volume 3, pages 666–671, 1992.

[31] C.J.C. Burges, O. Matan, Y. Le Cun, J.S. Denker, L.D. Jackel, C.E. Stenard, C.R. Nohl, and J.I. Ben. Shortest path segmentation : a method for training a neural network to recognize character strings. In *Proceedings of IJCNN Baltimore*, volume 3, pages 165–172, 1992.

[32] Peter J. Burt. Smart sensing within a pyramid vision machine. *Proceedings of the IEEE*, 76(8) :1006–1015, August 1988.

[33] Peter J. Burt, Tsai-Hong Hong, and Azriel Rosenfeld. Segmentation and estimation of image region properties through cooperative hierarchial comptutation. *IEEE Trans. on Systems, Man, and Cybernetics*, SMC-11(12) :802–809, December 1981.

[34] John Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(6) :679–698, November 1986.

[35] S. Carey and R. Diamond. From piecemeal to configurational representation of faces. *Science*, 195 :312–313, January 1977.

[36] M. Carlin. Radial Basis Function networks and non-linear data modelling. In EC2, editor, *Proceedings of Neuro-Nîmes'92*, pages 623–633, 1992.

[37] J. P. Changeux. *L'homme neuronal*. Fayard, 1983.

[38] N. Chaourar, A. Mellouk, Y. Bennani, and F. Fogelman. Etude comparative entre les cartes topologiques et les méthodes classiques de quantification vectorielle. In EC2, editor, *Proceedings of Neuro-Nîmes'90*, 1990.

[39] Chedsada Chinrungrueng and Carlo H. Séquin. Optimal adaptative K-Means algorithm with dynamic adjustement of learning rate. Technical Report 91-017, ICSI, Berkeley, California, March 1991.

[40] C. K. Chow. An optimum character recognition system using decision functions. *R.E. Trans. on Electronic Computers*, pages 247–254, December 1957.

[41] C. K. Chow. On optimum recognition error and reject tradeoff. *IEEE Trans. on Information Theory*, 16(1) :41–46, January 1970.

[42] Krystof J. Cios, Robert E. Tjia, Ning Liu, and Robert A. Langenderfer. Study of continuous ID3 and radial basis function algorithms for the recognition of glass defects. In *Proceedings of IJCNN Seattle*, volume 1, pages 49–54, 1991.

[43] G. W. Cottrell, P.W. Munro, and D. Zipser. Image compression by back-propagation : a demonstration of extensional programming. In N.E. Sharkey, editor, *Advances in Cognitive Science*, volume 2. Norwood NJ Ablex, 1989.

[44] Garrison W. Cottrell. Extracting features from faces using compression networks : Face, identity, emotion and gender recognition using holons. In D.S. Touretzky, J.L. Elman, T.J. Sejnowsky, and G.E. Hinton, editors, *Proc. of Connectionist Models Summer School*, volume 1, pages 328–337. Morgan Kaufman, 1991.

[45] Garrison W. Cottrell and Michael Fleming. Face recognition using unsupervised feature extraction. In *Proceedings of INNC Paris*, volume 1, pages 322–325, 1990.

[46] I. Craw, H. Ellis, and J.R. Lishman. Automatic extraction of face-features. *Pattern Recognition Letters*, 5 :183–187, February 1987.

[47] G. Cybenko. Approximation by superpositions of a sigmoidal function. *Math. Control Systems Signals*, 2 :303–314, 1989.

[48] C. Darken and J. Moody. Fast adaptative k-means clustering : Some empirical results. In *Proceedings of IJCNN Washington*, pages 233–238, June 1990.

[49] David DeMers and Garrison Cottrell. Non-linear dimensionality reduction. In *Advances in Neural Information Processing Systems*, 5, 1993. To appear.

[50] Rachid Deriche. Optimal edge detection using recursive filtering. In *First International Conference on Computer Vision*, pages 501–505, June 1987.

[51] Rachid Deriche and Jean Pierre Cocquerez. Extraction de composantes connexes basée sur une détection optimale des contours. In *Cognitiva 87*, May 1987.

[52] P.A. Devijver. New error bounds with the nearest neighbor rule. *IEEE Trans. on Information Theory*, 25 :749–753, November 1979.

[53] P.A. Devijver and J. Kittler. *Pattern Recognition, a Statistical Approach*. Prentice Hall, Englewood Cliff, 1982.

[54] T. Dietterich and G. Bakiri. A comparative study of ID3 and backpropagation for english text-to-speech mapping. In *Proceedings of the 1990 Machine Learning Conference*, Austin, TX., 1990.

[55] Xavier Driancourt. Fondation des systèmes d'apprentissage. Technical Report 653, Laboratoire de Recherches en Informatique, Paris XI, Orsay, March 1991.

[56] Xavier Driancourt, Léon Y. Bottou, and Patrick Gallinari. Learning Vector Quantization, Multi-Layer Perceptrons and Dynamic Programming : comparison and cooperation. In *Proceedings of IJCNN Seattle*, volume 2, pages 815–819, 1991.

[57] Xavier Driancourt and Patrick Gallinari. Empirical risk optimisation : neural networks and dynamic programming. In *Proceedings of the IEEE-SP 1992 Workshop*, Neural Networks For Signal Processing, pages 121–130, Denmark, 1992.

[58] Bernard Dubuisson. *Diagnostic et reconnaissance des formes*. Hermes, Paris, 1990.

[59] R.O. Duda and P.E. Hart. *Pattern Classification and Scene Analysis*. John Wiley & Sons, New York, 1973.

[60] H. D. Ellis. *The Neuropsychology of Face Perception and Facial Expression*. Lawrence Erlbaum Associates, New Jersey, 1986.

[61] Jean Christophe Feauveau. *Analyse multirésolution par ondelettes non orthogonales et bancs de filtres numériques*. PhD thesis, Université Paris XI, 1990.

[62] Jean Christophe Feauveau. Analyse multirésolution pour les images avec un facteur de résolution $\sqrt{2}$. *Traitement du Signal*, 7(2) :117–128, 1990. (in french).

[63] Jean Christophe Feauveau and Emmanuel Viennet. Multiresolution segmentation by neural networks. In *Proceedings of INNC Paris*, volume 1, pages 149–152, 1990.

[64] R.A. Fisher. The use of multiple measurements in taxonomic problems. *Ann. Eugenics*, 7, Part II :179–188, 1936. Also in *Contributions to Mathematical Statistics* (Wiley, New York, 1950).

[65] W. Fisher, V. Zue, J. Bernstein, and D. Pallett. An acoustic-phonetic data base. *J. Acoustic Soc. Amer.*, 81, S92, Suppl. (A), 1987.

[66] M.A. Fishler and R.A. Elschlager. The representation and matching of pictorial structures. *IEEE Trans. Computers*, c-22(1), January 1973.

[67] Michael Fleming and Garrison W. Cottrell. Categorization of faces using unsupervised feature extraction. In *Proceedings of IJCNN Washington*, volume 2, 1990.

[68] H. Von Foerster and H. Zopf. *Principles of self-organization*. Pergamon, New York, 1962.

[69] G. David Forney. The Viterbi algorithm. *Proceedings of the IEEE*, 61(3), March 1973.

[70] Jűrgen Franke. On the functional classifier. In *ICDAR*, pages 740–748, Saint Malo, France, 1991.

[71] Paolo Frasconi, Marco Gori, Soda Giovanni, Alberto De Curtis, and Fabrizio Innocenti. Face recognition using Multi-Layered Perceptrons which also reject never seen faces. In *Proceedings of 4th Italian Workshop on Neural Networks and Parallel Architectures*, Vietry (Italy), 1992.

[72] K. Fukunaga. *Statistical Pattern Recognition*. Academic Press, second edition, 1990.

[73] P. Gallinari, S. Thiria, F. Badran, and F. Fogelman Soulié. On the relations between discriminant analysis and multilayer perceptrons. *Neural Networks*, 4 :349–360, 1991.

[74] P. Gallinari, S. Thiria, and F. Fogelman Soulié. Multilayer perceptron and data analysis. In *Procs of IEEE 2nd ICNN San Diego*, volume 1, pages 391–401, 1988.

[75] Sir Francis Galton. Personal identification and description. *Nature*, 10 :173–177, 1888.

[76] Sir Francis Galton. Numeralized profiles for classification and description. *Nature*, 83 :127–130, 1910.

[77] M.D. Garris and R.A. Wilkinson. NIST Special Database 3 : handwritten segmented characters. NIST document, 1992. The NIST database can be obtained by writing at : NIST, 221/A223 Gaithersburg, MD 20899, USA.

[78] Philippe Gentric and Heini C.A.M. Withagen. Constructive methods for a new classifier based on a Radial Basis Function neural network accelerated by a tree, 1993. To appear, available on neuroprose ftp archive.

[79] Goldstein, Harmon, and Lesk. Identification of human faces. In *Proceedings of the IEEE*, volume 59, pages 748–760, 1971.

[80] B.A. Golomb, T.A. Lawrence, and T.J. Sejnowski. Sexnet : a neural network identifies sex from human faces. In R.P. Lippmann, J.E. Moody, and D.S. Touretzky, editors, *Advances in Neural Information Processing Systems*, 3, pages 572–577, 1991.

[81] Rafael C. Gonzalez and Paul Wintz. *Digital Image Processing.* Addison Wesley, 1977.

[82] V. Govindaraju, D.B. Sher, R. Srihari, and S.N. Srihari. Locating human faces in newspaper photographs. In *Proceedings IEEE-CS Conf. Computer Vision and Pattern Recognition*, pages 278–285, San Diego, CA, 1989.

[83] Venu Govindaraju, Sagur N. Srihari, and David B. Sher. A computational model for face location. In *Proceedings of the 3rd International Conference on Computer Vision*, pages 718–721, 1990.

[84] Hans P. Graf, Craig R. Nohl, and Jan Ben. Image segmentation with networks of variable scales. In J.E. Moody, S.J. Hanson, and R.P. Lippmann, editors, *Advances in Neural Information Processing Systems*, 4, pages 480–487, 1992.

[85] T.N.E. Greville. Some applications of the pseudo inverse of a matrix. *SIAM*, 2 :15–22, 1960.

[86] I. Guyon, P. Albrecht, Y. Le Cun, J. Denker, and W. Hubbard. Design of a neural network character recognizer for a touch terminal. *Pattern Recognition*, 24(2) :105–119, 1991.

[87] I. Guyon, V. Vapnik, B. Boser, L. Bottou, and S. A. Solla. Structural risk minimization for character recognition. In J.E. Moody, S.J. Hanson, and R.P. Lippmann, editors, *Advances in Neural Information Processing Systems*, 4, pages 471–479, 1992.

[88] J.B. Hampshire II and B.A. Pearlmutter. Equivalence proofs for multilayer perceptron classifiers and the Bayesian discriminant function. In D.S Touretzky, J.L. Elman, T.J. Sejnowski, and G.E. Hinton, editors, *Proc. of Connectionist Models Summer School*, volume 1, pages 159–172, 1991.

[89] C.C. Hand, M.R. Evans, and S.W. Ellacott. *A Neural Network Feature Detector Using a Multi-Resolution Pyramid*, chapter 4, pages 65–92. Volume 1 of Linggard et al. [140], 1992.

[90] D.J. Hand. *Discrimination and Classification.* J. Wiley & Sons, second edition, 1981.

[91] L.D Harmon. The recognition of faces. *Scientific American*, 229 :71–82, October 1973.

[92] L.D. Harmon. Automatic recognition of human face profile. In *Proc. 3rd Int. Conf. Pattern Recognition*, pages 183–188, 1976.

[93] L.D. Harmon, M.H. Khan, R. Lash, and P.F. Ramig. Machine identification of human faces. *Pattern Recognition*, 13 :97–110, 1981.

[94] L.D. Harmon, S.C. Kuo, Ramig P.F., and U. Raudkivi. Identification of human face profiles by computer. *Pattern Recognition*, 10 :301–312, 1978.

[95] Eric Hartman and James D. Keeler. Predicting the future : Advantages of semilocal units. *Neural Computation*, 3 :566–578, 1991.

[96] Eric Hartman and James D. Keeler. Semi-local units for prediction. In *Proceedings of IJCNN Seattle*, volume 2, pages 561–566, 1991.

[97] M.J.R. Healy. *Matrices for statistics.* Clarendon Press, Oxford, 1986.

[98] Donald Hebb. *Organization of behavior.* Science Edition, 1961.

[99] R. Hecht-Nielsen. Kolmogorov's mapping neural network existence theorem. In *Procs of IEEE first ICNN San Diego*, volume 3, pages 11–14, 1987.

[100] M. E. Hellman. The nearest neighbor classification rule with a reject option. *IEEE Trans. on System, Man and Cybernetics*, 6(3) :179–185, July 1970.

[101] J. Hertz, A. Krogh, and R.G. Palmer. *Introduction to the theory of neural computation*, volume 1 of *Lectures Notes of the Santa Fe Institute Studies in the Sciences of Complexities.* Addison-Wesley, 1991.

[102] J. Hopfield. Neural networks and physical systems with emergent collective computational abilities. *P.N.A.S. USA*, 79 :2554–2558, 1982.

[103] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2 :359–366, 1989.

[104] K. Hornik, M. Stinchcombe, and H. White. Universal approximation of an unknown mapping and its derivatives using multilayer feedforward networks. *Neural Networks*, 3 :551–560, 1990.

[105] *Proceedings of the IEEE*, 80(7), July 1992. Special issue on Optical Character Recognition.

[106] S. Impedovo and J.C. Simon, editors. *From Pixels to Features*, number 3 in Frontiers in Handwriting Recognition. North-Holland, 1992.

[107] Robert A. Jacobs and Geoffrey E. Hinton. Evaluation of adaptive mixtures of competing experts. In R.P. Lippmann, J.E. Moody, and D.S. Touretzky, editors, *Advances in Neural Information Processing Systems*, 3, pages 774–780, 1991.

[108] Robert A. Jacobs and Michael I. Jordan. A competitive modular connectionist architecture. In R.P. Lippmann, J.E. Moody, and D.S. Touretzky, editors, *Advances in Neural Information Processing Systems*, 3, pages 767–773, 1991.

[109] Anil K. Jain. *Fundamentals of digital image processing.* Prentice Hall, Englewood Cliff, 1989.

[110] R. D. Jones, Y. C. Lee, C. W. Barnes, C. W. Flake, K. Lee, P. S. Lewis, and S. Qian. Function approximation and time series prediction with neural networks. In *Proceedings of IJCNN Washington*, volume 1, pages 649–665, 1990.

[111] M. Kass, A. Witkin, and D. Terzopoulos. Snakes : Active contour models. In *Proc. First Int. Conf. on Computer Vision*, London, June 1987.

[112] S.N. Kavuri and V. Venkatasubramanian. Fault diagnostic using feedforward neural networks with ellipsoidal fan-in functions. In *Avignon'92 Conference on AI*, 1992.

[113] Jim Keeler and David E. Rumelhart. Integrated segmentation and recognition of hand-printed numerals. In R.P. Lippmann, J.E. Moody, and D.S. Touretzky, editors, *Advances in Neural Information Processing Systems*, 3, pages 557–563, 1991.

[114] Jim Keeler and David E. Rumelhart. A self-organizing integrated segmentation and recognition neural net. In J.E. Moody, S.J. Hanson, and R.P. Lippmann, editors, *Advances in Neural Information Processing Systems*, 4, pages 496–501, 1992.

[115] F. Kimura and M. Shridhar. Recognition of connected numeral strings. In *ICDAR*, volume 2, pages 731–739, Saint Malo, France, 1991.

[116] M. Kirby and L. Sirovich. Application of the Karhunen-Loeve procedure for the characterisation of human faces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1990.

[117] T. Kohonen. An adaptative associative memory principle. *IEEE Trans. on Computers*, April 1974.

[118] T. Kohonen. Learning Vector Quantization for pattern recognition. Technical Report TKK-F-A601, Helsinki University of Technology, Dpt. of Technical Physics, Laboratory of Computer and Information Science, 1986.

[119] T. Kohonen, G. Barna, and R. Chrisley. Statistical pattern recognition with neural networks : Benchmarking studies. In *Procs of IEEE 2nd ICNN San Diego*, volume 1, pages 61–68, 1988.

[120] T. Kohonen and M. Ruohonen. Representation of associated data by matrix operators. *IEEE Trans. on Computers*, C-22 :701–702, July 1973.

[121] Teuvo Kohonen. *Self-Organization and Associative Memory*. Springer-Verlag, third edition, 1989.

[122] Teuvo Kohonen. Statistical pattern recognition revisited. In R. Eckmiller, editor, *Advanced Neural Computers*, pages 137–144, 1990.

[123] Teuvo Kohonen, Jari Kangas, Jorma Laaksonen, and Kari Torkkola. LVQ_PAK : A program package for the correct application of Learning Vector Quantization algorithms. In *Proceedings of IJCNN Baltimore*, volume 1, pages 725–730, 1992.

[124] A. N. Kolmogorov. On the representation of continuous functions of many variables by superposition of continuous functions of one variable and addition. *Doklady Akademii Nauk SSR*, 144 :679–681, 1957.

[125] Makoto Kosugi. Robust identification of human face using mosaic pattern and BPN. In *Proceedings of the IEEE-SP 1992 Workshop*, Neural Networks For Signal Processing, pages 296–305, Denmark, 1992.

[126] K.J. Lang and G.E. Hinton. The development of TDNN architecture for speech recognition. Technical Report 152, CMU, 1988.

[127] Alan Lapedes and Robert Farmer. How neural nets work. In D.Z. Anderson, editor, *Advances in Neural Information Processing Systems*, 1, pages 442–456, 1989.

[128] Y. Le Cun. Generalization and network design strategies. In R. Pfeifer, Z. Schreter, F. Fogelman, and L. Steels, editors, *Connectionism in Persespective*, Zurich, Switzerland, 1989. Elsevier.

[129] Y. Le Cun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Handwritten digit recognition with a back-propagation network. In Touretzky, editor, *Advances in Neural Information Processing Systems*, 2, pages 396–404, 1990.

[130] Y. Le Cun, B. Boser, J.S. Denker, D. Henderson, R.E. Howard, W. Hubbard, and L.D. Jackel. Back-propagation applied to handwritten zip-code recognition. *Neural Computation*, 1(4), 1990.

[131] Y. Le Cun, J.S. Denker, and S.A. Solla. Optimal brain damage. In Touretzky, editor, *Advances in Neural Information Processing Systems*, 2, pages 598–605, 1990.

[132] Y. Le Cun, L. D. Jackel, H. P. Graf, B. Boser, J. S. Denker, I. Guyon, D. Henderson, R. E. Howard, W. Hubbard, and S. A. Solla. Optical character recognition and neural-net chips. In *Proceedings of INNC Paris*, volume 2, pages 651–655, 1990.

[133] Yann Le Cun. *Modèles Connexionnistes de l'Apprentissage*. PhD thesis, Université Paris 6, June 1987.

[134] Eric Lecollinet. *Reconnaissance d'écriture manuscrite mono-scripteur : description symboliques des mots par un graphe de primitives*. PhD thesis, Université Paris 6, June 1990.

[135] Eric Lecollinet and Jean Pierre Crettez. A grapheme-based segmentation technique for cursive script recognition. In *ICDAR*, volume 2, pages 740–748, Saint Malo, France, 1991.

[136] Dar-Shyang Lee, Sagur N. Srihari, and Roger Gaborski. Bayesian and neural network pattern recognition : a theorical connection and empirical results with handwritten characters. In *Artificial Neural Networks and Statistical Pattern Recognition*, pages 89–108. Elsevier Science, 1991.

[137] Yuchun Lee. Handwritten digit recognition using k nearest-neighbor, radial-basis function, and backpropagation neural networks. *Neural Computation*, 3 :440–449, 1991.

[138] T. K. Leen and N. Kambhatla. Fast non-linear dimension reduction. Poster at Snowbird, Neural Network for Computing Conference, April 1993. email : tleen@cse.ogi.edu.

[139] Y. Linde, A. Buzo, and R.M. Gray. An algorithm for VQ design. *IEEE Trans. on Communication*, 28 :84–95, 1980.

[140] R. Linggard, D.J. Myers, and C. Nightingale, editors. *Neural Networks for Vision, Speech and Natural Language*, volume 1 of *BT Telecommunications Series*. Chapman & Hall, 1992.

[141] B. Logan. Information in the zero-crossing of band pass signals. *Bell System Technical Journal*, 56, 1977.

[142] Jérôme Loncelle. *Contribution des réseaux connexionnistes au traitement d'image bas niveau*. PhD thesis, Université Paris XI, Nov 1990.

[143] David J.C. MacKay. Bayesian interpolation. *Neural Computation*, 4(3) :415–447, 1992.

[144] David J.C. MacKay. The evidence framework applied to classification networks. *Neural Computation*, 4(4) :720–736, 1992.

[145] J. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proc. of the fifth Berkeley Symposium on Mathematics, Statistics and Probabilities*, volume 1, pages 281–297, 1967.

[146] S. Mallat. Multiresolution representation and wavelets. Technical Report MS-CIS-88-68, University of Pennsylvania, Philadelphia, August 1988.

[147] S. Mallat. A theory for multiresolution signal decomposition : the wavelet representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2(7), july 1989.

[148] David Marr. *Vision.* Freeman, 1982.

[149] Gale L. Martin and Mosfeq Rashid. Recognizing overlapping hand-printed characters by centered-object integrated segmentation and recognition. In J.E. Moody, S.J. Hanson, and R.P. Lippmann, editors, *Advances in Neural Information Processing Systems*, 4, pages 504–511, 1992.

[150] O. Matan, J. Bromley, J.C. Burges, J.S. Denker, L.D. Jackel, Y. Le Cun, E.P.D. Pednault, W.D. Satterfield, C.E. Stenard, and T.J. Thompson. Reading handwritten digits : a ZIP code recognition system. *IEEE Computer Magazine*, 1992. (to appear).

[151] Ofer Matan, Christopher J.C. Burges, Yann Le Cun, and John S. Denker. Multi-digit recognition using a Space Displacement Neural Network. In J.E. Moody, S.J. Hanson, and R.P. Lippmann, editors, *Advances in Neural Information Processing Systems*, 4, pages 488–495, 1992.

[152] E. Mc Dermott and S. Katagiri. Shift-invariant, multi-category phoneme recognition using Kohonen's LVQ2. In *IEEE Proceedings of ICASSP*, volume S3.1, pages 81–84, Glasgow, U.K., 1989.

[153] W. McCulloch and W.L.R. Pitts. A logical calculus for the ideas immanent in nervous activity. *Bull. Math. Biophysics*, 5 :115–133, 1943.

[154] W. C. Mead, R. D. Jones, Y. C. Lee, C. W. Barnes, C. W. Flake, L. A. Lee, and M. K. O'Rourke. Using CNLS-net to predict the Mackey-Glass chaotic time series. In *Proceedings of IJCNN Seattle*, volume 2, pages 485–490, 1991.

[155] Yves Meyer. Principe d'incertitude, bases hilbertiennes et algèbres d'ope'rateurs. *Séminaire Bourbaki*, (662), 1985–1986.

[156] M. Minsky and S. Papert. *Perceptrons*. MIT Press, 1969.

[157] J. Moody and C.J. Darken. Fast learning in networks of locally-tuned processing units. *Neural Computation*, 1 :281–294, 1989.

[158] M. Mougeot, R. Azencott, and B. Angeniol. Image compression with back-propagation : Improvement of the visual restoration using different cost functions. *Neural Networks*, 4 :467–476, 1991.

[159] M. T. Musavi, W. Ahmed, K. B. Chan, and D. M. Hummels. On the training of Radial Basis Function classifiers. *Neural Networks*, 5 :595–603, 1992.

[160] M. T. Musavi, K.B. Faris, K.H. Chan, and W. Ahmed. On the implementation of RBF technique in neural networks. *ACM*, pages 110–115, 1991.

[161] S.J. Nowlan. Maximum likelyhood competitive learning. In Touretzky, editor, *Advances in Neural Information Processing Systems*, 2, 1990.

[162] Steven J. Nowlan and Geoffrey E. Hinton. Simplifying neural networks by soft weight-sharing. *Neural Computation*, 4 :473–493, 1992.

[163] J. Park and I. W. SandBerg. Universal approximation using Radial-Basis-Function networks. *Neural Computation*, 3 :246–257, 1991.

[164] R. Penrose. A generalized inverse for matrices. *Proceedings of Cambridge Philosophical Society*, 51 :406–413, 1955.

[165] P. Peretto. Collective properties of neural networks : a statistical physics approach. *Biological Cybernetics*, 50 :51–62, 1984.

[166] John L. Perry and Jeanne M. Carney. Human face recognition using a multilayer perceptron. In *Proceedings of IJCNN Washington*, volume 2, pages 413–416, 1990.

[167] E.D. Petajan. Automatic lipreading to enhance speech recognition. In *Proc. CVPR*, pages 40–47, 1985.

[168] Pierson. *Thèse*. PhD thesis, Université Paris XI, 1993. (à paraître).

[169] Matti Pietikanen and Rosenfeld Azriel. Image segmentation by texture using pyramid node linking. *IEEE Trans. on Systems, Man, and Cybernetics*, SMC-11(12) :822–825, December 1981.

[170] T. Poggio and F. Girosi. A theory of networks for approximation and learning. Technical Report 1140, MIT AI Lab, 1989.

[171] Tomaso Poggio and Federico Girosi. Networks for approximation and learning. *Proceedings of the IEEE*, 78(9) :1481–1497, September 1990.

[172] M.J.D. Powell. Radial Basis Functions for multivariate interpolation : A review. In Mason and Cox, editors, *Algorithms for Approximation*, pages 143–167, Oxford, 1987. Clarendon Press.

[173] W.H. Press, B.P. Flannery, et al. *Numerical Recipes*. Cambridge University Press, Cambridge, 1988.

[174] L.R. Rabiner and B. Gold. *Theory and application of digital signal processing*. Prentice Hall, Englewood Cliff, 1975.

[175] Sarunas J. Raudys and Anil K. Jain. Small sample size effects in statistical pattern recognition : Recommendations for practitioners. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(3) :252–264, 1991.

[176] J.R. Rice. *The Approximation of Functions*, volume 1. Addison-Wesley, Reading, MA, 1964.

[177] Michael D. Richard and Richard P. Lippman. Neural networks classifiers estimate Bayesian *a posteriori* probabilities. *Neural Computation*, 3 :461–483, 1991.

[178] F. Rosenblatt. The perceptron : a perceiving and recognizing automaton. Technical Report 85-460-1, Cornell Aeronautical Laboratory, Ithaca, N.Y., January 1957.

[179] F. Rosenblatt. *Principles of Neurodynamics : Perceptrons and the theory of brain mechanisms*. Spartan Books, Washington D.C., 1962.

[180] A. Rosenfeld and A. C. Kak. *Digital Picture Processing*. Academic Press, New York, 1982.

[181] Dennis W. Ruck, S.K. Rogers, M. Kabrisky, Oxley M.E., and Bruce W. Suter. The multilayer perceptron as an approximation to a Bayes optimal discriminant function. *IEEE Trans. on Neural Networks*, 1(4) :296–298, December 1990.

[182] W. Rudin. *Real and Complex Analysis*. McGraw Hill, New York, 1966.

[183] D.E. Rumelhart and J.L. Mc Clelland. *Parallel Distributed Processing : explorations in the microstructures of cognition*. MIT Press, 1986.

[184] Michael Sabourin and Amar Mitiche. Optical character recognition by a neural network. *Neural Networks*, 5 :843–852, 1992.

[185] Avijit Saha and D. Keeler, James. Algorithms for better representation and faster learning in Radial Basis Function networks. In Touretzky, editor, *Advances in Neural Information Processing Systems*, 2, pages 482–489, 1990.

[186] A. Samal. Minimum resolution for human face detection and identification. In *Proceedings of SPIE/SPSE Symposium on Electronic Imaging*, January 1991.

[187] H. Schwenk, P. Gallinari, and X. Driancourt. A modular system which improves the topological maps. In *Proceedings of IJCNN Baltimore*, volume 3, pages 352–357, 1992.

[188] T. Sejnowski and C.R. Rosenberg. Parallel networks that learn to pronounce english text. *Complex Systems*, 1 :145–168, 1987.

[189] T.J. Sejnowski, B.P. Yuhas, M.H. Goldstein, Jr, and R.E. Jenkins. Combining visual and acoustic speech signals with a neural network improves intelligibility. In Touretzky, editor, *Advances in Neural Information Processing Systems*, 2, pages 232–239, 1990.

[190] A.W. Senior. Off-line hanwriting recognition : A review and experiments. Technical Report 105, Cambridge University Engineering Departement, December 1992.

[191] M.A. Shackleton and W.J. Welsh. Classification of facial features for recognition. In *Proceedings of CVPR*, Hawai, 1991.

[192] Patrice Simard, Yann Le Cun, and John Denker. Efficient pattern recognition using a new transformation distance. 1992. To appear.

[193] Patrice Simard, Bernard Victorri, Yann Le Cun, and John Denker. Tangent prop – a formalism for specifying selected invariances in a adaptative network. In J.E. Moody, S.J. Hanson, and R.P. Lippmann, editors, *Advances in Neural Information Processing Systems*, 4, pages 895–903, 1992.

[194] Jean-Claude Simon. Off-line cursive word recognition. *Proceedings of the IEEE*, 80(7) :1150–1161, July 1992.

[195] Elliot Singer and Richard P. Lippman. Improved Hidden Markov Model speech recognition using Radial Basis Function networks. In J.E. Moody, S.J. Hanson, and R.P. Lippmann, editors, *Advances in Neural Information Processing Systems*, 4, pages 159–166, 1992.

[196] Elliot Singer and Richard P. Lippman. A speech recognizer using Radial Basis Function neural network in an HMM framework. In IEEE, editor, *Proceedings of the International Conference on Acoustic, Speech, and Signal Processing*, 1992.

[197] L. Sirovich and M. Kirby. Low-dimensional procedure for the characterization of human faces. *Journal Opt. Soc. Am. A*, 4(3) :519–524, March 1987.

[198] Padhraic Smyth. Probability density estimation and local basis function neural networks. In Hanson Petsche, Kearns and Rivest, editors, *Computational Learning Theory and Natural Learning Systems 2*. MIT Press, 1993.

[199] Inger Solheim, Tanya Payne, and Ralph H. Castain. Back Propagation neural networks for facial verification. Technical Report LA-12353-MS, Los Alamos National Laboratory, New Mexico, October 1992.

[200] Inger Solheim, Tanya L. Payne, and Ralph Castain. The potential in using backpropagation neural networks for facial verification systems. *SIMULATION*, 58(5) :306–310, May 1992.

[201] A.N. Tikhonov and V.Y. Arsenin. *Solutions of Ill-posed Problems*. W.H. Winston, Washington, D.C., 1977.

[202] Ya Tsypkin. *Adaptation and Learning in Automatic Systems*, volume 73 of *Mathematics in science and engineering*. Academic Press, 1971.

[203] Matthew Turk and Alex Pentland. Eigenfaces for recognition. Technical Report 154, MIT Media Lab Vision and Modeling Group, September 1990.

[204] Matthew Turk and Alex Pentland. Face processing : Models for recognition. In *Proc. of the SPIE.*, volume 1192, pages 22–32, 1990.

[205] V. Vapnik. *Estimation of Dependences Based on Empirical Data*. Springer-Verlag, 1982.

[206] Emmanuel Viennet. Application de la théorie des ondelettes pour la localisation de contours. Mémoire de DEA, Université Paris V, September 1989.

[207] Emmanuel Viennet and Francoise Fogelman Soulié. Multi-resolution scene segmentation using MLPs. In *Proceedings of IJCNN Baltimore*, volume 3, pages 55–60, 1992.

[208] Emmanuel Viennet and Francoise Fogelman Soulié. Scene segmentation using multiresolution analysis and MLP. In I. Aleksander and J. Taylor, editors, *Proceedings of ICANN'92 Brighton*, Artificial Neural Networks, pages 1599–1602. Elsevier, 1992.

[209] Emmanuel Viennet, Francoise Fogelman Soulié, and Bertrand Lamy. Multi-modular neural networks architectures for pattern recognition. Technical Report 827, Laboratoire de Recherches en Informatique, Paris XI, Orsay, March 1993.

[210] J.M. Vincent. Facial feature location in coarse resolution images by multi-layered perceptrons. In *Artificial Neural Networks*, pages 821–826. Elsevier Science, 1991.

[211] J.M. Vincent, D.J. Myers, and R.A. Hutchinson. *Image Feature Location in Multi-Resolution Images using a Hierarchy of Multilayer Perceptrons*, chapter 1, pages 13–29. Volume 1 of Linggard et al. [140], 1992.

[212] A. Waibel, T. Hanazawa, G.E. Hinton, K. Shikano, and K. Lang. Phoneme recognition : neural networks vs. Hidden Markov Models. In *IEEE Proceedings of ICASSP*, volume 1, pages 107–110, 1988.

[213] A. Waibel, T. Hanazawa, G.E. Hinton, K. Shikano, and K. Lang. Phoneme recognition using time-delay neural networks. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 37(3) :328, March 1989.

[214] J. B. Waite and W. J. Welsh. Head boundary location using snakes. *British Telecom Tech. Journal*, 8(3) :127–136, July 1990.

[215] Andreas S. Weigend, Bernardo A. Huberman, and Rumelhart David E. Predicting the future : a connectionist approach. *International Journal of Neural Systems*, 1(3) :193–209, 1990.

[216] Gérard Weisbuch. *Dynamique des systèmes complexes.* InterEditions CNRS, 1989.

[217] P. Werbos. *Beyond Regression : New Tools for Prediction and Analysis in the Behavioral Sciences.* PhD thesis, Harvard University, 1974.

[218] Dietrich Wettschereck and Thomas Dietterich. Improving the performance of Radial Basis Function networks by learning center locations. In J.E. Moody, S.J. Hanson, and R.P. Lippmann, editors, *Advances in Neural Information Processing Systems*, 4, pages 1133–1140, 1991.

[219] B. Widrow and M.E. Hoff. Adaptative switching circuits. *IRE WESCON Conv. Record, part 4*, pages 96–104, 1960.

[220] Andrew P. Witkin. Scale-space filtering. In *Proc. of the Eighth Int. Joint Conference on Artificial Intelligence*, pages 1019–1022, 1983.

[221] Jy Chyuan Wu and Jun S. Huang. Human face profile recognition by computer. *Pattern Recognition*, 23(3) :255–259, 1990.

[222] Allan L. Yuille, David S. Cohen, and Peter W. Hallinan. Feature extraction from faces using deformable templates. In *Proceedings of CVPR*, pages 104–109, San Diego C.A., June 1989.

©1993 Emmanuel Viennet

Juin 1993
Université de Paris-Sud
Centre d'Orsay - C.N.R.S. U.R.A. 410
Laboratoire de Recherche en Informatique

313

Architectures Connexionnistes Multi-Modulaires
Application à l'Analyse de Scène

©1993 Emmanuel Viennet

Juin 1993
Université de Paris-Sud
Centre d'Orsay - C.N.R.S. U.R.A. 410
Laboratoire de Recherche en Informatique