Architecture et Apprentissage d'un Système Hybride Neuro-Markovien pour la Reconnaissance de l'Écriture Manuscrite En-Ligne

Emilie Caillault

École Centrale de Nantes **Université de Nantes** **École des Mines de Nantes**

ECOLE DOCTORALE STIM

« SCIENCES ET TECHNOLOGIES DE L'INFORMATION ET DES MATERIAUX »

Année 2005

Architecture et Apprentissage d'un Système Hybride Neuro-Markovien pour la Reconnaissance de l'Écriture Manuscrite En-Ligne

---

THÈSE DE DOCTORAT

Discipline : Traitement du Signal et des Images
Spécialité : Automatique et Informatique Appliquée

Présentée et soutenue publiquement par

Émilie POISSON (épouse CAILLAULT)

le 12 décembre 2005, devant le jury ci-dessous

Rapporteurs :	M. Abdel BELAID	Professeur, Université Nancy 2
	M. Laurent HEUTTE	Professeur, Université de Rouen
Examinateurs :	M. Pierre-Michel LALLICAN	Ingénieur Recherche, Vision Objects
	M. Guy LORETTE	Professeur, Université Rennes 1
	M. Christian VIARD-GAUDIN	Maître de Conférences, HDR, IUT Nantes

Directeur de thèse : Christian VIARD-GAUDIN
Laboratoire IRCCyN
Adresse :Ecole polytechnique de l'université de Nantes
Rue Christian Pauc BP 50609
44306 NANTES cedex 3

N° ED 366-231

A Laurent et Pierre…

Remerciements

Je tiens tout d'abord à exprimer ma plus profonde estime à mon directeur de thèse, Christian Viard-Gaudin qui a accepté de diriger mes travaux et m'a introduite dans le monde de la recherche. Qu'il soit assuré de ma reconnaissance pour son implication, son soutien scientifique et humain et ses relectures efficaces des différents écrits.

Au commencement de mes travaux de recherche, je n'avais aucune connaissance et compétence dans les domaines de la classification et de la reconnaissance de l'écriture manuscrite. Je remercie Christian Viard-Gaudin, Pierre-Michel Lallican et Stefan Knerr pour m'avoir initiée à ces domaines et apporté leurs compétences et précieux conseils.

Je remercie très sincèrement les Professeurs Abdel Belaïd et Laurent Heutte pour avoir accepté d'examiner mon travail, et de rapporter ce manuscrit.
Je remercie Monsieur Guy Lorette, Professeur à l'Université de Rennes et Pierre-Michel Lallican, Responsable Recherche et Développement de la société Vision Objects, pour avoir accepté d'être membre de mon jury.

Je remercie tous mes collègues et amis - de l'équipe Image et VidéoCommunication, de l'IRCCyN, du CIES Grand Ouest et de l'École polytechnique de l'université de Nantes.

À mes parents, ma sœur pour leur confiance et leur soutien.
À Laurent et Pierre, les soleils de ces journées d'écriture.

Et un grand MERCI à tous ceux qui m'ont aidée
à concilier la thèse et la vie de maman.

TABLES DES MATIERES

LISTE DES FIGURES

LISTE DES ALGORITHMES

LISTE DES TABLEAUX

INTRODUCTION GENERALE

Même si les ordinateurs envahissent le monde de la communication, l'écrit et la parole demeurent deux modalités de communications privilégiées. A un horizon plus ou moins éloigné, nombreux travaux et prospectives cherchent à faire disparaître l'ordinateur de l'environnement de l'utilisateur. Néanmoins, celui-ci sans qu'il en ressente les contraintes sera connecté aux systèmes d'information par des interfaces naturelles : un geste de la main, un regard, la parole… et bien entendu l'écriture manuscrite. Du vieux concept utopique du bureau sans papier, on arrive sur le paradigme du bureau sans ordinateur perceptible.

Aujourd'hui, c'est encore bien souvent l'humain qui fait les efforts d'adaptation. Les mini-claviers des téléphones portables en sont un exemple. Leur ergonomie est assez limitée dès lors que l'on cherche, par exemple, à composer un petit message. Pourtant, la tendance est nette ; des progrès importants ont été réalisés pour rapprocher le monde du numérique et celui de l'écriture manuscrite. L'apparition des assistants personnels (PDA), des tablettes PC, des smart-phones, ces téléphones dernières générations qui combinent une multitude de fonctionnalités - agendas personnels, bloc-notes, jeux vidéos, appareil photo et caméra… - confirme cette situation. Des progrès très significatifs ont également été enregistrés dans le confort d'usage des stylos digitaux. Ils sont devenus aujourd'hui très comparables aux stylos usuels : leur poids et leur volume ont été divisés par deux en trois ans. Il est ainsi possible d'écrire avec un stylo quasi-ordinaire, sur du papier, et de traiter en temps-réel cette information éventuellement n'importe où sur la planète. En parallèle à tous ces progrès matériels, il reste à mieux maîtriser l'interprétation de ces traces écrites qui vont proliférer sur de multiples dispositifs. C'est dans le cadre de ces applications de saisie d'écriture en-ligne à capacités limitées que s'inscrivent nos travaux.

Les principales applications aujourd'hui imposent des contraintes fortes à l'utilisateur pour restreindre le taux d'erreur à un seuil acceptable par celui-ci. La contrainte principale étant le style d'écriture, nombreux systèmes imposent une écriture détachée, voire même un tracé particulier des caractères, ce qui est loin d'être naturel dans le cas d'une saisie de mots ou phrases. L'objectif de cette thèse est de contribuer à progresser dans le difficile problème de la reconnaissance de l'écriture manuscrite non contrainte en-ligne.

Pour diminuer les erreurs de reconnaissance, certains systèmes imposent une phase d'initialisation ou d'adaptation au scripteur. Or cette contrainte est souvent mal vécue par l'utilisateur qui souhaite utiliser son matériel efficacement dès la première seconde où il l'a en possession. Pour cela, il convient de construire un système omniscripteur performant originellement.

Nous présentons dans cette thèse deux moteurs de reconnaissance en-ligne omniscripteur, l'un pour les caractères manuscrits isolés, le second pour les mots cursifs. Pour cela, nous préconisons une approche s'appuyant sur des méthodes statistiques combinant des réseaux de neurones et des modèles de Markov cachés. Les premiers sont utiles pour leur fort pouvoir discriminant intrinsèque permettant de détecter des caractéristiques spécifiques (caractère par exemple, ou à un autre niveau, mots ou corps de mots,...). Les seconds ont une très bonne aptitude à modéliser les processus stochastiques tels que ceux conduisant aux séquences produites par de l'écriture manuscrite. L'apport des travaux se fera à la fois sur l'étude, la conception et la mise en œuvre d'un type spécifique de réseaux de neurones aux propriétés très intéressantes : les réseaux à convolution. Un des intérêts de leur structure réside dans l'intégration de l'étape toujours délicate de la définition des caractéristiques de la forme à

reconnaître. Ainsi, dans une certaine mesure, c'est le réseau qui fabrique ses propres caractéristiques. Toutefois, les difficultés du problème ne sont que déplacées, il s'agit alors d'étudier l'influence des contraintes sur la topologie retenue (choix du nombre de niveaux de représentation : du pixel au symbolique ; sur le partage des connexions,...). Les réseaux de neurones étant des boîtes noires difficiles à interpréter et paramétrer, nous pouvons craindre que les exigences en termes d'architecture et performances soient incompatibles avec les applications visées.

Cette thèse se décompose en trois parties : une partie introductive présentant le contexte de l'écriture manuscrite cursive en-ligne (chapitres 1 à 3) puis deux parties couvrant les axes majeurs de nos travaux de recherche, l'une sur les réseaux à convolution et leur couplage appliqués à la reconnaissance de caractères manuscrits isolés (chapitres 4 à 5), l'autre sur la reconnaissance de mots cursifs en-ligne (chapitres 6 à 8).

Dans la première partie, nous avons introduit la problématique de la reconnaissance de l'écriture cursive et plus particulièrement de l'écriture cursive en-ligne.

Le chapitre 1 présente les deux grandes classes de l'écrit et leurs différents thèmes de recherche. Nous définirons les caractéristiques de l'écriture naturelle et nous décrirons les diverses approches et classifieurs associées pour la reconnaître.

Le chapitre 2 est dédié à des classifieurs particuliers : les réseaux de neurones à convolution, qui nous serviront d'élément de base dans les systèmes développés de reconnaissances aussi bien pour les caractères isolés que pour les mots. Nous introduirons dans un premier temps quelques propriétés sur les réseaux de neurones et les architectures classiques rencontrées dans la littérature. Puis nous détaillerons l'architecture de base : le perceptron multi-couches pour ensuite bien comprendre les fondements des réseaux à convolution.

Le chapitre 3 est consacré aux différents schémas de couplage neuro-markovien proposés dans la littérature. Introduit en reconnaissance de la parole, ils ont ensuite été étendus au domaine de la reconnaissance de l'écriture, principalement hors-ligne. L'objectif du chapitre est de présenter dans un premier temps le mécanisme et l'intérêt d'une combinaison RNA-MMC, Réseau de Neurones Artificiels et Modèles de Markov Cachés puis dans un second temps les méthodes d'apprentissage de ces systèmes hybrides.

La partie II est consacrée à la reconnaissance de caractères isolés, brique de base de notre reconnaisseur mot. Nous détaillerons les différents systèmes, que nous avons implémentés, basés sur les réseaux de neurones à convolution et montrerons l'intérêt de ces architectures pour des applications à ressources limitées.

Le chapitre 4 détaille un premier système de reconnaissance de caractères en-ligne, basé sur une architecture TDNN, réseau de neurones à décalage temporel (Time Delay Neural Network). Nous analyserons l'impact de cette topologie particulière sur deux benchmarks de référence, les bases UNIPEN et IRONOFF [Poisson, 2001].

Le chapitre 5 présente une évolution de ce système de reconnaissance de caractères en-ligne basée sur l'exploitation uniquement de l'information temporelle de l'écriture. Dans le but d'accroître les taux de reconnaissance, nous avons cherché comment il était possible d'apporter au système initial de l'information complémentaire telle que la notion picturale de l'écriture [Poisson, 2002b]. La première étape a été d'analyser un réseau de neurones dual au TDNN, le

SDNN, réseau à décalage spatial (Space Displacement Neural Network) très adapté dans le cadre d'une reconnaissance de caractères statiques. La seconde étape est d'analyser les performances croisées du TDNN et du SDNN afin de démontrer l'intérêt de coupler de l'information temporelle et spatiale par une classification multiple. Une étude de la combinaison de ces deux réseaux est présentée et argumentée par différentes expérimentations [Poisson, 2002].

La partie III est l'extension au niveau mot.

Le chapitre 6 décrit en détails le système neuro-markovien que nous avons implémenté pour la reconnaissance de mots manuscrits en-ligne non contraints [Poisson, 2004]. Ce chapitre reprend les parties principales du système : normalisation de l'écriture pour une reconnaissance omni-scipteurs, extraction de caractéristiques, classification au niveau caractère et l'étage d'alignement temporel pilotés par des modèles de Markov cachés pour chacun des mots du lexique.

Le chapitre 7 correspond à l'analyse du processus d'apprentissage unifié du système hybride neuro-markovien. Nous présentons dans un premier temps le schéma général de l'apprentissage d'un tel hybride. Nous définirons un critère générique d'optimisation des paramètres du réseau de neurones à délai sans initialisation préalable au niveau caractère. Nous déclinerons les paramètres de ce critère dans le but de comparer les méthodes classiques de type maximisation de vraisemblance (ML : Maximum Likelihood), maximum d'information mutuelle (MMI : Maximum Mutual Information) et la méthode proposée. Nous comparerons les propriétés et résultats des différentes méthodes d'apprentissage [Caillault, 2005].

Le chapitre 8 présente deux voies d'optimisation du système de reconnaissance de mots que nous avons explorées et expérimentées. La première cherche à adapter au mieux la modélisation markovienne utilisée à l'écriture naturelle en introduisant une architecture TDNN multi-états [Caillault, 2005b]. La seconde vise à accroître la robustesse du système dans son processus de segmentation implicite par une initialisation de l'apprentissage avec un modèle de durée.

PARTIE I :
ÉTAT DE L'ART

1 LA RECONNAISSANCE DE L'ECRITURE MANUSCRITE

L'objet de ce chapitre est de décrire et faire le bilan des différentes approches utilisées dans le domaine de la reconnaissance de l'écriture manuscrite et des spécificités relatives à chacune de ces approches.

Nous aborderons dans un premier temps les deux grandes classes de l'écrit et leurs différents thèmes de recherche. Puis nous nous intéresserons aux différentes formes de l'écriture et leur incidence dans l'étape de reconnaissance. Enfin, nous nous focaliserons sur l'information utile portée par un signal d'écriture dans un but de reconnaissance et nous décrirons les diverses approches utilisées en reconnaissance dans le but d'introduire pour les chapitres suivants des classifieurs particuliers tels que les réseaux de neurones à convolution et les systèmes neuro-markoviens.

1.1 L'écriture manuscrite : définitions et domaines

1.1.1 Définitions

L'encadré ci-après donne les définitions séparées de chacun des termes du sujet de préoccupation « l'écriture manuscrite ». Ces définitions sont extraites du dictionnaire[1] de l'Académie, neuvième édition. Pour le vocable « écriture », nos travaux sont en relation avec les définitions 3 et 4, tandis que la première acception dans la définition du terme « manuscrit » convient parfaitement à nos travaux.

ÉCRITURE n. f. XI^e^ siècle, *scripture.* Du latin classique *scriptura,* « écriture, écrit, ouvrage ».
1. Représentation de la langue parlée par des signes graphiques. *L'écriture est un code de communication. Les origines de l'écriture. Les évolutions que l'écriture a connues au cours des millénaires.* **2.** Chacun des systèmes particuliers de signes employés pour cette représentation. *Écriture idéographique, pictographique, cunéiforme. Écriture alphabétique, écriture phonétique. L'écriture grecque, latine, cyrillique, gothique.* Spécialt. *Écriture chiffrée,* écriture secrète, qu'on ne peut comprendre que si l'on connaît son code particulier. **3.** Trace matérielle du fait d'écrire. *Une écriture indélébile. Une écriture qui s'efface.* **4.** Manière de tracer ces signes, ces caractères. *Une belle écriture. Une écriture droite, penchée, lisible, fine. Écriture ronde, bâtarde. Un expert en écritures. Les graphologues tentent d'analyser le caractère d'une personne par l'étude de son écriture.* Expr. fam. *Écriture en pattes de mouche,* écriture très menue.
5. Action d'écrire, rédaction. *L'écriture de ce billet lui a pris quelques minutes. L'écriture de cette thèse lui a demandé plusieurs années.* Spécialt. Action d'écrire une œuvre littéraire. *Se remettre à l'écriture. Se consacrer à l'écriture. L'écriture de ce roman lui a pris plusieurs années.* **6.** Manière d'écrire. *Un roman d'une écriture savante, relâchée.* Style caractéristique d'un écrivain. *L'écriture de Marcel Proust, de Louis-Ferdinand Céline.* (On préfèrera *Manière* ou *Style.*) S'applique, par ext., à une œuvre, à une école littéraire. *Écriture artiste,* style recherché, ayant pour ambition d'exprimer la complexité de la nature et de la vie, et s'écartant par principe de la langue usuelle. *L'écriture artiste des Goncourt. Écriture automatique,* procédé de création littéraire visant à traduire, sans passer par la

[1] http://www.academie-francaise.fr/dictionnaire/ Site web de l'académie française et de ses dictionnaires en ligne.

conscience, l'activité spontanée de l'esprit. *L'écriture automatique des surréalistes.* Par anal. et litt. BX-ARTS. Graphisme caractéristique d'un artiste. *L'écriture du Gréco.* - MUS. *L'écriture d'un musicien,* le style qui le distingue des autres. Ces emplois sont déconseillés. **7.** DROIT. Acte ou document constituant une preuve. *Écriture privée, écriture publique,* émanant soit de particuliers, soit d'officiers publics. Au pluriel. *Écritures,* actes de procédure nécessaires à la tenue d'un procès. Expr. vieillie. *Concilier les écritures,* s'efforcer d'accorder des pièces contradictoires. **8.** Au pluriel. COMMERCE. Enregistrement d'une opération dans les livres comptables d'une entreprise. Par méton. Livres de comptes, registres. *Tenir les écritures.* Par ext. Vieilli. Travaux mineurs de copie et de correspondance. *Un commis aux écritures.* **9.** Avec une majuscule. Ensemble des textes de l'Ancien et du Nouveau Testament. *L'Écriture. L'Écriture sainte. Les Saintes Écritures* ou, simplement, *les Écritures.*

(1)**MANUSCRIT, -ITE** adj. et n. XVI[e] siècle. Emprunté du latin *manu scriptus,* proprement « écrit à la main ».
1. Adj. Qui est écrit à la main, par opposition à ce qui est imprimé, dactylographié, etc. *Pièce, copie, lettre manuscrite. Des notes manuscrites.* **2.** N. m. Texte original ou copie d'un ouvrage écrit à la main. *Il a remis le manuscrit de son roman à l'éditeur. J'ai lu cette pièce en manuscrit, sur manuscrit. Établir un texte en se reportant au manuscrit. Un manuscrit corrigé de la main de l'auteur.* Par ext. Se dit aussi aujourd'hui, par un usage abusif et courant, d'un ouvrage dactylographié ou enregistré sur un support informatique, par opposition à la version imprimée. Se dit plus particulièrement de certains textes ou écrits remarquables par leur ancienneté, leur rareté, leur support, leur objet, etc., et qui sont généralement antérieurs à l'invention de l'imprimerie. *Manuscrit sur papyrus, sur parchemin, sur vélin. Manuscrits grecs, arabes. Un manuscrit à peintures, orné d'enluminures. Dans une bibliothèque, le département des manuscrits.* Spécialt. *Les manuscrits de la mer Morte,* ensemble de textes religieux, écrits en hébreu, en araméen, pour certains en grec, découverts à partir de 1947, en Jordanie. Titre célèbre : *Manuscrit trouvé à Saragosse,* de Jan Potocki (publié entre 1804 et 1814).

Figure 1. Définition des mots écriture et manuscrit

L'*écriture* est un moyen de communication encore essentiel et apprécié de nos jours dans nos civilisations. Elle peut être sous différents styles, imprimée ou *manuscrite*. Ces styles d'écriture diffèrent par le support sur lequel on trace les signes graphiques (pierre, papier, tablette digitale, ordinateur, etc.) et le moyen pour les tracer (poinçon, stylet, stylo, plume, clavier, machine à écrire, etc.). Même si l'imprimé prend une part grandissante dans diverses applications pour des raisons de lisibilité, stockage et sécurité, facilité et rapidité de communication, l'écrit reste encore très convoité que ce soit pour sa production naturelle qui nous a été enseignée ou pour son utilité dans le diagnostic de certaines maladies (parkinson, dyslexie, dyspraxie) ou d'autres tâches comme en psychologie expérimentale, dans l'éducation, l'expression des sentiments… Il en résulte que les travaux de recherche sur l'imprimé ou le manuscrit sont généralement bien séparés par leur nature. En effet l'imprimé est généralement codé selon une fonte particulière qui rend plus facile le développement des systèmes automatiques de traitement [Belaïd,1994].

Concernant l'écriture manuscrite, son traitement dans le cadre d'un système (semi-) automatique soulève de très nombreux problèmes, et même si des avancées significatives ont été enregistrées ces deux dernières décennies [Plamondon, 2000 ; Koerich, 2003], toutes les difficultés ne sont pas encore surmontées. Une des premières sources des difficultés touche à la très grande variabilité à la fois inter et intra scripteur. Il en résulte qu'un même caractère, a fortiori un même mot, n'est jamais reproduit à l'identique. Les solutions proposées vont alors imposer des contraintes pour pallier ces difficultés. Ces contraintes vont porter sur la langue traitée, le lexique et l'alphabet supportés, le style d'écriture autorisé (script, cursif, mixte), le nombre admissible d'utilisateurs (système mono, multi, omni scripteur) et bien entendu sur le type des tâches effectuées. Les tâches les plus basiques vont correspondre à la reconnaissance

de caractères isolés (digit, minuscules, majuscules), pour s'étendre à la reconnaissance de mots, et ensuite de phrases.

En complément des tâches de reconnaissance, sur lesquelles nos travaux ont portées, l'écriture manuscrite peut aussi se prêter à d'autres traitements. Il est ainsi envisageable d'identifier ou de vérifier l'identité d'un scripteur [Bensefia, 2004], de déterminer des caractéristiques biologiques (age, sexe) [Bandi, 2005] , d'aider à l'apprentissage de l'écriture [Teulings, 2005] ou à l'évaluation et/ou la correction de problèmes grapho-moteurs [Swett, 2005], de rechercher des mots-clés dans un texte dans une optique d'indexation.

1.1.2 Deux secteurs

La reconnaissance de l'écriture manuscrite est actuellement un domaine de recherche très actif où l'on distingue communément deux secteurs d'applications en reconnaissance de l'écriture manuscrite suivant le mode d'acquisition ou de saisie de l'écriture :

- La *reconnaissance dite dynamique ou temps-réel, ou encore en-ligne* (traduction anglaise : *on-line*). Le texte est reconnu par l'ordinateur à partir de la trajectoire du stylo décrite par une suite de points.
- La *reconnaissance dite statique ou encore hors-ligne* (traduction anglaise : *off-line*). Le texte a été écrit sur du papier et doit être reconnu par l'ordinateur à partir de son image.

1.1.2.1 L'écriture en-ligne

L'utilisateur écrit naturellement à l'aide d'un stylet sur une ardoise ou écran ou d'un stylo digital. Le logiciel de reconnaissance interprète les caractères ou les mots écrits pour les transformer en caractères numériques (type ASCII).

Trois propriétés caractérisent la reconnaissance en-ligne : la notion d'ordre d'écriture (enchaînement temporel de traits), la dynamique du tracé (vitesse, accélération, lever du stylo) et squelette du tracé (aucune épaisseur de trait).

Figure 2. Systèmes d'acquisition d'écriture en-ligne

La Figure 2 présente différents outils et supports d'acquisition. Les premiers logiciels de reconnaissance manuscrite commercialisées ont été intégrés à des organiseurs électroniques muni d'un stylet permettant la saisie de caractères voire de mots et phrases aujourd'hui et parfois muni de clavier. En dehors de ce marché important des petits assistants personnels, d'autres applications se sont développées à partir de tablette graphique et récemment des stylos digitaux :

- Dans le milieu médical, pour la saisie et stockage d'informations auprès du lit des malades[2], pour la saisie d'ordonnance médicale[3].
- Dans le monde de l'éducation, pour seconder l'enseignant dans sa tâche d'apprentissage de l'écriture qui ne peut surveiller qu'un enfant à la fois pendant son geste de production d'écriture [Teulings et al., 2005].
- Autant dans le milieu scolaire que médical, pour détecter rapidement les différentes causes liées aux troubles psychiques et moteurs (Parkinson, Sclérose…) et aux échecs scolaires (dyslexie, dyspraxie).
- Et dans l'univers des réunions, avec la possibilité de prise de notes, annotations, conservation de toutes traces écrites et orales [Liwicki, 2005] des différents intervenants de la réunion.

1.1.2.2 L'écriture hors-ligne

L'autre cadre applicatif de la reconnaissance de l'écriture manuscrite est la reconnaissance « hors-ligne », la reconnaissance a lieu après l'étape de numérisation du document papier et n'a pas forcément un objectif de traitement temps réels comme en reconnaissance en-ligne. Une difficulté importante est d'isoler l'écrit du reste du document qui peut être bruité par le système de numérisation mais aussi par son contexte. Le logiciel doit dans l'image bidimensionnelle séparer les mots avant de les reconnaître. Les difficultés sont loin d'être entièrement résolues et les logiciels qui fonctionnent actuellement sont encore très ciblés sur des applications précises telles que :

- La lecture automatique de documents manuscrits : lecture d'adresse postale manuscrite et tri automatique du courrier [Jarousse, 1998], lecture des montants de chèques [Gorski, 2001] …
- La reconnaissance et traitement automatique d'informations manuscrites précasées : analyse de formulaires [Héroux, 1998 ; Coüasnon, 2002] pour les grandes administrations principalement (INSEE, Caisses d'allocations familiales, Trésor Public, URSSAF…).
- La numérisation des collections patrimoniales, des archives, documents anciens, comme par exemple le projets DEBORA (pour Digital AccEss to Books of the RenAissance) dont l'objectif est de concevoir un ensemble d'outils permettant l'accès distant et collaboratif à des livres numérisés du XVIème siècle [Bouché, 2000] et BOVARY[4], programme de numérisation et d'édition hypertextuelle de l'ensemble du corpus des

[2] http://h41111.www4.hp.com/gomobile/fr/fr/customerstories/success/keh.html, article « Le Tablette PC HP fait de l'hôpital KEH un établissement sans fil ».
[3] http://www.visionobjects.com/IMG/pdf/CS_Medimediapro2.pdf, présentation d'Oedipsystem, application pour les docteurs qui permet d'écrire et de numériser leur ordonnance avec un papier et un stylo optique.
[4] Site du projet BOVARY : http://www.univ-rouen.fr/psi/BOVARY/presentation.htm

brouillons et manuscrits de l'œuvre *Madame Bovary* de Gustave Flaubert [Nicolas, 2003].

- La recherche d'information dans une base de documents manuscrits telle que l'identification du scripteur [Bensefia, 2003].

Ces logiciels de reconnaissance ont un coût très lourd en terme de puissance de calcul, mémoire et financier et ne s'appliquent pas aux particuliers ou petites infrastructures contrairement aux logiciels de reconnaissance en-ligne destinés davantage à des applications légères et à des ordinateurs, agendas de poche. Nous travaillerons majoritairement dans cet axe c'est-à-dire celui des applications ayant un bon compromis performance/taille du système.

1.1.2.3 Reconnaissance

La principale différence entre les deux domaines de l'écriture manuscrite en-ligne et hors-ligne, souvent traités séparément, réside dans la nature des données à traiter (temporelles ou spatiales) et de l'information pertinente que l'on peut extraire dans un but de reconnaissance. Les schémas de reconnaissance sont pour autant globalement communs et se déclinent en trois notions clés : les prétraitements, la reconnaissance, les post-traitements.

Les prétraitements concernent l'acquisition et la normalisation des données. Ils servent à supprimer les bruits induits par le contexte lui-même lors de l'acquisition (fréquence d'échantillonnage de la tablette graphique, qualité de numérisation du scanner, extraction du texte dans un document) et ceux générés par l'humain [Bengio, 1994]. Dans une application mono utilisateur on cherchera à rendre les données indépendantes des variabilités de l'utilisateur pendant sa production d'écriture : vitesse, forme. Dans une application omni scripteur, le système devra être à la fois indépendant des variabilités inter et intra scripteur.

La reconnaissance peut-être décomposée en deux parties : la segmentation et la reconnaissance propre des données. La partie segmentation consiste à découper les données en sous-éléments comme des lettres dans un mot, ou des mots dans un texte qui seront alors identifiés et étiquetés.

Les post-traitements permettent de simplifier ou corriger l'étape précédente par l'apport de connaissances lexicales, grammaticales et sémantiques. Les connaissances lexicales sont généralement les plus utilisées dans tout système de reconnaissance d'écriture. Ils permettent de vérifier la présence d'un mot dans un dictionnaire, cette vérification peut s'avérer coûteuse en fonction de la taille du lexique choisi. De nombreux travaux ont porté sur l'optimisation de cette opération : accès au lexique [Côté, 1997], architecture du lexique, réduction du lexique [Gilloux,1998], recherche de distance entre le mot observé et chacun des modèles mot du lexique [Carbonnel, 2004]. L'introduction de modèles de langages permet significativement d'améliorer les taux de reconnaissance de l'écriture manuscrite. Cet axe est relativement récent dans le domaine de l'écrit [Perraud, 2003] et se base sur des travaux antérieurs en reconnaissance de la parole.

Ces principales étapes formant la reconnaissance de l'écrit ont des déclinaisons particulières qui dépendent du contexte et principalement du style d'écriture que nous allons décrire par la suite. Nous constaterons en particulier la complexité de la séparation segmentation/labellisation.

1.2 Styles d'écriture et schémas de reconnaissance

L'écriture manuscrite est une expression libre propre à chaque personne même si certaines règles de production lui ont été enseignées. Les difficultés rencontrées par les logiciels pour reconnaître de l'écriture manuscrite tiennent d'une part de la grande variété de tracés possibles des mêmes caractères et d'autre part de l'enchaînement des caractères à l'intérieur du mot. Dans la suite du document nous nous focaliserons davantage sur la reconnaissance du domaine de l'écriture latine en-ligne qui nous intéresse.

1.2.1 Type d'écriture manuscrite reconnue.

Une grande majorité des applications disponibles actuellement sur le marché imposent une écriture « non naturelle » pour l'utilisateur : caractères spécifiques (Graffiti[5], SmARTtWriter[6]) , écriture en lettre capitale, écriture détachée (Jot[7]). Les systèmes de reconnaissance sans contrainte sont encore relativement restreints : on peut citer parmi les plus répandus Microsoft Transcriber[8], MyScript[9], Remus[10]. La **Erreur ! Source du renvoi introuvable.** illustre deux alphabets de production graphique contrainte.

Figure 3. Exemples d'écriture contrainte

On distingue principalement trois types d'écriture : le script, le cursif et le type mixte.

Le script caractérise l'écriture d'un mot en lettres séparées. Sa reconnaissance est ainsi largement simplifiée car une segmentation physique existe. La segmentation du mot en lettres se résume grossièrement à la détection des espaces, en correspondance avec l'absence de poser de stylet pendant un temps fixé et une reconnaissance basée caractères.

Le cursif correspond à un mot où toutes les lettres sont attachées. L'écriture cursive est dite naturelle, courante et rapide par opposition à l'écriture calligraphiée.

[5] http://www.escande.org/palm/GrfAnywhere/index_fr.html, logiciel libre pour des applications Palm.
[6] http://www.artcomp.com/smartwriter.htm, site officiel du logiciel SmartWriter (ART).
[7] http://www.cic.com/products/jot/, site officiel du logiciel Jot (CIC).
[8] http://www.microsoft.com/windowsmobile/downloads/transcriber.mspx, Microsoft transcriber.
[9] http://www.visionobjects.com/, site officiel de MyScript (Vision Objects).
[10] http://www-connex.lip6.fr/~lifchitz/Remus/, présentation de Remus, LIP6

Le type mixte est une combinaison des deux types précédents.

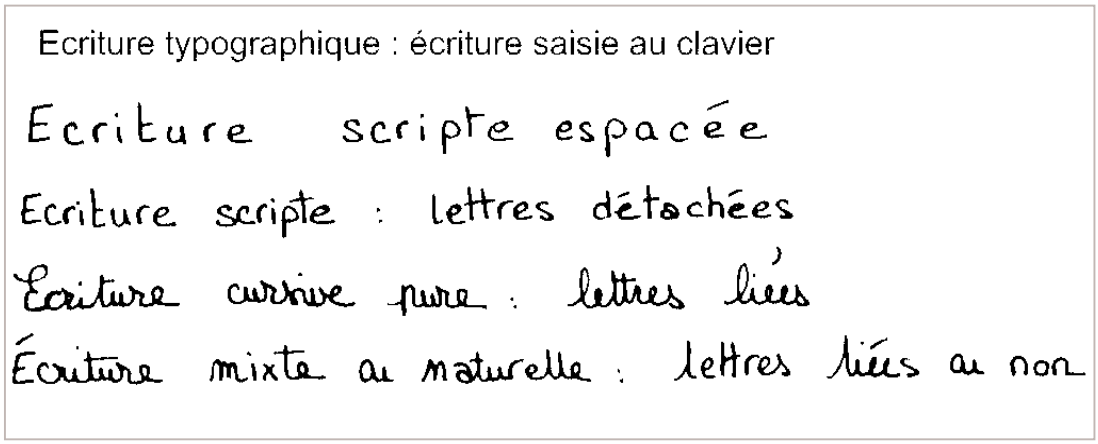


Figure 4. Catégories et Formes de l'écriture

L'écriture cursive présente l'avantage d'une liaison favorisant la fluidité et la rapidité du geste. Elle aide à la perception de l'écriture d'un mot comme un tout. L'enfant qui écrit directement en cursif perçoit les lettres comme un enchaînement : écrire en attaché. L'enfant qui a d'abord écrit en script puis en cursive perçoit les lettres comme un ensemble de tracés écrits puis attachés (2 gestes). Cette méthode peut entraîner une écriture cursive non fluide, saccadée due à une coupure dans le geste de communication, ralentissement de la production graphique et un risque de freinage dans la perception d'un mot comme un tout. L'apprentissage et la reconnaissance du cursif ou du type mixte sont beaucoup plus complexes que dans le cas du script et peuvent être décrits de la même façon que pour l'enfant soit d'une manière globale soit par une approche analytique.

1.2.2 Mode de reconnaissance

Le choix entre les différentes approches est fortement dépendant de l'application où s'exerce la reconnaissance de l'écriture manuscrite. Deux paramètres de l'application sont à prendre en compte pour ce choix : la taille du vocabulaire propre à l'application, et le nombre de scripteurs. La première approche consiste en une reconnaissance globale du mot, elle est davantage adaptée aux lexiques de taille restreinte. La deuxième approche correspond à une décomposition du mot en lettres et la reconnaissance du mot via une identification des lettres qui le composent. Elle est beaucoup plus adaptée pour une application multi et omni scripteurs et dans les cadres à vocabulaires étendus.

1.2.2.1 Approche globale

L'approche globale est dédiée principalement à la reconnaissance de lettres isolées, ou de mots isolés. Son principe est de décrire l'entité par un modèle global unique et indivisible. Ainsi un mot sera décrit par un ensemble de caractéristiques de taille fixe ou variable ne faisant pas référence explicitement à un modèle de lettres. Ce type d'approche est plus largement utilisé dans le domaine hors-ligne [Leroux, 1991 ; Gilloux, 1996 ; Guillevic, 1997] que dans le domaine en-ligne, notamment pour la robustesse de l'approche face aux bruits et aux déformations de l'écriture et les applications spécifiques telles que le traitement des chèques.

Pour la reconnaissance de caractères isolés en-ligne, différentes voies ont été explorées comme les techniques de mise en correspondance [Connell, 2001 ; Bahlmann, 2001] - alignement temporel, appariement élastique (elastic matching en anglais), distance entre

caractéristiques et des approches statistiques comme les k-plus proches voisins (k-ppv, [Schwenk, 1996]), les réseaux de neurones (RN, [Guyon, 1991 ; Lyon,1996 ; Poisson, 2001]) ou les machines à vecteur support (SVM, séparatrice à vastes marges, en anglais pour Support Vector Machine, [Vuurpijl, 2000 ; Ahmad, 2004]). Nous reviendrons dans la section suivante aux fondements des différents classifieurs utilisés.

Concernant la reconnaissance de mots manuscrits, une grande partie des travaux portent sur des modèles paramétriques tels que les modèles de Markov cachés (MMC, HMM en anglais pour Hidden Markov Models). Il s'agit en fait de calculer pour tous les mots du lexique un score de vraisemblance de la chaîne observée par rapport au modèle. Ceci reste concevable en temps-réel uniquement dans le cadre d'un vocabulaire restreint et figé ou en utilisant des approximations conduisant à des solutions sous-optimales.

1.2.2.2 Approche analytique

Contrairement aux approches globales où on ne peut pas reconnaître de mot non modélisé, l'approche analytique donne une plus grande souplesse et généricité au système reconnaissance de l'écriture manuscrite en-ligne (et hors-ligne). Elle s'appuie sur une description plus fine souvent liée à la structure alphabétique du vocabulaire : modèles lettres, modèles mots ou de phrases. Ainsi un modèle mot sera construit par concaténation des modèles lettres qui le composent. Une lettre pourra elle-même être décomposée en sous entités, graphèmes ou début, corps et fin de lettre.

Un nombre important de systèmes de reconnaissance en-ligne et hors-ligne sont basés sur cette approche et peuvent se distinguer par leur mode de localisation des caractères dans un mot. Le dilemme de Sayre [Sayre, 1973] introduit la problématique des approches analytiques : « Pour entraîner les modèles de lettres il faut pouvoir localiser ces dernières, et pour les localiser il faut avoir appris les modèles des lettres ». On distingue trois voies de localisation : la reconnaissance sans segmentation, la reconnaissance basée segmentation implicite et la reconnaissance basée segmentation explicite.

Reconnaissance sans segmentation

Ce modèle s'appuie sur une détection de la présence d'une lettre, et donc sur une modélisation statistique des 26 lettres de l'alphabet indépendamment de la segmentation. Pour cela, il est nécessaire d'utiliser des éléments invariants des traits caractéristiques élémentaires dans le but d'une comparaison ou discrimination de chaque classe. La méthode appelée k-ppv pour « k plus proches voisins » est une des méthodes les plus élémentaires. Elle consiste simplement à mesurer une distance entre les données à reconnaître et tous les exemples présentés dans la base d'apprentissage afin d'en déterminer les k plus proches. Il suffit alors de déterminer à quelle classe appartient l'objet en question. Dans les systèmes en-ligne, on peut noter la technique de Schwenk et Milgram qui utilise une série de réseau de neurones auto associatif Diabolo [Schwenk, 1998] pour chaque caractère. La particularité de ce réseau réside dans un nombre de sorties identiques au nombre d'entrées, le but étant alors d'obtenir une sortie la plus proche possible de l'entrée dans le cas d'une donnée appartenant à la bonne classe. Stéphane Gentric a étendu ces travaux et présenté un système de reconnaissance de mots manuscrits cursifs basé sur une technique "par modèles" et sur un apprentissage simultané de la segmentation en caractères et des modèles de caractères combinant programmation dynamique et apprentissage de réseaux connexionnistes [Gentric, 1998 ; Gentric, 2000].

La recomposition du mot en lettres est basée ensuite sur une stratégie de prédiction/vérification. Ces approches sont guidées par le lexique qui fournit un nombre réduit

de candidats. Ces méthodes présentent l'avantage de ne pas poser le problème difficile de la segmentation en lettres et sont optimales vis-à-vis des mots.

Les approches de reconnaissance sans segmentation sont limitées généralement aux modèles lettres, pour des raisons calculatoires et combinatoires. Les approches basées segmentation quant à elles permettent un niveau de description plus souple au niveau lettre et même inférieur à la lettre comme les graphèmes. Elles sont adaptées aux grandes variabilités de l'écriture naturelle. Les deux principales méthodes qui prédominent sont les réseaux de neurones et les modélisations markoviennes. D'un point de vue théorique et industriel, elles ont largement montré leur efficacité. Cependant leur apprentissage est encore aujourd'hui un sujet de recherche d'actualité. Nous reviendrons plus en détails sur ce sujet dans le chapitre 3. On retrouve également diverses méthodes de reconnaissance de forme basées sur la théorie de la logique floue, les classifieurs k-ppv, les machines à vecteur support. Quel que soit le classifieur, il est généralement couplé aux chaînes de Markov pour bien modéliser la notion de séquence temporelle de l'écriture.

Segmentation explicite

La segmentation explicite, dite aussi segmentation INSEG (Input Segmentation) effectue une segmentation a priori, elle construit un graphe de toutes les hypothèses de coupures du mot. Une segmentation stricte est rarement utilisée mais plutôt une légère sur-segmentation en lettres que le classifieur devra valider.

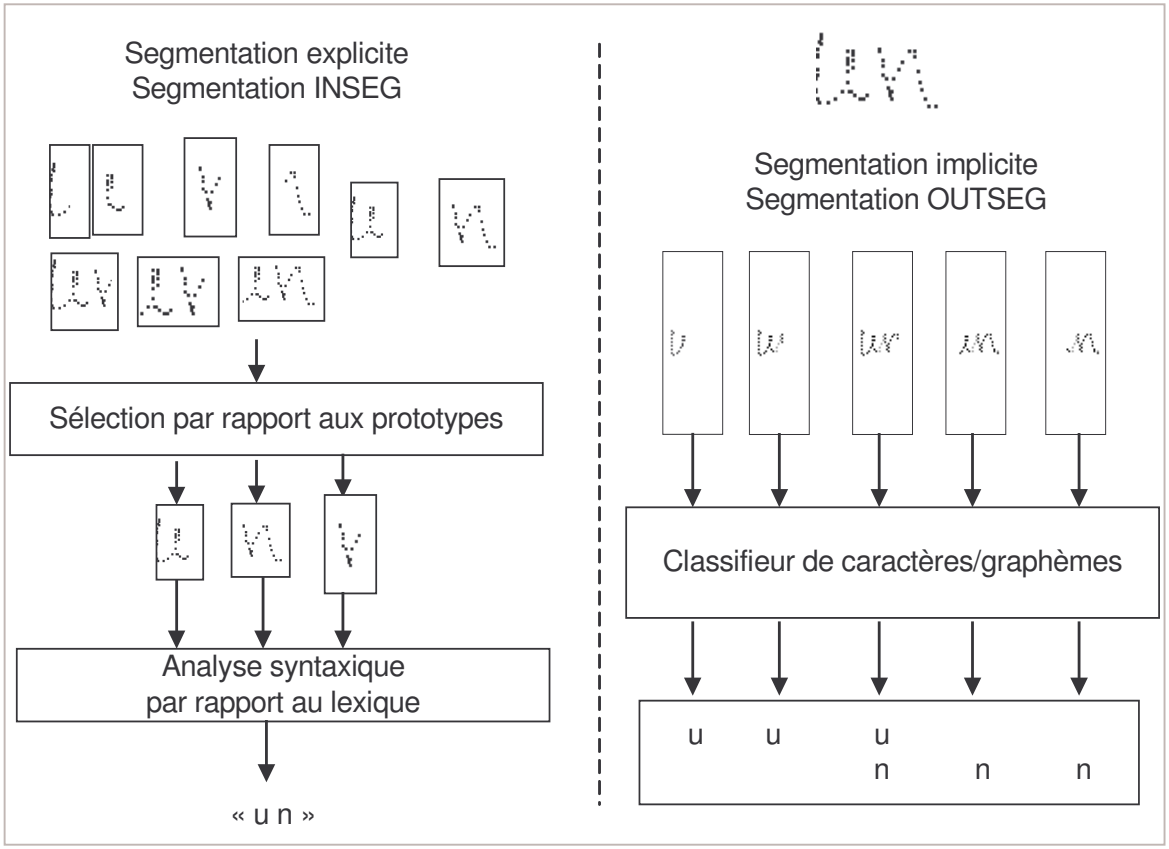


Figure 5. Illustration de la différence des processus de reconnaissance basée segmentation explicite et implicite du mot en-ligne « un »

La Figure 5 illustre ce processus sur la reconnaissance du mot « un ». En entrée diverses hypothèses de limites de caractères sont effectuées, les éléments obtenus sont comparés à la

base de données apprises. Seules quelques hypothèses de caractères sont retenues, celles de la lettre « u », la lettre « n » et la lettre « v ». Après une procédure de vérification par rapport aux modèles mots contenus dans le lexique, une liste de mot est proposée avec comme hypothèse principale celle du mot « un ».

Cette approche est très utilisée en reconnaissance hors-ligne [Chen2000 ; Knerr, 1998 ; Tay, 2002 ; Nosary, 2002], les points de segmentation potentiels s'appuient sur une analyse morphologique du mot. Ces points d'ancrage peuvent correspondre à des points de levers de stylet, points d'inflexion, point de rebroussement, extrema. Les primitives ainsi déterminées peuvent être représentées par des vecteurs caractéristiques ou des règles de modélisation floues [Anquetil, 1997 ; Ragot, 2003]. Cette approche est très appropriée dans le cas d'une écriture scripte [Oudot, 2004] mais peu dans le cadre d'une écriture naturelle si on ne possède pas de base de mots segmentés à la main.

Segmentation implicite

La segmentation implicite (appelée aussi segmentation OUTSEG, OUTput SEGmentation) ne procède pas à une segmentation *a priori* en entrée mais à une compétition des classes lettres ou graphèmes en sortie du classifieur. La Figure 5 illustre un découpage du mot selon une fenêtre d'observation glissante sur le mot « un ». Ceci conduit à une sur segmentation importante du mot pour n'oublier aucun point de segmentation et modéliser au mieux les liaisons inter-lettres. Cette approche utilise essentiellement des modèles de Markov cachés adéquats par leurs structures à modéliser des données séquentielles et leur capacité à modéliser justement les transitions et celle d'intégrer à la fois la segmentation et la reconnaissance. Plusieurs systèmes hybrides ont été construits sur cette approche : RN-MMC [Wimmer, 1999 ; Schenkel, 1995 ; Jaeger, 2000 ; Caillault, 2005], MMC-RN [Choisy, 2003].

Nous reviendrons au chapitre 3 sur ces segmentations et leurs conséquences importantes en terme d'apprentissage afin d'évaluer l'intérêt de notre approche basée sur une segmentation implicite. Afin de faciliter la tâche de segmentation et reconnaissance, il est important de procéder à un filtrage des bruits liés à l'écriture manuscrite cursive. En effet, plus elle est libre et plus elle est variée. Nous allons voir dans la section suivante les prétraitements effectués en amont de la reconnaissance et l'information utile que l'on peut extraire dans le cadre d'une reconnaissance en-ligne. Nous ne balayerons pas les spécificités du domaine hors-ligne qui ne concernent pas notre domaine d'application, nous pouvons citer comme référence l'état de l'art dressé par Vinciarelli [Vinciarelli, 2002].

1.3 Les prétraitements et caractérisation de l'écriture naturelle en-ligne

La reconnaissance de l'écriture naturelle est un problème relativement complexe et très riche, on peut noter :

- la diversité des langues à reconnaître,
- des productions graphiques possibles pour chaque langue et chaque caractère,
- mais aussi par la diversité des scripteurs (droitier/gaucher, enfant/adulte, métier : médecin…)

– et la grande variabilité d'écriture pour chaque personne dans leur aisance gestuelle, leur posture, leur psychisme.

Toutes ces diversités ne sont jamais prises en compte dans un même système de reconnaissance. Il est beaucoup plus aisé de construire des reconnaisseurs spécifiques en fonction des langues, de la latéralité des sujets, des métiers. Cependant quelques normalisations liées principalement aux règles de calligraphie peuvent être opérées. Nous allons décrire dans un premier temps quelques notions de base de l'écriture latine et les corrections principalement appliquées dans les systèmes de reconnaissance. Et nous introduirons ensuite la deuxième étape importante des prétraitements avant la phase de reconnaissance et les représentations possibles du signal d'écriture.

1.3.1 Variabilités de l'écriture naturelle.

Nous ne retracerons pas l'histoire de la forme de notre écriture latine actuelle, le site de la Bibliothèque nationale de France donne un très bel exposé sur l'écriture (disponible sur la toile à l'adresse suivante : http://classes.bnf.fr/dossiecr). Notons seulement que notre graphie est issue de l'écriture humanistique, créée à la fin du XIV^{e} siècle, par des humanistes florentins, jugeant les gothiques illisibles, et qui reprennent et adaptent la caroline.

1.3.1.1 Un geste d'écriture lié à un modèle d'apprentissage

A la base de son enseignement, l'écriture cursive est définie par quatre formes de bases que l'on écrit soit sur des pages blanches soit sur des lignes de cahier largement espacées. Ces formes de base sont la boucle (qui donne e, l pour commencer), la coupe – cf. coupe de fruits – (qui donne u, i, t) (i = demi-coupe, t = demi-coupe prolongée vers le haut), le rond (qui donne c, o, a, d, q, g), le pont (qui donne m, n, mais aussi p et h).

La deuxième étape de l'apprentissage de l'écriture des lettres est le respect de l'espace avec l'introduction du papier seyes. L'écriture suit un code graphique avec des règles statiques et une dynamique à respecter. La gestion statique est formée sous 5 composantes : espace de toutes sortes, axe horizontal orienté, tenue de la ligne, inclinaison des lettres et dimensions. Du côté de la dynamique, on peut noter essentiellement les règles suivantes : les boucles, les coupes et les ronds tournent dans le même sens (inverse des aiguilles d'une montre) ; les ponts tournent dans le sens inverse comme les jambages et le S et une partie du X. La dynamique n'est pas exploitée dans les systèmes actuels dédiés à la reconnaissance, ils partent du principe que l'écriture latine correspond à un tracé orienté gauche-droite. Seuls les logiciels éducatifs ou à vocation médicale prennent en comptent la dynamique, non pas dans un but de normalisation ou reconnaissance mais dans un but de contrôle, vérification, suivi du geste d'écriture.

1.3.1.2 Description et normalisation courante

Dans un contexte de reconnaissance, on se réfère au schéma standard illustré Figure 6 où quatre lignes de références caractérisent la position d'un caractère. Ces lignes correspondent à nos règles de production d'écriture apprises comme la taille des minuscules, majuscules, hauteur des hampes et jambages.

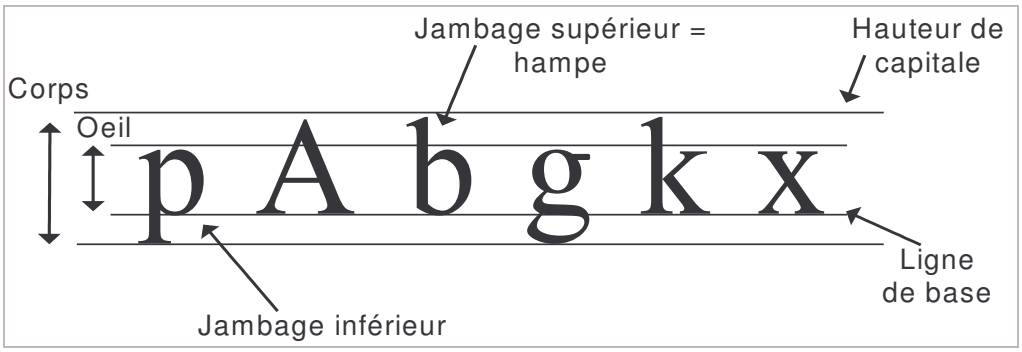


Figure 6. Anatomie d'un caractère

Ces lignes sont définies comme suit :

- la première ligne dite ligne de hampe, elle est située au-dessus des majuscules, des chiffres et des lettres telles que le t, le l, le h.
- la ligne directement inférieure dite ligne de corps, située au-dessus des minuscules.
- la troisième ligne dite ligne de base, elle correspond à la base inférieure des minuscules.
- la dernière dite ligne de jambage, comme par exemple pour les lettres j, p.

En se basant sur un critère morphologique, les caractères peuvent être regroupés en quatre catégories :

- médian, elle correspond aux caractères ne possédant ni ascendant ni descendant (a,c,e,i,m,n,o,r,s,u,v,w,x) ;
- hampe, pour les lettres possédant une hampe et pas de jambage (b,d,f,h,k,l,t). On y associe généralement les majuscules scriptes et certaines cursives (A B C D E F H I K L M N O P Q R S T U V W X Z, …) et les chiffres ;
- jambage, pour les caractères possédant uniquement un jambage (g,j,p,q,y,z) ;
- étendu, comme la lettre f , seule minuscule avec à la fois une hampe et un jambage et quelques majuscules cursives (G J Y ; …) .

Les caractères ne sont pas les seuls signes apparaissant dans l'écriture manuscrite. Il y a également les ponctuations et les signes diacritiques. Un *diacritique* est un signe graphique (point, accent, cédille) adjoint à un graphème simple de l'alphabet afin de transcrire un phonème différent de celui que transcrit ce graphème ou afin d'éviter la confusion entre homographes (comme par exemple : e, é, è, ê, ë).

1.3.1.3 Normalisation par rapport aux lignes de référence.

Les lignes de référence apportent une information importante pour des systèmes d'identification de l'écriture. Elles permettent de normaliser la taille de l'écriture et d'extraire des caractéristiques géométriques liés à la position, orientation des lettres dans le mot.

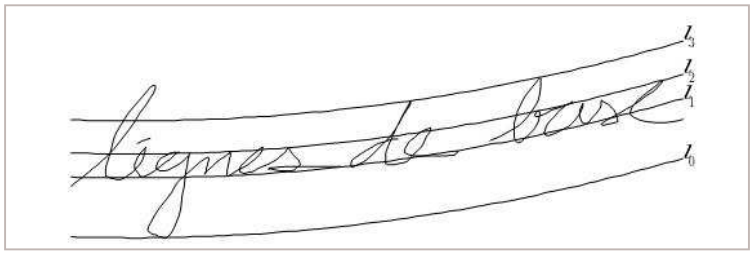


Figure 7. Modèles de lignes de bases paraboliques

Les méthodes de détection des lignes de référence les plus robustes sont basées sur des algorithmes de type EM (Expectation-Maximization) qui utilisent l'information des extremums du contour du mot et cherche à optimiser la position et l'orientation de quatre lignes. Elles dérivent en majeure partie de l'algorithme de Bengio et Lecun [Bengio, 1994] où les lignes de référence sont modélisées par quatre paraboles parallèles, cf. Figure 7. Dans le domaine hors-ligne, des approches supplémentaires sont utilisées, basées sur l'analyse de l'histogramme vertical de projections. La densité des points est alors analysée pour trouver ces lignes. Côté [Côté, 1996] propose une détection des lignes de référence de mots cursifs à l'aide de l'entropie. Le calcul de l'entropie est effectué sur les histogrammes obtenus par projection des ordonnées du contour du mot selon plusieurs plans de rotation et permet de trouver l'orientation selon laquelle le mot est le plus compact. La connaissance de cette orientation et l'analyse de la distribution de l'histogramme sélectionné permet de tracer des lignes de base qui suivent l'orientation du mot.

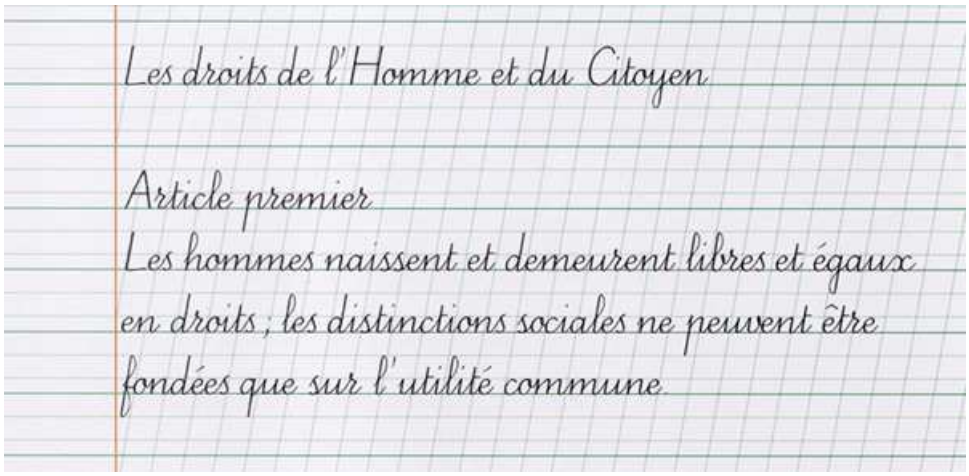
Ces corrections sont généralement bénéfiques pour les mots longs, soit des mots avec un nombre de lettres supérieurs à quatre lettres. Cependant, pour des mots de petite taille (nombre de lettres limité) la tâche risque d'avoir un effet contraire et perturber le système de reconnaissance. Pour pallier ce problème, il est peut-être utile d'estimer le nombre de caractères avant d'effectuer la normalisation. Certaines techniques se basent sur le comptage du nombre de traverses des lignes de référence connaissant le nombre moyen de traverses par caractères [Schenkel, 1995].

La localisation des lignes de référence s'accompagne d'une étape de normalisation qui implique le redressement de l'écriture, et éventuellement le redressement des caractères et une uniformisation de la taille de l'œil quel que soit le mot en entrée du système dans un contexte de reconnaissance omniscripteur.

1.3.1.4 Redressement de l'écriture

L'acte d'écriture ne respecte pas forcément les règles définies en théorie. Les zones de corps des minuscules peuvent présenter des variations importantes de hauteur à l'intérieur du même mot. De plus, même si l'écriture dite droite est une règle par convention, elle est souvent peu respectée d'une part pour des raisons d'aisance gestuelle et d'autre part de liberté artistique. Il faut donc prendre en compte les écritures penchées, écriture généralement plus adaptée à une écriture naturelle rapide et moins fatigante pour la main.

On distingue en général trois grandes classes : l'écriture inclinée vers la gauche, l'écriture inclinée vers la droite et l'écriture droite ou verticale. L'inclinaison de l'écriture est, dans un premier temps, fonction de l'éducation reçue. En France, on considérait l'inclinaison à droite faisant un angle de 54° avec l'horizontale comme la règle, voir Figure 8.



(http://www.education.gouv.fr/presse/2002/ecriture/ecrituredp.htm)

Figure 8. Présentation du modèle d'écriture cursive destiné à l'apprentissage

Mais on enseigne maintenant plus volontiers l'écriture verticale, ou même libre, permettant alors à l'enfant d'adopter l'inclinaison qui lui vient naturellement. De plus, l'inclinaison de l'écriture varie de manière souvent importante tout au long de la vie. Car cette caractéristique de l'écriture est directement porteuse des évènements que l'on subit et de leurs conséquences, ainsi que de l'influence sur nous du milieu dans lequel nous évoluons. La correction de l'inclinaison permet ainsi de normaliser l'écriture à une écriture dite verticale.

La correction de l'inclinaison des caractères intervient après le redressement vertical de l'écriture. Plusieurs méthodes existent pour estimer l'inclinaison des lettres. Elles sont basées sur la détection des traits quasi-verticaux, par l'utilisation de la transformée de Hough [Marukatat, 2004] par exemple, ou bien directement par optimisation d'un critère basé sur le profil de projection verticale. L'inclinaison moyenne étant déterminée, on corrige globalement l'inclinaison avec une transformation de type cisaillement [Vincieralli, 2000].

1.3.1.5 Gestion des marques diacritiques

Les langues latines introduisent elles-mêmes de grandes variabilités sur leurs formes et notamment sur leurs diacritiques. La Figure 9 montre un aperçu de trois langues différentes : le français, l'allemand et le tchèque.

Alphabet français :
A À Â B C D E É Ê Ë È F G H I Ï Î J K L M N O Ô Œ P Q R S T U Ù Û V W X Y Z
a à â b c ç d e é è ê ë f g h i î ï j k l m n o ô œ p q r s t u ù û v w x y z

Alphabet allemand :
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z Ä Ö Ü
a b c d e f g h i j k l m n o p q r s t u v w x y z ß ä ö ü

Alphabet tchèque :
A Á B C Č D Ď E Ě F G H I Í J K L M N Ň O Ó P Q Ř S Š T Ť U Ú Ů V W Y Ý Z Ž
a á b c č d ď e é ě f g h i í j k l m n ň o ó p q r ř s š t ť u ú ů v w x y ý z ž

Figure 9. Exemples d'alphabets latins avec diacritiques

Dans une grande partie de la littérature sur la reconnaissance dite de l'écriture manuscrite cursive latine, les systèmes de reconnaissance sont évalués sur un dictionnaire de mots composés uniquement des lettres minuscules anglaises où les rares signes diacritiques sont les points du i et du j et la barre du t ! Peu de systèmes ont été évalués sur des alphabets latins très ponctués comme la langue française, il est à noter que cette langue n'est pas isolée : Pedro Inigo Yanez[11] illustre l'ensemble des alphabets européens et un nombre important de langues et dialectes internationaux.

Une manière simple de s'affranchir de ce problème délicat est soit de contraindre l'utilisateur à marquer ces diacritiques juste après la saisie du graphème associé soit d'imposer une saisie sur un clavier virtuel. Ce qui n'est pas naturel. Différentes techniques consistent à détecter et supprimer toutes marques diacritiques en insérant un codage spécifique [Schenkel, 1995 ; Garcia-Salicetti, 1996, Jaeger, 2000] de présence de signe diacritique en chaque point du signal d'entrée. Certaines techniques basées sur des modélisations markoviennes [Hu, 2000] permettent de modéliser les retours en arrière correspondant au tracé des marques diacritiques, ces méthodes sont plus généralisées dans le domaine hors-ligne.

Les corrections apportées au signal d'écriture sont liées à un modèle de production avec des bases géométriques, les notions de type d'écriture ne sont pas normalisées et le reconnaisseur doit pouvoir s'adapter à tous les allographes possibles de l'écriture cursive en-ligne. Cette notion d'allographe est très importante dans un contexte mono-scripteur ou dans un schéma de reconnaissance basé sur l'appariement de prototypes. Nous allons maintenant nous intéresser aux variabilités directement liées au scripteur.

1.3.2 Variabilités liées au scripteur.

En graphologie on a coutume de discriminer une personne par 7 aspects de l'écriture [Huteau, 2004]:

- La dimension, examen de la taille de l'écriture et des différences de taille entre les différents signes. La dimension donne des indications sur les rapports du sujet avec lui-même, et son adaptabilité.
- La direction permet de cerner la sensibilité, l'équilibre. Exemple: une écriture droite montre un sujet maître de lui et de ses émotions; penchée à droite, l'écriture révèle quelqu'un qui va de l'avant, ouvert aux autres; penchée à gauche, elle dénote un repli sur soi.
- La continuité, ou comment les lettres sont reliées entre elles. Cette étude permet d'appréhender la persévérance, ou les hésitations...
- La forme, ou manière dont les lettres sont "dessinées". Elle renseigne sur le degré d'originalité, d'imagination, de création.
- L'ordonnance, ou manière dont le texte s'inscrit sur la page, importance des marges, espace entre les lignes...etc. L'ordonnance donne des indications sur l'organisation du sujet ainsi que sur son mode de fonctionnement avec autrui.
- La pression, ou force du trait. Elle renseigne sur l'énergie psychique, l'intensité des sentiments...
- La vitesse révèle la vitalité motrice, la rapidité de pensée.

[11] http://pedroiy.free.fr/alphabets/index.htm, le site de tous les alphabets.

Nous ne rentrerons pas dans ces notions subjectives de l'écriture, cependant elles montrent à la fois les différences entre deux scripteurs différents (dimension, forme, continuité, forme, ordonnance, vitesse) et la diversité du style d'écriture d'une même personne (direction, pression, vitesse).

1.3.2.1 Allographes de caractères, sélection ou généralisation.

Les allographes de caractère représentent toutes les traces possibles résultant de la représentation d'un même symbole. Pour un même caractère, il existe un nombre très important d'allographes dans l'écriture naturelle comme on le peut le constater sur l'image ci-dessous qui caractérise quelques formes d'écriture possibles de la lettre f.

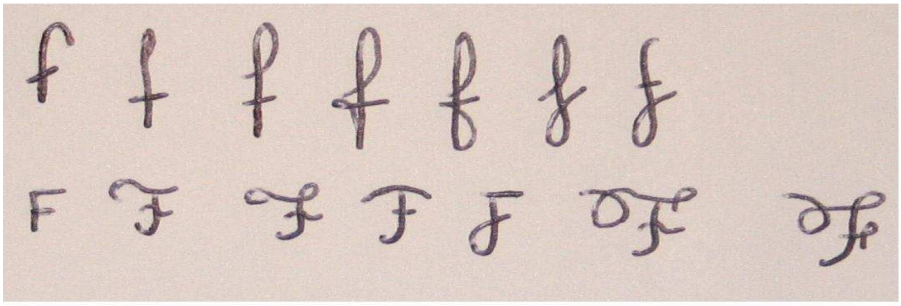


Figure 10. Échantillons d'allographes possibles de la lettre « f »

La Figure 10 ne donne qu'une idée de différents allographes statiques d'un même caractère. En complément, dans le domaine en-ligne, il faut considérer les variantes selon la dimension temporelle. Par exemple concernant un trait vertical, il a pu être tracé de bas en haut ou inversement de haut en bas, ou bien une boucle, tracée dans le sens des aiguilles d'une montre ou dans le sens trigonométrique.

La reconnaissance de l'écriture naturelle, même dans une langue donnée, peut devenir très difficile face à cette grande variété d'allographes. Certes dans un cadre mono-scripteur, il sera plus aisé de sélectionner certains prototypes ou caractéristiques propres au scripteur et il suffira de rechercher le plus proche voisin dans la base de connaissances enregistrées au préalable. Cependant dans le cadre d'une reconnaissance omni-scripteur, nous pouvons buter sur le problème de la malédiction de la dimensionnalité où l'apprentissage peut devenir de plus en plus difficile au fur et à mesure que le nombre de caractéristiques augmente. Et, il est évident qu'aucune base d'apprentissage ne peut contenir toutes les variantes possibles d'un caractère, tellement le processus de collecte de l'écriture est longue et coûteuse et le nombre d'individus astronomique. Il est donc nécessaire dans le cadre omniscripteur de généraliser au mieux les différentes classes, regarder le plus proche voisin ne suffit plus. Certains algorithmes d'apprentissage modernes cherchent à relever ce défi tels que les réseaux de neurones artificiels, les machines à vecteurs supports. L'idée de base est d'utiliser un modèle flexible mais qui permet de résumer l'information dans les données et de capturer les régularités dans les données.

1.3.2.2 Ré-échantillonnage spatial de l'écriture.

L'écriture est le fruit d'un mouvement, d'un geste commandé par le cerveau à notre main pour transmettre une pensée à un lecteur. Chaque écriture est unique et transmet une part de personnalité de l'individu qui en est l'auteur. Elle retranscrit, de manière plus ou moins apparente, l'état d'esprit de son auteur, son environnement, son mental, mais aussi son physique,

son état de santé... Ces variations mais aussi les différents bruits liés aux systèmes d'acquisition sont en général portés par la notion de vitesse du tracé, source de confusions. Dans la plupart des systèmes de reconnaissance omniscripteur elle est éliminée par un ré-échantillonnage spatial du signal en points équidistants.

1.3.3 Représentation du signal d'écriture en-ligne

La définition et sélection d'un ensemble de caractéristiques pertinent pour un système de reconnaissance de l'écriture cursive représentent un travail délicat et important pour la suite de l'étape de reconnaissance. Le choix de la représentation du signal d'écriture doit intégrer différents critères :

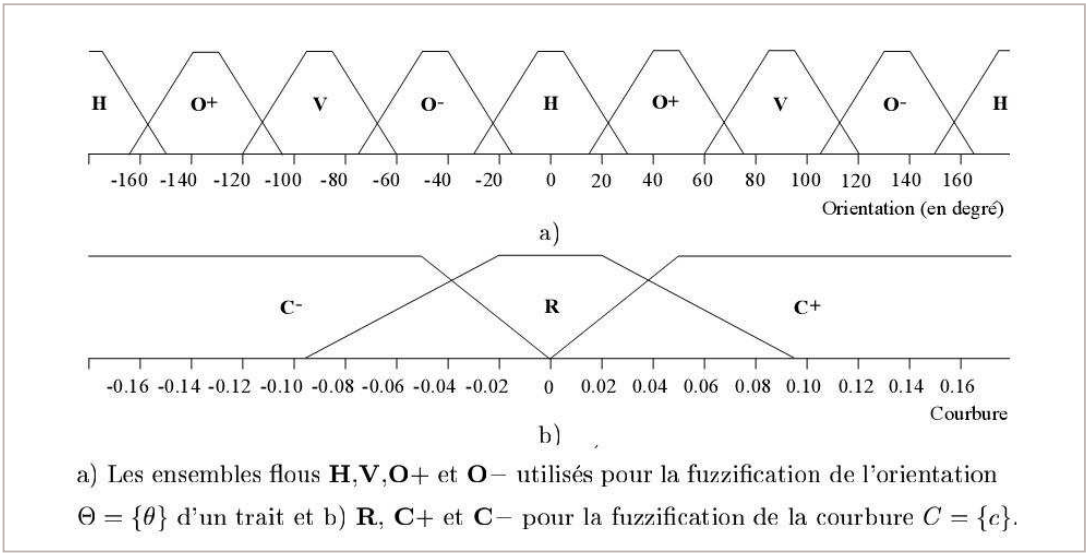
- Il doit être adapté à la méthode de classification.
- La représentation doit être à la fois suffisamment discriminante pour identifier la bonne classe et englobante pour caractériser la variabilité intra-classe.
- Les caractéristiques sélectionnées doivent être complémentaires.
- La complexité (calcul, mémoire) doit être judicieuse en fonction du matériel visé et du temps de traitement.

La représentation du signal en-ligne peut être divisée en deux parties qui dépendent du niveau de reconnaissance. Dans le cadre d'une reconnaissance basée lettre ou symbole, l'entité étant généralement normalisée en taille, sa représentation est en dimension fixe. Tandis que pour la reconnaissance de mots, la représentation la plus courante est une séquence temporelle de vecteurs caractéristiques, mesures locales en chaque point du tracé normalisé par rapport à la hauteur de la ligne de base. Nous présentons les caractéristiques les plus utilisées en fonction des contraintes utilisées par les systèmes de reconnaissance et nous nous intéresserons à l'exploitation de la dimension spatiale de l'écriture.

1.3.3.1 Reconnaissance de symboles, représentation en dimension fixe.

Il existe différentes approches soit basées sur des caractéristiques locales soit basées sur des caractéristiques dites globales. La première représentation est la plus courante pour les systèmes de reconnaissance de caractères. Elle consiste à ré-échantillonner tout signal en un nombre fixe de points par des méthodes d'interpolation du tracé et de le positionner par rapport à son centre de gravité [Guyon, 1991 ; Schenkel, 1996 ; Poisson, 2001].

Les principales caractéristiques extraites en chaque point du tracé sont les coordonnées, l'orientation de la trajectoire par rapport à l'horizontale, la courbure et l'information de poser ou de lever de stylet. Nous détaillerons cette approche et ces caractéristiques dans nos schémas de prétraitement.



a) Les ensembles flous **H**,**V**,**O**+ et **O**− utilisés pour la fuzzification de l'orientation $\Theta = \{\theta\}$ d'un trait et b) **R**, **C**+ et **C**− pour la fuzzification de la courbure $C = \{c\}$.

Figure 11. Représentation floue de Parizeau [Parizeau, 2001]

La deuxième approche consiste à extraire un nombre fixé de caractéristiques globales du signal d'écriture. Parizeau et Lemieux [Parizeau, 2001] utilise une transformation de l'espace 2D des caractères vers un nouvel espace vectoriel flou basé région (précisément un découpage en 6 zones) où sept caractéristiques floues sont extraites, Figure 11. Les trois premières mesures caractérisent la courbure selon sa valeur : trait rectiligne (R), trait courbé positivement (C+), trait courbé négativement (C-). Les quatre autres concernent l'orientation du tracé : plus ou moins horizontal (H), vertical (V), oblique positif ou négatif (O+ ou O-). Le signal est représenté alors par une concaténation des vecteurs flous de chaque région couplée à des mesures de densités horizontales et verticales pour chaque région.

1.3.3.2 Représentation pour la reconnaissance de mots.

Représenter un mot par une séquence de points est dans le domaine en-ligne, la représentation la plus naturelle. Toutefois, cette représentation présente l'inconvénient d'être dépendante de la longueur du tracé. La Figure 12 illustre différents tracés du mot longueur avec des tailles différentes bien que la hauteur d'œil soit constante.

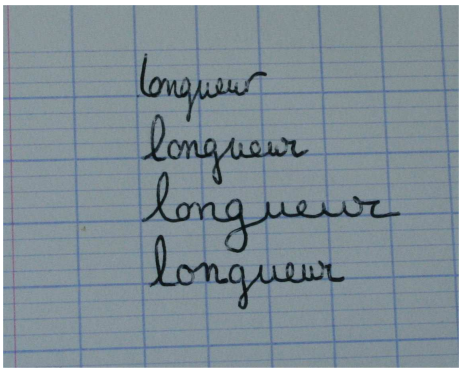


Figure 12. Ecritures de longueurs différentes du mot longueur.

De cette séquence initiale, on peut dériver des caractéristiques purement locales [Schenkel, 1995 ; Jaeger, 2000], semi-locales ou globales. Voici une liste (non exhaustive) des caractéristiques les plus utilisées :

- Déplacement relatif sur les abscisses, Δx
- Ordonnée du point par rapport à la ligne de base (y0), y – y0
- Orientation de la trajectoire par rapport à l'axe horizontal, cosinus et sinus de l'angle
- Courbure de la trajectoire au point courant, cosinus et sinus de l'angle
- Lever ou poser de stylet (appelé Penup)
- Indicateur de marques diacritiques
- Vitesse instantanée du stylet avant ré échantillonnage spatial
- Ratio de la boîte englobante
- Indicateur d'ascendant et descendant.

Le nombre de caractéristiques à sélectionner est une étape fondamentale. Quelques travaux [Alleau, 2003 pour le domaine en-ligne ; Günter, 2003] se sont intéressés à vérifier l'intérêt et la complémentarité des caractéristiques mais leur évaluation est difficile car elle nécessite de disposer d'un système de reconnaissance.

D'autre part, certains travaux cherchent à caractériser le signal par des informations structurelles comme par exemple le codage de Freeman, le codage directionnel du tracé selon 16 directions [Lee, 2001], codage en 36 tracés élémentaires [Artieres, 2002 ; Marukatat 2004] illustrés par la Figure 13.

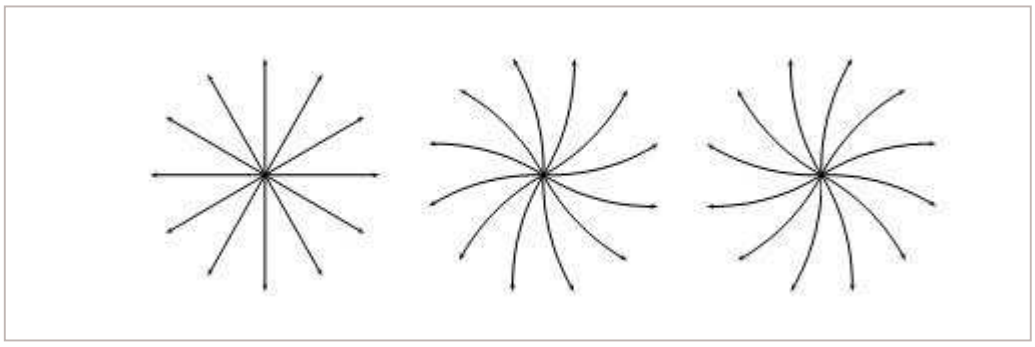


Figure 13. Ensemble de 36 tracés élémentaires utilisés pour représenter les formes en deux dimensions [Artieres, 2002]. De gauche à droite : 12 tracés droits, 12 convexes, 12 concaves.

1.3.3.3 Modélisation de la représentation spatiale du signal dynamique.

Les représentations citées ci-dessus intégraient la notion dynamique de l'écriture. Or le processus de lecture humain se base tout d'abord sur une image du mot puis sur un balayage temporel du mot. Dans le cadre d'un système de reconnaissance en-ligne, différents auteurs ont étudié l'apport et la complémentarité de la représentation statique du tracé par rapport au signal dynamique de celui-ci. En effet, deux trajectoires différentes du stylet peuvent conduire à la même forme graphique, dans ce cas, la représentation statique sera plus robuste, et inversement un même caractère peut avoir des représentations graphiques distinctes qui soient produites par des mouvements très voisins, donnant cette fois un avantage à la représentation dynamique. On peut espérer dans ces conditions qu'une approche combinant les deux types d'information permette d'améliorer les performances de reconnaissance [Alimoglu, 1997 ; Poisson, 2002].

Pour des approches procédant par segmentation implicite, une représentation statique de faible résolution décrivant la forme du mot dans le voisinage d'un point est ajoutée à la représentation séquentielle basée sur les caractéristiques dynamiques locales du tracé. Dans le système Npen++ de Jaeger et Manke [Jaeger, 2000] une fenêtre de taille 3*3 est ajoutée pour chaque point. Elle correspond au nombre de points du tracé situés dans une fenêtre centrée sur le point courant. La Figure 14 illustre le calcul des niveaux de gris pour la fenêtre d'observation du t. Cette fenêtre permet de capturer une information spatiale et notamment de détecter des marques diacritiques.

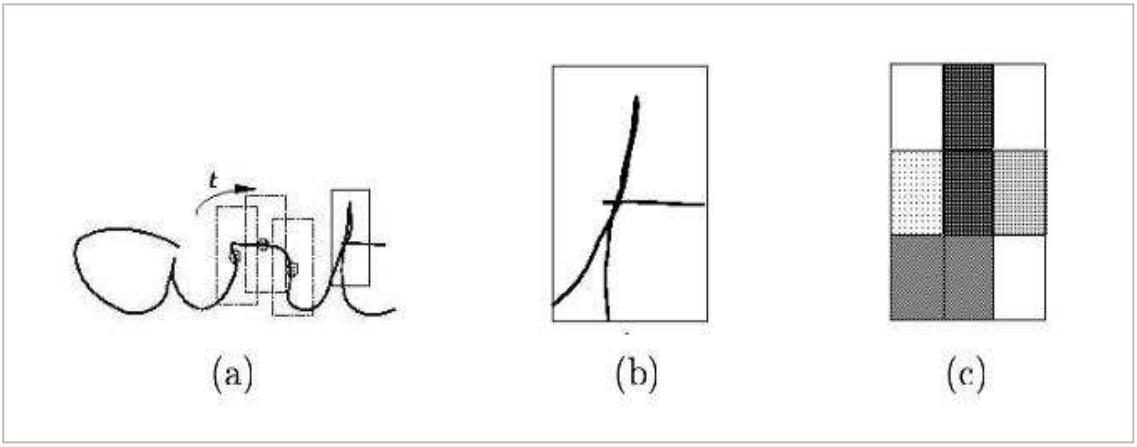


Figure 14. Npen ++ : incorporation d'information spatiale. (a) balayage temporel du signal (b) fenêtre d'observation englobant la lettre t (c) fenêtre en niveaux de gris

Dans LeRec [Bengio, 1995], la notion picturale de l'écriture est nettement plus importante. La représentation appelée AMAP est une image 2D du signal où chaque pixel contient des informations locales dynamiques.

Les prétraitements et le choix de la représentation sont une étape fondamentale pour construire un système de reconnaissance. Ils peuvent, s'ils sont bien choisis, avantager nettement le classifieur, ou à l'inverse, le pénaliser fortement. Il est en général admis qu'il faut 10 fois plus d'exemples par classe que de caractéristiques pour estimer correctement les paramètres d'un classifieur. On peut alors chercher à réduire le nombre de caractéristiques parmi celles disponibles ou construire de nouvelles caractéristiques à partir de celles déjà existantes. L'espace des caractéristiques dépend fortement de la technologie du classifieur et du contexte de l'application. Dans la suite de ce chapitre, nous allons revenir sur les fondements et principes des classifieurs les plus utilisés en reconnaissance de l'écriture manuscrite en-ligne.

1.4 Les classifieurs en-ligne les plus classiques

Les méthodes d'apprentissage statistiques (Modèles de Markov et réseaux de neurones) sont prépondérantes dans la littérature par rapport aux méthodes de modélisation globale telle que l'appariement de patrons ou prototypes (template). Ces dernières sont restreintes à une application soit mono-scripteur soit à vocabulaire limité et fixé. Par opposition, les méthodes d'apprentissage statistiques permettent de modéliser et apprendre la variabilité de problèmes réels et de valider les approches sur des bases de données significatives. De plus elles intègrent les concepts de généricité du système : adaptabilité à des écritures différentes et adaptabilité à

des vocabulaires ouverts. On peut donc distinguer deux types d'approches, celles à vocabulaire fixé et celles à vocabulaire paramétrable sans nécessiter de nouvelle phase d'apprentissage. Nous allons maintenant présenter les différents classifieurs.

1.4.1 Techniques de reconnaissance basée prototypes.

1.4.1.1 Appariement de graphe

Une des méthodes les plus simples de reconnaissance est la comparaison. Il s'agit de mettre en correspondance le signal d'entrée avec un ensemble de prototypes. La difficulté réside dans la modélisation de la déformation entre le signal d'entrée et chacun des prototypes et l'évaluation de cette déformation. Différentes techniques sont utilisées comme l'alignement de points, l'appariement souple ou élastique (elastic matching en anglais, illustré Figure 15), appartenance à des zones spatiales, mesure d'invariants…

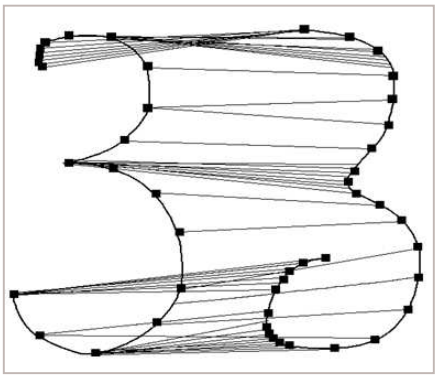


Figure 15. Recherche de correspondance entre le prototype à gauche du label 3 et le signal d'entrée (à droite)

Il est nécessaire de trouver un compromis entre la vitesse de reconnaissance et la robustesse du système. Afin de réduire la mise en correspondance avec le nombre de prototypes un changement d'espace peut être opéré : transformée de Fourier [Fujisaki, 1971].

1.4.1.2 Les k-plus proches voisins (k-ppv).

La méthode par k plus proches voisins est une méthode simple qui ne nécessite pas d'apprentissage. La modélisation utilisée pour la classification est la base d'apprentissage qui constitue l'ensemble des connaissances du système de reconnaissance. Le principe est d'attribuer à la donnée en entrée du reconnaisseur la classe la plus représentée dans son plus proche voisinage. Il faut donc déterminer le nombre k d'individus considérés et la métrique utilisée qui peut être une distance élastique, telle que présentée à la section précédente, pour comparer les individus et sélectionner les plus proches voisins.

L'efficacité de cette méthode dépend de la qualité de sa base d'apprentissage et de la capacité de stockage de celle-ci. Elle est donc limitée à une reconnaissance globale avec un vocabulaire restreint pour ne pas alourdir les temps de traitement et les coûts de stockage. Elle est donc particulièrement adaptée à une reconnaissance mono-scripteur, l'utilisateur devra alors, avant d'utiliser le système, procéder à la saisie de la base d'apprentissage.

De telles méthodes, basées prototypes, sont propices à des techniques d'adaptation au scripteur en cours d'utilisation. Les prototypes non utilisés sont remplacés progressivement par

de nouveaux échantillons plus spécifiques du scripteur, et cela de façon non supervisée grâce à l'appui du lexique [Perraud, 2003].

1.4.2 Les machines à vecteurs supports (SVM, support vector machines)

Ces méthodes sont proches de celles basées prototypes. En effet, elle repose sur la détermination de vecteurs supports, forme remarquable de la base d'apprentissage qui permettront de discriminer les formes.

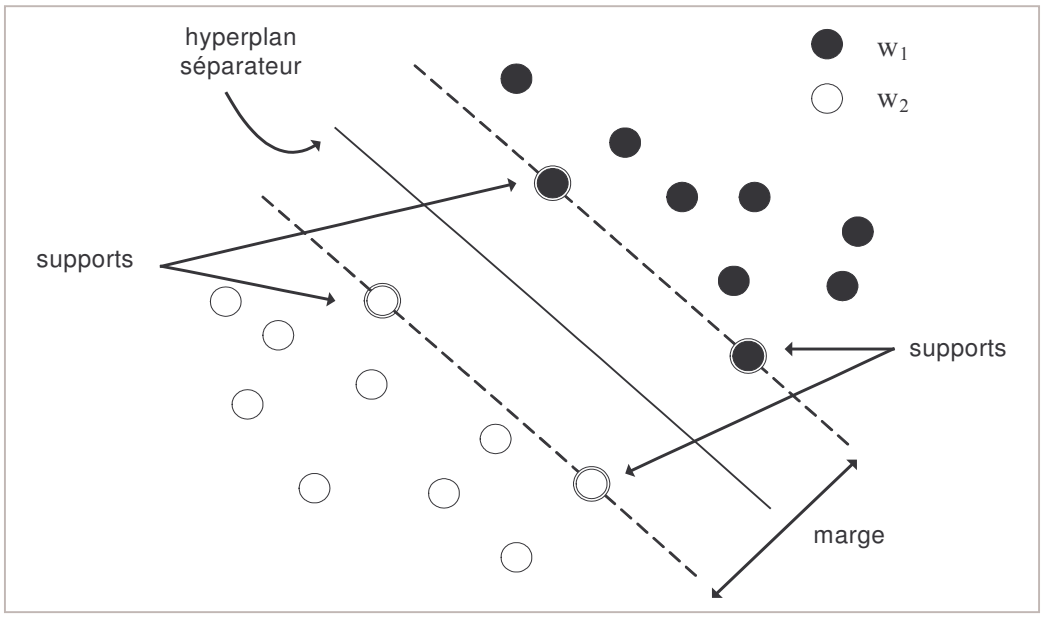


Figure 16. Séparation souple par SVM : marge et hyperplan séparateur

Ces méthodes sont aussi appelées classifieurs à marge optimale ou encore séparateurs à vaste marge. Il s'agit d'une famille d'algorithmes d'apprentissage mis en avant par le mathématicien Vladimir Vapnik [Vapnik, 1995] et détaillés dans les tutoriaux de Chris Burges de Lucent Bell Labs [Burges, 1998].

L'idée principale est que deux classes (voir Figure 16, la classe des points noirs et la classe des points blancs) peuvent être linéairement séparées dans un espace de grande dimension $\Phi(X)$. Si les points d'apprentissage sont séparables, il existe une infinité d'hyperplans séparateurs.

On définit la **marge** comme étant la distance minimale entre la surface de décision et les exemples d'apprentissage (qui sont supposés être tous correctement classifiés par cette surface de décision). La difficulté est alors de trouver le meilleur hyperplan séparateur dans $\Phi(X)$ en utilisant les propriétés des fonctions à noyaux et de le transcrire dans un cas non séparable. Il s'agit d'un problème d'optimisation quadratique sous contraintes, ce qui est assez complexe (en termes algorithmiques) mais qui donne des garanties sur le temps de convergence (à un minimum global), contrairement à l'optimisation numérique d'une fonction de coût non-quadratique (comme pour des réseaux de neurones ou des mixtures de Gaussiennes). On veut maximiser la marge sous la contrainte que les exemples soient correctement appris.

Les SVM travaillent avec des données en dimension fixe, ils sont donc plus utilisés dans des applications restreintes à la reconnaissance de caractères ou de mots basée sur une segmentation

explicite. L'inconvénient majeur de cette méthode est le coût important de mémorisation des vecteurs supports qui sont très nombreux pour absorber la variabilité de l'écriture.

1.4.3 Approches markoviennes

Les modèles de Markov cachés ont été tout d'abord introduits principalement en reconnaissance de la parole [Rabiner, 1993]. Ces travaux ont ensuite été adaptés à l'écriture hors-ligne cursive qui peut être modélisée par une suite séquentielle d'observations. La principale différence entre la reconnaissance de la parole et celle de l'écriture cursive réside dans la nature différente des caractéristiques extraites.

Dans les modélisations markoviennes, une suite d'observations est modélisée par une suite d'états avec des probabilités d'émission. La suite d'états est conditionnée par des probabilités de transitions. A chaque état est associé une probabilité d'émission de symboles observables. Ces états ne sont pas directement observables, c'est pourquoi, ils sont dits cachés, d'où le terme de modèles de Markov cachés. La Figure 17 illustre l'adéquation entre cette description et le signal d'écriture, nous reviendrons au chapitre 3 sur le formalisme mathématique des modèles de Markov [Rabiner, 1989].

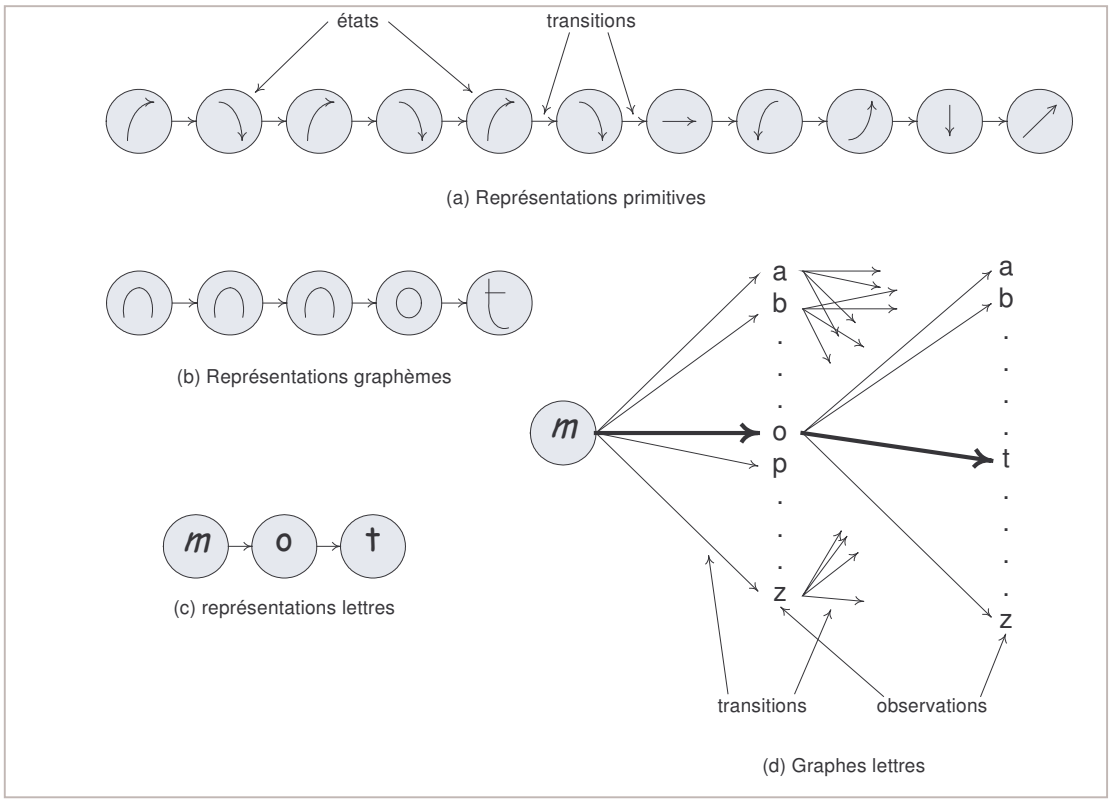


Figure 17. Différentes représentations markoviennes du mot

On distingue trois types de modèles de Markov correspondantes aux trois principales stratégies de segmentation.

1. Modèle de mot à partir de ses constituants lettres ou graphèmes (approche analytique explicite). L'étape préalable à l'utilisation de ce modèle est la segmentation du mot en lettres ou en graphèmes. Le modèle est construit essentiellement sur les bigrammes ou les trigrammes (deux ou trois lettres consécutives dans un mot).

2. Modèle de lettre (approche analytique implicite). Il y a un modèle de Markov caché par mot du lexique. Ce modèle de mot est généralement construit par concaténation des modèles de lettres.

3. Modèle de mot à partir de ses primitives (approche globale).

Un mot peut être représenté par une suite d'éléments orientée gauche droite tels que des primitives, ou des graphèmes voire des lettres. Pour reconnaître le mot, on cherche à retrouver dans le lexique le mot qui lui correspond. Pour trouver le mot le plus vraisemblable, on calcule à partir des données une probabilité d'observation de chaque élément dans chaque état et on probabilise les transitions. Les probabilités peuvent être définies à partir d'un modèle de durée, d'un modèle de langage (bigramme, trigramme : fréquences d'occurrences de lettres consécutives). Ainsi la probabilité d'un élément est la probabilité de la transition qui a amené à ce caractère multipliée par la probabilité d'observation de ce caractère dans cet état.

Une modélisation markovienne est bien adaptée à l'écriture car elle permet d'intégrer les variabilités de longueurs des mots et permet d'absorber les distorsions. Cependant ce sont des modèles à vision locale qui prennent en compte uniquement un contexte local et un contexte limité ce qui n'est pas optimal dans le cadre d'une reconnaissance mot.

1.4.4 Modèles connexionnistes

Les approches basées réseau de neurones correspondent quant à elles à des modèles à vision globale. Les réseaux de neurones sont utilisés pour séparer des classes, principalement de caractères. L'idée principale est qu'un neurone formel, modèle mathématique issu du modèle biologiste (cf. Figure 18), est capable de réaliser quelques calculs élémentaires notamment la séparation de vecteurs en deux classes. Les classes sont déterminées par les poids du neurone. Le problème est alors de choisir quels coefficients affecter aux poids pour résoudre une séparation optimale.

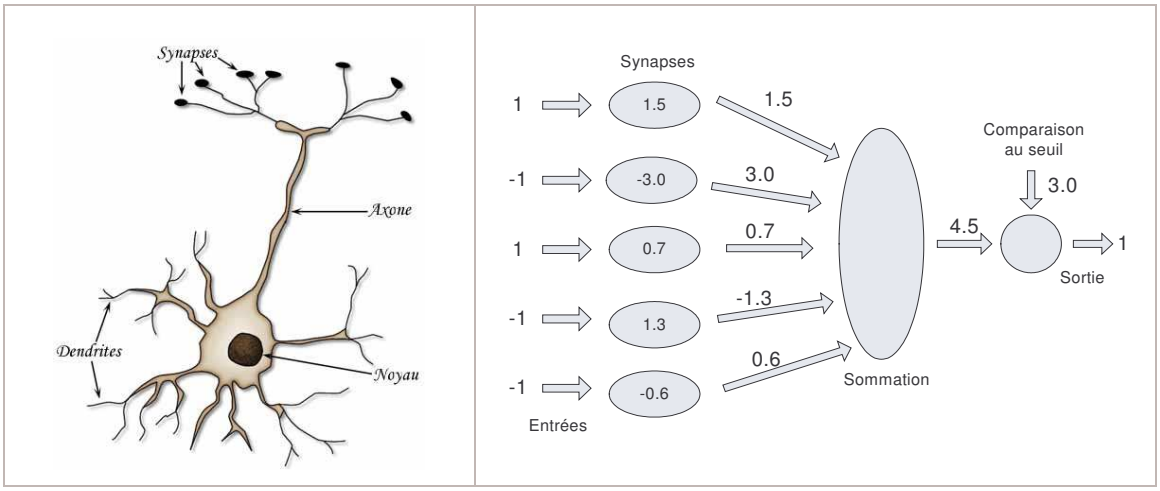


Figure 18. Du neurone biologique (à gauche) au neurone formel (à droite)

La multiplication des neurones permet de séparer plusieurs classes, la difficulté réside alors dans le choix de la topologie du réseau en fonction du problème en adéquation avec le nombre de données en apprentissage.

Les réseaux de neurones sont particulièrement adaptés pour reconnaître des formes globales telles que les lettres isolées et ont une bonne capacité de généralisation. Cependant, ils se

limitent à la classification de formes simples car ils sont basés sur une représentation en dimension fixe. Dans le chapitre suivant nous détaillerons quelques topologies de réseaux de neurones et notamment les réseaux à convolution qui permettent d'avoir un contexte local.

1.4.5 Techniques de combinaisons de classifieurs

Dans le paragraphe précédent, nous avons présenté les principales techniques de classifications utilisées en reconnaissance de l'écriture manuscrite en-ligne et leurs intérêts et limites. Pour pallier les limitations, telles que des performances insuffisantes, une instabilité et/ou une complexité trop importante, différents travaux dans la littérature ont été conduits sur des systèmes à classifieurs multiples. L'idée principale est d'exploiter au maximum la complémentarité des modélisations afin d'obtenir des décisions plus robustes et compenser les faiblesses de chaque classifieur.

On distingue deux types d'approches : les approches de combinaison de classifieurs fonctionnellement indépendants et celles de combinaison dépendante.

1.4.5.1 Combinaison de classifieurs fonctionnellement indépendants

Chaque classifieur est considéré comme un acteur indépendant dans le mécanisme de décision. Comme l'illustre la Figure 19, les données sont traitées en parallèle par chaque entité qui apportent leurs décisions sans influence des autres classifieurs. La décision finale dépend de la nature des sorties de chaque classifieur. Dans le cas de sorties homogènes, la combinaison peut-être réalisée par des méthodes par votes majoritaires ou pondérés. Dans le cas de sorties hétérogènes (comme par exemple les rangs ou distances), une étape de normalisation des sorties est nécessaire, un classifieur supérieur est généralement utilisé pour traiter ces données.

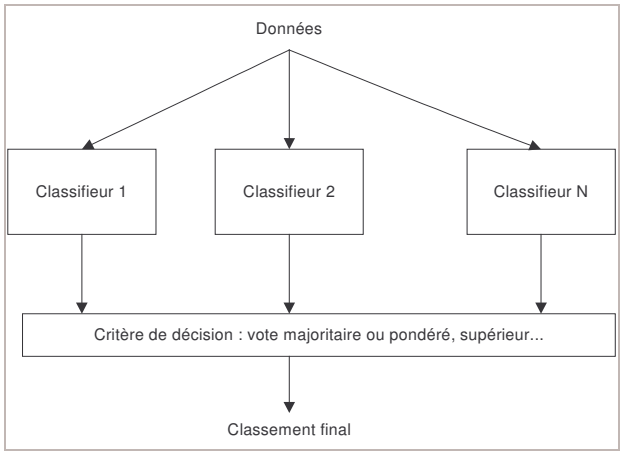


Figure 19 : Combinaison parallèle de classifieurs indépendants

La méthode de Bagging [Breiman, 1996] permet de construire des ensembles de classifieurs, chaque classifieur étant entraîné sur une réplique différente de la base d'apprentissage. Les sorties de chaque classifieur sont ensuite combinées selon différentes méthodes de vote. L'intérêt de cette méthode est de minimiser l'impact d'un classifieur instable.

La combinaison de classifieurs fonctionnellement indépendants est aussi utilisée pour manipuler des sous-espaces de l'espace des caractéristiques original. Ces sous-espaces peuvent être liés à la nature de l'information à extraire : dans le cas de la reconnaissance de l'écriture, il est intéressant d'analyser la complémentarité entre représentations en-ligne et hors-ligne de

l'écrit [Alimoglu, 1997 ; Poisson, 2002 ; Velek, 2003]. Dans le cas d'un problème avec un espace de caractéristiques de grande dimension contenant des informations souvent redondantes, il est intéressant de découper l'espace original en sous-espaces de taille réduite, chaque classifieur sera entraîné sur un sous-espace particulier. Ce découpage est principalement intéressant lorsque les données d'apprentissage sont insuffisantes par rapport aux paramètres du classifieur (problème de la dimensionnalité). Des techniques de découpage en sous-espaces aléatoires ont été détaillées dans [Ho, 1998].

La diversité des points de vue des classifieurs obtenus à partir de données – identiques, prétraitées voire sélectionnées, permet d'améliorer les performances des systèmes de reconnaissance.

1.4.5.2 Combinaison de classifieurs fonctionnellement dépendants

Une deuxième méthode consiste à attribuer un rôle particulier à chaque entité pour effectuer une sous-partie de la tâche de classification.

La topologie la plus simple est l'approche séquentielle qui consiste à appliquer une méthode de reconnaissance sur les résultats obtenus par une méthode précédente. Cette organisation, illustrée Figure 20a, permet notamment de gérer les refus de classement ou les décisions non fiables (comme un taux de classification majoritaire considéré insuffisant).

A ces approches de combinaisons correctives, on peut ajouter les approches basées sur des ensembles de classifieurs [Opitz, 1999]. Elles reposent sur l'utilisation d'un classifieur principal dont les performances sont jugées insuffisantes et la création d'autres classifieurs afin de corriger les décisions erronées du classifieur de base ou précédent. Ces méthodes suivent la méthode d'apprentissage d'ensemble AdaBoost [Freund, 1996] qui utilise comme règle de combinaison un vote pondéré prenant en compte la précision de chaque classifieur [Singer, 1999], cf. Figure 20b. Cette méthode a été appliquée à la reconnaissance de caractères en-ligne à partir de réseaux de neurones dans [Schwenk, 1997].

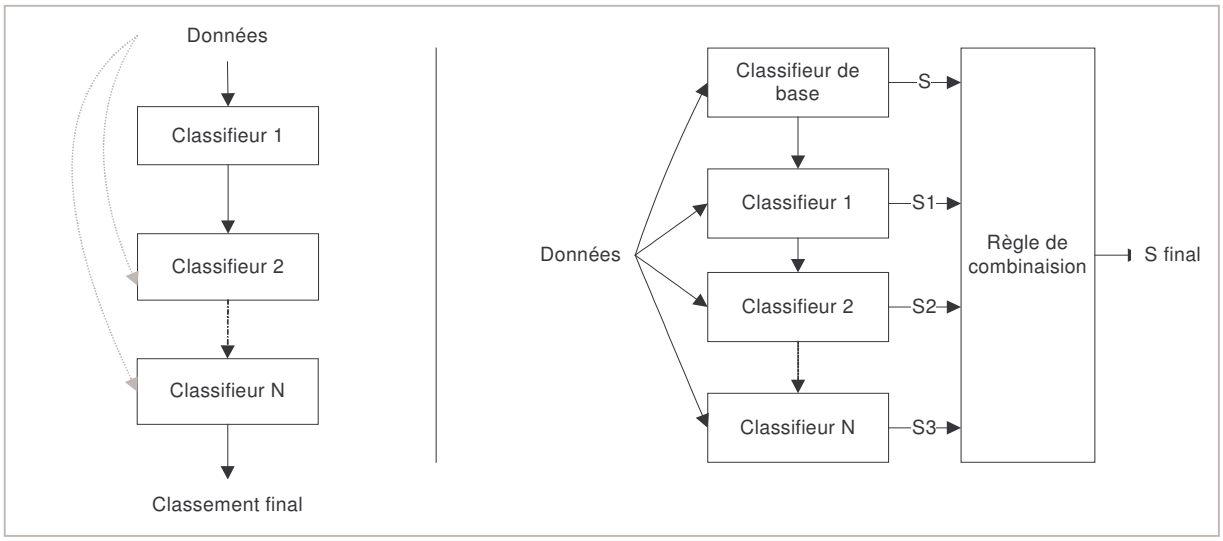


Figure 20 : a -Combinaison séquentielle *b- Combinaison corrective AdaBoost*

Une approche différente est celle de la combinaison par experts qui cherche à décomposer le problème en sous-problèmes comme par exemple une entité dédiée à la caractérisation d'une classe particulière [Oudot, 2004b ; Schwenk, 1998], illustré sur le

schéma Figure 21. La classe finale C correspond à la sortie de distance minimale parmi les N distances fournies par chaque classifieur.

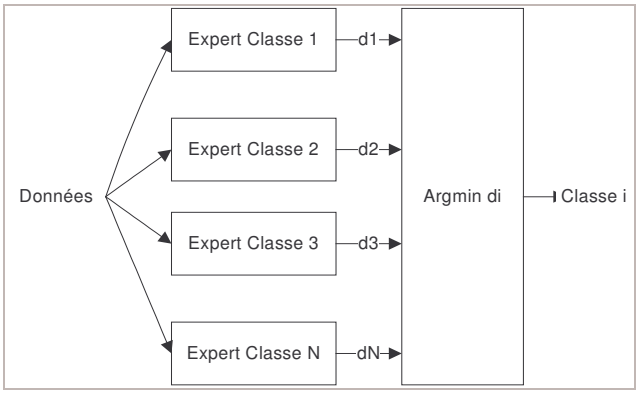


Figure 21 : Combinaison d'experts.

Outre l'intérêt de combiner des classifieurs pour leur vision différente sur des données identiques ou pour pallier leur point faible, il est intéressant de combiner des classifieurs pour traiter des informations de nature différente. Nous nous appuierons au chapitre 5 sur ce type de combinaison pour gérer des informations temporelles et spatiales.

1.5 Conclusion

Dans ce chapitre, nous avons présenté le contexte de nos travaux c'est-à-dire la reconnaissance de l'écriture manuscrite cursive non contrainte pour une application omniscripteur. Nous avons pu souligner les variabilités importantes engendrées par l'écriture naturelle et leurs conséquences sur le choix de leur représentation et d'une technique de reconnaissance.

Cette vision permet de valider nos choix en terme de techniques utilisées : les réseaux de neurones couplés à une modélisation markovienne. La coopération de deux approches, l'une basée niveau caractères et l'autre basée niveau mot, est communément appelée modélisation hybride (RN-MMC, MMC-RN, MMC-MMC , k-ppv-MMC, SVM-MMC). L'apprentissage global de tels systèmes est encore aujourd'hui un problème délicat, contourné aujourd'hui soit par des approches INSEG ou des contraintes fortes demandées à l'utilisateur. Avant d'aborder plus précisément les principes liés à la conception de notre approche, nous nous focaliserons dans le chapitre suivant sur la topologie des réseaux de neurones choisis c'est-à-dire les réseaux de neurones à convolution. Ceux-ci présentent l'avantage de simplifier considérablement les coûts de stockage de l'information nécessaire à une bonne généralisation.

2 LES RESEAUX DE NEURONES A CONVOLUTION

Ce chapitre est consacré aux réseaux de neurones et plus particulièrement aux réseaux de neurones à convolution que nous avons choisis pour la reconnaissance de caractères et qui sera aussi utilisé comme module important dans notre système de reconnaissance de mots.

Afin de comprendre notre solution, nous introduirons dans un premier temps quelques propriétés sur les réseaux de neurones et les architectures classiques rencontrées dans la littérature. Nous détaillerons une architecture de référence : le perceptron multi-couches, qui sera une des principales références de comparaison de nos différents reconnaisseurs. Puis nous aborderons les approches que nous avons retenues aux travers deux travaux de référence dans la littérature : le Pénacée [Guyon, 1991 ; Guyon, 1995] et LeNet [LeCun, 1998a].

Nous conclurons sur les conséquences qu'impliquent ces choix sur la conception du système complet de reconnaissance de caractères et celui de reconnaissance de mots.

2.1 Les réseaux de neurones artificiels

2.1.1 Définition du neurone formel

Un modèle neuromimétique, appelé aussi réseau de neurones artificiels (RNA), ou modèle connexionniste [McClelland, 1986], est formé d'un grand nombre de cellules élémentaires simples, fortement interconnectées. La cellule élémentaire est appelée « *neurone* », son fonctionnement est fondé sur celui d'un automate proposé comme une approximation du fonctionnement du neurone biologique [McCulloch, 1943].

La définition mathématique d'un neurone formel est donnée par la Figure 22.

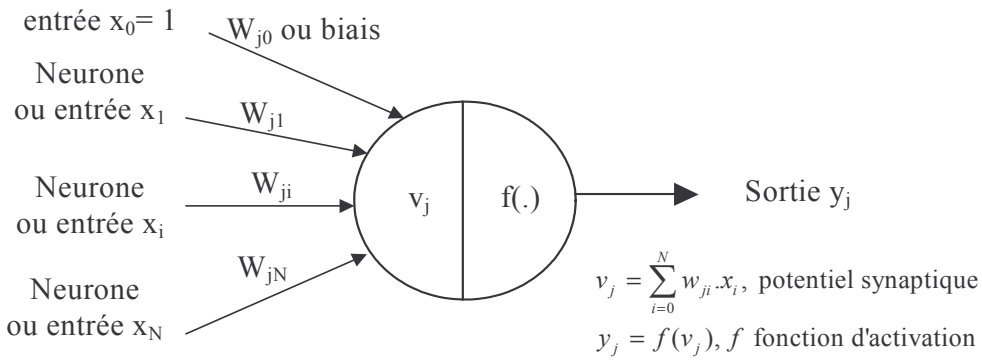


Figure 22. Modélisation d'un neurone

Un neurone j est défini par les paramètres suivants :

1. un vecteur d'entrées $X_j=(x_i)_N$ de taille N fixée
2. un vecteur de poids $W_j=(w_{ji})_N$

3. et un biais, dit aussi seuil qui peut être traité comme un poids supplémentaire auquel on présente une activité constante égale à 1
4. un potentiel synaptique v_j, avec $v_j = \sum_i w_{ji} \times x_i$
5. une fonction d'activation appelée aussi fonction de transfert f. Une forme analytique très courante pour la décision est la fonction sigmoïde, mais d'autres fonctions peuvent également être utilisées (comme la fonction Softmax).
6. une sortie $Y_j=f(v_j)$. La sortie de cette cellule est ainsi une fonction non linéaire de la somme pondérée de ses entrées.

Un neurone calcule l'équation d'un hyperplan : il convient donc pour les données linéairement séparables. Sa capacité de classification est donc très fortement limitée : partition de l'espace en deux régions, illustrée Figure 23. Nous allons voir dans la suite de ce chapitre comment exploiter des neurones dans le cadre de la reconnaissance de caractères manuscrits, qui est un problème non linéairement séparable. En effet, deux lettres peuvent être à la fois très voisines et très éloignées dans l'espace de représentation choisi.

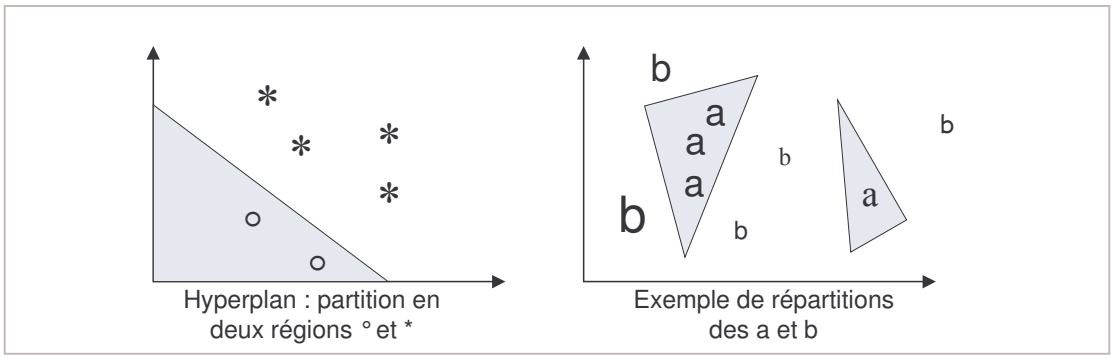


Figure 23. Partitionnement avec un seul hyperplan à gauche et exemple de représentation de lettres « a » et « b » à droite.

2.1.2 Définition d'un RNA

Un RNA est un ensemble de neurones formels (d'unités de calcul simples, de noeuds processeurs) associés en couches (ou sous-groupes) et fonctionnant en parallèle. Selon l'architecture du RNA il sera possible de classifier différentes formes comme l'illustre la Figure 24 et d'obtenir un approximateur universel (dernier cas à droite).

Les RNA ont la capacité de stocker de la connaissance empirique et de la rendre disponible à l'usage. Les habiletés de traitement (et donc la connaissance) du réseau vont être stockées dans les poids synaptiques, obtenus par des processus d'adaptation ou d'apprentissage. L'objectif de l'apprentissage est de fournir une méthode au réseau afin qu'il puisse ajuster ces paramètres lorsqu'on lui présente des exemples à traiter. On distingue habituellement trois paradigmes d'apprentissage : supervisé, non supervisé et hybride.

Dans le cas d'un apprentissage supervisé, on fournit au réseau la donnée à traiter mais aussi la réponse attendue. Le réseau effectue une évaluation de la donnée, puis compare la valeur obtenue avec la valeur désirée, il va ensuite modifier ses paramètres internes afin de minimiser l'erreur constatée. L'apprentissage par renforcement (on parle aussi d'apprentissage par Récompense et Pénalité) est une variante de l'approche supervisée, dans ce cadre on fournit au réseau une critique qui qualifie la réponse calculée.

Dans le second cas, non supervisé, aucune information (en plus des données à apprendre) n'est fournie au système. Celui-ci est amené à découvrir la structure sous-jacente des données afin de les organiser en clusters.

Certains auteurs, utilisent le terme d'apprentissage hybride pour parler d'un couplage supervisé, non supervisé ; dans ce cas il s'agit d'un réseau qui met en parallèle ou en série un réseau entraîné en mode supervisé et un autre en mode non supervisé.

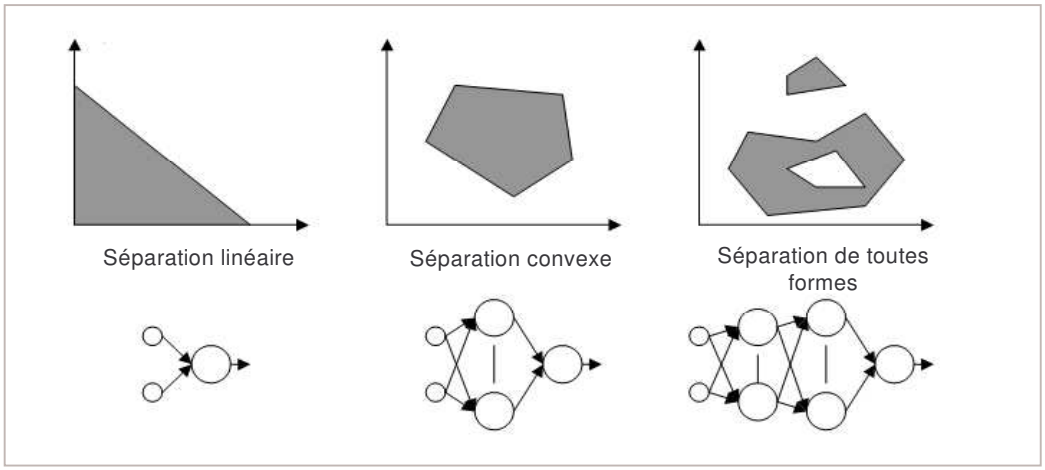


Figure 24. Partitionnements possibles en fonction de l'architecture du RNA

2.1.3 Présentation de quelques RNA.

Un exemple de réseaux de neurones artificiels simples est un réseau à une couche tel que le Percetron de Rosenblatt et Adaline de Widrow et Hoff [Hérault, 1994]. Leurs règles d'apprentissage reposent sur la minimisation d'un critère d'erreur quadratique instantanée. Dans Adaline (ADAptive Linear Neuron) le neurone est linéaire : sa fonction de transfert correspond à la fonction identité f(v)=v, alors que celle du Perceptron est une fonction échelon soit f(v)=Echelon(v-θ) avec θ un seuil de décision. L'apprentissage se fait de manière supervisée, c'est-à-dire que l'on présente un vecteur à l'entrée du réseau et en même temps on fournit la sortie désirée (potentiel désiré $v_{désiré}$ pour Adaline, sortie désirée $y_{désirée}$ pour le Perceptron). Pour un neurone, à chaque pas d'apprentissage, l'erreur quadratique E à minimiser s'écrit de la façon suivante :

$$E = (v - v_{désiré})^2, \text{erreur multivaluée dans le cas d'Adaline}$$
$$E = (y - y_{désirée})^2, \text{erreur binaire dans le cas du Perceptron}$$

Le principe de la minimisation de l'erreur repose sur une méthode de gradient. On calcule le gradient de l'erreur par rapport aux poids w_k, et l'on effectue une correction en sens inverse sur les poids. Dans l'algorithme du Perceptron, les poids sont modifiés selon le principe du gradient stochastique :

$$\Delta w_k = -\mu(y - y_{désirée})x_k$$
$$\text{car } \frac{\partial E}{\partial w_k} = 2(y - y_{désirée})\frac{\partial y}{\partial w_k} \text{ et } \frac{\partial y}{\partial w_k} = \frac{\partial \sum_j w_j x_j}{\partial w_k} = x_k$$

avec μ un gain d'adaptation .

Ce type de réseau mono couche très simpliste a été vivement critiqué notamment par Minski et Paper [Minski, 1969] mais fut à la base des travaux futurs en classification, identification, prédiction… En effet ces mémoires associatives linéaires simples sont contraintes par leurs topologies à résoudre des problèmes linéairement séparables. Une solution à cette limitation sont les réseaux multicouches considérés comme des approximateurs universels dont les plus étudiés et appliqués en général sont les perceptrons multicouches [Bishop, 1995].

Parmi les modèles les plus utilisés en reconnaissance de mots, on peut citer une majorité de réseaux dit non bouclés tels que les perceptrons multicouches, les réseaux à convolution, particulièrement les réseaux multicouches à retard (ou dit à délai), les réseaux bayésiens et quelques réseaux bouclés dits aussi récurrents.

Les perceptrons sont des réseaux sans contre réaction : les sorties des neurones d'une couche forment les entrées de la couche suivante. La Figure 25 montre la structure d'un perceptron à trois couches dont une couche cachée. Ce modèle est issu des travaux de Rosenblatt sur le perceptron monocouche [Rosenblatt, 1962].

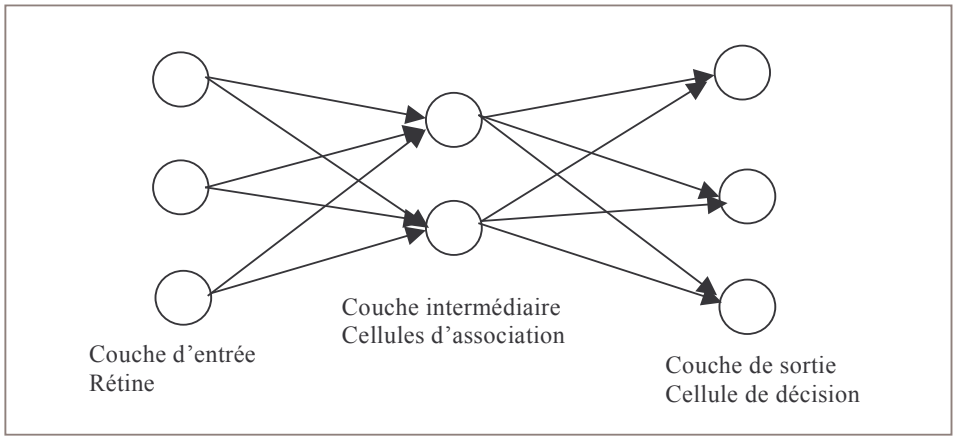


Figure 25. Perceptrons à trois couches

Les réseaux multicouches à retard, ou *Time-Delay Neural Networks* (TDNN) (introduit par Waibel en reconnaissance de la parole [Waibel, 1988] puis en écriture [LeCun, 1995 ; Guyon, 1995], sont des perceptrons capables de traiter des séquences de vecteurs d'écriture grâce à l'introduction de retards temporels fixes sur les entrées. Chaque neurone d'un TDNN traite ainsi le vecteur d'écriture présenté à l'instant t sur la couche d'entrée et les N vecteurs précédents. L'apprentissage des poids des différentes connexions est effectué à l'aide d'un algorithme adapté de l'algorithme classique de rétropropagation de l'erreur [LeCun, 1998b].

Une autre approche pour le traitement de séquences de formes consiste à utiliser des réseaux récurrents. Comme le montre la Figure 26 dans le cas particulier d'un réseau multi-couches, l'état d'un neurone est ici fonction du vecteur d'entrée courant et de l'état du réseau à l'instant précédent. Ces réseaux apportent une réponse plus générale que les réseaux à retard au problème du traitement du temps.

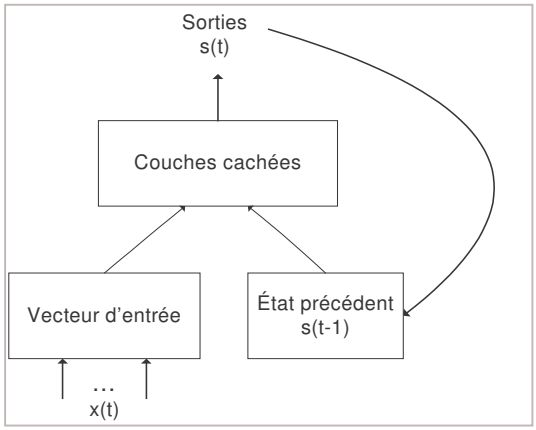


Figure 26. Principe d'un réseau récurrent

Différents réseaux récurrents ont été conçus et testés en reconnaissance de la parole et de l'écriture [Bourlard, 1989 ; Senior, 1998]. Ces réseaux sont le plus souvent seulement partiellement récurrents et constitués d'un perceptron multicouches avec une boucle de retard permettant de réinjecter à l'entrée la valeur de la couche de sortie et, parfois, des couches cachées.

2.2 Le perceptron multi-couches

Même si les perceptrons multi-couches sont souvent qualifiés de boîtes noires difficiles à interpréter, implémenter et à régler, ils ont pris une place importante dans les domaines de la reconnaissance (écrit, parole, visage) et de l'identification (signature). Pour bien comprendre la philosophie de cet outil, nous allons détailler la structure d'un perceptron multi-couches (PMC) et son mode d'apprentissage le plus classique dans le domaine de l'écrit. Ainsi nous pourrons le comparer à notre approche dérivée du PMC, les réseaux à convolution que nous aborderons à la suite.

2.2.1 Structure d'un PMC

Un perceptron multi-couches se caractérise par trois types de couche : la couche d'entrée, la ou les couches cachées et la couche de sortie. Chaque neurone d'une couche est connecté à tous les neurones de la couche inférieure, on dit que le réseau est complètement connecté.

Le schéma suivant représente un PMC à 4 couches. La couche d'entrée comporte 3 caractéristiques. La première couche cachée contient cinq neurones, la seconde quatre. La couche de sortie, étape de décision ne contient ici qu'un seul neurone. Les couches cachées ont pour but d'extraire de l'espace d'entrée de l'information pertinente pour résoudre le problème de décision, les entrées pouvant être brutes (données originales bien que normalisées) ou déjà de plus haut niveau (type caractéristiques en-ligne vu dans le chapitre précédent).

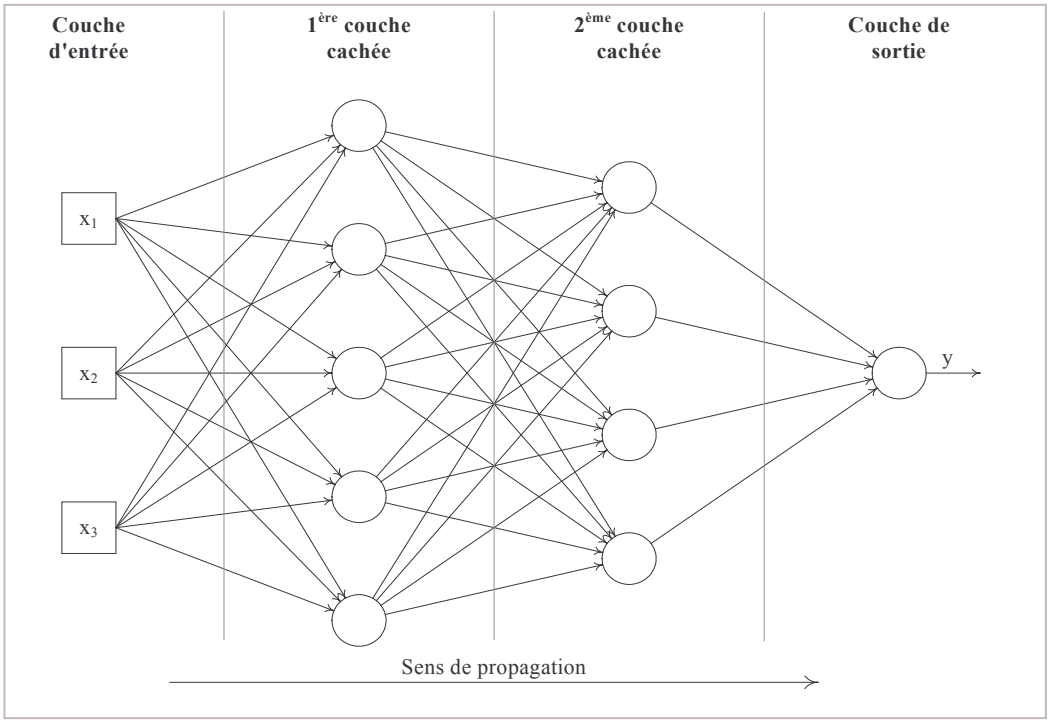


Figure 27. Schéma d'un perceptron multi couche

Le calcul de la sortie se fait en propageant de gauche à droite les calculs, calcul des sorties de chaque neurone de la première couche cachée puis ceux de la seconde couche cachée et enfin la sortie y. Dans un problème de reconnaissance de caractères ou chiffres isolés, il y a en sortie du réseau en général autant de neurones que de classes à reconnaître (par exemple 10 dans le cas des chiffres de 0 à 9). La décision finale en reconnaissance de l'écriture se caractérise en général par un vecteur des probabilités d'une classe par rapport à l'entrée. Pour obtenir des sorties de type probabilités (entre 0 et 1) [Bridle, 1990], on utilise en sortie comme fonction de transfert la fonction dite Softmax.

$$\text{Softmax}(v_i) = P(Classe = Ci \,|\, X = x) = \frac{e^{v_i}}{\sum_k e^{v_k}}$$

avec

i étant l'indice du neurone correspondant à la classe Ci

k étant l'indice balayant tous les neurones de la même couche

v le potentiel synaptique du neurone.

x l'entrée des neurones

Tandis que pour les couches cachées le choix de la fonction de transfert réside dans sa dérivabilité et sa simplicité de calcul pour l'apprentissage comme par exemple la fonction sigmoïde ou la fonction tangente hyperbolique, sigmoïde ramenée entre –1 et 1.

$$a - Sigmoide(v_i) = \frac{1}{1+e^{-v_i}} \text{ entre [0 et 1]}$$

$$b - Tangente\ hyperbolique(v_i) = \frac{\sinh(v_i)}{\cosh(v_i)} = \frac{e^{2v_i}-1}{e^{2v_i}+1}$$

La Figure 28 illustre les fonctions sigmoïde et tangente hyperbolique.

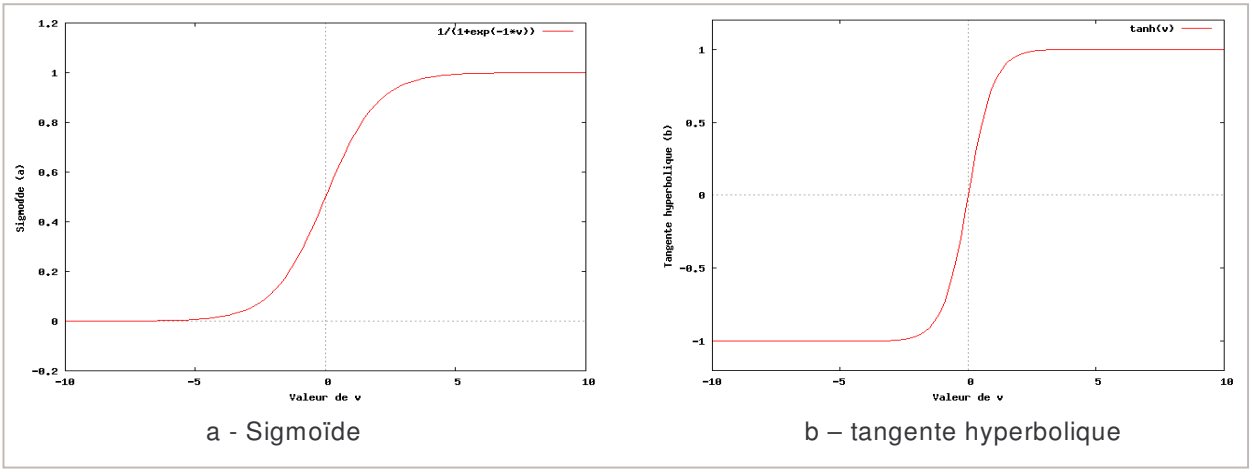


Figure 28. Fonctions de transfert de type sigmoïde

2.2.2 Apprentissage

Avant de pouvoir associer un élément à une classe, il faut avoir appris à discriminer cette classe et être capable de généraliser le problème avec un nombre réduit d'échantillons en apprentissage car il n'est pas possible dans un contexte omni scripteur de récupérer tous les prototypes possibles.

2.2.2.1 Base d'apprentissage

L'apprentissage se fait en règle générale de manière supervisée. On possède une base de données, suffisamment importante par rapport au nombre de poids du PMC. Cette base est découpée en deux voire trois sous-bases. La première sous-base, appelée base d'apprentissage, permet comme son nom l'indique d'apprendre les poids du réseau de neurones. La seconde sous-base ou base de test sert à régler les méta-paramètres de l'algorithme d'apprentissage (nombre de couches cachées, nombres de neurones, pas de gradient, …). Elle fournit une évaluation optimiste de l'erreur, quant à la troisième sous-base, lorsqu'elle est utilisée, elle sert de base de validation afin d'évaluer le système sur des exemples non encore vus. Le découpage varie selon le nombre d'échantillons et le nombre de paramètres du PMC. On trouve différents types de découpage :

- Méthode 2/3 1/3, soit 2/3 des échantillons pour la base d'apprentissage et de validation et 1/3 pour la base de test, mesure finale de la performance. C'est le cas des bases de données en-ligne issues de IRONOFF [Viard-Gaudin, 1999] et UNIPEN [Guyon, 1994].
- Méthode des (X-1)/X 1/X. On peut citer la base MNIST [LeCun, 1998a], base de chiffres hors-ligne avec X égal à 7 (60 000 échantillons en apprentissage et 10 000 en test).
- Validation croisée, généralement quand la base est de taille petite par rapport au nombre de poids du réseau de neurones. Il s'agit de séparer la base en L segments : B_1, B_2, … B_L et on crée L ensembles d'apprentissage et ensembles de test de manière cyclique. C'est-à-dire que pour chaque création de nouvel ensemble on va retirer un segment de la base d'apprentissage, celui-ci servira de base de test, et réintroduire le segment de test précédent dans la base.

2.2.2.2 Algorithme d'apprentissage

L'apprentissage est basé sur la correction des erreurs de classements des données. Pendant l'entraînement supervisé, le système modifie graduellement ses paramètres ajustables pour que sa sortie tende vers la sortie désirée.

Notons :

- X_e, le vecteur d'entrée présenté au PMC
- e, un indice de balayage des échantillons de la base d'apprentissage
- w, le vecteur de paramètres du PMC,
- $Y(X_e,w)$ le vecteur de sortie du PMC
- et $Y_d(X_e)$ le vecteur de sortie désiré.

Apprendre les exemples de la base de données revient à minimiser l'erreur globale sur toute la base donnée par l'équation ci-dessous :

$$Erreur\ locale : d(Y(X_e,W),Y_d(X_e))$$
$$Erreur\ globale\ : E(W)=\sum_e d(Y(X_e,W),Y_d(X_e))$$

En général, on utilise principalement, pour adapter les poids, les méthodes utilisant le gradient qui sont très efficaces. Les algorithmes de gradient conjugué sont nombreux mais coûteux en temps, calculs et mémoires. On préfère la méthode de descente de gradient stochastique qui est une méthode itérative et dont la règle de mise à jour est la suivante :

1. initialisation de départ des poids $W^{(0)}$
2. règle de mise à jour :

$$w_{l,j,i}^{n+1}=w_{l,j,i}^{n}-\mu^n * \nabla E(w_{l,j,i}^{n})$$

μ^n étant le pas de descente

l, l'indice de la couche et i indice du neurone.

$$\text{Gradient local : } \nabla E_{loc_e}(w_{l,j,i}^{n})=\frac{\partial d(Y(X_e,W^n),Yd(X_e))}{\partial w_{l,j,i}^{n}}=\frac{\partial d(Y(X_e,W^n),Yd(X_e))}{\partial Y(X_e,W^n)}\frac{\partial Y(X_e,W^n)}{\partial w_{l,j,i}^{n}}$$

$$\text{Gradient total : } \nabla E(w_{l,j,i}^{n})=\sum_e \nabla E_{loc_e}(w_{l,j,i}^{n})$$

La méthode stochastique, soit par optimisation de l'erreur par le gradient local, entraîne bien la convergence, comme la méthode classique. Certains auteurs comme Yann LeCun ont montré dans la pratique que la méthode stochastique est bien plus rapide que le gradient global.

La rétro propagation [LeCun, 1987] est très efficace pour calculer le gradient de la fonction d'erreur d'un perceptron multi couche, comparée à la méthode directe [Rossi, 1996]. Nous allons expliciter cette méthode que nous utiliserons par la suite dans nos systèmes. Pour cela, introduisons dans un premier temps l'algorithme d'apprentissage et ensuite sa mise en œuvre d'un point de vue mathématique.

```
Soit e, indice des exemples de la base d'apprentissage
Soit Nb_e le nombre d'exemples de la base d'apprentissage
Soit c, l'indice de couche cachée du PMC
Soit Nb_c, le nombre de couches cachées du PMC
Tant que le critère d'arrêt n'est pas satisfait
  Pour e=1 à Nb_e
     Tirage d'un exemple de la base de façon aléatoire
      Propagation des caractéristiques de l'échantillon de la
      couche d'entrée vers la sortie
      Mesure de l'erreur en sortie du réseau
     Pour c=Nb_c à 1 (rétro propagation)
        Estimation de l'erreur pour chaque neurone de la couche
        cachée c
     Fin Pour c
     Pour c=1 à Nb_c
        Correction des poids du réseau
     Fin Pour c
  Fin Pour e
  Mesure du critère d'arrêt
Fin Tant que
```

Algorithme 1. Algorithme d'apprentissage du perceptron multi-couches

L'algorithme d'apprentissage se fait donc en trois passages :

1. Le premier de la gauche vers la droite pour propager les données et connaître en sortie l'estimation du réseau, on obtient ainsi un vecteur Y pour l'entrée X_e.
2. Le deuxième de la droite vers la gauche pour calculer les erreurs commises par chaque neurone en commençant par la couche de sortie et en remontant vers la couche d'entrée.
3. Le troisième suit le sens de la propagation pour corriger les poids du réseau.

Le problème du contrôle de l'apprentissage d'un RN est crucial. Un des dangers importants est celui du sur-apprentissage. Cela se traduit par une erreur très faible sur la base d'apprentissage mais par un comportement très médiocre en généralisation sur la base de test. Pour pallier cette situation, essentiellement deux techniques sont disponibles ; celles basées sur l'observation expérimentale de l'évolution de l'erreur (validation croisée) et celles intégrant un terme de généralisation à la fonction de coût (penalty-based).

La méthode *early-stopping* correspond au premier type de méthodes. Elle consiste à garder en mémoire les paramètres du réseau qui ont réalisé le minimum de l'erreur de généralisation (base de test) lors des différentes itérations de l'apprentissage.

Pour les méthodes basées pénalité, on distingue deux types de pénalités : « *weight decay* » et « *weight elimination* ».

Il existe différents critères pour stopper l'apprentissage, les plus utilisées sont la méthode *early stopping* et les méthodes « *penalty-based* ».

Le terme de pénalité pour *weight decay* est :

$$penalty = lambda * \sum_i w_i^2$$

où la somme est prise sur tous les poids de toutes les couches *à l'exception des seuils (biais)*.

Le terme de pénalité pour *weight elimination* est :

$$penalty = lambda * \sum_i \frac{w_i^2}{w_i^2 + c^2}$$

où à nouveau la somme ne comprend pas les seuils. Un choix de c grand favorise beaucoup de petits poids, tandis qu'un choix de c petit avantage quelques poids importants.

2.2.2.3 Calcul du gradient

Dans cette partie nous allons détailler les calculs du gradient pour illustrer l'implémentation de la rétropropagation. Le schéma suivant permet d'introduire les notations nécessaires à la compréhension des équations et explications.

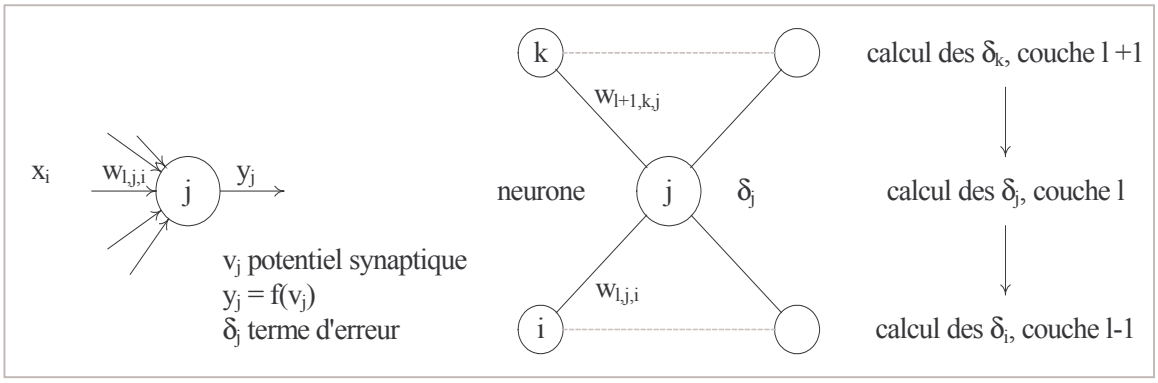


Figure 29. Notations des neurones et poids pour le calcul du gradient

En reconnaissance de caractères, la fonction de coût la plus utilisée est l'erreur quadratique moyenne qui est déterminée pour un exemple e, par le carré de la distance euclidienne entre le vecteur yd^k des sorties désirées (1 ou 0) et le vecteur y^k des sorties calculées (C classes) par le réseau de neurones. Connaissant le label du caractère présenté à l'entrée du réseau de neurones, on peut construire le vecteur des sorties désirées du réseau : mise à 1 de la sortie correspondante à la classe du caractère et mise à zéro pour les autres classes.

$$E_e(w) = \frac{1}{2} \times \sum_{c=1}^{C} (yd_c^e - y_{l,c}^e)^2 \text{ avec } l \text{ indice de la dernière couche du PMC}$$

correction des poids : $w_{l,j,i}^{n+1} = w_{l,j,i}^n - \mu^n \times \nabla E_e(w_{l,j,i})$

l, l'indice de la couche et i,j indice du neurone.

Gradient $\nabla E_{loc_e}(w_{l,j,i}^n) = \frac{\partial}{\partial w_{l,j,i}^n}(E_e(w))$

Par souci de clarté, nous ne gardons pas l'indice de l'exemple.

Gradient $\nabla E_{loc}\ (w^n_{l,j,i}) = \frac{\partial E(w)}{\partial v_{l,j}} \times \frac{\partial v_{l,j}}{\partial w^n_{l,j,i}}$

et $\frac{\partial v_{l,j}}{\partial w^n_{l,j,i}} = \frac{\partial}{\partial w^n_{l,j,i}} (\sum_k w^n_{l,j,k} \times x_{l-1,k}) = x_{l-1,i}$

d'où $\nabla E_{loc}\ (w^n_{l,j,i}) = \frac{\partial E(w)}{\partial v_{l,j}} \times x_{l-1,i}$ on pose $\delta_{\mathrm{l,j}} = \frac{\partial E(w)}{\partial v_{l,j}}$

Pour calculer le gradient, on mémorisera l'activation de sortie `x`, le potentiel synaptique `v` et son terme d'erreur δ. Le calcul du terme d'erreur diffère en fonction du niveau de la couche. Dans un premier temps, on calcule ce terme sur la couche de sortie dont l'équation est différente de celles des couches cachées.

Couche de sortie :

$$\delta_{\mathrm{l,j}} = \frac{\partial E(w)}{\partial v_{l,j}} = \frac{\partial E(w)}{\partial x_{l,j}} \times \frac{\partial x_{l,j}}{\partial v_{l,j}}$$

$$\frac{\partial x_{l,j}}{\partial v_{l,j}} = \frac{\partial}{\partial v_{l,j}} fs(v_{l,j}) = fs'(v_{l,j})$$

avec fs fonction Softmax $x_{l,j} = fs(v_{l,j}) = \frac{e^{v_{l,j}}}{\sum_k e^{v_{l,k}}}$ et fs' sa dérivée

$$\frac{\partial E(w)}{\partial x_{l,j}} = \frac{\partial}{\partial x_{l,j}} (\frac{1}{2} \times \sum_{c=1}^{C} (yd_c - y_{l,c})^2) = -(yd_j - y_{l,j}) \qquad (\text{car } x_{l,j} \equiv y_{l,j})$$

d'où $\delta_{l,j} = (y_{l,j} - yd_j) \times fs'(v_{l,j})$

L'erreur en sortie du réseau est visible, elle est égale à la différence entre le vecteur désiré et la sortie obtenue en propageant l'entrée. Pour les couches cachées, les erreurs sont cachées, il faut évaluer l'erreur qu'un neurone j d'une couche l a propagée dans les neurones de la couche supérieure l+1 (cf. Figure 29).

Couches cachées :

$$\delta_{\mathrm{l,j}} = \frac{\partial E(w)}{\partial v_{l,j}} = \sum_k \frac{\partial E(w)}{\partial v_{l+1,k}} \times \frac{\partial v_{l+1,k}}{\partial v_{l,j}}$$

$$or\ \frac{\partial E(w)}{\partial v_{l+1,k}} = \delta$$

et $\frac{\partial v_{l+1,k}}{\partial v_{l,j}} = \frac{\partial v_{l+1,k}}{\partial x_{l,j}} \times \frac{\partial x_{l,j}}{\partial v_{l,j}}$

$$\frac{\partial x_{l,j}}{\partial v_{l,j}} = \frac{\partial}{\partial v_{l,j}} fsig(v_{l,j})$$

avec $fsig$ fonction Sigmoide $x_{l,j} = fsig(v_{l,j})$

et $fsig'(v_{l,j}) = fsig(v_{l,j}) \times (1 - fsig(v_{l,j}))$ sa dérivée

$$\frac{\partial v_{l+1,k}}{\partial x_{l,j}} = \frac{\partial}{\partial x_{l,j}} (\sum_m (w_{l+1,k,m} \times x_{l,m})) = \sum_m (w_{l+1,k,m} \times \frac{\partial x_{l,m}}{\partial x_{l,j}}) = w_{l+1,k,j}$$

d'où $\delta_{l,j} = \frac{\partial E(w)}{\partial v_{l,j}} = \sum_k \delta_{l+1,k} \times w_{l+1,k,j} \times fsig'(v_{l,j})$

soit $\delta_{l,j} = \frac{\partial E(w)}{\partial v_{l,j}} = fsig'(v_{l,j}) \times \sum_k \delta_{l+1,k} \times w_{l+1,k,j}$

Ainsi pour les couches cachées, l'erreur est égale à la somme des erreurs en aval pondérées par les poids correspondant aux liaisons et la pente de son activation.

Une fois tous les termes d'erreur calculés en commençant par la couche de sortie et en remontant vers la couche d'entrée on peut appliquer la formule de correction des poids tout en choisissant un pas adapté. Le choix du pas est un réglage essentiel pour assurer la bonne convergence de la minimisation de la fonction de coût choisie.

2.2.2.4 Apprentissage stochastique élargi

Il a été montré de manière empirique que l'apprentissage stochastique est plus rapide et moins coûteux qu'un apprentissage global. La correction des poids pour chaque exemple présenté est concevable et rapide, avec les technologies des processeurs et capacités de stockage actuelles, dans le cadre de la reconnaissance de caractères ou chiffres isolés. Dans le cadre de la reconnaissance de mots ou phrases avec des lexiques de taille importante, un apprentissage peut s'avérer très coûteux en temps et besoins de stockage. Une solution est donc de faire une correction tous les N échantillons en intégrant les erreurs locales sur ces N échantillons.

2.2.3 Paramétrage

La rétropropagation peut être particulièrement lente pour des réseaux multicouches. Aucune formule ne garantit la convergence du réseau vers l'optimum global. Cependant, il est possible d'augmenter les chances du réseau à converger en paramétrant plusieurs facteurs tels que :

- normalisation des entrées,
- initialisation des poids,
- fonction de transfert de type sigmoïde,
- apprentissage global ou stochastique…

2.2.3.1 Initialisation des poids et choix du pas

L'initialisation [LeCun, 1998] des poids peut avoir une influence significative sur l'apprentissage du réseau. Les poids doivent être choisis de manière aléatoire mais inclus dans la partie linéaire de la sigmoïde. Si leurs valeurs initiales sont trop grandes, la sigmoïde saturera

et entraînera un apprentissage trop lent pour les faibles gradients. L'intérêt de normaliser les poids initiaux dans la partie linéaire de la sigmoïde est important, les gradients seront ainsi suffisamment importants pour permettre un apprentissage correct. En sortie de chaque couche cachée ou sortie on souhaite obtenir une somme unitaire des activations de chaque neurone, une activation proche de un en sortie d'un neurone signifiant la prépondérance de cette caractéristique ou label. En supposant que les entrées x_i soient non corrélées et de variance unitaire, l'écart type de la somme pondérée d'un neurone doit être égal à :

$$\sigma_{x_i} = \sqrt{(\sum_j w_{ij}^2)}$$

Pour assurer un écart type σ_{xi} unitaire, les poids doivent être initialisés aléatoirement à partir d'une répartition de moyenne nulle et un écart type donné par

$$\sigma_w = m^{-1/2}$$

où m , appelé « fan-in » est le nombre d'entrées d'un neurone.

Le pas de gradient, dans la correction des poids, est aussi un facteur déterminant dans l'apprentissage d'un réseau de neurones. Il peut être ajusté automatiquement : on diminue le pas de gradient lorsque la matrice des poids oscille et à l'inverse on l'augmente lorsque celle-ci suit toujours la même direction. Malheureusement, cette méthode n'est appropriée ni à l'emploi d'un gradient stochastique ni pour un apprentissage dynamique. Or le gradient stochastique est le plus usité car un apprentissage stochastique est plus rapide qu'un apprentissage global et donne en général de meilleurs résultats.

Au lieu de choisir un pas de gradient unique pour tous les poids, imposer différents pas pour chaque poids peut accélérer la convergence de la matrice poids vers la solution optimale. Une méthode employée par Yann Le Cun [LeCun, 1993] est l'utilisation de la matrice Hessienne qui mesure la courbure de la fonction de coût à minimiser. La philosophie est de s'assurer que tous les poids convergent grossièrement à la même vitesse.

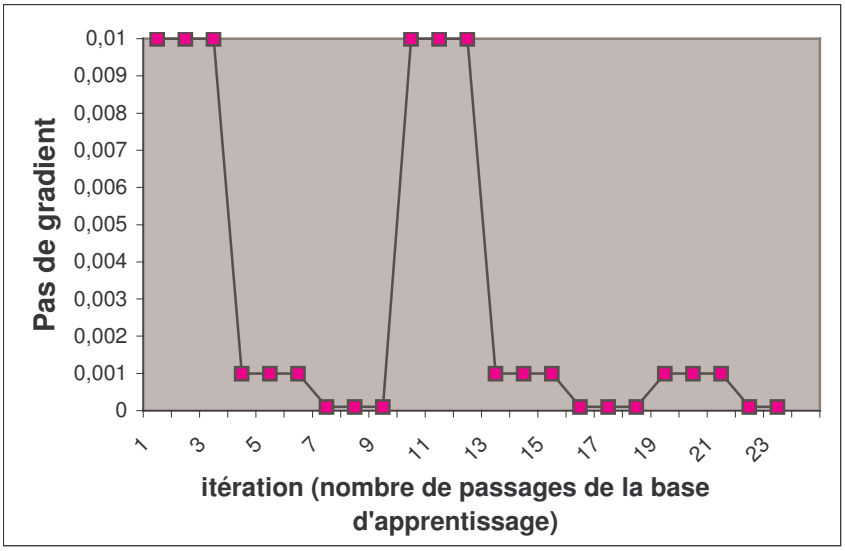


Figure 30. Choix de la valeur du pas pendant l'apprentissage (0,01 - 0,001- 0,0001)

Une autre solution, basée sur des résultats d'expériences [Tay, 2002] est de régler le pas de manière cyclique. Pour commencer, on choisit un pas relativement fort pour accélérer la convergence, ensuite on divise par pas de 10 sa valeur pour affiner la solution puis on réitère ce

schéma afin d'éviter de tomber dans un minimum local. La Figure 30 donne un exemple d'évolution du pas de convergence donné dans la thèse de Tay.

2.2.3.2 Architecture du réseau de neurones

La capacité du réseau de neurones dépend du nombre de paramètres libres (poids du réseau). Pour être capable d'apprendre des surfaces de décision complexe, le PMC doit avoir un nombre de couches suffisant (un minimum de deux couches cachées selon les théorèmes de Kolmogorov et Kurkova) et un nombre suffisant de neurones sur cette (ces) couche(s) cachée(s). Cependant, la qualité des résultats dépend beaucoup de la représentativité de l'ensemble d'apprentissage, des paramètres d'entrée et de sortie.

Il n'existe pas une structure unique du réseau. Différentes topologies du réseau, en termes de taille et type de connexions, peuvent conduire aux mêmes performances. Afin d'optimiser effectivement le RN, il faut faire converger les configurations – initialisation des poids, choix de la fonction d'activation, nombre de neurones, nombre de couches cachées,…- vers une performance maximale. Pour cela, on définit un score, par exemple en tenant compte en sortie du RN du nombre minimum d'itérations qu'il a fallut pour obtenir 100% de résultat (pour la tolérance donnée), du nombre minimum de couches et de neurones, de l'erreur moyenne en sortie, de la stabilité de l'apprentissage (reproductibilité de l'apprentissage avec cette configuration). Nos travaux étant destinés à des applications embarquées de faibles ressources mémoire et calcul, nous définirons le critère d'optimisation de la structure suivant : l'architecture optimale sera celle combinant la plus petite taille et la plus petite complexité de calculs pour réaliser la fonction demandée.

Pour définir l'architecture du réseau, deux méthodes sont utilisées :

- la première, la plus courante est la méthode empirique,
- la deuxième consiste à utiliser des algorithmes d'optimisation automatique.

La première méthode consiste, par l'expérience, à comparer les performances obtenues pour différentes architectures, et à ne conserver ensuite que celle reproductible et correspondant au meilleur compromis performance/coût architectural. La taille de la couche d'entrée est souvent choisie égale à la taille des données. Toutefois, une projection des données dans un espace de dimension plus élevée peut conduire à des résultats intéressants. C'est par exemple le cas des architectures « Square-MLP » où chaque entrée est dédoublée en ajoutant son carré [Flake, 1998]. Le nombre de neurones de la couche de sortie est égal au nombre de classes que l'on cherche à distinguer avec une possibilité d'un neurone supplémentaire pour qualifier les éléments n'appartenant à aucune des classes, appelé classe de rejet [Tay, 2002] ou classe « nil » (classe de non présence de caractères) dans [Schenkel, 1995]. Le nombre de couches cachées et de neurones associés varient selon la précision souhaitée. L'intérêt d'une optimisation empirique réside dans la possibilité d'intégrer des connaissances préalables des données comme par exemple, l'intégration de connaissances temporelles ou spatiales pour les réseaux à convolution que nous présentons à la section suivante.

Concernant les algorithmes d'optimisation automatique de la structure d'un réseau de neurones, on peut citer les algorithmes de croissance et de dégénérescence de réseaux de neurones. Comme son nom l'indique le but de l'algorithme de croissance est d'effecteur l'apprentissage avec au départ une architecture très petite et de l'accroître en fonction des performances obtenues sur la base d'apprentissage. Les algorithmes de dégénérescence à l'inverse partent d'un réseau surdimensionné que l'on simplifie au cours de l'apprentissage par diminution du nombre de neurones et de connexions. Nous n'approfondirons pas à ces méthodes, nous pouvons citer les travaux de Torres-Moreno dont la thèse [Torres-Moreno,

1997] expose les différentes méthodes de calculs des capacités d'un réseau de neurones et propose des algorithmes constructifs.

Une autre approche, le méta-apprentissage, est une passerelle entre la méthode empirique et la méthode d'optimisation automatique. Les principaux opérateurs utilisés en reconnaissance d'écriture sont le boosting [Schwenk, 1997] et le bagging, qui correspondent au lancement de plusieurs sessions d'apprentissage en pondérant au fur et à mesure, et de manière sélective, les individus de la base. Nous renvoyons à l'article [Opitz, 1999] qui détaille le fonctionnement de ces différents opérateurs et les compare. Le méta-apprentissage consiste à apprendre à apprendre : on utilise par exemple un méta-réseau de neurones pour apprendre à configurer un réseau de neurones classique, qui lui s'occupe d'un problème concret. Le méta-réseau[12] est un RN qui apprend à reconnaître une bonne configuration (sans la tester) afin d'optimiser les tirages aléatoires. En entrée du méta-RN on indique les configurations testées, et en sortie les scores obtenus correspondants. Ensuite, seules les configurations (toujours tirées au hasard) dont le méta-RN juge intéressantes seront réellement testées.

2.3 Les réseaux de neurones à convolution (RNC)

Le réseau de neurones est un outil de représentation des connaissances qui présente les avantages d'une part d'une bonne tolérance aux bruits et un apprentissage automatique des poids avec une forte capacité de généralisation. Nous avons souligné les principes du perceptron multi-couches qui s'appuient sur des entrées de dimension fixe et une vision globale des entrées. Les aspects temporels et 2D ainsi que les traits locaux de l'écriture sont peu exploités dans cette architecture alors que dans un réseau à convolution, il est possible d'intégrer ces notions que nous allons détailler dans la suite de ce chapitre.

2.3.1 Caractérisation des RNC

La principale différence entre les réseaux de neurones classiques de type perceptron et les réseaux de neurones à convolution réside dans leur architecture et plus précisément dans leur approche connexionniste. La Figure 31 illustre cette différence.

Un neurone d'une couche d'un perceptron est connecté à tous les neurones de la couche précédente tandis que pour un réseau de neurones à convolution, un neurone est connecté à un sous-ensemble de neurones de la couche précédente. Chaque neurone peut être vu comme une unité de détection d'une caractéristique locale.

Les réseaux de neurones à convolution (RNC) incorporent des contraintes et réalisent un certain degré d'invariance de décalage et de déformation en utilisant trois idées : zones réceptives locales, poids partagés, et sous-prélèvement spatial. L'utilisation des poids partagés réduit le nombre de paramètres dans le système facilitant la généralisation. Ce type de réseau a été avec succès appliqué à la reconnaissance des chiffres [LeCun, 1990].

Le concept de poids partagés [Shawe-Taylor, 1994 ; Ridder, 1996] se base sur le principe que dans le cerveau humain des neurones détectent certains traits dans de petites régions de la rétine, essentiellement de la même manière dans toutes ces régions. On a ainsi plusieurs neurones qui calculent la même fonction sur des entrées différentes. Ainsi découle le fait qu'il soit nécessaire qu'ils partagent les mêmes poids.

[12] RN autoconfigurant sur http://patrice.dargenton.free.fr/ia/ialab/rnautoconfigurant.html

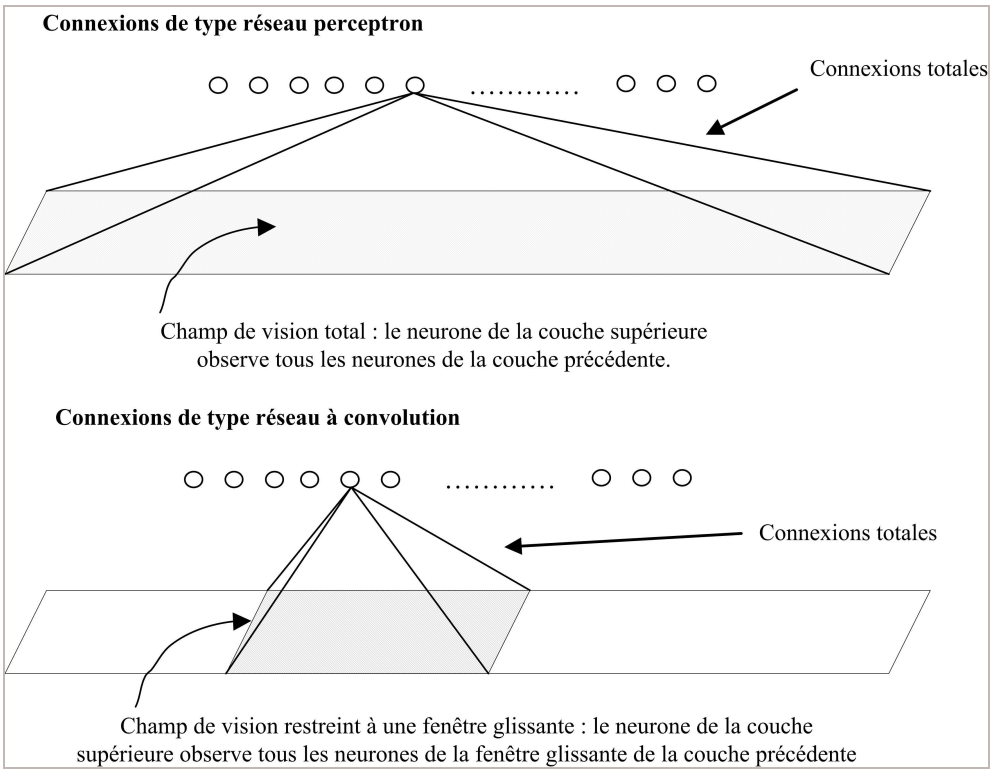


Figure 31. Illustrations des connexions dans un PMC et dans un RNC

2.3.2 Deux topologies : TDNN – SDNN

Deux grands types de réseaux à convolution sont développés dans la littérature pour la reconnaissance de caractères manuscrits : les TDNN, Time Delay Neural Networks (soit réseau de neurones à délais temporels) et les SDNN, Space Displacement Neural Networks (soit réseau de neurones à déplacement spatial). Le TDNN est un réseau à délai utilisé pour des données de nature séquentielle, donc adapté pour la reconnaissance de l'écriture en-ligne tandis que les SDNN sont dédiés à des données de nature spatiale, soit pour l'écriture hors-ligne. La Figure 32 illustre les entrées de chaque type de réseau et le sens de déplacement de la fenêtre de convolution, appelée aussi fenêtre glissante.

A chaque couche du réseau à convolution, différentes fenêtres de convolution peuvent être paramétrées en taille et en délai (surface de recouvrement). A chaque fenêtre correspond une matrice de poids. Pour compenser la réduction importante du nombre de paramètres libres liée à la taille de la fenêtre glissante, on multiplie le nombre de traits caractéristiques dans les couches cachées. Ainsi un TDNN aura une couche cachée à 2 dimensions, un axe temporel (axe t sur le schéma suivant) et un axe des caractéristiques (axe f) et respectivement trois pour un SDNN, deux axes selon les directions horizontale et verticale (axes x et y) et un selon celui des caractéristiques.

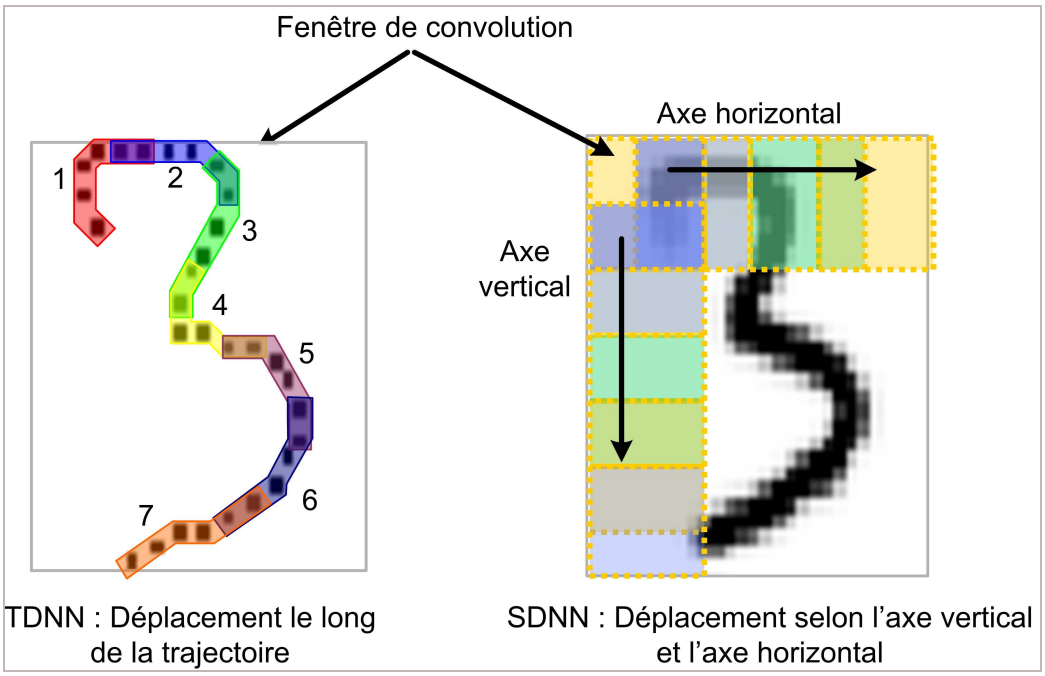


Figure 32. Déplacement de la fenêtre de convolution dans les cas temporel et spatial

La topologie des RNC se caractérise par deux parties illustrées à la Figure 33. La première, correspondant aux couches basses, implémente les convolutions successives permettant de transformer progressivement une séquence de vecteurs caractéristiques en une autre séquence de vecteurs caractéristiques d'ordre supérieur. La seconde correspond à un PMC classique, elle reçoit en entrée l'ensemble des sorties de la partie extraction.

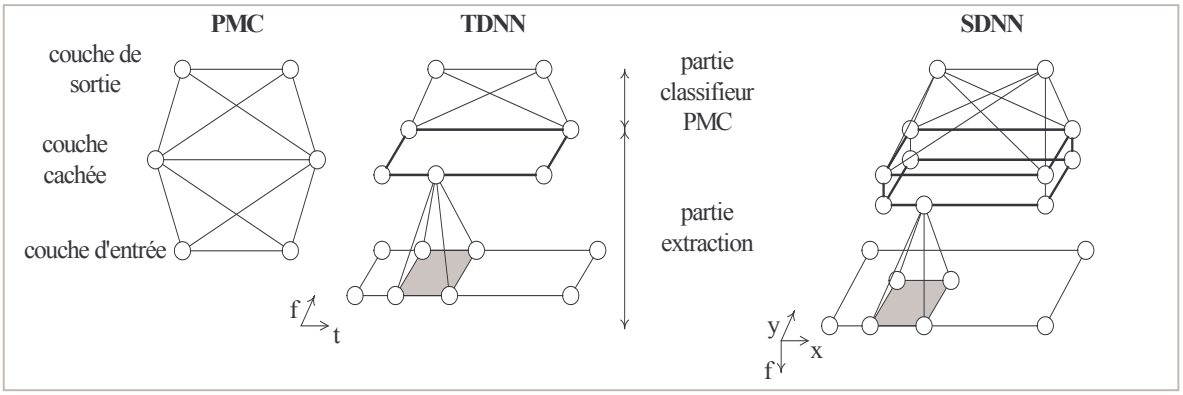


Figure 33. Différences de structures entre PMC, TDNN et SDNN

Afin de bien illustrer ces deux topologies, nous allons dans la suite détailler deux architectures de référence sur lesquelles se sont basées nos études : Le Pénacée et LeNet.

2.3.3 TDNN : Le Pénacée

Isabelle Guyon et Yann LeCun ont développé un TDNN pour classifier des chiffres et lettres majuscules saisis en ligne avec une architecture particulière décrite ci-après [Guyon, 1991]. Le réseau a été entraîné à partir d'une base de 12 000 chiffres et lettres majuscules pour l'apprentissage et 2 500 pour la généralisation d'environ 250 scripteurs différents.

L'entrée du TDNN correspond à une séquence de caractéristiques des points d'un caractère. A partir des coordonnées fournies par la tablette, on rééchantillonne spatialement ces points pour obtenir un nombre fixe de points équidistants. Ensuite un pré-processeur calcule les caractéristiques normalisées de chacun de ces points. Isabelle Guyon exploite les caractéristiques suivantes : la position du stylet (levé ou posé), les coordonnées du point, la direction et la courbure de la trajectoire, soit 7 neurones sur l'axe des caractéristiques de la première couche.

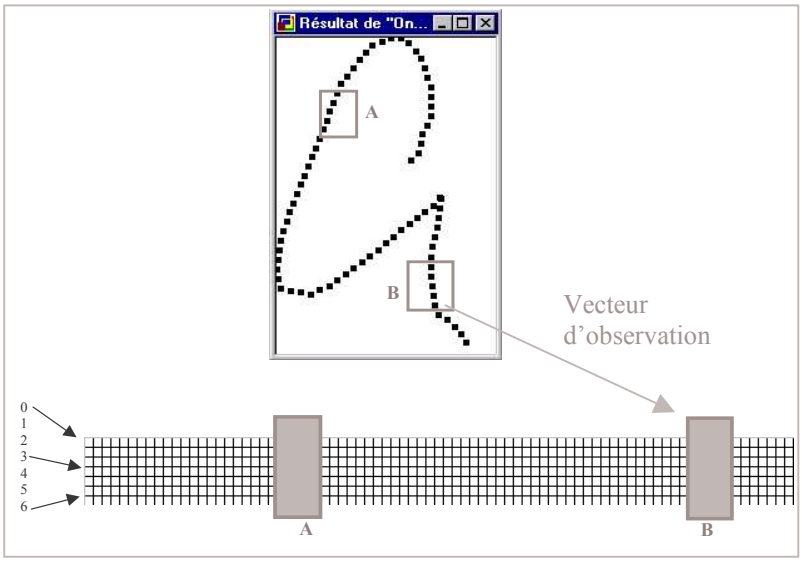


Figure 34. Illustration de l'entrée du TDNN avec le caractère « a »

La Figure 34 illustre le caractère « a » rééchantillonné et sa représentation intermédiaire correspondant à la séquence temporelle des vecteurs comprenant les sept caractéristiques de chaque échantillon. Le temps augmente selon l'axe horizontal de gauche à droite. Chaque vecteur est représenté par une colonne quadrillée.

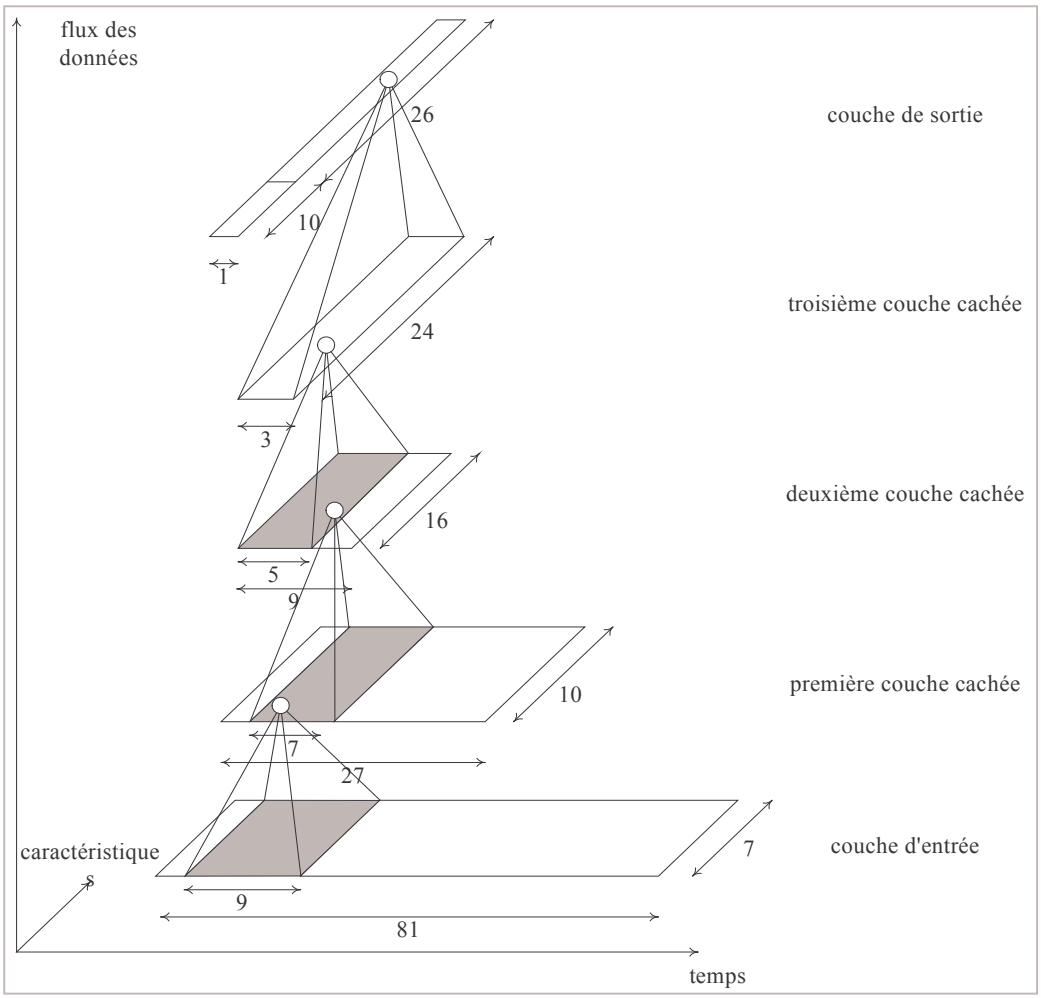


Figure 35. Représentation du TDNN de [Guyon, 1991]

La Figure 35 donne la configuration finale présentée dans [Guyon, 1991]. Le TDNN est composé au total de 5 couches : une couche d'entrée de 81 points avec 7 caractéristiques, trois couches cachées et une couche de sortie liée à un classifieur linéaire. La partie TDNN, extraction des caractéristiques se caractérise par 3 couches :

La première couche cachée est caractérisée par

- 27 neurones sur l'axe temporel
- 10 neurones sur l'axe des caractéristiques
- une fenêtre de convolution de longueur 9 neurones selon l'axe temporel
- un délai temporel entre chaque fenêtre de 3 neurones

La deuxième couche cachée est caractérisée par

- 9 neurones sur l'axe temporel
- 16 neurones sur l'axe des caractéristiques
- une fenêtre de convolution de longueur 7 neurones selon l'axe temporel
- un délai temporel entre chaque fenêtre de 3 neurones

La troisième couche cachée est caractérisée par

- 3 neurones sur l'axe temporel
- 24 neurones sur l'axe des caractéristiques.
- une fenêtre de convolution de longueur 5 neurones selon l'axe temporel
- un délai temporel entre chaque fenêtre de 3 neurones

Et la partie TDNN classifieur se caractérise par 2 couches dont une d'adaptation :

- La première couche du classifieur est en fait la duplication de la dernière couche de la phase extraction d'où 3x24 neurones sur cette couche.
- La deuxième couche correspond à la couche de sortie, elle comprend 36 neurones de sortie, 10 pour les chiffres et 26 pour les majuscules.

Le réseau avec cette configuration possède 35 964 connexions mais seulement 6 345 poids indépendants. Les performances annoncées sur une base de test de 2500 exemples sont un taux d'erreur de 3,4% ; 2,3% si on ne considère que les chiffres et 3,8% si on ne considère que les lettres majuscules.

D'autres simulations ont été réalisées avec pour seuls paramètres le nombre de neurones selon l'axe temporel de la première couche et celui de la seconde couche du TDNN. Les résultats sont résumés dans le Tableau 1.

Tableau 1 . Illustration de l'influence du nombre de connexions sur les taux de reconnaissance

Nombre de neurones selon t sur la 1ère couche	Nombre de neurones selon t sur la 2ème couche	Taux approché d'erreur en test %	Nombre de poids indépendants	Nombre total de connexions
24	8	8	4 748	38 516
16	16	3.8	7 668	36 036
8	16	3.9	6 740	20 996
8	24	3.7	9 948	27 156

Ce tableau illustre, comme il a été cité au paragraphe précédent, l'importance du nombre de paramètres libres, nombre de poids indépendants, dans l'efficacité d'un réseau de neurones. Réduire le nombre de neurones (axe temporel) de la première couche de 16 à 8 neurones ne montre pas une dégradation significative. Inversement, réduire le nombre de neurones sur la deuxième couche de 16 à 8 neurones a des conséquences évidentes : doubler le nombre d'erreurs de reconnaissance. Toutefois, augmenter à 24 le nombre de neurones sur la seconde couche n'améliore pas considérablement les performances du TDNN mais augmente le coût de calcul soit la taille du TDNN.

2.3.4 SDNN : LeNet

Comme le TDNN, un SDNN est un réseau de neurones à convolution. Ce réseau n'exploite plus la notion temporelle et les contraintes de causalité mais s'intéresse à des données spatiales. Le SDNN est une généralisation du TDNN à une topologie 2D.

LeNet et ses successeurs sont basés sur une structure SDNN à l'origine destinés à la reconnaissance de chiffres numérisés (application postale). Ces réseaux sont généralement organisés en couches de différentes sortes : les couches de convolution et les couches de sous échantillonnage.

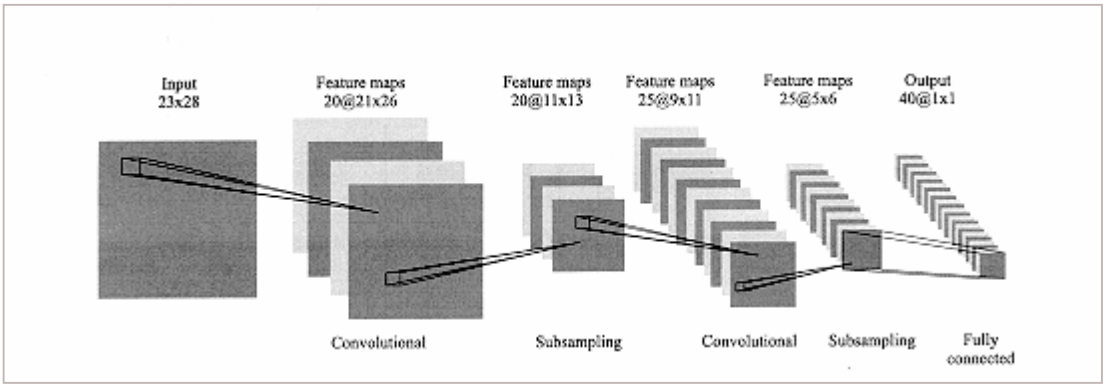


Figure 36. Représentation de système LeNet5 : SDNN [LeCun, 1998a]

La Figure 36 illustre LeNet5. Il est composé de 5 couches. La première couche du SDNN extrait 20 primitives différentes pour 28*28 localisations différentes sur l'image d'entrée, chaque neurone d'une primitive a 5*5 entrées et partage ses poids avec les neurones de la même primitive pour limiter le nombre de connexions libres. La deuxième couche correspond à un sous échantillonnage des primitives. Les couches suivantes reprennent l'idée d'extraction de primitives et sous échantillonnage. Les dernières couches sont complètement connectées avec des sorties gaussiennes pour la classification.

2.4 Conclusion

Dans ce chapitre nous avons détaillé l'architecture des réseaux de neurones de type perceptron multi couches. Ces réseaux relativement faciles à implémenter nous serviront de référence pour tous nos travaux. C'est la raison pour laquelle nous avons tenu à détailler son architecture et son mode d'apprentissage. Nous avons opté pour une solution basée sur les réseaux de neurones à convolution qui sont des cas particuliers des PMC. Ils présentent l'avantage d'avoir le même mode d'apprentissage et bénéficie d'un nombre réduit de paramètres par rapport au perceptron classique. Ce gain de stockage est important pour des applications de faible capacité qui veulent offrir un maximum de fonctionnalités possibles comme c'est par exemple le cas aujourd'hui avec le smart-phone, téléphone portable incluant de la reconnaissance de l'écriture manuscrite, reconnaissance vocale, l'acquisition et stockage photo et de vidéo…

Au-delà de la reconnaissance de caractères isolés, les réseaux de neurones à convolution sont très intéressants pour la reconnaissance de mots cursifs. En effet dans le Pénacée [Guyon, 1995] et dans la synthèse de LeCun en 1998 [LeCun, 1998a], il a été montré qu'il est possible de répliquer le réseau sur un mot dynamique complet ou encore sur une image complète d'un montant, sans segmenter au préalable en chiffres ou en lettres.

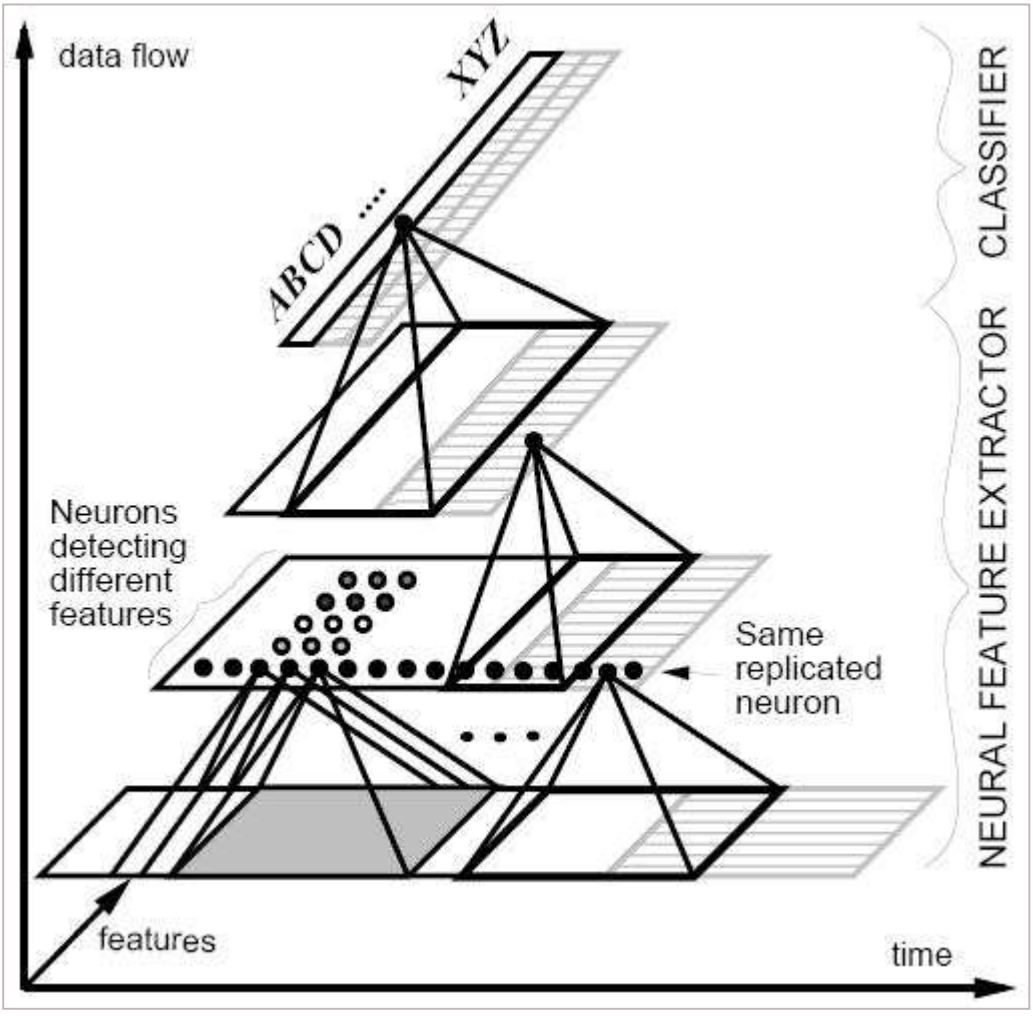


Figure 37. Illustration du Pénacée tirée de [Guyon, 1995].

La Figure 37 représente l'architecture du Pénacée. Le TDNN de base est représenté en noir, il calcule le vecteur des probabilités d'appartenance de la première observation à chaque classe. Sur la couche d'entrée est matérialisée en gris (rayé) la suite du mot sur lequel on fait glisser la fenêtre d'observation du TDNN. En sortie, on obtient donc une séquence de vecteurs des probabilités des classes obtenue à partir de chaque observation qu'a balayé le TDNN. Le découpage implicite du TDNN se caractérise en sortie par un certain nombre d'observations, dont trois observations sont représentées à la Figure 37.

Pour retrouver l'alignement temporel des classes en sortie du Pénacée, on utilise les chaînes de Markov pour modéliser les mots du lexique et principalement les liaisons inter et intra-lettres dans le mot. Nous allons voir dans le chapitre suivant l'intérêt d'un tel couplage réseau de neurones et modélisation markovienne.

3 LES APPROCHES NEURO-MARKOVIENNES

Dans le chapitre 1, nous avons introduit les principaux concepts relatifs aux systèmes de reconnaissance de l'écriture manuscrite et présenté les systèmes de classification les plus classiques qui y sont mis en œuvre.

Le chapitre 1 a permis d'approfondir une structure particulière, les réseaux de neurones à convolution, qui nous servira d'élément de base, d'abord dans la reconnaissance de caractères isolés, mais aussi ensuite, pour concevoir un système hybride de reconnaissance de mots cursifs. De tels systèmes hybrides, combinant Réseaux de Neurones Artificiels (RNA) et Modèles de Markov à états Cachés (MMC), ont d'abord été introduits en reconnaissance de la parole [Bourlard, 1998 ; Bottou, 1991], ils ont ensuite été étendus au domaine de la reconnaissance de l'écriture, principalement hors-ligne. L'objectif de ce chapitre est de présenter dans un premier temps le mécanisme et l'intérêt d'une combinaison RNA-MMC. Ensuite, nous aborderons un point essentiel : les méthodes d'apprentissage de ces systèmes hybrides.

3.1 Présentation générale

Il est aisé de faire un parallèle entre Reconnaissance Automatique de la Parole (RAP) et reconnaissance automatique de l'écriture manuscrite (RAE) en-ligne. Alors que la tâche élémentaire en RAP est de retrouver un mot à partir d'un flux d'informations acoustiques, en RAE, il s'agit de retrouver un mot à partir d'une séquence issue du tracé manuscrit. Ainsi, la Figure 38, issue de [Bourlard, 1998] qui illustre les composants principaux d'un système de RAP, peut-être facilement transposée à la RAE.

Le premier bloc constitue l'acquisition du signal avec son environnement, il a un effet important sur la représentation du signal généré. Par exemple, différents bruits viennent s'ajouter aux données utiles. Ils proviennent par exemple de l'équipement d'acquisition utilisé, des conditions physiques et morales de la personne. Dans le cas de la RAP, les bruits dépendent essentiellement du bruit additif, du type et de la position du microphone, de la résonance de la pièce. En RAE, les bruits dépendent à la fois du support (papier, écran, tablette), de l'outil d'écriture et du scripteur. Ils peuvent aussi être de type inter et intra scripteurs (vitesse d'écriture, calligraphie, orientation de l'écriture).

Le second bloc cherche à s'affranchir de ces différents bruits et à extraire des caractéristiques pertinentes pour bien regrouper/séparer les mêmes/différentes classes d'écriture ou de sons. Les deux blocs suivants illustrent le cœur des opérations de reconnaissance. Dans la plupart des systèmes de RAP, une représentation de la parole telle que son spectre ou cepstre[13] est calculée à des intervalles successifs. En RAE, les représentations sont plus variées, cf. chapitre 1 : on extrait différentes caractéristiques du tracé, soit en chaque point, soit sur les graphèmes obtenus après segmentation. Ces représentations sont comparées aux caractéristiques des données utilisées en apprentissage en utilisant des mesures de similarité ou de distance. Chacune de ces comparaisons peut être assimilée à un appariement local. L'appariement global consiste en la recherche de la meilleure séquence du mot selon une liste donnée de propositions et est déterminé en intégrant tous les appariements locaux. La reconnaissance de forme locale ne

[13] Le cepstre permet d'extraire la fréquence fondamentale d'un signal de la parole et de déterminer la fréquence des formants (traits caractéristiques des voyelles et de certaines consonnes), il est obtenu par transformation inverse du logarithme de la transformée de Fourier du signal vocal.

fournit pas généralement un seul choix parmi l'ensemble des classes possibles mais plutôt une liste de distances ou probabilités correspondant aux différents phonèmes (RAP) ou graphèmes (RAE) possibles. Ces mesures sont ainsi utilisées pour déterminer la séquence du mot la plus probable. Une fonction importante du décodeur est de compenser les distorsions temporelles que ce soit dans l'écriture naturelle ou le langage naturel de la parole. Par exemple, un même caractère pourra avoir une taille différente même dans un même mot.

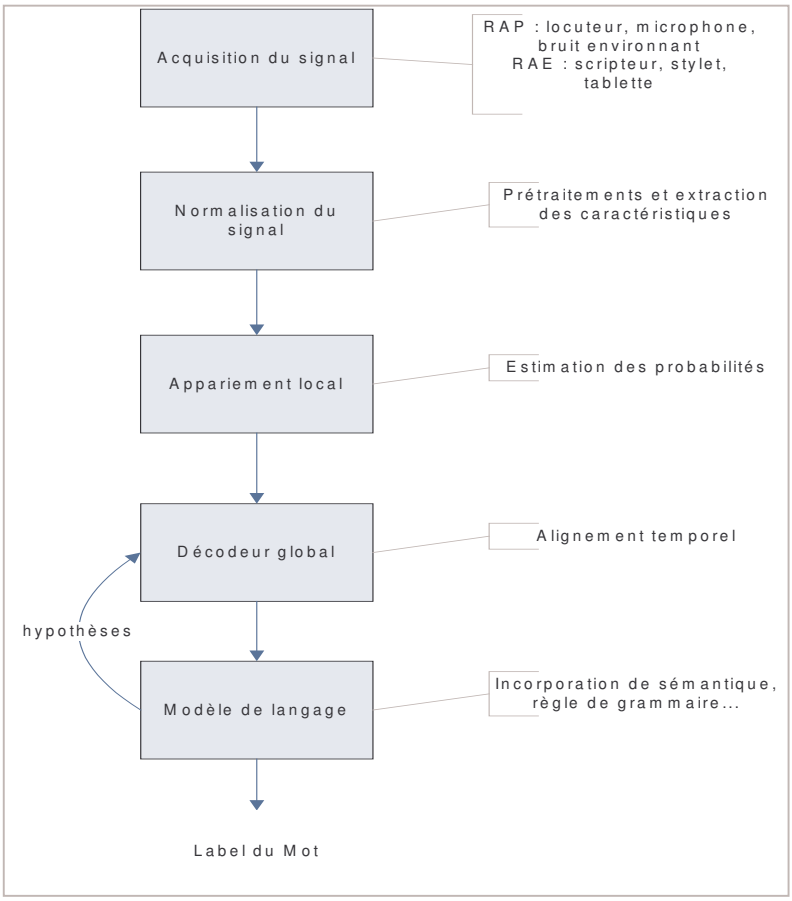


Figure 38 : Bloc diagramme d'un système de RAP/RAE

L'approche la plus utilisée pour le décodeur global se base sur les méthodes de Programmation Dynamique (DP pour Dynamic Programmation) ou Dynamic Time Warping (DTW). Il permet de comparer efficacement des mots de tailles différentes en cherchant un chemin d'appariement optimal. Un mot conserve son sens même après certaines distorsions temporelles. La programmation dynamique appliquée à la reconnaissance de la parole ou de l'écriture consiste à distordre dans le temps les mots de façon à les faire coïncider au mieux. Il existe différents algorithmes possibles pour réaliser la DP [Sutton, 1998] cependant le plus usuel est basé sur l'algorithme de Viterbi. Il permet, une fois définie une distance locale, de trouver la distance globale du coût minimum entre séquences de longueur éventuellement différente selon le mécanisme suivant :

1. A chaque intervalle de temps, considérer les transitions possibles avec les phonèmes/graphèmes de l'intervalle précédent.
2. Pour chaque transition possible, prendre le coût de la séquence de phonèmes/graphèmes à l'instant t-1 et l'ajouter au coût de la transition à l'instant t.

3. Choisir la transition la moins coûteuse par rapport à ce nombre et l'ajouter au coût d'appariement local, en conservant trace du pointeur de la séquence antérieure gagnante. La somme est le coût global de la séquence qui peut être retrouvée à partir de la position des pointeurs qui ont été sauvegardés.
4. A la fin du signal émis, revenir en arrière à partir du coût le plus bas pour générer la séquence de parole ou d'écriture correspondante.

Nous ne rentrerons pas plus en détails dans le cadre de la reconnaissance de la parole qui a largement été approfondie et décrite par Léon Bottou, Hervé Bourlard et al. Nous redéfinirons par la suite plus en détails l'étape de reconnaissance. Il est tout de même à noter que les coûts locaux et ceux de transitions sont généralement implémentés comme des logarithmes de probabilités pour faciliter les calculs (stabilité numérique) où les produits sont remplacés par des sommes.

Le dernier bloc de la Figure 38 représente un modèle de langage qui permet de mettre en correspondance une suite d'éléments acoustiques ou d'écriture avec une forme lexicale. Il permet d'obtenir une information *a priori* sur l'occurrence d'un mot dans le signal à reconnaître par différentes techniques de modélisation soit à base de grammaire, soit purement statistique, soit à base d'approches mixtes telles que les grammaires probabilistes. Ce bloc peut ainsi ré-estimer les N sorties les plus probables données par le décodeur global. Nous ne nous focaliserons pas sur les modèles de langage par la suite, différents travaux en reconnaissance de l'écriture sont en cours dans notre équipe de recherche [Perraud, 2005].

La formule générale, dans le cadre d'un système entièrement probabiliste, s'exprime sous la forme d'une équation bayésienne. Elle a été d'abord énoncée dans le cadre de la reconnaissance de la parole par Bahl et al. en 1983 [Bahl, 1983]. Le but du système est de trouver l'hypothèse W^* qui maximise pour toutes les séquences de mots W possibles et pour un signal observé O, l'équation suivante :

$$W^* = \underset{W}{\operatorname{argmax}}\, P(W|O) = \underset{W}{\operatorname{argmax}}\, \frac{P(W)P(O|W)}{P(O)} = \underset{W}{\operatorname{argmax}}\, P(W)P(O|W)$$

Dans cette équation, nous pouvons identifier plusieurs facteurs liés à la description précédente du système de RAP/RAE :

P(O) est la probabilité de l'observation. Elle ne fait pas intervenir W, et donc peut disparaître dans la maximisation précédente.

P($O|W$) représente la vraisemblance de l'observation O pour un mot (voire une séquence de mots) donné W. Celle-ci sera fournie par la sortie du bloc précédent, cette valeur représente le résultat du processus de reconnaissance de formes. Par exemple, elle pourra être calculée par l'algorithme Forward-Backward, ou encore l'algorithme de Viterbi, dans le cadre d'une modélisation MMC (cf. 3.2).

P(W) est la probabilité *a priori* de la séquence de mots W, sans aucune notion d'acoustique ou de calligraphie, dans le langage considéré. C'est la probabilité générée par le modèle de langage. Celui-ci devra au préalable avoir été défini en s'appuyant sur un corpus textuel représentatif du domaine de discours étudié.

3.2 Les Modèles de Markov Cachés

Les Modèles de Markov Cachés (MMC, acronyme anglais HMM pour Hidden Markov Models) sont des automates stochastiques permettant de modéliser des données séquentielles telles que l'écriture ou la parole.

Il est important de définir cet outil mathématique avec ses intérêts et limites.

3.2.1 Définition

Rabiner et Juang [Rabiner, 1989 ; Rabiner 1993] ont largement détaillé et illustré cet outil mathématique. Nous ne présentons ici qu'une description succincte. Le principe de la modélisation par des modèles de Markov discrets est le suivant : on mesure une variable aléatoire discrète X(t) et on cherche à modéliser le processus qui en est à l'origine par un modèle stochastique.

Un modèle de Markov Caché λ(A, B, Π, T) est défini par l'ensemble de données suivantes :

- une matrice *A* qui permet la définition de la topologie du MMC en indiquant les probabilités de transition d'un état S_i vers un autre état (ou lui-même), soit $p(q_t=S_j| q_{t-1}=S_i) = a_{ij}$. La taille de cette matrice est NxN, où N est le nombre d'états du modèle.
- une matrice *B* qui contient les probabilités d'émission des observations dans chaque état $b_j(O_t) = p(O_t| q_t=S_j)$. Chaque état de la séquence $q_1, q_2, ..., q_T$ émet une observation selon un second processus stochastique. Il en résulte une séquence $O=O_1, O_2, ..., O_T$ de T observations. Les modèles les plus couramment utilisés sont les modèles de Markov d'ordre 1, pour lesquels la probabilité d'être dans un état à un instant donné dépend uniquement de l'état à l'instant précédent.
- un vecteur $\Pi=\{\pi_i\}=\{P(q_{t=0}=S_i)\}$ qui donne la distribution de départ des états, c'est-à-dire pour chaque état la probabilité d'être atteint à partir de l'état initial q_I. Et une matrice $\Phi=\{\tau_i\}=\{P(q_{t=fin-1}=S_i)\}$ qui correspond aux probabilités de l'état final q_F donne la distribution de sortie de la chaîne. Ces états q_I et q_F sont particuliers puisqu'ils ne peuvent émettre d'observations.

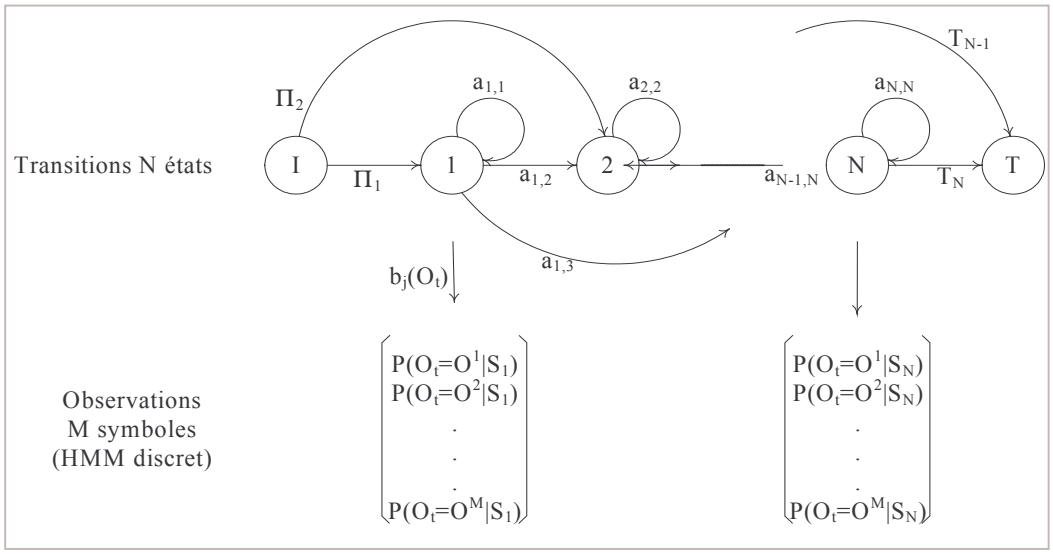


Figure 39. Représentation d'un modèle de Markov caché gauche droite avec saut

Dépendant de la topologie du modèle, différents types de MMC existent (l'état de l'art de la thèse Pierre-Michel Lallican [Lallican, 1999] reprend ces modèles). Deux types sont couramment utilisés en reconnaissance de parole ou d'écriture : les modèles gauche-droite qui ne permettent pas de transition arrière, et les modèles ergodiques qui permettent les transitions entre tous les états. En reconnaissance d'écriture et de la parole, le modèle gauche-droite d'ordre 1 dit de Bakis est le plus largement utilisé. Cette topologie permet la modélisation des variations temporelles au sein du signal d'écriture. Ainsi, sur une écriture très large, il peut y avoir une répétition d'états, c'est pour cela que ce modèle de MMC permet de boucler sur un état (transitions a_{ii} sur la Figure 39). *A contrario*, si l'écriture est serrée, il est aussi possible de sauter l'état suivant (arc a_{ij}).

Les différentes distributions de probabilité à l'intérieur des états peuvent être interprétées comme les caractéristiques du signal d'écriture ou de parole modélisées. Selon le formalisme classique des MMC [Rabiner, 1989], la probabilité d'une séquence d'observations O se décompose sur tous les chemins Γ possibles dans un modèle λ_k :

$$P(O \mid \lambda_k) = \sum_{\Gamma} P(O, \Gamma \mid \lambda_k) = \sum_{\Gamma} P(O \mid \Gamma, \lambda_k) P(\Gamma \mid \lambda_k) \qquad (1)$$

où un chemin représente une séquence particulière des états de la chaîne de Markov ayant émis la séquence d'observations.

Etant dans le cas de modèles de Markov d'ordre 1, la transition vers l'état suivant ne dépend que de l'état courant d'où la simplification du second terme de l'équation précédente :

$$P(\Gamma \mid \lambda_k) = P(q_1, q_2, ..., q_T \mid \lambda_k) = P(q_1 \mid \lambda_k) \cdot P(q_2 \mid q_1, \lambda_k) \cdots P(q_T \mid q_1, ..., q_{T-1}, \lambda_k)$$
$$P(\Gamma \mid \lambda_k) = P(q_1 \mid \lambda_k) \cdot P(q_2 \mid q_1, \lambda_k) \cdots P(q_T \mid q_{T-1}, \lambda_k)$$
$$P(\Gamma \mid \lambda_k) = P(q_1 \mid \lambda_k) \cdot \prod_{t=2}^{T} P(q_t \mid q_{t-1}, \lambda_k)$$

Avec la deuxième hypothèse stipulant que chaque observation ne dépend que de l'état courant, on obtient pour le premier terme de $P(O \mid \lambda_k)$:

$$P(O \mid \Gamma, \lambda_k) = \prod_{t=1}^{T} P(O_t \mid q_t, \lambda_k) \qquad (2)$$

L'implantation du calcul de l'équation (2) doit être envisagée avec précautions. Une implantation naïve de cette équation conduit à une complexité en $O(T.N^T)$, ce qui est rédhibitoire. De plus, lorsque T augmente, le nombre de produits de probabilité augmente. Cela conduit rapidement à un débordement sur les représentations numériques (underflow). L'algorithme Forward-Backward, avec étape de normalisation, permet une programmation efficace de ce calcul qui redescend en $O(T.N^2)$.

3.2.2 Usage et Intérêts

Ces modèles sont optimaux dans le sens où, si les distributions de probabilité des classes de formes étudiées sont connues, leur classification par une méthode de type bayésien donnera le taux d'erreur minimal. En pratique, ces distributions doivent être estimées à partir de corpus de données d'apprentissage.

L'un des principaux problèmes de l'utilisation des MMC réside dans la phase d'apprentissage : une fois la topologie du graphe choisie, il faut estimer tous les paramètres du modèle, c'est-à-dire les coefficients de la matrice de transition, les coefficients du vecteur des probabilités initiales et les distributions de probabilité dans les états. Il s'agit avec un corpus

d'apprentissage, contenant un étiquetage par sous-unités du signal temporel, de maximiser la vraisemblance que le modèle MMC ait produit la suite d'observations. Il existe plusieurs algorithmes éprouvés pour faire cela : l'algorithme de Baum-Welch est le plus répandu. Il s'appuie sur un schéma itératif de type EM (Expectation-Maximisation) permettant une estimation des paramètres basée sur une approche de type MLE (Maximum Likelihood Estimation). Cet algorithme est détaillé dans les articles de Rabiner et al. ([Rabiner, 1989 ; Rabiner 1993])

Après avoir entraîné autant de modèles que de formes à reconnaître, on peut calculer la vraisemblance d'une séquence d'observations donnée vis-à-vis de chacun des modèles (algorithme Forward-Backward) et retenir le modèle qui maximise cette vraisemblance :

$$\lambda_{\max} = \arg\max_{\lambda_k} P(O \mid \lambda_k)$$

On peut alors extraire du modèle λ_{max} le chemin Γ ayant le plus contribué à $P(O|\lambda_{max})$, grâce à l'algorithme de Viterbi, et se servir des instants de transitions entre états pour décoder la séquence d'observations et l'étiqueter par les états du modèle. Une telle segmentation peut être utile dans un schéma hybride pour affiner la qualité de l'étiquetage au sein d'un processus itératif alternant les étapes étiquetage/apprentissage/reconnaissance.

L'existence d'algorithmes d'apprentissage par des exemples, et la preuve de convergence ont contribué au succès de cette approche. Néanmoins, plusieurs difficultés subsistent comme le choix de la topologie (nombre d'états principalement), la distribution de probabilité à l'intérieur des états (mesure discrète, mixtures de gaussiennes...) ou encore la modélisation de la durée. De plus, lorsque le volume des données en apprentissage est limité, il peut être intéressant de se tourner vers les méthodes neuromimétiques, ces méthodes possédant une capacité de généralisation à partir de données incomplètes.

Les modèles stochastiques de type modèle-générateur tels que les MMC présentent quelques limitations, notamment dues aux hypothèses restrictives en général introduites dans les algorithmes d'optimisation associés. D'un autre côté, les réseaux de neurones (RN) se révèlent utiles pour la classification de formes statiques tout en se montrant faibles en ce qui concerne le traitement de la temporalité du signal de parole ou d'écriture. Dans la section suivante, nous allons montrer l'intérêt de coupler les réseaux de neurones aux modèles de Markov Cachés en les utilisant soit comme estimateurs des probabilités *a posteriori*, soit comme approximateurs des fonctions de densité de probabilité. Et nous verrons ensuite les modes d'apprentissage associés à ces hybrides.

3.3 Les systèmes hybrides RN MMC

Les progrès réalisés ces dernières années dans le domaine de la reconnaissance de l'écriture manuscrite sont dus en grande partie à l'utilisation d'approches statistiques. Parmi celles-ci, les deux techniques précédemment décrites ont plus particulièrement été mises à contribution. Les approches connexionnistes sont particulièrement intéressantes par leur fort pouvoir discriminant et leur capacité à construire des frontières de décisions complexes dans des espaces de grande dimension. Les techniques basées sur les modèles de Markov cachés modélisent par une approche paramétrique les séquences d'observations générées par des processus stochastiques tels que, par exemple, l'écriture manuscrite. La notion de séquence prend toute son importance lorsque l'on s'intéresse à la reconnaissance au niveau mot. Les MMC apportent alors leur

capacité à modéliser la distribution des observations pour chaque classe de forme à reconnaître. Alors que, lorsqu'il s'agit de la reconnaissance de caractères isolés, c'est d'avantage une forme globale qu'il s'agit d'apprécier, les réseaux de neurones sont alors particulièrement adaptés. Certaines approches, parmi les plus évoluées, tentent de combiner les avantages de chacune de ces techniques en proposant des systèmes de reconnaissance hybrides: le réseau de neurones calcule les probabilités P(classe/observation), les séquences d'observation résultant de la concaténation de ces probabilités sont traitées par les MMC.

3.3.1 Intérêts d'un couplage hybride RN-MMC

De nombreux travaux comparent les RN et les MMC [Bottou, 1991] et ce que chacun peut apporter. La principale difficulté avec les réseaux de neurones est l'intégration du contexte temporel à l'échelle d'une séquence d'écrit, de longueur variable. Nous avons vu au chapitre précédent deux types de réseaux qui ont été étudiés pour résoudre ce problème : les réseaux récurrents et les réseaux à délais. Ils permettent en effet d'extraire une information de la séquence et de la propager dans le temps soit par réintroduction du passé en entrée dans le cas des réseaux récurrents soit par décalage partiel de la fenêtre d'observation d'un TDNN. Cependant la dimension du passé réintroduit est relativement restreinte par rapport à la taille globale d'un mot. De plus les RN ne sont pas robustes aux perturbations telles que la variabilité de durée et de longueur d'une lettre ou d'un mot, contrairement à une modélisation markovienne.

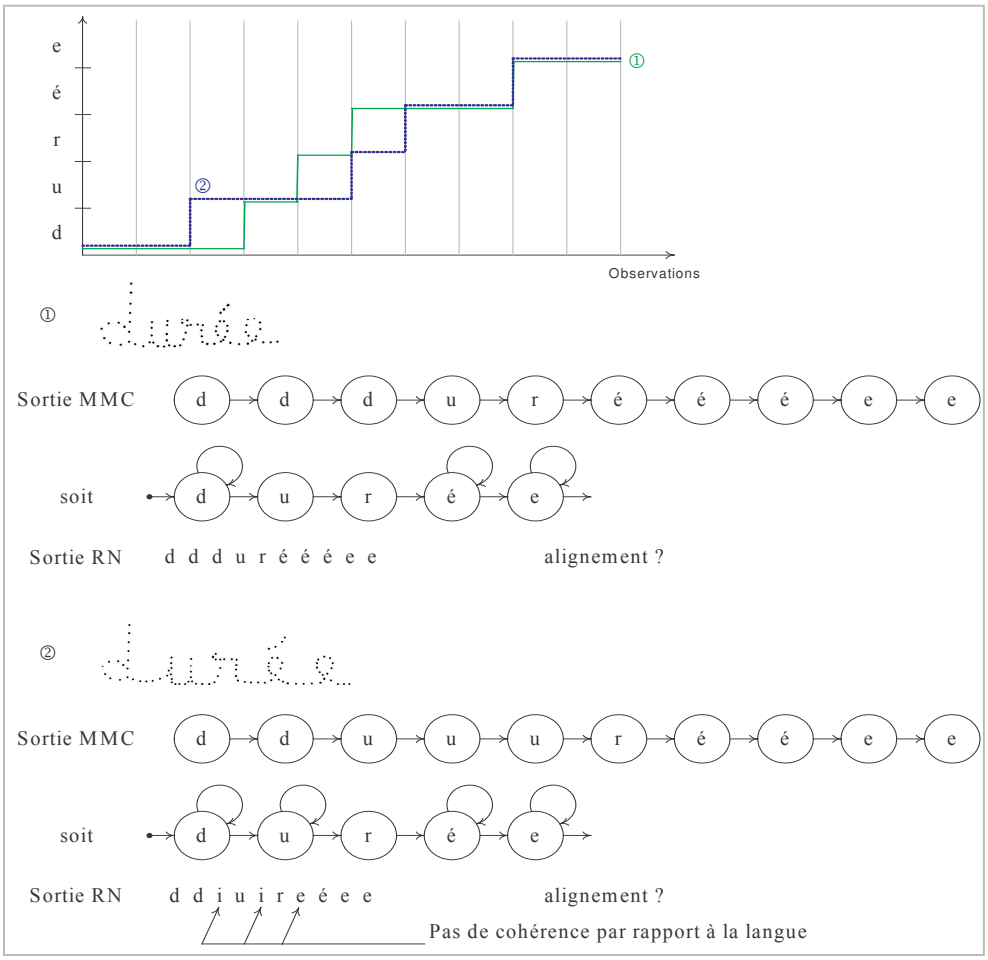


Figure 40. Reconnaissance du mot "durée" selon 2 écritures par un MMC et un RN

La Figure 40 illustre ce problème d'alignement temporel sur le mot durée. Pour les deux écritures possibles du mot « durée », une modélisation markovienne permet de gérer les passages d'une lettre à la suivante et la durée d'une même lettre (cas de la lettre d). Tandis que le réseau de neurones n'est capable que de donner une suite de classes correspondantes à la segmentation qui, concaténée, ne donne pas forcément un mot du dictionnaire. Une étape d'alignement temporel est nécessaire.

On peut noter les différents avantages énoncés ci-dessous à coupler les réseaux de neurones, fort de leur pouvoir de discrimination de classes et les modèles de Markov cachés, utiles pour l'identification de séquence temporelle.

- Exactitude / Performance du modèle : L'estimation des probabilités par un réseau de neurones ne requiert aucune hypothèse sur la forme de la distribution statistique à modéliser.
- Sensibilité au contexte : dans le cas de réseaux récurrents et à délais, les corrélations locales des vecteurs d'entrées peuvent être prises en compte dans la distribution des probabilités. Celle-ci a été démontrée tant en reconnaissance de la parole [Bengio, 1993] et de l'écrit que de l'estimation de qualité d'une vidéo [Le Callet, 2005].
- Discrimination : Les RN permettent un apprentissage discriminant. Dans une grande majorité des hybrides standards MMC/RN que ce soit dans la littérature ou dans les applications industrielles, la discrimination est seulement locale. Cependant, quelques travaux [Chen, 2000 ; Tay, 2001] ont démontré la faisabilité et l'intérêt d'effectuer un apprentissage global discriminant des systèmes hybrides.
- Structure réduite : toutes les probabilités de distribution sont représentées par un même jeu de paramètres partagés.
- Flexibilité : L'emploi d'un RN comme estimateur statistique permet une combinaison facile de diverses caractéristiques, telle que la combinaison de mesures continues ou catégoriques (discrètes).
- Complémentarité : Les RN peuvent aussi apporter des informations complémentaires à un système préexistant basé sur des vraisemblances. Par exemple, pour la parole ou l'écriture, la combinaison de MMC avec un RN (réseau de neurones segmentaire) apporte quelques améliorations par rapport au système original. Dans ce cas, un problème à N meilleures solutions est utilisé pour générer les N meilleures hypothèses de segmentation qui sont ré estimées et pondérées par un réseau de neurones.

Par ailleurs, il a été montré [Koerich, 2003] que l'estimation des probabilités *a posteriori* locales permettent une meilleure classification pour des systèmes de reconnaissance à large vocabulaire, autant dans la parole que l'écriture. Et les probabilités a posteriori sont indépendantes de la dimension de l'espace d'entrées, contrairement aux valeurs des vraisemblances qui dépendent de la taille de l'espace des caractéristiques.

3.3.2 Types de couplage

Combiner les capacités respectives des MMC et de RN est une solution viable avec les puissances de calcul actuelles même pour des applications mobiles embarquées. Cependant l'apprentissage de ces deux structures n'est pas simple à réaliser et reste encore un vaste sujet de recherche. Nous présentons dans ce qui suit deux couplages différents :

- Les réseaux de neurones en amont d'un MMC,
- Les réseaux de neurones effectuant un post traitement des sorties des MMC.

3.3.2.1 Réseau neuronal en amont d'un MMC

La stratégie de couplage la plus courante dans les hybrides RN-MMC est d'utiliser un RN comme estimateur local et les MMC dans leur formalisme habituel.

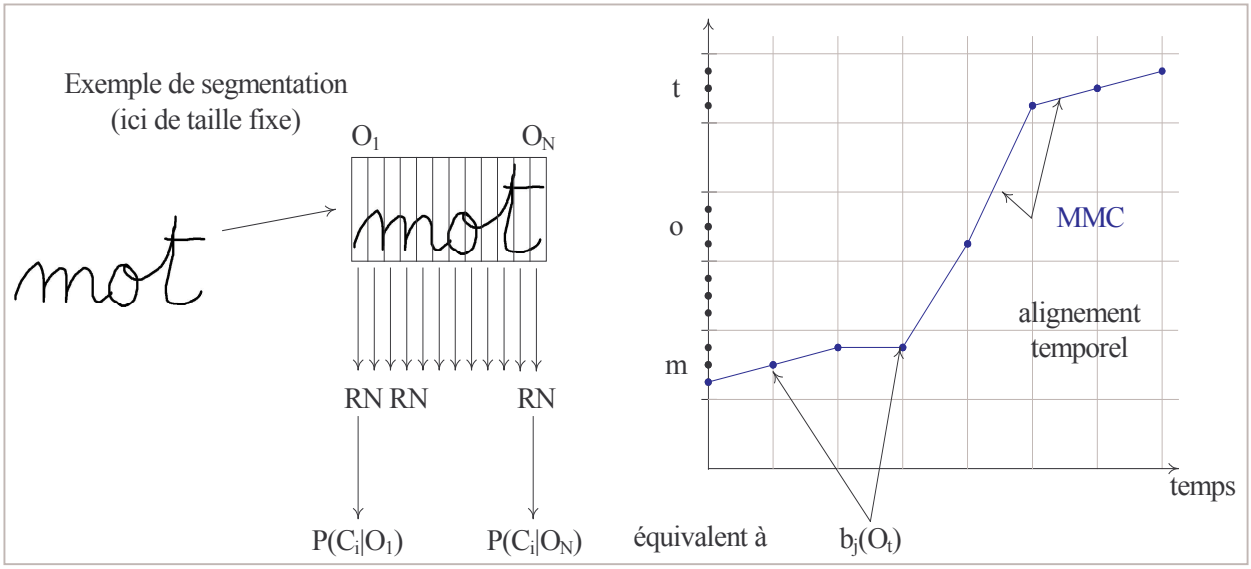


Figure 41. Schéma classique RN-MMC, à gauche le RN fournit pour chaque observation les probabilités d'appartenance à une classe, et à droite, à partir du treillis où chaque point désigne les probabilités a posteriori des états du MMC provenant du RN, le MMC détermine le meilleur alignement temporel.

A partir d'un mot segmenté ou sur-segmenté en observations qui peuvent être des lettres entières ou non, le réseau de neurones va calculer les probabilités *a posteriori* $P(C_i|O_t)$ des états des MMC en fonction de l'observation. Selon la règle de Bayes, on peut exprimer la probabilité d'une classe Ci sachant l'entrée Ot par l'équation suivante :

$$P(C_i \mid O_t) = P(q_i \mid O_t) = \frac{P(O_t \mid q_i)P(q_i)}{P(O_t)}$$

Le terme $P(O_t)$ est indépendant de la classe il peut donc être ignoré pour la classification. $P(q_i)$ quant à lui représente la probabilité a priori de la classe, elle permet d'obtenir des vraisemblances normalisées. Et $P(O_t|q_i)$ est la probabilité de l'observation O_t dans l'état q_i soit $b_i(O_t)$ dans la notation standard donnée à la section 1.

A partir des vraisemblances $b_i(O_t)$ normalisées, le HMM effectue la tâche de reconnaissance au niveau mot en trouvant le meilleur alignement temporel sur les différents mots du lexique, cf. Figure 41.

De nombreux systèmes ont été développés à partir de ce principe, notamment pour l'écrit en-ligne on peut citer [Bengio, 1995 (LeRec, méthode OUTSEG) ; Schenkel, 1995 ; Jaeger, 2000 ; Caillault, 2005]. Diverses expériences ont démontré que de tels systèmes hybrides améliorent les performances des MMC tout en restant efficaces en ce qui concerne le temps calcul et l'encombrement en mémoire [Système de reconnaissance de la parole ABBOT : [Hochberg, 1995]. Cependant l'implantation et la mise en œuvre ne sont pas simples du fait du nombre de paramètres à ajuster et de la quantité de données d'apprentissage nécessaire pour assurer la bonne convergence globale du processus. Nous verrons dans la suite les différents types d'apprentissage et leurs contraintes.

Une autre solution a été proposée, elle consiste à utiliser le RN non plus pour calculer les probabilités a posteriori mais comme un étiqueteur d'un MMC discret. L'idée dans le cadre de la RAP, consiste à donner une étiquette phonétique [Le Cerf, 1994a ; Le Cerf, 1994b] à chaque vecteur de parole entré dans le MMC ce qui revient à une sorte de quantification vectorielle phonétique : le signal acoustique est remplacé par une séquence de symboles produite par les RN (cf. Figure 42). Une solution analogue, fondée sur un apprentissage non supervisé est décrite dans [Rigoll, 1994].

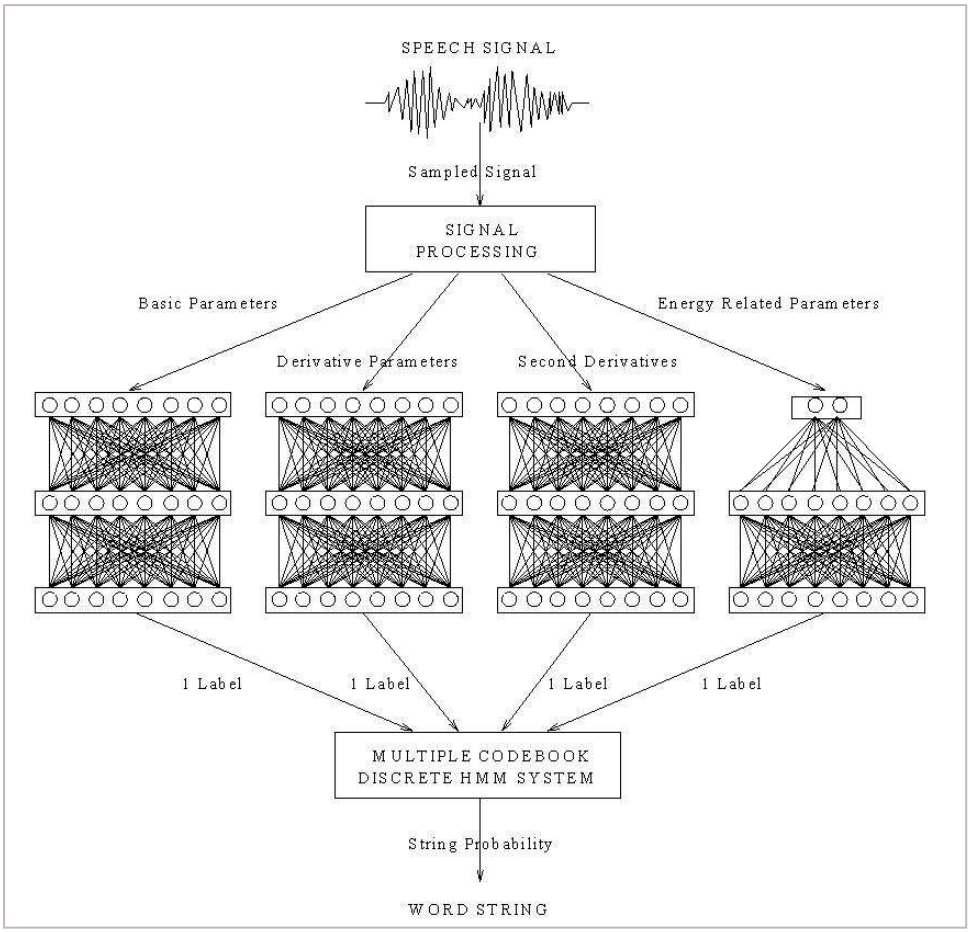


Figure 42. Hybride multi PMC- MMC, PMC étiqueteur du MMC de [Le Cerf, 1994b]

3.3.2.2 Réseau neuronal en aval d'un MMC

Une autre solution pour la réalisation de systèmes hybrides consiste à utiliser le réseau de neurones comme post processeur d'un MMC.

Une première méthode consiste à mettre en entrée du réseau toutes les entités reconnues par le MMC [Guo, 1993]. Le RN est un classifieur de phonèmes/graphèmes dans un système MMC à segmentation interne. Cette solution n'est viable que pour des solutions à vocabulaire limité. Une autre solution revient à ne fournir au réseau que les quelques meilleures réponses du MMC obtenues par un algorithme de recherche de type N-Best [Boiteau, 1993].

Un exemple est fourni dans [Mari, 1994-a] pour la reconnaissance de lettres épelées en continu. Le système correspondant est formé d'un MMC du second ordre [Mari, 1994-b] entraîné pour la reconnaissance des lettres et d'un réseau neuronal sélectif. Ce dernier est un perceptron multicouche qui se focalise sur la partie discriminante de mots pour aider à lever l'ambiguïté sur la reconnaissance de paires telles que /m, n/ ou /p, t/. Comparé au schéma RN-HMM, le MMC est utilisé pour fournir une liste des N meilleures solutions, et repérer des frontières de mots. Ces frontières sont utilisées par le réseau neuronal pour localiser la partie discriminante et fournir finalement la meilleure solution. L'expérience montre que les scores d'identification des lettres sont améliorés tant en anglais qu'en français.

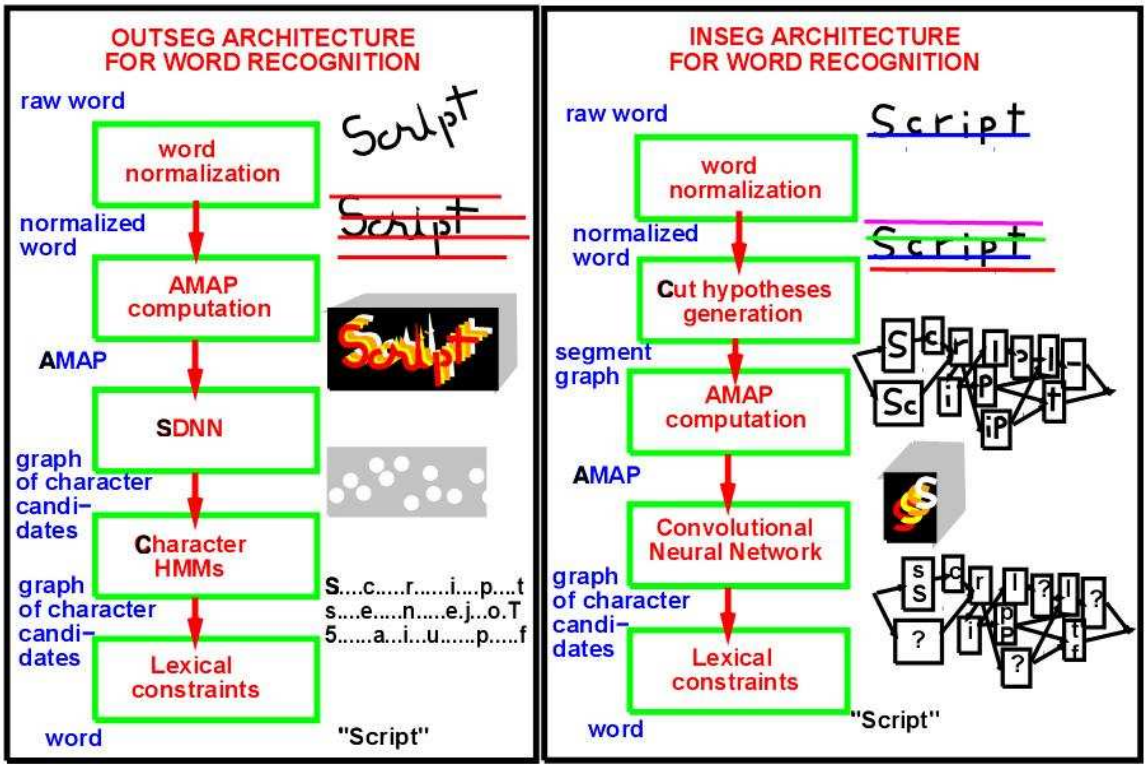


Figure 43. LeRec, système hybride : segmentation OUTSEG et INSEG [Bengio, 1995]

En reconnaissance en-ligne, on peut citer le formalisme INSEG de LeRec, cf. Figure 43 à droite tirée de [Bengio, 1995]. Un graphe de segmentation du mot est construit. Ensuite un réseau de neurones ici de type SDNN, est appliqué à chaque nœud du graphe. Une grammaire est ensuite utilisée pour chercher le meilleur chemin dans le graphe des observations.

En écriture hors-ligne, Choisy et Belaïd s'appuient sur une architecture hybride MMC-RN dans le cadre d'une reconnaissance de mots cursifs hors-ligne. La Figure 44, extraite de [Choisy, 2002a] illustre le principe de reconnaissance. Le MMC est utilisé comme modèle à vision locale (MVL) pour absorber les distorsions et variations de longueur. Le MVL a une architecture particulière de type NSHP-HMM (Non symetric Half-Plane Hidden Markov Model : [Saon, 1997] qui agit directement sur l'image binaire en observant les colonnes de pixels dans les états de HMM. L'image est alors normalisée en fonction de ces distorsions. Ensuite des réseaux de neurones, exemples de MVG (modèles à vision globale) sont appliqués pour effectuer une analyse de manière globale de la forme normalisée. Choisy a utilisé différents MVG tels que les machines à vaste marge [Choisy, 2002b]. Cependant d'après les auteurs, les réseaux de neurones sont plus faciles à interpréter car leurs sorties estiment des probabilités *a posteriori*.

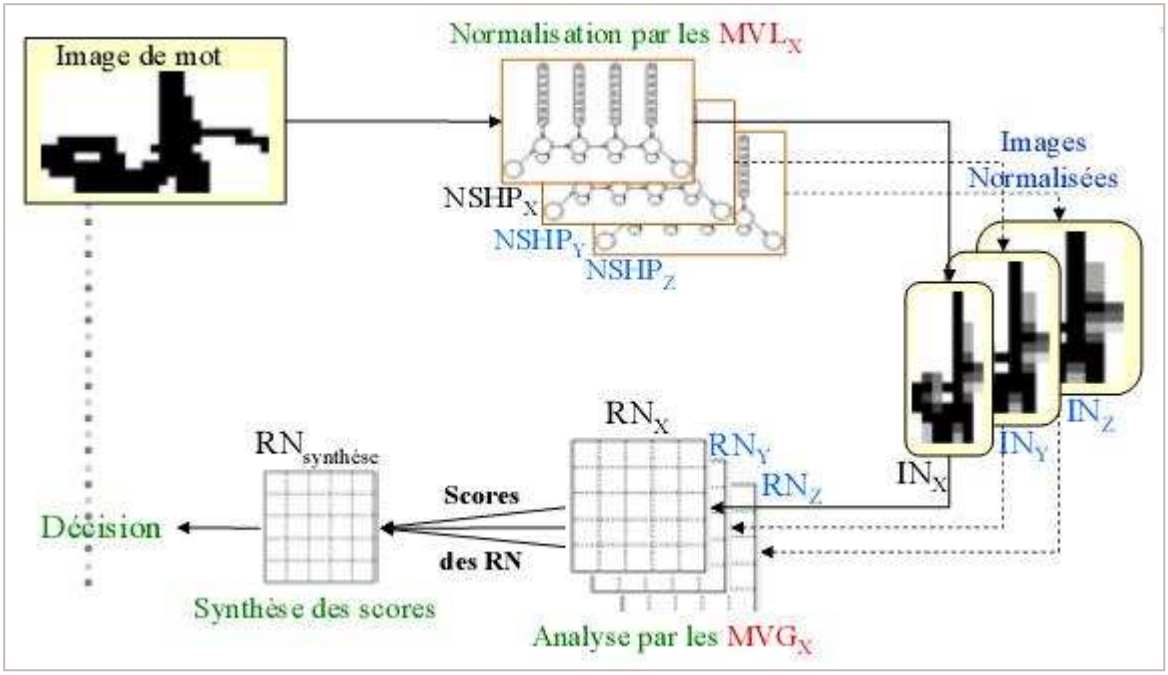


Figure 44. Schéma de reconnaissance d'un hybride MMC RN proposé dans [Choisy, 2002a]

Dans tous les systèmes ci-dessus, la coopération entre les réseaux de neurones et les modèles de Markov cachés a lieu au niveau de la décision de reconnaissance et non au niveau des modèles eux-mêmes. L'hybridation est la plus souvent réduite à sa plus simple expression, c'est-à-dire que les deux modèles RN et MMC sont appris séparément avec des méthodes et fonctions de coût séparées. Dans le cas d'une hybridation RN-MMC, il est possible de concevoir des modèles avec un formalisme unique. Dans cette voie, il est démontré [Bridle, 1990 ; Brown, 2001 ; Brown, 2002] que l'algorithme d'apprentissage de MMC *forward-backward* est équivalent à l'algorithme de rétropropagation du gradient d'erreur. Les MMC peuvent être implémentés à l'aide d'un réseau récurrent (ce parallèle est détaillé dans la partie 7.2 du livre [Rojas, 1996]).

Les réseaux de neurones sont entraînés à discriminer des classes de formes, tandis que les MMC sont normalement entraînés pour estimer, selon le principe du maximum de vraisemblance, les lois de probabilités de chaque modèle, c'est-à-dire de chaque classe. Il faut noter également que plusieurs méthodes d'apprentissage discriminant pour les MMC, fondées sur les critères de maximum d'information mutuelle [Bahl, 1986] ou de minimum d'information discriminante [Ephraim, 1990] ont été introduites et ont permis d'améliorer sensiblement les performances d'un MMC avec les mêmes données d'apprentissage. Encore peu de systèmes de reconnaissance de l'écriture manuscrite cursive en-ligne ou hors-ligne se basent sur cette unification de l'apprentissage d'un hybride. Nous allons dans la suite énumérée les différents types d'apprentissage que l'on retrouve dans la littérature dans le cadre de la reconnaissance de l'écriture manuscrite.

3.4 Les différents types d'apprentissage.

Comme nous l'avons présentée dans la section précédente, l'hybridation des modèles à vision globale tels que les réseaux de neurones avec une modélisation markovienne est un concept très intéressant. Cependant leur paramétrage et leur mode d'apprentissage reste une tâche délicate à mener conjointement.

3.4.1 Apprentissage local versus global

On distingue essentiellement deux approches pour entraîner le système hybride comme un reconnaisseur : soit par apprentissage séparé, soit par apprentissage global. La configuration des deux systèmes est quasi-identique. La Figure 45 illustre ces deux approches.

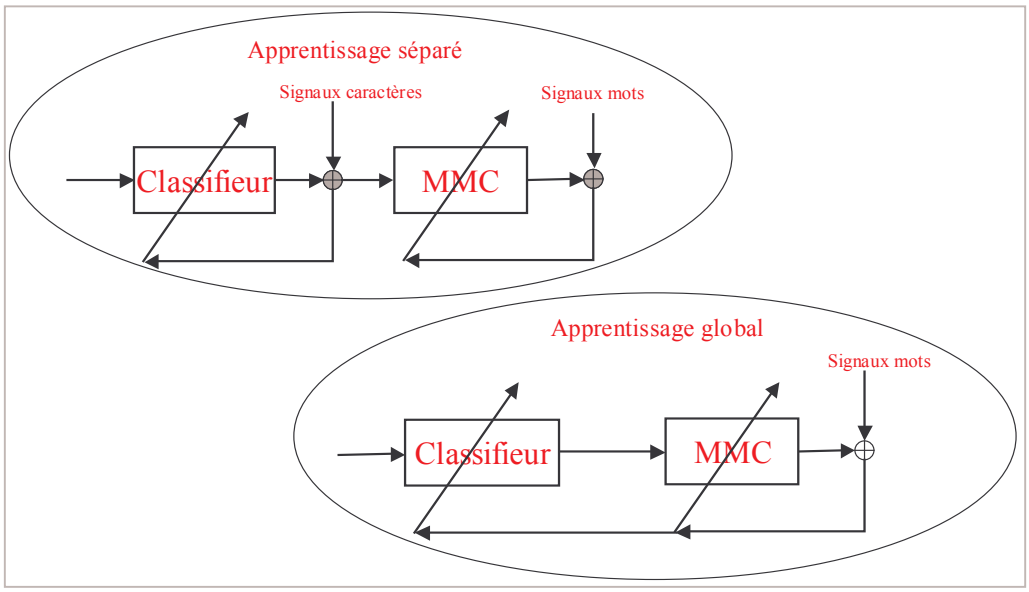


Figure 45. Apprentissage local versus apprentissage global

3.4.1.1 Apprentissage séparé

La première solution, en haut de la Figure 45, est d'effectuer un apprentissage au niveau caractère. C'est l'approche la plus simple à réaliser. Cependant, comme le rôle du classifieur est d'estimer les probabilités de reconnaissance au niveau caractère, elle nécessite une segmentation préalable de la base de mots en caractères. Par ailleurs, cette procédure demande plusieurs itérations du processus apprentissage/reconnaissance/étiquetage pour obtenir une segmentation correcte et des résultats satisfaisants. Ensuite les probabilités de transitions des états des MMC peuvent être estimées en utilisant les algorithmes de Baum-Welch ou Forward-Backward à partir de données mots comme dans un apprentissage classique de MMC.

Un tel schéma pose différentes difficultés.

1. Bootstraping

La première est l'initialisation (en anglais bootstraping) du système avec un premier étiquetage. On distingue principalement trois approches pour effectuer cette tâche.

La solution la plus simple consiste à utiliser une base de caractères isolés pour entraîner le classifieur de caractères. Cependant, cette approche conduit à des résultats très médiocres. En effet, toute l'information contextuelle liée à l'écriture cursive au sein d'une séquence de lettres est perdue. De plus, la notion de lignes de référence du mot, permettant la normalisation en taille et position de caractères, n'est pas accessible par cette méthode.

Il est donc préférable de pouvoir s'appuyer directement sur des caractères issus de la segmentation des mots. Or, l'écriture étant non contrainte, scripte ou cursive, la segmentation n'est ni unique, ni facile à obtenir.

La segmentation manuelle n'est envisageable que sur une base de taille limitée étant donné le caractère laborieux de cette tâche. De plus, une telle procédure ne prend pas en compte les spécificités du système de reconnaissance.

Il est donc plus intéressant de générer automatiquement cette étape d'initialisation du système. La solution consiste alors à s'appuyer sur un système de reconnaissance, de type analytique, existant. On peut, par exemple, se baser sur une approche de type MMC discrets et utiliser l'algorithme de Viterbi pour retrouver le meilleur chemin de segmentation de la séquence connaissant le label du mot et ainsi fournir les points de segmentation et les différents labels lettres.

2. Vraisemblances normalisées (scaled Likelihood en anglais)

Un second problème est de transformer les probabilités a posteriori estimées par le RN en vraisemblances normalisées. Selon le théorème de Bayes, la vraisemblance de l'observation O_t pour la classe C s'écrit :

$$p(O_t \mid C) = \frac{P(C \mid O_t)\, p(O_t)}{P(C)}$$

Or il est difficile d'estimer la densité de probabilité $p(O_t)$. Il est souvent défini une pseudo vraisemblance normalisée :

$$\hat{p}(O_t \mid C) = \frac{P(C \mid O_t)}{P(C)}$$

3. Données non apprises

Le troisième problème est de savoir traiter les entrées non rencontrées pendant l'apprentissage car elles ne correspondent à aucune étiquette réelle de caractère. Une technique est d'utiliser en sortie du réseau de neurones une classe de rejet, le problème est alors reporté sur la sélection des exemples devant illustrer cette classe de rejet.

3.4.1.2 Apprentissage global

Même s'il est possible de spécifier et apprendre une classe de rejet au réseau de neurones, un apprentissage séparé est tout de même en contradiction avec une reconnaissance de mots et montre certaines limitations.

- Du fait d'un apprentissage non unifié, les probabilités de transition peuvent être apprises en utilisant l'algorithme de Baum-Welch avec une optimisation du maximum de vraisemblance et la fonction de coût pour optimiser le réseau de neurones n'est pas directement liée à cette optimisation et donc aux performances de la reconnaissance mot. Or une optimisation du RN au niveau caractère n'entraîne pas forcément une bonne reconnaissance et généralisation au niveau mot.
- Dans l'adaptation des sorties du RN, aux probabilités d'émission des états des MMC, il faut normaliser les sorties du RN par les probabilités des classes pour obtenir des vraisemblances normalisées. Or cette normalisation pose un problème si peu de labels caractères sont représentés. C'est principalement le cas des majuscules comparées aux lettres minuscules.
- L'étage d'apprentissage des non-caractères (rejets) est délicate et peut entraîner une mauvaise interprétation : une mauvaise classification d'un caractère réel peut être interprété comme un faux caractère.

Pour éviter les limitations énoncées ci-dessus, il est préférable d'un point de vue théorique d'unifier le processus d'apprentissage et d'introduire une fonction d'optimisation globale à injecter dans les deux modèles RN et MMC. C'est le cas de la représentation de la Figure 45 avec un apprentissage global. Il est possible d'entraîner globalement le système de reconnaissance mot par un algorithme de descente de gradient. Alpha-nets [Bridle, 1990] est un des premiers systèmes en reconnaissance de la parole utilisant cette méthode de rétropropagation du gradient. L'idée sous-jacente à ce modèle consiste à considérer comme un type de réseau récurrent la partie *forward* de l'algorithme d'apprentissage de Baum-Welch utilisé dans les MMC.

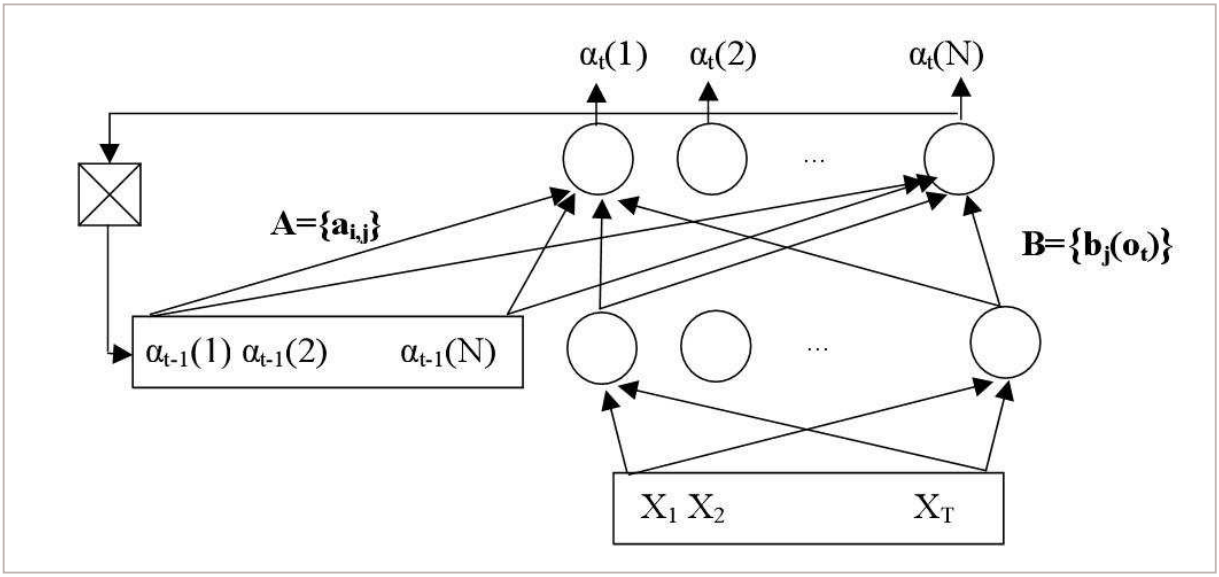


Figure 46. Schéma représentant un alpha-net, où (Xi) représente le vecteur d'observation à l'instant t et $\alpha_t(i)$ sont les sorties avec les probabilités des N classes cumulées sur toute la séquence

Ce système, cf. Figure 46, est constitué, d'une part d'un réseau à propagation qui calcule les probabilités d'observation $b_j(O_t)=p(x_t|q_i)$ du signal en entrée, et d'autre part d'une récurrence sur les sorties pour accumuler les probabilités $\alpha_t(j)$ pour chacun des états qi. Ces sorties ne comportent pas de non-linéarités. On obtient l'algorithme Forward des MMC. On peut remarquer que c'est directement la probabilité $p(x_t|q_i)$ qui est évaluée et non $p(q_i|x_t)$ comme dans la plupart des systèmes hybrides unificateurs. Bridle propose de lever les contraintes sur les probabilités de transition ($\Sigma_j a_{ij}=1$) et laisser les poids de transition dériver selon l'apprentissage.

Bottou [Bottou, 1991] propose une méthodologie très générale aux Alpha-nets, permettant de combiner différents modules, appris conjointement en propageant les dérivées partielles. Le but de cette coopération des modules optimise le critère global, par rapport aux systèmes dont chaque module a été repris séparément. Son système était basé sur une combinaison TDNN et programmation dynamique (DTW) par LVQ2+. Nous reprendrons ce type de schéma dans notre application sur la reconnaissance de mots au chapitre 6. Cette méthode a été reprise dans les transducteurs et les réseaux de transformations de graphes dans l'article de LeCun et Bengio [LeCun, 1998a].

3.4.2 Méthodes d'optimisation.

Dans ce qui suit, nous allons maintenant nous intéresser aux différents modes d'apprentissage et fonctions d'optimisation associées et nous conclurons sur un tableau

reprenant différents systèmes hybrides RN-MMC en reconnaissance de l'écriture et leur schéma d'apprentissage.

3.4.2.1 L'apprentissage MLE

L'apprentissage par maximum de vraisemblance est l'apprentissage le plus courant rencontré dans les systèmes de reconnaissance de l'écriture manuscrite basés sur des MMC. Il est possible de l'étendre à un système hybride : on conserve le même type de fonction de coût et l'on entraîne le réseau de neurones par l'algorithme de rétropropagation du gradient de cette fonction. On rappelle que la fonction de coût du maximum de vraisemblance (MLE pour Maximum Likelihood Estimation en anglais) est la moyenne géométrique de la vraisemblance de tous les exemples e, elle s'exprime par l'équation suivante :

$L_{MLE} = \prod_e P(O^e \mid \Lambda)$, Λ étant l'ensemble des paramètres du système hybride.

Dans le cadre d'un apprentissage stochastique avec une rétropropagation du gradient, on utilise le logarithme de la vraisemblance, soit la fonction suivante :

$L_{MLE} = \log P(O \mid \lambda_{vrai})$, λ_{vrai} étant le modèle vrai (modèle à apprendre).

Ainsi, on optimise les paramètres du système en augmentant la vraisemblance du vrai modèle par rapport aux autres modèles.

L'entraînement des paramètres d'un système hybride, transition ou émission, peut être réalisé par optimisation par descente de gradient avec pas adapté :

$$\Lambda^{k+1} = \Lambda^k + \mu^k \times \frac{\partial L}{\partial \Lambda^k},$$ μ^k le pas de descente.

Dans le cas d'un apprentissage unifié avec des transitions de probabilités équiprobables ou unitaires, seuls les poids du réseau de neurones sont adaptés. On calcule le gradient de la fonction d'optimisation par rapport aux poids en utilisant la règle des dérivées en chaîne :

$$W^{k+1} = W^k + \mu^k \times \frac{\partial L}{\partial W^k},\ \mu^k \text{ le pas de descente}$$

$$\text{et } \frac{\partial L}{\partial W} = \sum_j \left(\frac{\partial L}{\partial b_j(O_t)} \times \frac{\partial b_j(O_t)}{\partial W}\right) \qquad \forall O_t$$

L'indice j permet de balayer les classes en sortie du réseau.

La vraisemblance d'un modèle mot λ pour une séquence d'observation donnée O peut être défini par :

$$P(O \mid \lambda) = \sum_{\Gamma} \prod_{t=1}^{T} a_{q_{t-1}q_t} b_{q_t}(O_t)$$

où $a_{q_{t-1}q_t}$ et b_{q_t} sont respectivement les probabilités de transitions et les probabilités d'émissions. La vraisemblance correspond à la somme sur tous les chemins possibles Γ à travers le MMC λ [Rabiner, 1989].

En fixant les probabilités de transitions, seules les dérivées des probabilités d'observation influent sur le calcul du premier terme du gradient de L_{MLE}.

$$\frac{\partial L_{MLE}}{\partial b_j(O_t)} = \frac{\partial}{\partial b_j(O_t)}(\log P(O \mid \lambda_{vrai})) = \frac{1}{P(O \mid \lambda_{vrai})} \cdot \frac{\partial P(O \mid \lambda_{vrai})}{\partial b_j(O_t)}$$

$$avec \ \frac{\partial b_{q_t}(O_t)}{\partial b_j(O_t)} = \delta_{j,q_t} \cdot \frac{b_{q_t}(O_t)}{b_j(O_t)} \ \text{ où } \ \delta_{i,j} = \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases}$$

Ainsi la dérivée de la vraisemblance d'un mot par rapport aux probabilités des observations s'écrit :

$$\frac{\partial P(O \mid \lambda)}{\partial b_j(O_t)} = \sum_{\Gamma} \prod_{t=1}^{T} a_{q_{t-1}q_t} \delta_{j,q_t} \cdot \frac{b_{q_t}(O_t)}{b_j(O_t)}$$

$$\begin{aligned} \frac{\partial P(O \mid \lambda)}{\partial b_j(O_t)} &= \frac{1}{b_j(O_t)} \sum_{\Gamma} \prod_{t=1}^{T} a_{q_{t-1}q_t} \delta_{j,q_t} b_{q_t}(O_t) \\ &= \frac{1}{b_j(O_t)} \sum_{\Gamma} P(O, \Gamma, q_t = j \mid \lambda) \\ &= \frac{1}{b_j(O_t)} P(O, q_t = j \mid \lambda) \end{aligned}$$

où $P(O, q_t = j \mid \lambda)$ peut être calculé par un algorithme de programmation dynamique. On obtient donc pour le premier terme :

$$\frac{\partial L_{MLE}}{\partial b_j(O_t)} = \frac{1}{b_j(O_t)} \times \frac{P(O, q_t = j \mid \lambda_{vrai})}{P(O \mid \lambda_{vrai})}$$

$$où \ \sum_j P(O, q_t = j \mid \lambda) / P(O \mid \lambda)$$

est la probabilité que l'observation donnée correspond à un caractère du MMC mot λvrai.

Pour le second terme du gradient ∂L/∂W, $b_j(O_t)=P(O_t|q_t=j)$ correspond à la $j^{ème}$ sortie du réseau de neurones pour l'observation O_t. La dérivée de $b_j(O_t)$ par rapport aux poids s'obtient par l'algorithme classique de rétropropagation, cf. chapitre 2.

La Figure 47 illustre cette idée d'apprentissage au niveau mot. Le signal d'erreur rétro propagé dans le réseau de neurones (par exemple PMC sur le schéma) est basé sur une fonction de coût au niveau mot et modifie les sorties associées aux classes du mot vrai.

Avec cette illustration on constate qu'un apprentissage MLE ne propage que des gradients positifs ou nuls dans le réseau, aucun gradient négatif. Pour une observation particulière, on augmente les probabilités des classes de caractères (ou graphèmes) appartenant au mot considéré vrai. Pour assurer la convergence de l'apprentissage, il est nécessaire d'utiliser une fonction Softmax comme fonction d'activation des neurones de sorties, ainsi celle-ci normalise la somme des sorties à 1 et tire vers le bas, par un effet balancier, toutes les classes de caractères n'appartenant pas au mot.

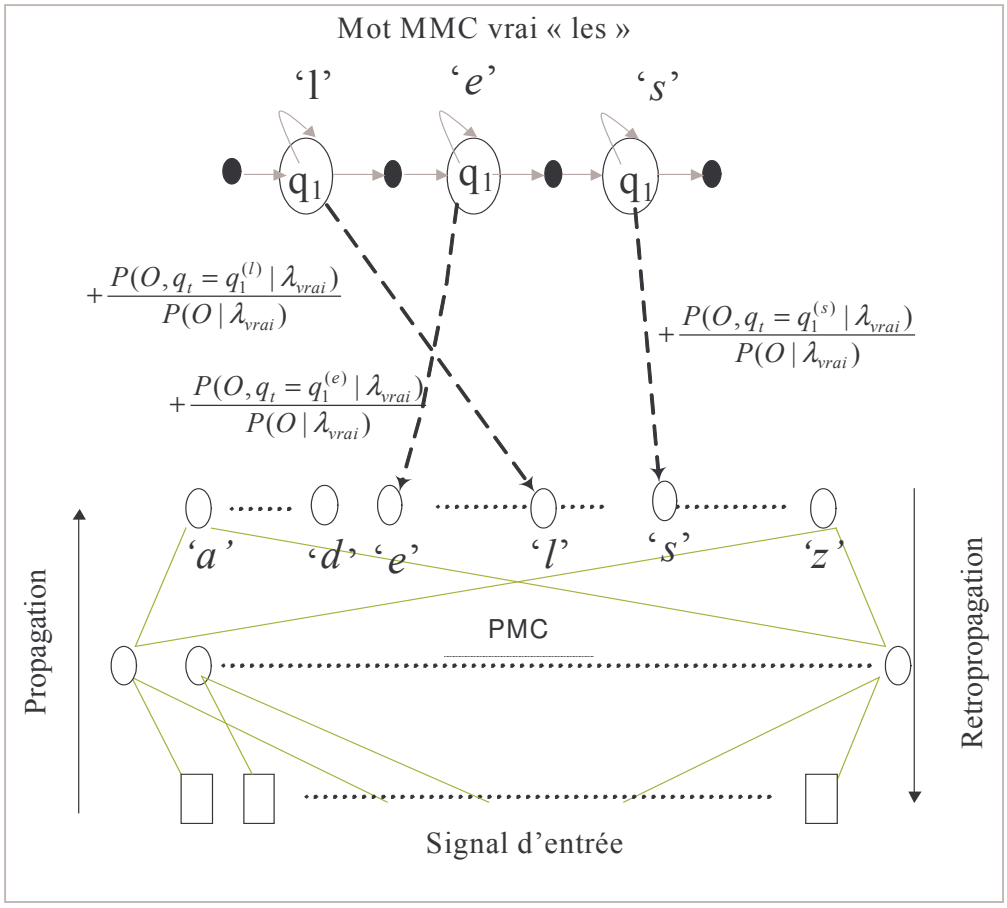


Figure 47. Rétropropagation du gradient dans un apprentissage MLE

Le critère du maximum de vraisemblance est un critère de modélisation : chaque modèle maximise les probabilités des seuls exemples de sa classe et n'est jamais optimisé pour minimiser les probabilités des occurrences des autres classes. Pour assurer des performances correctes, cet apprentissage nécessite des bases de données d'apprentissage conséquentes en taille et requiert une phase d'initialisation au niveau caractère pour assurer sa stabilité.

L'apprentissage MMI présenté ci-après (MMI : Maximum Mutual Information) est au contraire un critère discriminant.

3.4.2.2 L'apprentissage MMI

Ce critère appliqué initialement en reconnaissance de la parole a ensuite été appliqué à la reconnaissance de l'écriture manuscrite hors-ligne [Tay, 2002]. Optimiser un critère d'information mutuelle consiste pour une séquence d'observation donnée d'une part à augmenter la vraisemblance du modèle mot MMC vrai et d'autre part à discriminer tous les autres modèles (M modèles).

$$L_{MMI} = \log \frac{P(O \mid \lambda_{vrai})}{\sum_i P(O \mid \lambda_i)}$$
$$= \log P(O \mid \lambda_{vrai}) - \log \sum_i P(O \mid \lambda_i)$$
$$avec\ i = 1, 2, ..., M$$

Pour réduire les coûts de complexité de ce critère, une version simplifiée notée L_{MMIs} est la plus souvent utilisée ne considérant une discrimination que par rapport au modèle de plus forte vraisemblance.

$$L_{MMIs} = \log P(O \mid \lambda_{vrai}) - \log P(O \mid \lambda_{TOP1})$$
$$où\ \lambda_{TOP1} = \arg\max_i P(O \mid \lambda_i)$$

Ainsi la fonction du critère MMI simplifié est basée sur la différence des logarithmes du modèle MMC vrai, λ_{vrai} et du modèle reconnu en première position, λ_{TOP1}. Ainsi si le modèle reconnu en TOP 1 est le modèle vrai, la fonction L_{MMIs} est égale à zéro : elle a atteint son maximum et aucune erreur n'est rétropropagée dans le réseau. Sinon on applique comme pour le critère précédent de maximum de vraisemblance l'algorithme de descente de gradient. Les calculs étant très similaires nous ne donnerons que les grandes lignes.

$$W^{k+1} = W^k + \mu^k \times \frac{\partial L_{MMIs}}{\partial W^k},\ \mu^k \text{ le pas de descente}$$

$$\text{et } \frac{\partial L}{\partial W} = \sum_j \left(\frac{\partial L_{MMIs}}{\partial b_j(O_t)} \times \frac{\partial b_j(O_t)}{\partial W}\right) \qquad \forall O_t$$

$$\frac{\partial L_{MMIs}}{\partial b_j(O_t)} = \frac{1}{b_j(O_t)} \times \left[\frac{P(O, q_t = j \mid \lambda_{vrai})}{P(O \mid \lambda_{vrai})} - \frac{P(O, q_t = j \mid \lambda_{TOP1})}{P(O \mid \lambda_{TOP1})} \right]$$

La Figure 48 caractérise la rétropropagation du gradient lors de l'apprentissage du mot « les », le mot reconnu en première position étant par exemple mot « des ». En addition à la correction des classes caractères du modèle vrai, on pénalise les sorties correspondantes aux classes du mot reconnu en TOP 1.

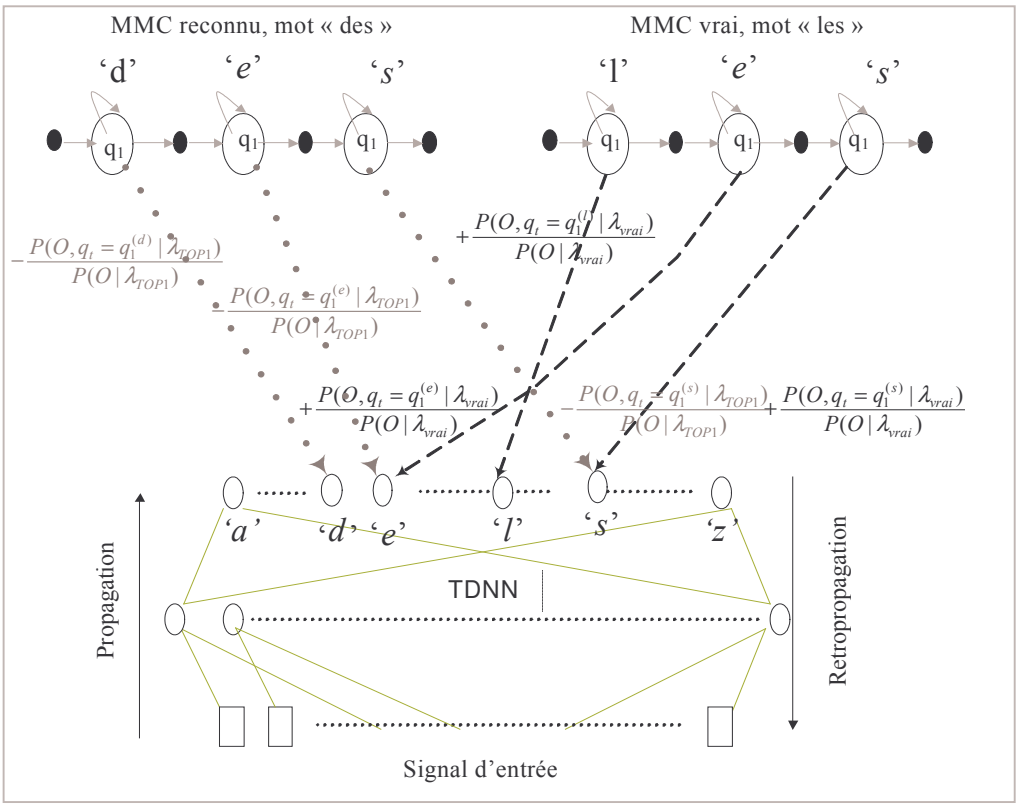


Figure 48. Rétropropagation du gradient d'un critère MMI simplifié

Les expérimentations dans des systèmes de type INSEG, notamment dans [Tay, 2002; Chen, 2000] montrent que ce critère de discrimination permet d'accélérer la convergence de la fonction de coût et accroître les résultats en généralisation par rapport à un critère classique de type maximum de vraisemblance.

3.4.2.3 L'apprentissage MAP et REMAP

Il est possible de relier le critère de Maximum d'Information Mutuelle (MMI) avec le critère par Maximum de Probabilité à Posteriori (MAP). Dans le cas d'une optimisation MMI, les MMC évaluent la vraisemblance de la séquence d'observations p(O|λ). A l'opposé, les MMC MAP, appelés aussi MMC discriminant, évaluent directement la probabilité a posteriori p(λ|O) du modèle à la vue de la séquence. Cette probabilité s'exprime par l'équation suivante :

$$p(\lambda \mid O) = \sum_{\Gamma_\lambda} P(\Gamma_\lambda \mid O_1 ... O_T)$$

par hypothèse d'un processus de Markov, on a:

$$p(\lambda \mid O) = \sum_{\Gamma} \prod_{t=1}^{T} p(q_t \mid q_{t-1}, O_t)$$

où Γ représente un chemin $q_1 \ldots q_T$ à travers le modèle MMC λ. Les probabilités $p(q_t|q_{t-1},O_t)$ sont habituellement séparées en probabilité d'observation $p(O_t|q_t)$, sortie du RN et probabilité de transition $p(q_t|q_{t-1})$ évaluée par les MMC. Or l'idée de l'apprentissage compétitif pour un MMC consiste à utiliser un réseau pour optimiser une mesure de discrimination fondée sur la probabilité *a posteriori* d'occupation d'un état $p(q_t|q_{t-1},O_t)$. Cette méthode permet d'améliorer les performances de l'algorithme classique de Baum-Welch. REMAP (Bourlard, 1995) est une méthode hybride également à rapprocher de ces techniques.

3.5 Conclusion

Dans ce chapitre nous nous sommes intéressés aux systèmes hybrides neuro-markoviens et leur apprentissage, notre but étant d'utiliser des architectures couplant un réseau de neurones à convolution en amont d'un modèle de Markov caché avec un apprentissage unifié.

De tels schémas ont été mis en œuvre dans un nombre restreint de systèmes de reconnaissance de l'écriture en-ligne. On peut en noter principalement trois : Pénacée [Schenkel, 1995], Npen++ [Jaeger, 2000] et Remus [Wimmer, 1999]. Dans ce dernier cas, le RN est un PMC standard, réellement il y a autant de PMC distincts que d'états lettres et ils jouent le rôle de prédicteurs de trames (à partir de la connaissance des 3 trames précédentes). Le MMC cherche alors à minimiser l'erreur de prédiction sur le treillis construit sur les différences entre trames prédites et trames observées le long de la séquence complète d'observations.

Les deux autres systèmes intègrent quant à eux une coopération TDNN-MMC. En ce sens, ils se rapprochent davantage du système que nous avons développé et qui sera présenté au chapitre 6. Cependant, comme une grande majorité des systèmes de reconnaissance de mots cursifs, ils sont appris selon une optimisation du maximum de vraisemblance. Or, comme nous l'avons cité, un critère MLE nécessite un premier apprentissage au niveau caractère et donc requiert une base de données caractères soit issue de la base de mots segmentés à la main, soit étiquetée par un autre classifieur au préalable.

La table, page suivante, illustre quelques systèmes hybrides RN-MMC de reconnaissance de l'écriture manuscrite cursive avec leur schéma d'apprentissage. Le premier système [Knerr, 1998] cité correspond à un système perceptron multicouche MMC dédié à la reconnaissance de mots cursifs hors-ligne. Les deux classifieurs sont appris séparément.

Les deux systèmes suivants sont dédiés à la reconnaissance de mots en-ligne. Dans ce cadre la plupart des classifieurs sont entraînés avec un critère de type modèle générateur qui cherche à maximiser la vraisemblance du vrai modèle sans prendre en compte les autres modèles. L'apprentissage des systèmes Pénacée et Npen ++ basé sur un critère de maximum de vraisemblance se déroule en trois étapes dont les deux premières se basent sur des données d'apprentissage de type caractères isolés et dont seule la dernière phase concerne l'apprentissage des mots. Dans le cas du Pénacée, la première étape consiste à entraîner le réseau de type TDNN sur des caractères isolés, la seconde a entraîné le réseau à caractériser la classe de non-présence de caractères. Le système Npen++ est basé sur une architecture TDNN multi état, une lettre est représentée par un modèle gauche-droite à 3 états (début, milieu et fin de la lettre), à un état correspond une sortie du TDNN. La première phase d'apprentissage consiste à apprendre des caractères sur une base segmentée à la main avec une contrainte de durée égale pour chaque état du modèle lettre (si un caractère est découpé en 6 observations, il devra tout d'abord passer 2 fois dans le premier état puis 2 fois dans le deuxième état), et enfin 2 fois dans le dernier état. La deuxième phase consiste à apprendre ces mêmes caractères en relâchant la contrainte sur la durée des états.

Les systèmes suivants [Tay, 2001 ; Chen, 2000] basés sur une architecture PMC-MMC avec segmentation explicite mettent en évidence l'intérêt des critères d'information mutuelle en reconnaissance de mots cursifs hors-ligne par rapport au critère MLE. Tay compare un apprentissage global discriminant basé sur un critère de maximum d'information mutuelle simplifié avec un apprentissage au niveau caractère, les résultats montrent qu'un apprentissage global permet de réduire de 15 pourcent l'erreur de classification sur la base IRONOFF [Viard-Gaudin, 1999]. Chen et Gader procèdent tout d'abord à une initialisation des paramètres du réseau de neurones avec un apprentissage au niveau caractère et ensuite entraîne le schéma hybride avec une fonction de coût de type MMI.

Tableau 2. Caractérisation de systèmes hybrides RN-MMC par leur domaine de reconnaissance de l'écriture, classifieur et mode d'apprentissage.

Auteurs	Domaines	Classifieur	Mode d'apprentissage
Knerr, Augustin [Knerr, 1998]	Hors-ligne	MLP-MMC Seg. implicite 44 classes	Apprentissage séparé : -apprentissage du MLP -apprentissage des MMC
Pénacée Schenkel, Guyon, Henderson [Schenkel, 1995]	En-ligne	TDNN MMC 26 classes	Cross entropie (MLE), 3 étapes : 1. Apprentissage du TDNN par des caractères isolés 2. Apprentissage de la classe de non-présence de caractères (nil-class) 3. Apprentissage au niveau mot
Npen++ Jaeger, Manke, Reichert, Waibel [Jaeger, 2000]	En-ligne	MsTDNN MMC 26 classes x 3	Cross entropie (MLE), 3 étapes : 1. Apprentissage sur la base segmentée à la main par le chemin Viterbi du modèle vrai avec la contrainte de la même durée pour chaque état. 2. Même apprentissage avec aucune contrainte sur le chemin Viterbi. 3. Apprentissage au niveau mot
Tay, Lallican [Tay, 2001]	Hors-ligne	MLP-MMC segmentation explicite	Apprentissage global discriminant MMI simplifié Comparaison avec un apprentissage au niveau caractère
Chen , Gader [Chen, 2000]	Hors-ligne	NN-MMC Segmentation explicite	Apprentissage MMI + fonction discriminative Apprentissage au niveau caractère puis au niveau mot
Caillault, Viard-Gaudin [Caillault, 2005]	En-ligne	Ms-TDNN MMC 66 classes × 3	Apprentissage global discriminant au niveau mot Fonction de coût générique combinant MLE,MMI et discrimination TDNN

Dans ce cadre, nous avons cherché à intégrer aussi bien les modèles générateurs que les modèles de discrimination, dernier cas proposé dans le tableau 2 ; nous détaillerons une fonction de coût générique proposée au chapitre 7.

PARTIE II : RECONNAISSANCE DE CARACTÈRES ISOLÉS

4 TDNN POUR LA RECONNAISSANCE DE CARACTERES ISOLES EN-LIGNE

Cette partie est consacrée à la reconnaissance de caractères isolés. Nous détaillerons les différents systèmes, que nous avons implémentés, basés sur les réseaux de neurones à convolution que nous avons introduits au chapitre 2. Le premier système, basé sur une architecture TDNN est appliqué à la reconnaissance de caractères en-ligne, il sera développé dans ce chapitre. Le second est le parallèle dédié à la reconnaissance de caractères hors-ligne, il est basé sur un réseau de type SDNN. Le chapitre suivant présente une nouvelle architecture permettant de coupler de l'information temporelle et spatiale basée sur les deux reconnaisseurs précédents, TDNN et SDNN, dans le cadre d'une reconnaissance en-ligne.

Le but de ce chapitre est de présenter la brique de base de notre reconnaisseur mot, le TDNN, et d'analyser l'impact de ces paramètres sur une application de reconnaissance de caractères isolés en-ligne.

4.1 Présentation générale d'un reconnaisseur de caractères isolés

Un système de reconnaissance de caractères (SRC) accepte les données provenant d'un équipement en ligne ou hors ligne comme des données d'entrée, en assure le traitement et produit des données en sorties compréhensibles. On rappelle que le schéma général du système fait apparaître plusieurs composants, Figure 49. L'un de ces composants se charge des fonctions de prétraitement comme la normalisation. Une fois que la forme a été prétraitée, un autre composant en extrait les attributs caractéristiques. Les caractéristiques ainsi extraites sont utilisées par un composant de « classification » (comme un réseau de neurones) pour attribuer une étiquette à la forme. Il est bon de noter que l'étape d'extraction des caractéristiques n'est pas toujours nécessaire.

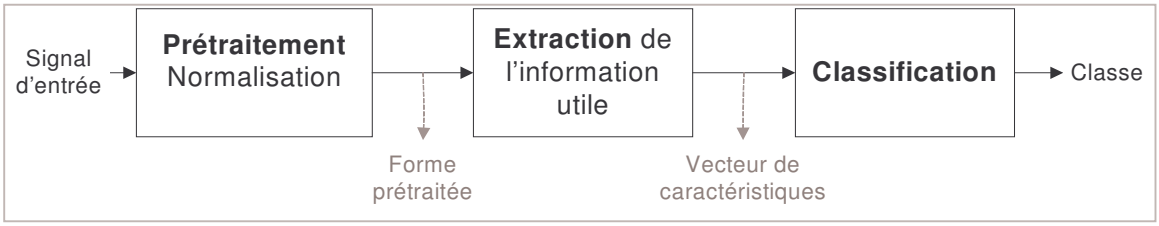


Figure 49. Synoptique du système de reconnaissance de caractères.

Notre système comporte ces trois composants que nous allons détailler séparément par la suite. Il reçoit en entrée un signal en-ligne (exactement le fichier des coordonnées des points d'un caractère au format Unipen [Guyon, 1994]). Ce signal est alors rééchantillonné en un nombre de points fixes puis normalisé. De ce signal normalisé, on extrait une séquence de vecteurs de sept caractéristiques par point qui sera présentée à un réseau de neurones à délai temporel, TDNN qui fournira en sortie un vecteur des probabilités *a posteriori* de chaque classe.

Le système présenté rejoint le reconnaisseur de Guyon et al. [Guyon, 1991] présenté au chapitre 2. Nous avons cherché à optimiser la structure d'un tel système : nous avons cherché le

meilleur compromis performances/architecture pour des applications à ressources réduites telles que les assistants personnels, téléphones portables.

Les spécifications du système de reconnaissance de caractères isolés que nous avons développé sont les suivantes :

- Les caractères sont entrés sur une tablette digitale. Ils sont représentés comme une séquence de coordonnées [x(t), y(t)].
- Les caractères sont dessinés dans des fenêtres disjointes.
- Tous les chiffres et lettres de l'alphabet français sont considérés.
- Le système doit être complètement automatique.
- Aucune analyse syntaxique, sémantique ou autre information ne sera utilisée.

4.2 Prétraitement et extraction de caractéristiques

Nous avons choisi de préserver la nature séquentielle de l'information acquise par la tablette électronique. L'ensemble du processus de normalisation et extraction des caractéristiques est représenté par la Figure 50.

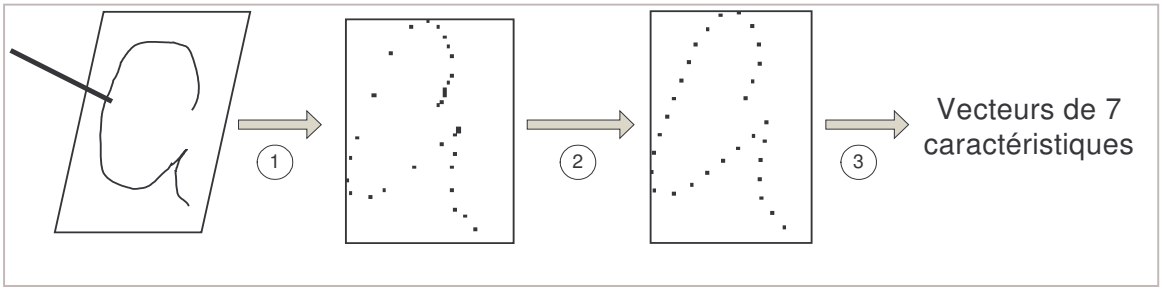


Figure 50. Représentation du processus d'extraction en ligne (1- acquisition du signal 2- rééchantillonage spatial 3- normalisation et extraction des caractéristiques)

Nous allons maintenant détailler ces étapes.

4.2.1 Acquisition du signal

L'entrée du reconnaisseur est un fichier au format UNIPEN, un exemple tiré de la base IRONOFF (base développée dans notre laboratoire) est donné à la *Figure* 51. Les données brutes du signal contiennent d'importantes variations selon la vitesse d'écriture. Entraîner un réseau de neurones avec de telles données conduit à de faibles performances (60% d'erreurs de reconnaissance). Echantillonner pour obtenir un nombre constant de points espacés régulièrement le long de la trajectoire du stylet apporte un meilleur taux de reconnaissance.

```
.DATE          08 07 1998          ← date de l'écriture
.COMMENT       Declarations        ← dimension du cadre permis pour écrire
.X_DIM            3600
.Y_DIM            3600
.X_POINTS_PER_INCH 300
.Y_POINTS_PER_INCH 300
.POINTS_PER_SECOND 100             ← fréquence d'acquisition
.COORD         X Y P T             ← contenu des données
.COMMENT       Data
```

```
.WRITER_ID   sk18091962          ← informations sur le scripteur
.COUNTRY     Germany
.HAND        R
.AGE         35
.SEX         M
.SEGMENT DIGIT ? ? "4"           ← label du caractère écrit
.PEN_DOWN                        ← début de l'acquisition : poser du stylet
70 85 4 0                        ← coordonnées :
70 85 16 11                      1ère est la position horizontale X(t)
68 90 27 23                      2ème la position verticale Y(t)
65 95 34 35                      3ème la pression du stylet sur la tablette
64 98 41 46                      4ème le temps t.
63 100 42 57
63 105 45 66
65 108 47 76
69 110 48 88
75 110 48 97
81 109 45 107
87 107 21 117
92 106 8 127
.PEN_UP                          ← détection d'un lever du stylet
.PEN_DOWN                        ← détection d'un nouveau poser du stylet
98 92 14 148
96 95 23 158
91 101 44 168
88 109 48 178
85 116 51 189
84 122 51 200
84 128 43 212
.PEN_UP                          ← détection d'un lever du stylet et fin
```

Figure 51. Fichier au format UNIPEN d'un chiffre 4 saisi sur une tablette.

4.2.2 Ré-échantillonage spatial

Cette étape consiste à réduire l'information redondante (répétition de points) ou non pertinente telle que les distorsions temporelles et/ou spatiales et ne conserver que l'information pertinente. En effet, selon les scripteurs, le temps de formation d'une lettre peut être plus ou moins long et la forme des caractères très différente. De même la pression est peu utile.

Du fichier UNIPEN, on analyse seulement les coordonnées de position horizontales et verticales et les informations de lever et poser de stylet (.Pen_UP et .Pen_Down) qui caractérise un nouveau trait. On détermine la matrice des caractères échantillonnés [X, Y, PenUpDown] où X est le vecteur des abscisses du signal échantillonné en un nombre fixé de points N donné en paramètre, respectant l'ordre temporel de la formation de la lettre. De la même façon, Y est le vecteur des ordonnées verticales. On distingue deux styles de points selon l'état du stylet, les points où le stylet appuie sur la tablette et les points ne caractérisant pas l'appui du stylet mais le mouvement de la main dans l'écriture. Nous spécifions l'état du stylet au point n, PenUp (stylet levé) ou PenDown (stylet posé) en le codant par l'ajout d'une variable supplémentaire PenUpDown(n) de la façon suivante : PenUpDown=-1 pour le stylet levé et PenUpDown=1. Sur la Figure 52, qui illustre quelques exemples de caractères avant et après rééchantillonage spatial, les points caractérisant l'appui du stylet (PenUpDown=-1) sont matérialisés par les points, les autres par des carrés.

Les caractères échantillonnés sont ainsi représentés comme une séquence de points [X(n),Y(n)] (n numéro du point) espacés par une distance régulière selon la longueur de l'arc, par opposition au temps dans la séquence d'entrée. Par expérimentation, nous avons fixé le nombre de points à 50 pour chaque caractère (30 points étant suffisant pour catégoriser

uniquement les chiffres, 50 points sont nécessaires dans le cadre des lettres minuscules et majuscules). Nous appliquons une interpolation linéaire avec comme métrique la distance euclidienne dans l'algorithme d'échantillonnage donné ci-après ().

```
  //Définition du trait : séquence de points d'un lever/poser de
stylet
  N : paramètre, nombre de points échantillonnés
  Allocation mémoire des sorties : vecteurs X,Y, PenUpDown
  //Calcul de la longueur totale du signal
  Balayage des traits du signal
        Balayage des points du trait
                Longueur = Longueur + distance entre les points
        Si plusieurs traits
                Alors Longueur = Longueur + Longueur entre les traits
  //Calcul de la distance d'échantillonnage dist_ech
  dist_ech = Longueur/(N-1);
   Conservation du 1er point
   Détermination des N-1 autres points par interpolation
```

Algorithme 2. Algorithme de ré-échantillonnage spatial

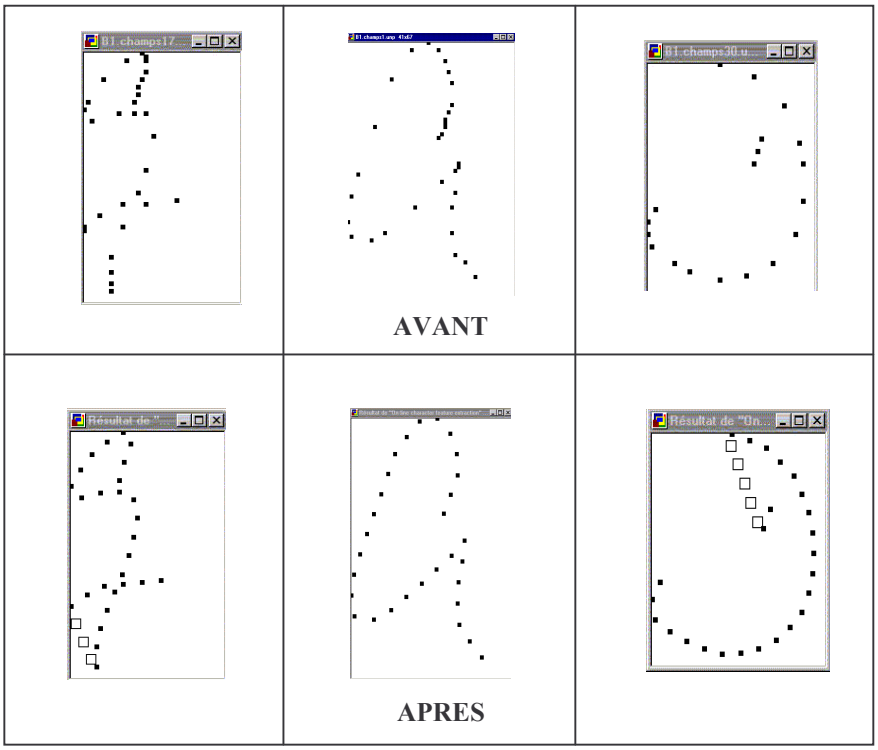


Figure 52. Exemples de caractères avant et après échantillonnage(chiffre 7, minuscule a, majuscule D ; les carrés représentent les levers de stylet).

4.2.3 Normalisation du signal et extraction des caractéristiques

Cette dernière phase du processus est l'extraction d'information importante, pertinente du signal acquis. Avant d'analyser l'information acquise, il est nécessaire de recentrer le caractère manuscrit et de le normaliser. Le but du recentrage et de la normalisation est d'obtenir une représentation invariante aux translations et aux distorsions spatiales, la taille des caractères et le style d'écriture (italique, droit…) varient. Le signal est donc recentré par rapport au centre de

l'image puis normalisé par rapport à la taille maximale du caractère. Ensuite dans le but de faciliter la tâche du classifieur, on extrait de cette séquence de points des informations géométriques locales telles que la direction du mouvement et la courbure de la trajectoire. On obtient alors une séquence de vecteurs de 7 caractéristiques - Coordonnées, direction, courbure, lever ou poser du stylet – par point.

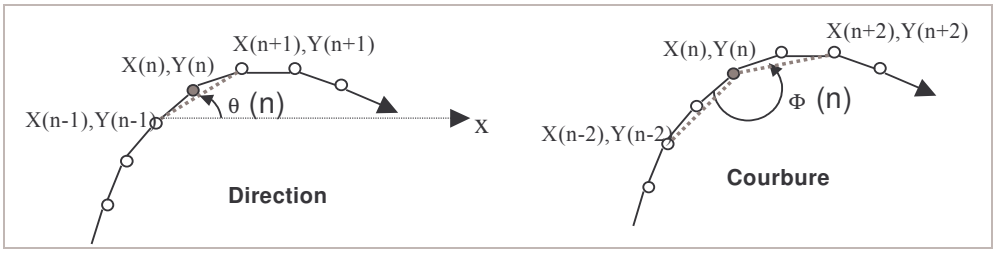


Figure 53. Illustration des angles dans le calcul de la direction (angle θ) et courbure de la trajectoire (angle Φ)

Ainsi, nous avons fixé le nombre de points à 50 pour chaque caractère. En chacun de ces points, un vecteur de 7 caractéristiques est extrait. Il comporte les coordonnées x et y, les cosinus directeurs de la direction (cosθ, sinθ) et de la courbure (cosΦ, sinΦ) de la trajectoire et l'état poser ou lever du stylet en ce point, cf. Figure 53.

L'algorithme général est donné ci-dessous.

```
allocation de la sortie : matrice x
calcul du centre de l'image [x0,y0]
initialisation xmax=x(0) , xmin=x(0)et ymin=y(0), ymax=y(0)
Balayage sur tous les points, n de 0 à N-1
    Si x(n) supérieur à xmax alors xmax ← x(n)
    Si x(n) inférieur à xmin alors xmin ← x(n)
    Si y(n) supérieur à ymax alors ymax ← x(n)
    Si y(n) supérieur à ymin alors ymin ← x(n)
    X0 ← (xmin+xmax)/2 ;
    y0 ← (ymin+ymax)/2 ;
//calcul du facteur d'échelle delta
delta_y=(ymax-ymin);
delta_x=(xmax-xmin);
si delta_x<delta_y alors delta=delta_y;
sinon delta = delta_x;
//calcul des nouvelles coordonnées x[n][0] (coordonnée en x) et
x[n][1] (coordonnée en y)
Balayage des points (de X et de Y)
    x[n][0] = (X[n]-x0)/delta ;
    x[n][1] = (Y[n]-y0)/delta ;
    x[n][6]=PenUpDown[n] ;
    //calcul de la direction x[n][2] x[n][3] (cosinus directeurs)
    Balayage des points
          // calcul de la longueur de la corde dS
          d_x=X[i+1]-X[i-1]
          d_y=Y[i+1]-Y[i-1]
          dS= sqrt(d_x*d_x+d_y*d_y)
          si cette distance est nulle
                alors x[i][2]=0;  x[i][3]=0;
          sinon x[i][2]=arc_x/dS; x[i][3]=arc_y/dS
    //points particuliers : point initial et final
```

```
point initial = point suivant
point final = point précédent
//calcul de la courbure x[n][4] et x[n][5] (cosinus directeurs)
Balayage des points
      x[i][4]=x[i+1][2]*x[i-1][2]+x[i+1][3]*x[i-1][3];
      x[i][5]=x[i+1][2]*x[i-1][3]-x[i+1][3]*x[i-1][2];
//points particuliers
point initial = point suivant
point final = point précédent
```

Algorithme 3. Algorithme de normalisation et extraction des caractéristiques

On obtient ainsi une matrice de taille fixe 50×7 représentant un caractère tel que l'illustre la Figure 54.

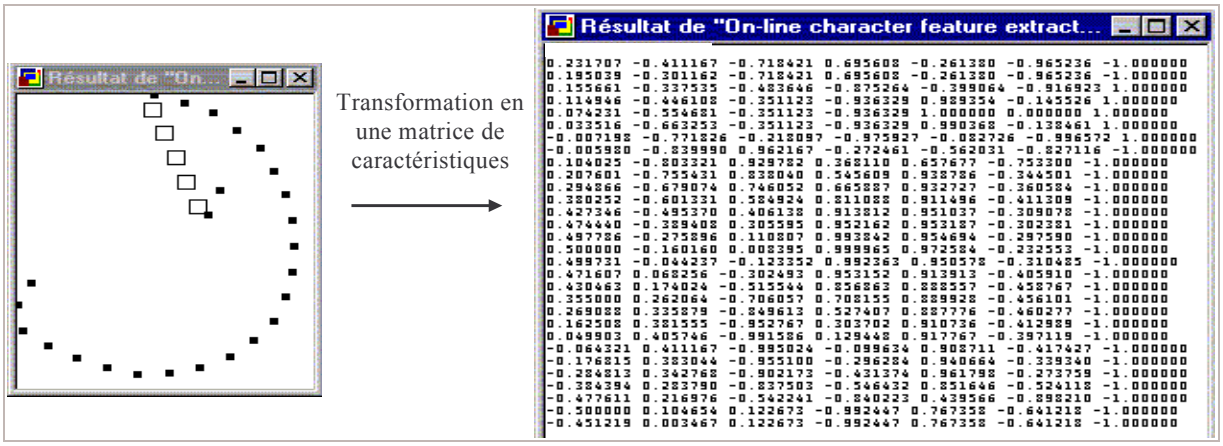


Figure 54. Résultat de l'extraction des caractéristiques (lettre D)

Cette matrice est ensuite propagée dans le TDNN. A la sortie du réseau de neurones, chaque neurone fournit en sortie la probabilité du label associé. Le neurone dont la sortie est la plus proche de la valeur 1 définira le label du caractère reconnu.

4.3 Architecture et apprentissage du TDNN

Nous allons maintenant introduire l'implémentation que nous avons réalisée de l'architecture du TDNN et présenter les algorithmes associés de reconnaissance et d'apprentissage. Le réseau, illustré Figure 55 est composé d'un réseau de neurones à décalage temporel avec poids partagés ayant pour objectif l'extraction de caractéristiques suivi d'un perceptron linéaire ou multicouche.

4.3.1 Implémentation du TDNN

L'architecture retenue comporte deux parties principales, extraction et classification. La première, correspondant aux couches basses, implémente les convolutions successives permettant de transformer progressivement les caractéristiques en grandeurs de plus en plus significatives vis-à-vis du problème. La seconde correspond à un perceptron multicouche (PMC) classique, il reçoit en entrée l'ensemble des sorties de la partie extraction du TDNN. Ces

deux blocs sont complètement paramétrables, ils sont décrits par les grandeurs présentées ci-après.

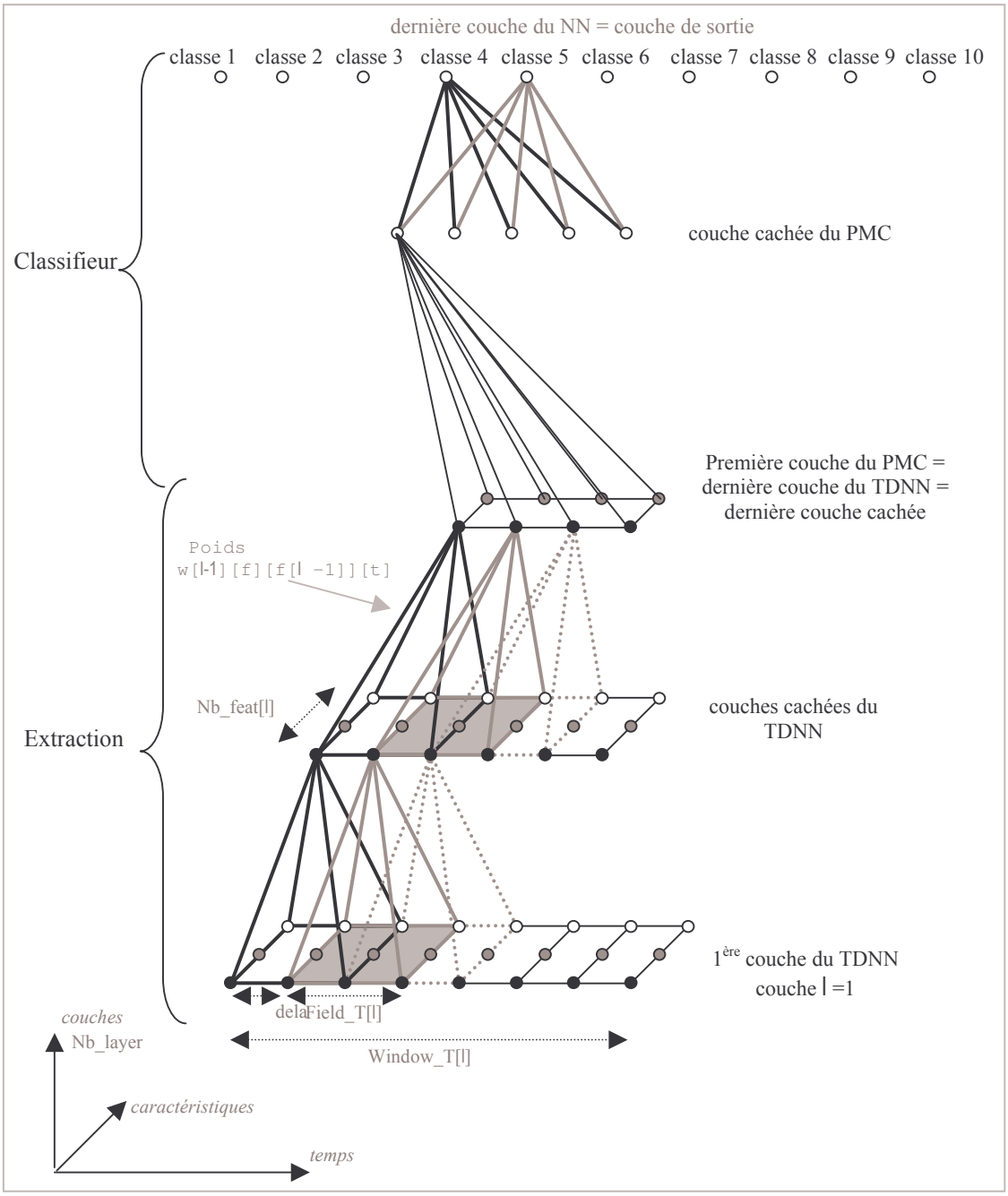


Figure 55. Architecture du TDNN. Il est à noter que par souci de clarté les biais n'ont pas été mentionnés. Le nombre de classes est égal au nombre de labels à discriminer comme par exemple 10 sorties pour une reconnaissance de digits.

La partie TDNN se caractérise par :

- Le nombre de couches, nb_layer
- Le nombre de neurones de chaque couche selon la direction temporelle, window_T ; ce paramètre se déduit du nombre de retards (Delay[l-1]) dans les connexions de la couche

précédente (Window_T[l-1]) et de la largeur de la fenêtre de spécialisation (field_T[l-1]) de la manière suivante

$$Window\_T[l] = \frac{Window\_T[l-1] - nb\_feat[l-1]}{delay[l-1]} + 1$$

- Le nombre de neurones de chaque couche selon la direction caractéristique, nb_feat
- La taille de la fenêtre temporelle vue par chaque couche (sauf celle d'entrée) soit le nombre de neurones de la couche l vu par un neurone de la couche l+1, field_T
- Le délai temporel (nombre de neurones) entre chaque fenêtre, delay

Un neurone de la partie TDNN est identifié par sa couche l, sa caractéristique f, et son emplacement temporel t. Pour chaque neurone sont définis :

- une sortie, ou encore activation du neurone, x[l][f][t],
- une matrice des poids des entrées, w[l-1][f][f[l-1]][t],
- un vecteur de poids des biais, w_biais[l-1][f],
- la somme pondérée des entrées, v[l][f][t],
- le terme d'erreur pour la rétro-propagation du gradient, y[l][f][t],
- le gradient, delta[l-1][f][f[l-1]][t],
- une fonction d'activation de type tangente hyperbolique.

Et la partie PMC se caractérise par :

- le nombre de couches, NN_nb_layer,
- le nombre de neurones de chaque couche NN_nb_neuron.

Un neurone de cette partie est identifié par sa couche l, et son emplacement temporel t. Pour chaque neurone sont définis :

- une sortie, ou encore activation du neurone, x[l][t],
- une matrice des poids des entrées, w[l-1][t][t[l-1]],
- un vecteur de poids des biais, w_biais[l-1],
- la somme pondérée des entrées, v[l][t],
- le terme d'erreur pour la rétropropagation du gradient, y[l][t],
- le gradient, delta[l-1][t][t[l-1]],
- une fonction de type tangente hyperbolique pour les couches cachées,
- une fonction d'activation de type Softmax pour la couche de sortie.

Par rapport aux couches de la partie extraction, la dimension des caractéristiques , [f], a ici disparu.

La première couche du réseau acquiert les caractéristiques du signal. Une ou plusieurs couches cachées du réseau de neurones (phase d'extraction) transforment une séquence de vecteurs caractéristiques en une autre séquence de vecteurs caractéristiques d'ordre supérieur. Un neurone donné détecte une caractéristique topologique locale de la trajectoire du stylet. Le champ de vision du neurone est restreint à une fenêtre temporelle limitée. Avec la contrainte des poids partagés, le même neurone est dupliqué dans la direction temps (soit la même matrice des

poids dupliquée) pour détecter la présence ou l'absence de la même caractéristique à différentes places le long de la trajectoire du signal. En utilisant plusieurs neurones (nb_feat) à chaque position temporelle, le réseau de neurones effectue la détection de caractéristiques différentes : les sorties des différents neurones produisent un nouveau vecteur caractéristique pour la couche supérieure.

Les opérations réalisées par une couche du TDNN sont de type convolution. Chaque neurone k de la couche l +1 a un noyau de taille w (nombre de neurones de la fenêtre temporelle de la couche l) × f (nombre de caractéristiques de la couche l).

La composante temporelle de la représentation du signal d'origine est éliminée au fur et à mesure en sous-échantillonnant la convolution à chaque couche. Pour compenser cette perte d'informations, le nombre de caractéristiques est multiplié. Nous avons donc une architecture de type bipyramidal. Ce réseau bipyramidal convertit progressivement des informations temporelles en information de type caractéristiques.

Enfin, la première couche de la partie classifieur (PMC entièrement connecté) correspond à la dernière couche de la partie extraction. En sortie, le PMC donne les probabilités *a posteriori* de l'entrée pour chaque classe définie.

4.3.2 Algorithmes d'apprentissage et de reconnaissance

L'algorithme de propagation d'un exemple est le suivant.

```
Entrée : TDNN_net    réseau
Sortie :   Float          matrice caractéristique de l'exemple

/* Phase d'extraction : TDNN  */
1. Phase d'acquisition des données TDNN
Balayage temporel des neurones de la 1ère couche du TDNN
    Balayage sur tous les neurones caractéristiques de la couche
          Acquisition des données
2. Extraction : couches cachées et couche de sortie du TDNN
Balayage sur toutes les couches l
    Balayage temporel des neurones de la couche t
          Balayage selon les caractéristiques de la couche f
                Initialisation des sommes pondérées v de chaque
                neurone au biais
                Balayage sur tous les neurones de la fenêtre associée
                      Calcul des sommes pondérées v de chaque neurone
                      Calcul  des  activations  x  de  chaque  neurone
                   (tanh)
                Décalage de la fenêtre suivant le délai associé

/* phase classifieur : PMC  */
3. Adaptation du TDNN au PMC (1ère couche, copie des sorties x)
4. Classifieur : Couches cachées du PMC
Balayage sur toutes les couches
    Balayage temporel des neurones de la couche t
      Initialisation  des  sommes  pondérées  v  de  chaque  neurone  au
  biais
      Balayage de tous les neurones de la couche précédente
          Calcul des sommes pondérées v de chaque neurone
          Calcul des sorties x de chaque neurone (tanh)
5. Résultat : dernière couche du PMC (Softmax)
Balayage temporel sur tous les neurones
    Calcul des sommes pondérées v de chaque neurone
    Calcul des activations x de chaque neurone (Softmax)
```

Algorithme 4. Algorithme de propagation d'un exemple

Un des points importants dans la mise en œuvre d'une solution à base de réseau de neurones concerne la méthode d'apprentissage [Hérault, 1994] retenue. Idéalement, celle-ci doit permettre de converger rapidement vers le minimum global de la fonction de coût sélectionnée. Suivant la complexité du problème d'optimisation, plusieurs types de méthodes peuvent être envisagés. D'un point de vue théorique, les méthodes de second ordre [Minoux, 1983] (Newton, Quasi-Newton, Gauss-Newton, Levenberg-Marquardt) sont considérées comme étant les plus performantes. Toutefois dans le cadre de l'application envisagée, vu le nombre de paramètres à estimer, ces méthodes conduisent à des charges de calcul et des occupations mémoires rédhibitoires. Nous avons donc dans un premier temps utilisé une méthode du premier ordre avec le classique algorithme de la rétropropagation du gradient, toutefois la version utilisée est la version dite du gradient stochastique qui permet de converger souvent plus rapidement que le gradient global [Sarle, 1997].

L'algorithme d'apprentissage utilisé est le suivant.

```
Allocation et définition du TDNN
Initialisation aléatoire (contrôlée) des poids du réseau
Construction d'une base normalisée
Pour chaque exemple de la base d'apprentissage e
    Rééchantillonnage de l'exemple e
    Normalisation de l'exemple e
    Extraction et sauvegarde des caractéristiques de l'exemple e
    Sauvegarde du label de l'exemple e
Tant que le critère d'arrêt n'est pas satisfait
    Pour chaque exemple de la base d'apprentissage e
          Propagation de l'exemple e
                Calcul de l'erreur locale
                Si le critère d'erreur locale non satisfait
                Alors Rétropropagation de l'erreur
    //Calcul du critère
    Pour chaque exemple e de la base d'apprentissage e
    Propagation de l'exemple e
    Calcul de l'erreur locale et de l'erreur cumulée
```

Algorithme 5. Algorithme d'apprentissage du TDNN

Cet algorithme est généralisable pour n'importe quel réseau de neurones de type perceptron ou à convolution. La différence réside dans les fonctions de propagation et de rétropropagation, dont l'algorithme est donné ci-après, car il faut tenir compte de la taille de la fenêtre observée par les neurones des couches intermédiaires de la partie extraction des caractéristiques des réseaux de neurones à convolution tels que le TDNN, réseau à décalage temporel.

```
1. Rétropropagation sur le réseau PMC
Balayage temporel des neurones de la dernière couche du PMC
    Initialisation des erreurs y de chaque neurone
    Calcul et sauvegarde des erreurs y de chaque neurone de sortie
Balayage des couches cachées l et celle d'entrée du PMC, en
décrémentant
    Balayage temporel sur les neurones j de chaque couche
          Balayage temporel sur les neurones k de la couche
      supérieure
                Initialisation des erreurs y de chaque neurone
```

```
                Calcul de la somme totale des poids
            w[l][k][j]*y[l+1].
                Calcul de l'erreur y
          Modification des poids des biais
Balayage des couches cachées l et celle d'entrée du PMC, en
incrémentant
    Balayage temporel sur les neurones j de chaque couche
          Balayage temporel sur les neurones k de la couche
       inférieure
                Calcul du gradient delta
                Modification des poids
2. Adaptation des 2 réseaux : passage du PMC au TDNN (copie de x et
v)
3. Rétropropagation sur le réseau TDNN
Balayage sur les couches cachées du TDNN, en décrémentant
    Balayage temporel des neurones j de chaque couche t
          Balayage sur les neurones k des fenêtres temporelles
          associées, couche supérieure selon t et f
                Initialisation des erreurs y de chaque neurone
                Calcul de la somme totale des w[l][k][j]*y[l+1].
                Calcul de l'erreur y
          Modification des poids des biais
Balayage sur les couches cachées du TDNN, en incrémentant
    Balayage temporel des neurones j de chaque couche t
          Balayage sur les neurones k des fenêtres temporelles
          associées, couche inférieure selon t et f
                Calcul du gradient delta
                Modification des poids
```

Algorithme 6. Algorithme de rétropropagation du TDNN pour un exemple

4.4 Protocole d'expérimentations et tests fonctionnels

Dans cette section, nous décrivons les bases utilisées et le scénario des tests fonctionnels permettant de valider le comportement de la solution à base de réseaux de neurones développés et les tests de performances.

4.4.1 Bases de données de caractères isolées

Le système décrit précédemment a été implémenté et mis en œuvre pour classifier des caractères isolés provenant des bases IRONOFF [Viard-Gaudin, 1999] et UNIPEN [Guyon, 1994]. Nous avons considéré séparément les chiffres, les lettres minuscules et les lettres majuscules.

4.4.1.1 Description de la base IRONOFF

La base de données IRONOFF (IRESTE ON/OFF database) est une base de données duales en-ligne et hors-ligne collectées et distribuées par notre laboratoire. Elle contient un nombre important de caractères isolés, de chiffres et de mots en français et anglais au format UNIPEN. Cette base de données a été créée de telle sorte qu'un point en-ligne puisse être projeté sur sa position correspondante dans l'image scannée, et inversement chaque élément du tracé hors-ligne peut être temporellement indexé.

Cette base a été acquise avec une tablette Wacom UltraPad, la résolution spatiale typique est de l'ordre de 300 points par pouce tandis que la fréquence d'échantillonnage des données est de

l'ordre de 100 points à la seconde. Elle a été collectée auprès d'environ 700 scripteurs différents.

Dans ces expérimentations, nous avons exploité les bases de chiffres, lettres minuscules et majuscules isolés dont la répartition est donnée dans le Tableau 3. Chaque base est découpée en deux sous-bases, l'une étant la base d'apprentissage et l'autre la base de généralisation. Le découpage utilisé, selon la méthode des 2/3-1/3 (cf. chapitre 2), est celui fourni avec la base originale afin d'obtenir des résultats unifiés avec les différents utilisateurs de cette base.

Tableau 3. Taille des sous-bases de caractères IRONOFF

Base	Nombre de Classes	Nombre total d'exemples	Nombre d'exemples en apprentissage	Nombre d'exemples en test
Chiffres	10	4108	3 059	1 510
Minuscules	26	11868	7 952	3 916
Majuscules	26	11879	7 953	3 926

4.4.1.2 Description de la base UNIPEN

La base UNIPEN est la base de référence pour la communauté de la reconnaissance en-ligne latine. Cette base présente l'avantage de contenir un grand nombre de données détaillées dans le Tableau 4 provenant de plus de 2 200 scripteurs ; cependant, la qualité de l'acquisition des tracés, de la segmentation et de l'étiquetage n'est pas constante. Il est important de préciser que nous avons utilisé la base originale, sans aucun nettoyage.

Tableau 4. Taille des sous-bases de caractères UNIPEN

Base	Nombre de Classes	Nombre total d'exemples	Nombre d'exemples en apprentissage	Nombre d'exemples en test
Chiffres	10	15 635	10 423	5 212
Minuscules	26	52 267	34 844	17 423
Majuscules	26	26 605	17 736	8 869

4.4.2 Protocoles d'expérimentation

Pour comparer notre système de reconnaissance de caractères par rapport aux résultats disponibles dans la littérature, nous avons évalué notre système sur chacune des bases UNIPEN et IRONOFF séparément, ces résultats seront présentés à la fin de ce chapitre. Cependant dans la phase d'évaluation des paramètres du système, il est important d'avoir une base de données de taille conséquente, nous avons donc décidé de former une base unique intégrant les exemples de la base UNIPEN à ceux de la base IRONOFF. Cette base a été découpée selon un partage de deux tiers des exemples pour la base d'apprentissage et un tiers pour la base de généralisation.

Nous disposons donc des bases suivantes avec une répartition quasi-équitable.

Tableau 5. Taille des sous-bases de caractères pour l'optimisation du système

Base	Nombre de Classes	Nombre d'exemples en apprentissage	Nombre d'exemples en test
Chiffres	10	13 482	6 722
Minuscules	26	42 796	21 339
Majuscules	26	25 689	12 795
Caractères	62	81 970	40 854

Pour l'analyse des performances du réseau, nous sommes partis de la matrice de confusion à chaque itération en apprentissage et généralisation pour évaluer la qualité de la classification [Egmont-Peterson, 1994]. Elle est obtenue en comparant les données classées (labels associés par le réseau) avec les données de référence (labels des caractères saisis connus). Le nombre de points de contrôle doit être suffisant et, si possible, de même importance dans chaque classe pour que la comparaison ait un sens. Si cela n'est pas possible, il faut prendre un plus grand nombre de points de contrôle pour les classes qui ont le plus d'importance thématique. Les bases d'apprentissage et de test utilisées dans ce projet sont quasi-uniformes.

La matrice de confusion se construit en mettant respectivement sur les lignes et sur les colonnes les données de référence et la classification. Ceci permet de calculer :

- La précision totale : nombre de caractères bien classés divisé par le nombre total de caractères. Elle représente le taux de reconnaissance qui sera utilisé dans les graphiques et les tableaux de ce chapitre.
- L'erreur d'excédents : pourcentage de caractères d'une classe reconnue appartenant en fait à d'autres classes dans les données de référence.
- L'erreur de déficits : pourcentage de caractères d'une classe de référence affectés à d'autres classes par la classification.
- Les erreurs d'affectation rassemblent les erreurs d'excédents et les erreurs de déficits.

Nous avons essentiellement retenu comme outils d'évaluation les informations suivantes :

- Le taux de reconnaissance pour l'ensemble des exemples à chaque itération de la base en phase d'apprentissage et de généralisation.
- L'évolution de la fonction de coût à minimiser, à chaque itération. Si l'apprentissage est correct, cette erreur doit décroître, puis se stabiliser. Elle permet de contrôler la vitesse de convergence du réseau.

La fonction de coût choisie tant pour le PMC que pour la structure TDNN est la distance de Kullback-Leibler, donnée par l'équation :

$$KL^k = -\sum_i d_i^k * \log(d_i^k) + \sum_i d_i^k * \log(s_i^k)$$

avec d_i la sortie désirée et s_i la sortie réelle du neurone de la classe i pour l'exemple k.

La distance de Kullback-Leibler fournit une évaluation de la différence des distributions de probabilités sur la base d'apprentissage entre les labels vrais et les classes reconnues.

- La position moyenne, performance normalisée de reclassement exact, ou bien de déviation des valeurs attendues.
- Et la matrice de confusion.

Dans le cadre de la validation du comportement du TDNN, nous avons arrêté l'algorithme de rétropropagation du gradient à un nombre fixé d'itérations de la base d'apprentissage, ici 300 itérations avec un pas de gradient égal à 0,003. Par la suite, nous avons géré l'arrêt de l'algorithme selon la technique dite « early stopping », arrêt de l'algorithme lorsque la fonction de coût sur la base d'exemples de validation ayant atteint son minimum le dépasse.

4.4.3 Architectures de références

L'architecture de base testée du TDNN est un réseau à 3 couches, la première étant la couche d'entrée, la seconde la couche cachée de la partie extraction, la dernière la couche de sortie du PMC (la couche cachée du TDNN étant la couche d'entrée du PMC).

La couche d'entrée possède 350 neurones (50 points × 7 caractéristiques) balayée par une fenêtre de convolution de 4 neurones décalée selon l'axe temporel de 2 neurones. La couche cachée de la partie extraction possède 16 neurones selon la direction temporelle et 20 neurones selon la direction caractéristique.

La partie PMC est caractérisée par les 16 × 20 unités de la couche d'entrée. Le nombre de neurones de la couche de sortie dépend du type de classification choisie : 10 sorties pour les chiffres, 26 pour les lettres minuscules ou majuscules et 62 si on additionne tous ces caractères.

TDNN à 1 couche cachée pour la partie extraction et classifieur PMC linéaire :

Couche d'entrée du TDNN à poids partagés, l = 0 :
nombre de neurones selon la direction temporelle, window_T[0] = 50
nombre de neurones selon la direction caractéristique, nb_feat[0] = 7
taille du champ perceptuel (fenêtre) par la couche, field_T[0] = 4
délai temporel (nombre de neurones) entre chaque fenêtre, delay[0] = 2

Couche cachée du TDNN, l = 1 (couche d'entrée du classifieur NN, nn_l = 0) :
nombre de neurones selon la direction temporelle, window_T[1] = 16
nombre de neurones selon la direction caractéristique, nb_feat[1] = 20

Couche de sortie MLP, nn_l = 2
nombre de neurones NN_nb_neuron[2] = nombre de classes
(10 pour les chiffres, 26 pour les lettres).

Afin d'évaluer l'intérêt d'une structure de type TDNN, nous l'avons comparée à un perceptron multi-couches (une couche cachée de 100 neurones). Les courbes présentées Figure 56 et Figure 57 montrent les comportements obtenus avec le PMC et le TDNN, et le Tableau 6 résume les performances atteintes.

Les courbes Figure 56 illustrent les performances du PMC sur la base de chiffres, avec dans l'ordre le taux de reconnaissance des caractères en première position, la fonction de coût et la position moyenne. Le trait gras vertical est un indicateur du sur-apprentissage, le problème du "**sur-apprentissag**e" (over fitting) des modèles neuronaux est bien connu. La convergence en apprentissage de la position moyenne vers la position première et la décroissance de la position moyenne en test jusqu'à une valeur plafond montre la limite du meilleur compromis apprentissage/généralisation. A cause de leur grande plasticité (nombre de paramètres libres), les modèles neuronaux peuvent dans certains cas prédire avec une grande précision l'ensemble des données sur lequel l'ajustement a été fait. Cependant, de très fortes erreurs de prédictions peuvent être constatées dès lors que les modèles sont utilisés pour prédire de nouvelles données.

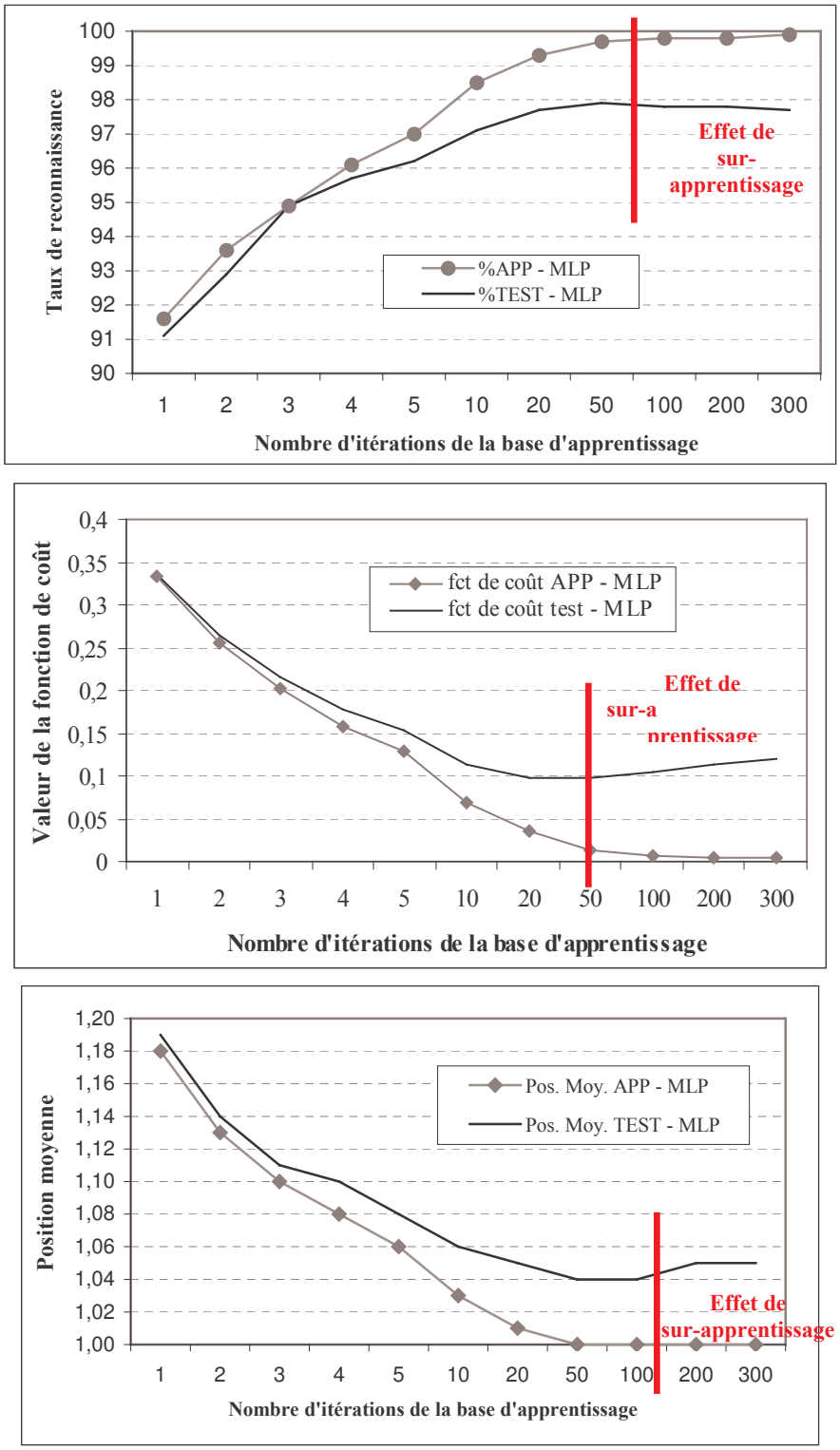
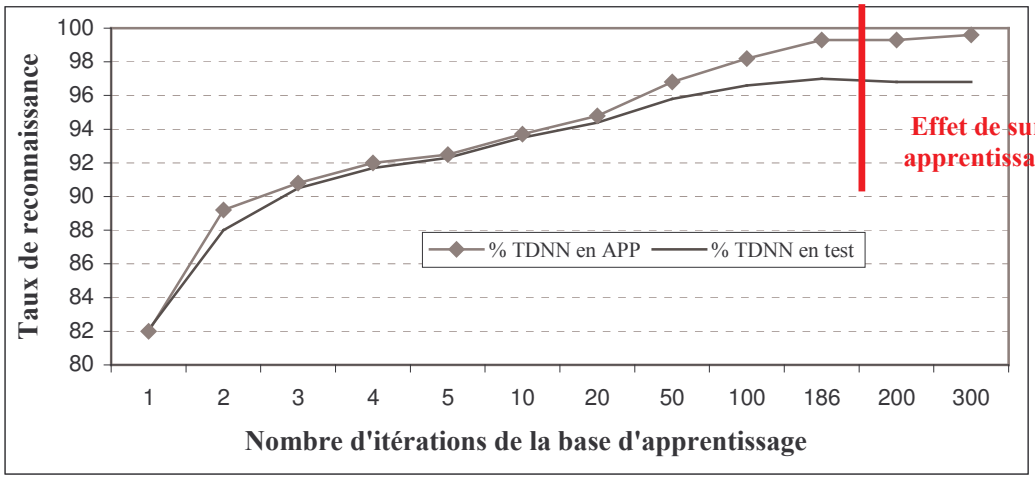
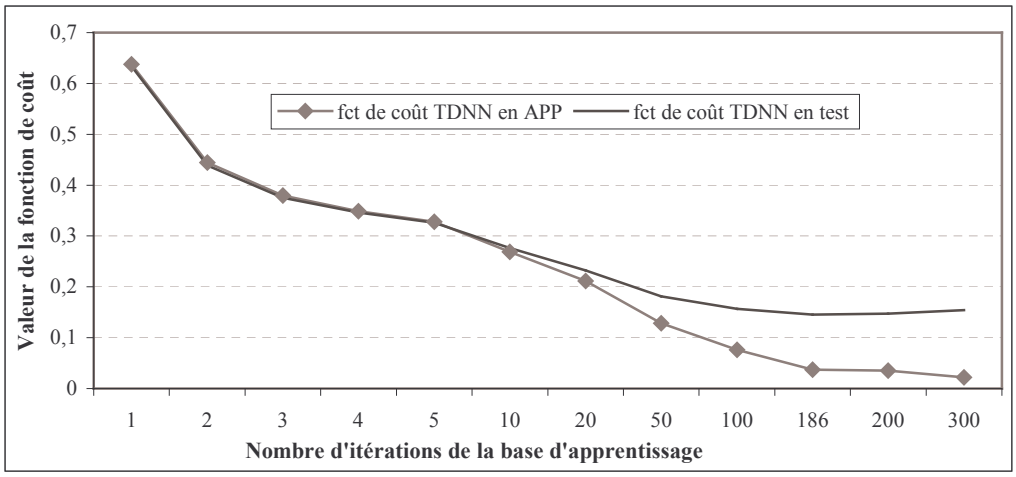


Figure 56. Performances du PMC 1 couche cachée sur la base de chiffres.

La courbes présentées Figure 57 illustrent les résultats obtenus avec le réseau de neurones TDNN dont l'architecture a été décrite ci-dessus. On constate une convergence sensiblement plus lente qu'avec le PMC.





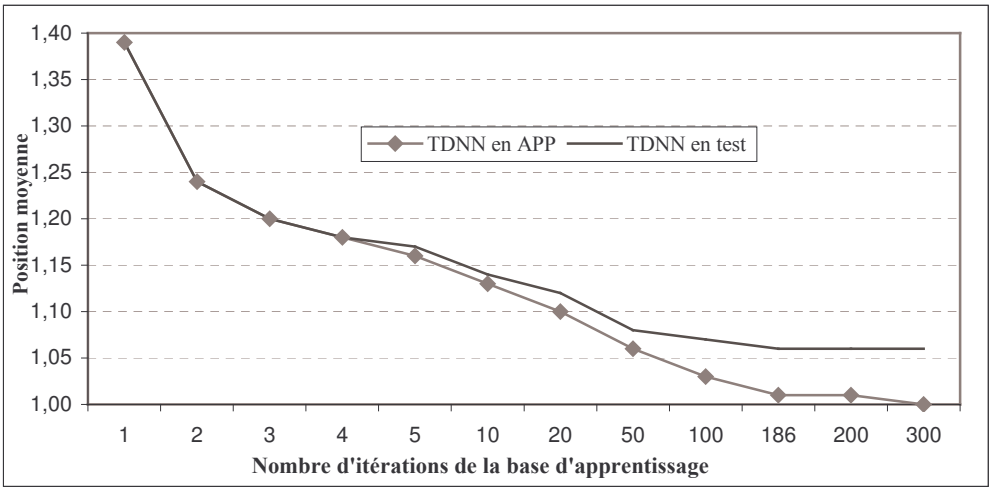


Figure 57. Performances du TDNN, 1 couche cachée, sur la base de chiffres.

Le TDNN converge bien vers le minimum de la fonction de coût (Kullback Leibler) après 186 itérations de la base d'apprentissage en reconnaissance de chiffres. Par ailleurs, il converge bien en apprentissage vers la position moyenne idéale. En généralisation, il obtient une performance normalisée de reclassements exacts sur les chiffres de 1,06.

4.4.4 Résultats de référence

Les résultats suivants donnés au Tableau 6 serviront de références de base pour comparer les performances des différentes architectures du TDNN et ensuite les améliorer. La première partie du Tableau 6 présente les performances d'un perceptron à une couche cachée de 100 neurones utilisant la méthode du gradient stochastique avec un pas fixe de 0,003. La seconde correspond aux premiers résultats obtenus avec l'architecture citée à la section 4.4.3 avec la même configuration : méthode du gradient, pas fixe de 0,003, bases identiques d'apprentissage et de test.

Tableau 6. Comparaison de deux architectures de référence PMC et TDNN

	Perceptron 1 couche cachée (100 neurones)				TDNN 1 couche cachée 4/2/16/20 classifieur linéaire			
Classes	PL	Iter.	Taux %	PM	PL	Iter.	Taux %	PM
Chiffres	**36 110**	35	97,8	1,04	**3 790**	186	97,0	1,06
Majuscules	**37 726**	22	93,1	1,20	**8 926**	85	92,5	1,28
Minuscules	**37 726**	13	91,2	1,25	**8 926**	59	91,0	1,24
62 Caractères	**41 362**	11	77,1	1,78	**20 482**	54	77,5	1,71

Une première comparaison peut être établie entre les deux architectures et leurs résultats présentés ci-dessus. On remarque globalement que les performances obtenues (Taux % = taux de reconnaissance exacte, PM : position moyenne) par le TDNN sont légèrement inférieures à celles du perceptron à 3 couches. La première explication est le nombre de paramètres libres (PL), de 2 à 10 fois inférieur à celui du perceptron. Cependant, si on analyse les positions moyennes (PM) les deux architectures sont sensiblement identiques.

La deuxième remarque est le nombre d'itérations sur la base d'apprentissage nécessaire pour atteindre l'optimum global (Iter.) plus important pour le TDNN. Il sera nécessaire d'analyser le pas de gradient et l'initialisation des poids.

4.5 Paramétrage et Optimisation du TDNN

L'intérêt d'une optimisation « manuelle » réside dans la possibilité d'intégrer dans l'architecture des connaissances préalables sur les données. Ceci peut être effectué par exemple par l'utilisation de connexions partielles plutôt que totales, et/ou de poids partagés. Dans cette section, est décrit l'analyse des paramètres à partir d'une démarche empirique. La démarche a consisté à comparer les performances obtenues pour différentes architectures, et à ne conserver ensuite que celles qui correspondent au meilleur compromis performance/coût architectural.

4.5.1 Définition des paramètres

Les paramètres à optimiser pour l'architecture TDNN sont :
- Le nombre de couches de la partie extraction.
- Leur nombre de neurones caractéristiques (fréquences) pour chaque couche.
- La largeur des fenêtres de spécialisation (champ perceptuel d'un neurone de la couche supérieure).
- Les délais associés aux fenêtres de spécialisation
- Le nombre de couches de la partie classification.
- Et leur nombre de neurones.

Mis à part une analyse approfondie de l'architecture et du comportement d'un TDNN pour la reconnaissance de l'écriture manuscrite non contrainte, l'objectif final du système est une réalisation matérielle embarquable, temps-réel et destinée à ne reproduire que la phase d'utilisation du réseau. Il est souhaitable de développer une architecture neuronale aussi petite que possible. Ceci sous-entend avec un nombre minimum de neurones, de connexions et de bits pour le codage des données dans le réseau. Ainsi par diminution du nombre de calculs nécessaires à la simulation, on peut non seulement écourter le temps de propagation, mais aussi contribuer à réduire le coût matériel.

Avant d'entreprendre la conception d'une solution, il convient de déterminer les caractéristiques que l'on veut optimiser. Les critères d'optimisation les plus courants sont les suivants :

- L'amélioration du pouvoir de généralisation (performances du système neuronal).
- La minimisation de l'architecture (pour réduire la quantité de calculs en phase d'utilisation).
- La prise en compte de caractéristiques destinées à l'implantation matérielle.
- La minimisation du temps d'apprentissage.

Bien que le dernier critère ne concerne que la phase de conception et de mise au point, il reste très important, la diminution du temps de calcul est un facteur à ne pas négliger pour pouvoir valider les résultats par un nombre suffisant d'expérimentations. Le temps d'apprentissage peut se compter en heures voire en jours. En développant l'ensemble des sources en langage C et en optimisant le code de la solution développée, un gain de temps d'un facteur quatre a été obtenu en utilisant la librairie Intel (voir Manuel du programmeur en annexe). L'utilisation de plates-formes logicielles de simulation (SNNS, Matlab) aurait à l'inverse conduit à des temps d'apprentissage beaucoup plus importants, et aurait rendu impossible toutes les expérimentations reportées ici, sans vouloir anticiper sur le passage au niveau mot qui demande une maîtrise complète du cœur du système. Le Tableau 7 ci-dessous donne un aperçu des temps d'apprentissage pour un TDNN à poids partagés (1 couche cachée TDNN, classifieur linéaire) avec un Pentium II-450 et l'utilisation des fonctions optimisées de la bibliothèque Intel.

Tableau 7. Exemples de temps d'apprentissage d'un TDNN (architecture la plus réduite : une couche cachée - classifieur linéaire)

Temps d'apprentissage	**Chiffres** APP : 13482 TEST : 6722	**Majuscules** APP : 25689 TEST : 12795	**Minuscules** APP : 42796 TEST : 21339
1 itération de la base d'apprentissage	~50 secondes	~90 secondes	~110 secondes
300 itérations de la base d'apprentissage (performance en apprentissage > 99%)	3 à 4 heures	5 à 7 heures	8 à 10 heures
Temps moyen pour un bon compromis apprentissage / généralisation	1 heure	1 heure 15	1 heure 30

En complément à la recherche de la topologie optimale du TDNN, cités précédemment, d'autres améliorations destinées à augmenter le pouvoir de généralisation et/ou la rapidité de l'apprentissage ont été analysées et détaillées ci-après :

- Contrainte ou non des poids partagés.
- Influence de l'initialisation des poids.
- Gain d'adaptation.

4.5.2 Accélération de l'apprentissage

Nous avons étudié la sensibilité de la vitesse de convergence vis-à-vis du choix du pas d'adaptation et de l'initialisation des poids lors du lancement de l'apprentissage. Ce sont les deux caractéristiques les plus importantes pour contrôler l'apprentissage.

4.5.2.1 Le pas d'adaptation.

Dans le cas de l'algorithme de rétropropagation du gradient, le calcul du gradient consiste à définir dans un espace contenant autant de dimensions qu'il y a de poids partagés, la direction dans laquelle doit s'effectuer la modification des poids. La valeur du pas de gradient détermine l'amplitude du déplacement dans cette direction : un pas trop grand provoque des oscillations, voire une divergence, un pas trop faible ralentit la convergence vers le minimum de la fonction de coût.

Les courbes présentées Figure 58 illustrent les taux de reconnaissance obtenus sur la base des chiffres (échantillonnés sur 50 points, soit 350 points en entrée du réseau) en s'intéressant à la sensibilité au pas du gradient. Le réseau testé est un TDNN avec une couche cachée pour la partie extraction et une de 100 neurones pour la partie classifieur. Les paramètres de la partie extraction sont : 20 caractéristiques, fenêtre de 4 et délai de 2 neurones.

La Figure 58 illustre bien les comportements attendus. Un pas de 0.1 ne permet pas au réseau de converger de manière satisfaisante, cette valeur est manifestement trop grande. Inversement, la valeur de 0.001 tarde à faire converger le réseau, tandis que les valeurs comprises entre 0.003 et 0.01 ont un comportement assez proche.

Nous avons cherché à accélérer la convergence de l'apprentissage en implémentant la méthode Levenberg-Marquardt [Amari, 1998], celle-ci n'a apporté aucun gain significatif par rapport au pas obtenu de façon empirique (pas de 0,003).

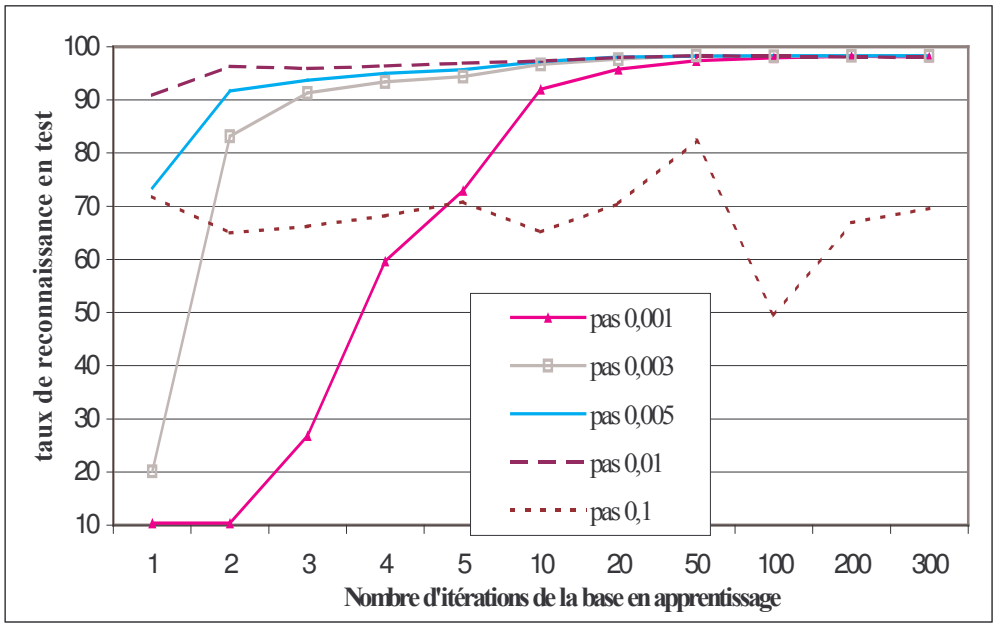


Figure 58. Taux de reconnaissance des chiffres en généralisation – Analyse du pas.

4.5.2.2 Initialisation des poids.

Au lancement de l'apprentissage, les valeurs initiales des poids doivent être différentes de zéro pour que l'algorithme de rétropropagation puisse fonctionner. D'autre part l'utilisation de valeurs élevées peut provoquer un phénomène de saturation prématurée qui contribue à diminuer la vitesse de convergence de l'apprentissage. Ce phénomène est fonction de l'amplitude des poids, de la pente de la tangente hyperbolique et du nombre de neurones dans chaque couche.

Afin de se situer dans la zone linéaire de la tangente hyperbolique, l'initialisation est faite aléatoirement dans l'intervalle [-1,1] de la manière suivante :

$$\text{poids} = \frac{2*(\frac{\text{alea}\%10000}{10000})-1}{\alpha*\sqrt{fan-in}}$$

avec aléa, un entier généré aléatoirement dans l'intervalle [0, 32767].

Le fan-in correspond aux nombres d'entrées du neurone. Pour la partie extraction, il correspond à la taille de la fenêtre de spécialisation, pour la partie classifieur à la taille de la couche précédente. Le paramètre α permet de contrôler l'effet de saturation de la sigmoïde, il sera fixé à 10 dans le Tableau 8 et variera de 001 à 100 sur la Figure 59.

Afin de connaître la précision des résultats donnés et l'influence de l'initialisation des poids, nous avons testé dans un premier temps la même architecture de TDNN avec des initialisation différentes. L'initialisation des poids étant aléatoire, elle peut influencer le résultat, donc entacher celui-ci d'une certaine imprécision. Le réseau testé est un TDNN avec une couche cachée pour la partie extraction et une de 100 neurones pour la partie classifieur. Les paramètres de la partie extraction sont : 20 caractéristiques, fenêtre de 4 et délai de 2 neurones.

Tableau 8. Tableau des performances d'un TDNN avec différentes initialisations

	Taux de reconnaissance sur les chiffres α=10				Meilleure position moyenne en test
initialisation aléatoire	1ère itération		meilleure itération		
N° des différents tests	%APP	%TEST	n° itération	%TEST	
t1	28,07	27,33	42	98,19	1,04
t2	19,54	19,13	38	98,20	1,04
t3	31,64	31,06	48	98,27	1,04
t4	18,49	18,79	49	98,23	1,04
t5	31,92	31,60	43	98,26	1,04
t6	26,78	26,84	46	98,21	1,03
t7	29,53	29,66	41	98,20	1,04
t8	29,28	28,59	36	98,08	1,04
t9	29,42	28,79	43	98,30	1,03
t10	31,23	30,66	51	98,27	1,03
Moyenne	27.59	27.25		**98,22**	**1,04**
Ecart-type	4.79	4.62		**0,06**	**0,00**

Comme on peut le constater en analysant les performances du réseau aux premières itérations de la base d'apprentissage, elles sont extrêmement sensibles à l'initialisation (valeurs initiales aléatoires données aux connexions). Pour cette raison, les simulateurs de réseaux de neurones auront des performances différentes d'un apprentissage à l'autre. On ne peut alors jamais être sûr d'avoir obtenu la meilleure prédiction possible.

Toutefois, au vu du Tableau 8, on peut constater que l'on obtient une bonne stabilité des résultats au terme de l'apprentissage, manifestement le réseau doit converger vers le même minimum local. On peut également estimer que les résultats obtenus seront donnés avec une incertitude inférieure au $1/10^{e}$ concernant les taux de reconnaissance affichés.

Ainsi l'architecture du TDNN testé pour cette analyse obtient des performances supérieures (98,2 % de caractères bien reconnus) à celles de références du perceptron (97,8%) et de la première architecture donnée du TDNN (97,0%, tests fonctionnels).

L'initialisation aléatoire des poids joue un rôle important dans la sensibilité du réseau, précédemment le paramètre α était pris égal à 10, il joue lui aussi un rôle prépondérant car il contrôle l'effet de saturation comme on peut le voir ci-dessous.

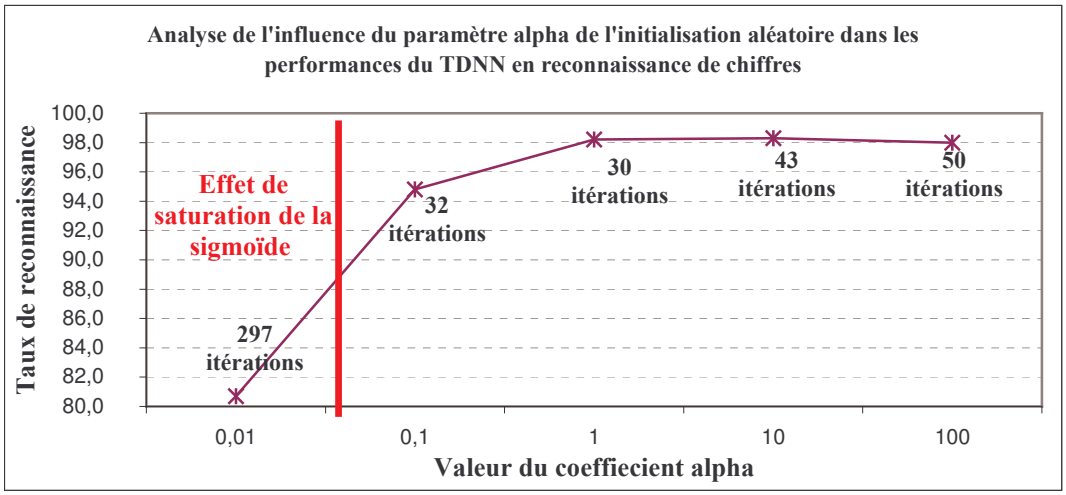


Figure 59. Influence de l'initialisation des poids – effet de saturation de la tangente hyperbolique.

4.5.3 Optimisation de l'architecture du TDNN

4.5.3.1 Analyse du nombre de couches

Mis à part les couches d'entrée et de sortie, il est possible de jouer sur le nombre de couches cachées du réseau. Sans couche cachée, le réseau n'offre que de faibles possibilités d'adaptation. Avec une couche cachée, il est capable, avec un nombre suffisant de neurones, d'approximer toute fonction continue (Hornik,1991). Une seconde couche cachée prend en compte les discontinuités éventuelles.

Tableau 9. Performances de diverses configurations du TDNN – analyse du nombre de couches

TDNN	1 couche cachée (cc)				1 cc + 1 cc au classifieur				2 couches cachées				2 cc + 1 cc au classifieur		3 couches cachées	
TDNN	16/20/20/2 (A/B/C/D)		7/20/20/5		16/20/20/2 nn : 100		7/20/20/5 nn : 100		24/20/4/2 11/20/4/2		21/20/10/2 17/20/5/1		21/20/10/2 17/20/5/1 nn : 100		21/20/10/2 17/20/5/1 13/20/5/1	
	iter	TEST	iter	TEST	iter	TEST	iter	TEST	iter	TEST	iter	TEST	iter	TEST	iter	TEST
digit	400	97,7	79	98	55	98,2	**45**	**98,4**	300	97,4	94	97,7	60	97,8	156	97,7
Maj	54	93,7	47	93	40	94,3	**74**	**94,5**	102	93,1	74	93,5	49	93,2	108	93,4
min	53	91,4	52	91	47	92,7	**49**	**92,7**	73	91,4	55	91,6	37	91,7	89	92,0

Notations du Tableau 9

A = Window_T : nombre de neurones selon la direction temporelle
B = Nb_feat : nombre de neurones de type caractéristiques
C = Field_T : taille de la fenêtre de spécialisation
D = Delay : Délai temporel entre les fenêtres de spécialisation.
nn : nombre de neurones sur la couche cachée (cc) du classifieur
Maj, min : majuscules, minuscules

La topologie constituée d'une couche cachée pour la partie extraction (TDNN) et une couche cachée pour la partie classifieur (PMC) repérée en gras dans le Tableau 9 correspondra à l'architecture finale retenue.

Parallèlement à la recherche du nombre idéal de couches cachées, il est important de bien paramétrer la taille de celles-ci. Un nombre trop grand de cellules peut conduire à une mauvaise généralisation (variance élevée de l'estimateur), un nombre trop faible amène de grandes difficultés dans la phase d'apprentissage, avec trop peu de capacité d'apprentissage (biais important de l'estimateur).

4.5.3.2 Analyse du nombre de neurones de type caractéristiques

Chaque neurone supplémentaire permet de prendre en compte des configurations spécifiques des formes en entrée. Un nombre plus important permet donc de mieux coller aux données présentées mais augmente la sensibilité au bruit du réseau. Pour fixer ce nombre, il n'existe pas de règle générale mais quelques règles empiriques. Celles-ci préconisent de choisir cette taille :

- soit égale à celle de la couche d'entrée (Wierenga et Kluytmans, 1994),
- soit égale à 75% de celle-ci (Venugopal et Baets, 1994),

– soit égale à la racine carrée du produit du nombre de neurones dans la couche d'entrée et de sortie (Shepard, 1990).

Le Tableau 10 et la Figure 60 illustrent les règles citées ci-dessus et montrent l'influence du nombre de caractéristiques sur le nombre de degré de liberté du TDNN (1 couche cachée avec fenêtre de 20 et délai de 5 neurones + classifieur linéaire) et l'impact importance sur ses performances en reconnaissance de chiffres.

Le nombre de degré de libertés se calcule de la manière suivante :

$$Pl_{total} = PL(\text{partie extraction}) + PL(\text{partie classification})$$

$$PL(\text{partie extraction}) = \sum_{l=2}^{nb\_couches\_extraction} (fenetre(l-1)*caractéristiques(l-1)+1)*caractéristiques(l))$$

$$PL(\text{partie classification}) = \sum_{l=2}^{nb\_couches\_classification} (nb\_neurone\_t(l-1)+1)*nb\_neurone\_t(l))$$

avec pour la couche d'entrée de la partie classification :

$$nb\_neurone\_t(1) = caractéristiques(nb\_couches\_extraction)*nb\_neurone\_t(nb\_couches\_extraction)$$

Par exemple pour un TDNN à 1 couche cachée avec 20 caractéristiques, une fenêtre de 20 neurones et un délai de 5, le calcul est le suivant : [20*7 +1]*20+(7*20+1)*10 = 4 230.

Tableau 10. Tableau d'analyse de l'impact du nombre (nb.) de caractéristiques (car.)

Nb. de car.	nb_iter	%app	%test	Position moyenne	Paramètres libres	Remarque
5	92	96,5	95,4	1,09	1 065	Biais important
10	68	98,3	96,7	1,06	2120	Biais important
20	78	99,2	97,8	1,04	4 230	Bon compromis (temps)
38	90	99,5	97,8	1,04	8 028	Règle du nbre total d'entrées
50	76	99,3	**98,0**	1,04	10 560	Règle du 75% des entrées
60	68	99,3	97,9	1,04	12 670	Règle de la racine carrée
100	98	99,5	97,9	1,04	21 110	Sur-apprentissage
200	103	99,5	97,5	1,05	42 210	Sur-apprentissage
500	120	99,5	97,2	1,05	105 510	Sur-apprentissage

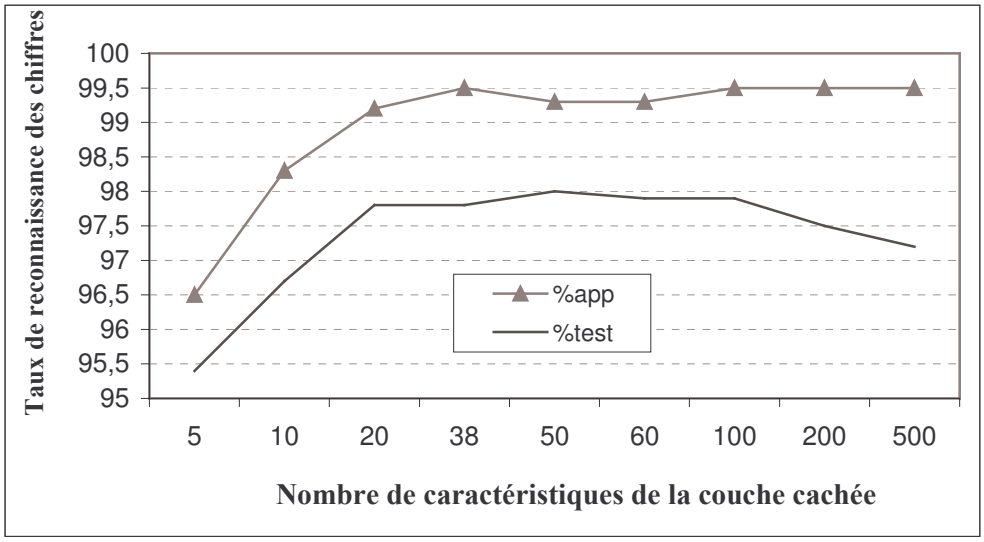


Figure 60. Illustration de l'impact du nombre de caractéristiques de la couche cachée

L'écart entre la courbe d'apprentissage et la courbe du taux de reconnaissance des chiffres en généralisation est un indice pour évaluer l'effet de minima locaux ou de sur-apprentissage. Un écart trop important traduit un sur-apprentissage dû à un nombre de paramètres libres trop important. Inversement un écart trop faible traduit l'impossibilité même en apprentissage de s'approcher des exemples à apprendre : soit pas assez de degrés de liberté (informations sur la trajectoires insuffisantes). Un nombre de caractéristiques égal à 20 donne un bon compromis nombre de paramètres libres /performances, nous opterons donc pour cette solution.

4.5.3.3 Analyse des paramètres de convolution : fenêtre et délai

Les analyses des paramètres présentés précédemment ont montré l'intérêt d'utiliser un TDNN par rapport au perceptron classique : meilleures performances, temps d'apprentissage plus court (nombre de paramètres libres nettement moins important). Ceci est la conséquence de l'architecture de la partie extraction du TDNN qui correspondent aux couches basses de celui-ci. La partie extraction implémente les convolutions successives permettant de transformer progressivement les caractéristiques d'entrées en grandeurs de plus en plus significatives vis-à-vis du problème.

Les différentes architectures de réseaux de neurones artificiels (RNA) ne prennent pas en compte le temps de la même manière. Ainsi, les réseaux non récurrents doivent utiliser une fenêtre temporelle de longueur fixe en entrée pour chaque variable du problème. La fenêtre temporelle garde un nombre plus ou moins important mais fixé a priori de valeurs successives de la variable. Pour les réseaux non récurrents, plusieurs problèmes se posent. D'abord, il est très difficile de choisir la taille de la fenêtre temporelle : selon la position dans la série, la fenêtre utilisée peut s'avérer trop longue ou trop courte. Ensuite, pour des problèmes de reconnaissance de séquences, la position de la séquence d'intérêt dans la fenêtre aura une importance disproportionnée. Le Time Delay Neural Networks (TDNN), grâce au partage des poids entre les différentes positions de la fenêtre, arrive à mieux traiter les problèmes de reconnaissance de séquences mais la taille de la fenêtre d'entrée reste fixe.

Comme le montre la Figure 61, les paramètres de convolution, fenêtre de spécialisation et délai temporel, ont un rôle important dans les performances du réseau à généraliser n'importe

quel caractère saisi. Pour analyser l'influence des paramètres de convolution, nous avons testé un TDNN à une seule couche cachée dont le nombre de caractéristiques est fixé à 20 (classifieur linéaire) et nous avons fait varier la largeur de la fenêtre de spécialisation et le pas de décalage de cette fenêtre.

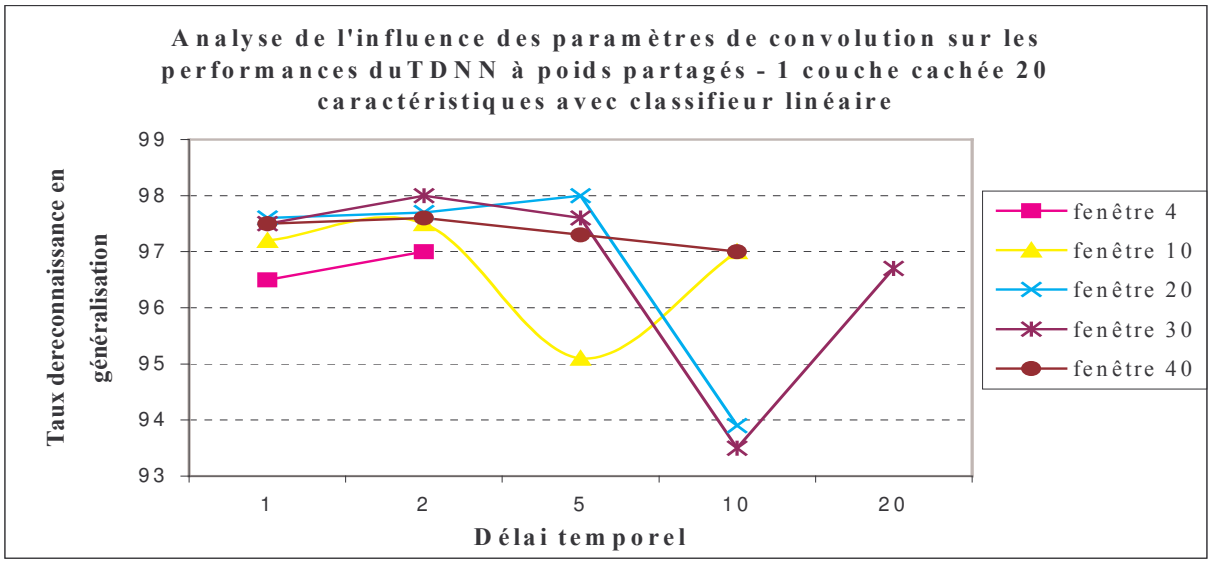


Figure 61. Illustration de l'influence des paramètres de convolution dans le cadre de la reconnaissance des chiffres

Par dimensionnement de la taille de la fenêtre et du délai pour chaque couche, on influence la capacité de mémorisation du réseau et le type de caractéristiques qui peut être extrait. Plus on augmente la taille de la fenêtre, plus les couches basses du réseau auront une vison globale et seront à même de détecter des formes complexes à l'intérieur du caractère. Inversement, avec des fenêtres de petites tailles, on s'intéressera uniquement à faire ressortir des caractéristiques très locales. Le délai quant à lui, va régler le sous-échantillonnage réalisé entre les couches. Un délai unitaire conserve la même résolution d'analyse au niveau supérieur, tandis qu'un délai élevé réduit cette résolution et donc simplifie l'architecture.

En se reportant à la Figure 61 une fenêtre de 20 points (par rapport aux 50 points représentant la totalité du caractère) associée à un délai de 5 apparaît réaliser le meilleur compromis performance/architecture.

4.5.3.4 Influence de la contrainte des poids partagés

Les tests et optimisations présentées ont été menés jusque là uniquement à partir d'un TDNN à poids partagés. Ce principe de poids partagés confère au TDNN une aptitude à gérer les invariances par translations dans le temps. Le nombre de poids partagés représente le nombre de paramètres libres du réseau, il dépend de la taille de la fenêtre et du délai de chaque couche de la partie extraction du TDNN (illustré Figure 62). Si l'on relâche cette contrainte de poids partagés pour ne conserver que la faculté d'analyse locale, on va augmenter le nombre de paramètres libres.

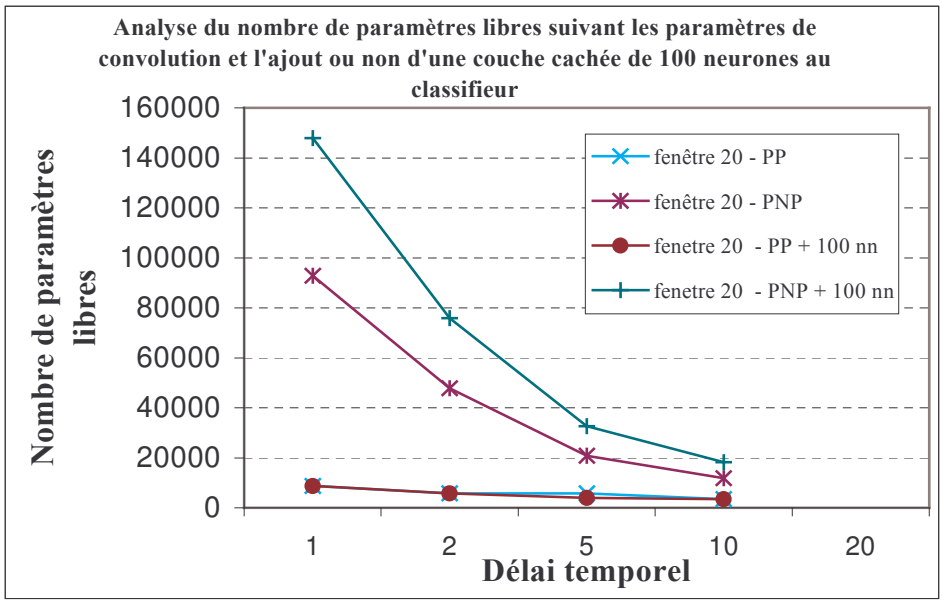


Figure 62. Illustration du nombre de paramètres libres en fonction des paramètres de convolution

Pour comparer, la différence entre un TDNN à poids partagés (PP) et un TDNN à poids non partagés (PNP), il est important de connaître le nombre de paramètres libres. Illustré Figure 62 à topologie identique (largeur de la fenêtre égale 20) un TDNN à poids partagés possède beaucoup moins de degrés de liberté qu'un TDNN à poids non partagés. Néanmoins, dans la plupart des topologies, il obtiendra de meilleures performances de reconnaissance et donc un compromis performance/architecture d'autant meilleur.

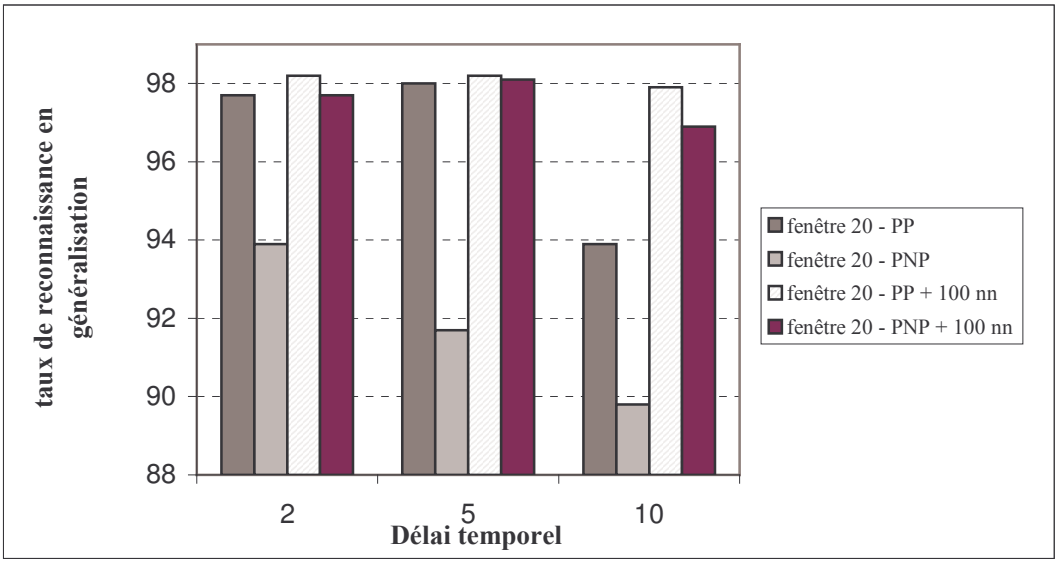


Figure 63. Analyse de la contrainte des poids partagés et impact du classifieur sur les performances d'un TDNN 1 couche cachée (20 car. - classifieur avec ou sans couche cachée (100 neurones) en reconnaissance de chiffres en généralisation

4.6 Définition de la topologie optimale

Les expériences menées ont abouti à une définition de la topologie d'un TDNN à poids partagés qui semblerait être la plus performante (vis-à-vis de celles testées). Son architecture est donnée dans l'encadré ci-après.

TDNN à 1 couche cachée pour le TDNN et 1 couche cachée pour le classifieur MLP :

Couche d'entrée du TDNN à poids partagés, l = 0 :

nombre de neurones selon la direction temporelle, window_T[0] = 50

nombre de neurones selon la direction caractéristique, nb_feat[0] = 7

taille du champ perceptuel (fenêtre) par la couche, field_T[0] = 20

délai temporel (nombre de neurones) entre chaque fenêtre, delay[0] = 5

Couche cachée du TDNN, l = 1 (couche d'entrée du classifieur NN, nn_l = 0) :

nombre de neurones selon la direction temporelle, window_T[1] = 7

nombre de neurones selon la direction caractéristique, nb_feat[1] = 20

Couche cachée du classifieur type MLP, nn_l = 1

nombre de neurones NN_nb_neuron[1] = 100

Couche de sortie MLP, nn_l = 2

nombre de neurones NN_nb_neuron[2] = nombre de classes (10 pour les chiffres, 26 pour les lettres).

Au Tableau 6 nous avons donné deux architectures, la première est basée sur TDNN et la seconde est un perceptron multi-couches. Après optimisation de la topologie du réseau (soit un gain de 50% sur la taille des poids), on obtient les performances suivantes donné au Tableau 11.

Tableau 11. Comparaison TDNN/PMC sur les 3 bases : chiffres, minuscules et majuscules

Base	Classes	Exemples APP	Exemples TEST	Nombres de paramètres libres	Nombre d'itérations TDNN	**% Taux TDNN**
Chiffres	10	13 482	6 722	17 930	45	**98.4**
Minuscules	26	42 796	21 339	18 546	49	**92.7**
Majuscules	26	25 689	12 795	18 546	74	**94.5**
Caractères	62	81 970	40 854	22 882	105	**86,3**

Les résultats obtenus avec l'architecture du TDNN présenté sont supérieurs à celle d'une architecture classique telle que le perceptron. C'est le fruit de la contrainte des poids partagés qui confère des informations plus pertinentes à analyser par le réseau. Par ailleurs la contrainte des poids partagés présente aussi l'avantage de demander à l'architecture d'un système de reconnaissance d'écriture manuscrite beaucoup moins de place mémoire pour stocker les poids du réseau. Ceci est très intéressant pour des applications comme par exemple les ordinateurs de poche, applications en pleine croissance à l'heure actuelle.

4.7 Résultats sur les bases de référence UNIPEN et IRONOFF

Les résultats donnés dans le Tableau 11 correspondaient à l'évaluation du système décrit au paragraphe précédent sur des bases d'apprentissage et de tests résultant de la fusion des bases IRONOFF et UNIPEN, cf. section 4.4.2. Dans le Tableau 12, l'architecture optimale du TDNN est comparée avec une architecture PMC sur les bases séparées. Par rapport aux résultats présentés dans le Tableau 6, on note cette fois-ci que le TDNN surpasse les performances du perceptron tout en ayant une architecture moins complexe : gain d'un facteur 2 environ entre le nombre de paramètres libres du TDNN et du PMC.

Tableau 12. Comparaison TDNN/PMC sur les bases UNIPEN et IRONOFF : chiffres, minuscules et majuscules

	Exemples APP	Exemples TEST	**TDNN %**	**PMC %**
UNIPEN				
10 chiffres	10 423	5 212	**97,9**	**97,5**
26 minuscules	34 844	17 423	**92,8**	**92,0**
26 majuscules	17 736	8 869	**93,5**	**92,8**
IRONOFF				
10 chiffres	3 059	1 510	**98,4**	**98,2**
26 minuscules	7 952	3 916	**90,7**	**90,2**
26 majuscules	7 953	3 926	**94,2**	**93,6**

Les Tableau 13 et Tableau 14 comparent les résultats disponibles dans la littérature sur chacune des bases UNIPEN et IRONOFF avec les performances du TDNN optimisé.

Tableau 13. Performances connues sur la base IRONOFF

Auteurs	Systèmes	Performances	Remarques
Chiffres			
Notre système	**TDNN**	**98,4**	**PL : 17 930**
[Ragot, 2003]	ResifCar	94,3	PL : 6 084
[Ragot, 2003]	Mélidis	95,8	PL : 12 416
[Ahmad,2004]	SVM	98,8	PL : 1 054 900
Minuscules			

Notre système	**TDNN**	**90,7**	**PL : 19 546**
[Ahmad,2004]	SVM	92,4	PL : 5 493 600
Majuscules			
Notre système	**TDNN**	**94,2**	**PL : 19 546**
[Ahmad,2004]	SVM	95,4	PL : 3 512 350

Tableau 14. Performances connues sur la base UNIPEN

Auteurs	Systèmes	Performances	Remarques
Chiffres			
Notre système	**TDNN**	**97,9**	**PL : 17 930**
[Ratzlaff, 2003]	Sn-tuple	98,3	
[Marukatat, 2004]	MMCHN - 50	96,4	
[Hu, 2000]	MMC	96,4	
[Oudot, 2003]	k-ppv	98,8	
[Ahmad,2004]	SVM	98,3	PL : 452 100
Minuscules			
Notre système	**TDNN**	**92,8**	**PL : 19 546**
[Ragot, 2003]	Mélidis	89,7	PL : 82 506 (mode multi-scripteurs)
[Ratzlaff, 2003]	Sn-tuple	90,5	
[Marukatat, 2004]	MMCHN - 50	87,2	
[Oudot, 2003]	k-ppv	96,3	
[Ahmad,2004]	SVM	94,0	PL : 2 354 400
Majuscules			
Notre système	**TDNN**	**93,5**	**PL : 19 546**
[Ratzlaff, 2003]	Sn-tuple	94,4	
[Marukatat, 2004]	MMCHN - 50	91,8	
[Hu, 2000]	MMC	93,6	Découpage de la base différent (1/2-1/2)
[Oudot, 2003]	k-ppv	96,6	
[Ahmad,2004]	SVM	94,8	PL : 1 505 250

4.8 Conclusion

A partir des résultats présentés dans ce chapitre, nous pouvons conclure que notre système de reconnaissance de caractères isolés a des performances très correctes par rapport aux systèmes récents dans la littérature. En intégrant le compromis performances/mémoire, propre à l'application visée, la solution TDNN permet de se positionner à un très bon niveau comparativement à des solutions basées sur des machines à vastes marges [Ahmad, 2004] trop gourmandes en ressources ou par rapport à des architectures réduites comme le Médilis [Ragot, 2003] ayant la même finalité.

Dans ce chapitre nous avons détaillé le système de reconnaissance de caractères manuscrits isolés en-ligne dans un cadre omniscripteur basé sur une architecture TDNN. Nous avons analysé chacun des paramètres influençant le pouvoir de généralisation de ce réseau afin de l'optimiser. Toutefois seules les caractéristiques dynamiques ont été prises en compte, dans le chapitre suivant nous allons proposer une solution pour intégrer des caractéristiques statiques.

5 COMBINAISON TDNN/SDNN : EN-LIGNE/HORS-LIGNE

Dans ce chapitre, nous allons présenter un système de reconnaissance de caractères isolés hors-ligne que nous avons implémenté à partir d'une architecture de type SDNN, réseau de neurones artificiels à décalage spatial (SDNN : Space Displacement Neural Network en anglais). Même si le domaine de l'écriture statique n'est pas au centre de nos travaux, nous avons cherché à améliorer le système présenté au chapitre précédent en intégrant la représentation spatiale de l'écriture au signal dynamique, cette dernière étant initiale dans le processus de lecture de l'œil humain. Nous avons souligné au chapitre 1 (cf. paragraphe 1.3.3.3) différents travaux sur la modélisation de l'information picturale de l'écriture du signal qui améliorent les performances de reconnaissance des systèmes basés uniquement sur une représentation séquentielle. Les principales solutions dans la littérature sont une combinaison des deux informations en entrée d'un même classifieur : ajout de caractéristiques globales hors-ligne, ajout d'un voisinage local de type image en chaque point, ou encore la construction d'une image 2D où chaque pixel est en fait l'addition d'un niveau de gris et d'informations locales en-ligne.

La solution proposée ici est de combiner deux réseaux : un TDNN et un SDNN, dual du réseau temporel pour le traitement d'une image d'un caractère. L'étude de la combinaison de ces deux réseaux est l'objet du chapitre suivant mais avant de pouvoir les coupler, il est nécessaire de construire une architecture SDNN valide et compréhensible et d'analyser l'apport de ces deux réseaux.

5.1 SDNN : Reconnaisseur de caractères isolés hors-ligne

Avec le TDNN, la nature temporelle des données est exploitée par le système de reconnaissance, cela permet souvent de lever des ambiguïtés et d'identifier plus facilement certains caractères. A l'inverse, certains ordonnancements temporels sont perturbateurs, en particulier des signes diacritiques ou des retouches faites sur un tracé. Dans ce cas, la représentation picturale est plus stable, c'est donc ce type de signal que se propose d'exploiter le SDNN. Le SDNN est un réseau de neurones à convolution qui n'exploite plus la notion temporelle mais s'intéresse aux positionnements spatiaux des données. Le SDNN est une généralisation du TDNN à une topologie 2D.

5.1.1 Architecture SDNN

La Figure 64 illustre la structure typique de ce réseau à convolution avec des couches cachées et leurs fenêtres de convolution associées à 3 dimensions (dimensions spatiales horizontale et verticale, dimension des caractéristiques). Les entrées du SDNN correspondent à la matrice 2D des pixels des niveaux de gris des images centrées de caractères, de taille fixe égale à 28x28, les niveaux de gris étant normalisés entre –1 et 1 pour s'adapter à la fonction d'activation des neurones.

Les algorithmes d'entraînement et de reconnaissance du TDNN sont généralisables dans le cas du SDNN, nous ne les détaillerons pas. Pour les pas de descente du gradient, nous avons utilisé l'algorithme de Levenberg-Marquardt.

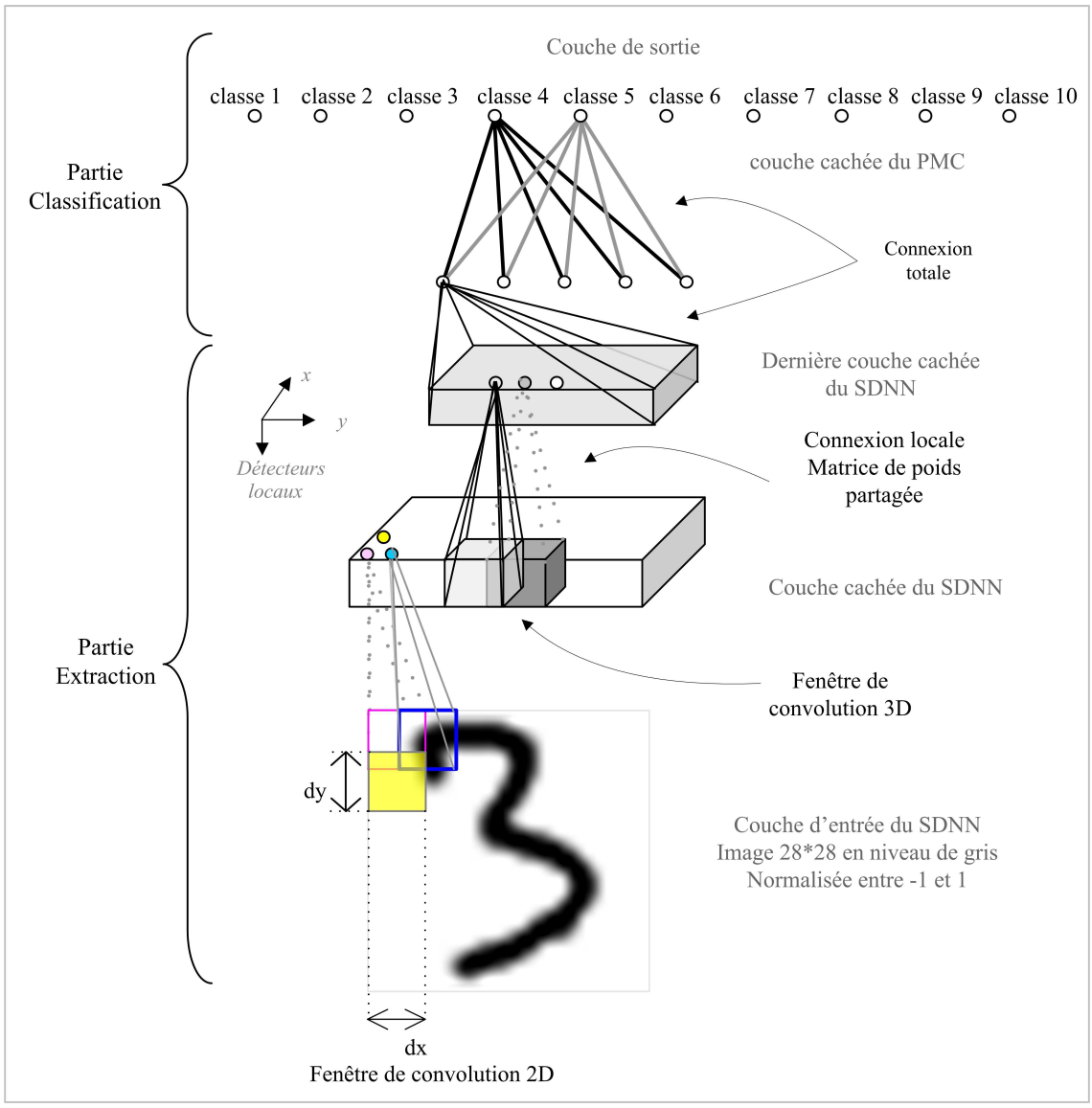


Figure 64. Structure d'un SDNN

5.1.2 Expérimentations

5.1.2.1 Base de données

Le benchmark de référence utilisé pour tester et paramétrer le SDNN est la base de chiffres MNIST qui est découpée en deux sous-bases : une base d'apprentissage de 60 000 exemples de chiffres et une base de test de 10 exemples provenant d'environ 250 scripteurs. Cette base est une version modifiée de la base de chiffres de l'institut NIST[14] (National Institut of Standard and Technology), elle est gratuite et disponible sur la toile[15]. Quelques exemples d'images de la base MNIST sont affichés à la Figure 65.

[14] adresse du site officiel de la base OCR NIST : http://www.nist.gov/srd/optical.htm
[15] adresse du site officiel de la base MNIST : http://yann.lecun.com/exdb/mnist/index.html

Figure 65. Exemples de chiffres de 0 à 9 provenant de la base MNIST

Les chiffres sont déjà prétraités : normalisés en taille et centrés dans une image de taille fixe, 28x28 pixels. Les images originales en noir et blanc provenant du package NIST ont été normalisées en taille pour être contenu dans une boîte englobante de 20x20 pixels en préservant le ratio hauteur/largeur du chiffre. Les images ont ensuite été filtrées afin d'obtenir une image en niveau de gris.

Comme le montre le Tableau 15, la répartition des classes de chiffres est quasi-uniforme.

Tableau 15. Répartition du nombre d' exemples de chiffres dans les bases MNIST (APP : base d'entraînement, TEST : base de généralisation)

Label	0	1	2	3	4	5	6	7	8	9
APP	5923	6742	5958	6131	5842	5421	5918	6265	5851	5949
TEST	980	1135	1032	1010	982	982	958	1028	974	1009

5.1.2.2 Architecture optimisée

Les méta-paramètres à fixer pour ce réseau concernent la taille de la fenêtre de convolution, le décalage spatial, le nombre de caractéristiques locales, le nombre de couches cachées pour la partie extraction et pour la partie classification. Ceux-ci ont été déterminés expérimentalement selon le même processus que le TDNN sur la base MNIST, le meilleur compromis a été obtenu avec l'architecture suivante :

- 2 couches cachées pour la partie extraction, chaque couche ayant une de fenêtre de convolution de taille 6×6 neurones avec un pas de décalage horizontal et vertical identique égal à 2, et 20 caractéristiques
- et un classifieur linéaire avec en sortie le nombre de classes correspondant à la forme à reconnaître.

5.1.2.3 Résultats

Nous avons tout d'abord comparé les résultats obtenus, cf. première partie du Tableau 16, avec l'architecture SDNN proposée à ceux d'un perceptron à une couche cachée de 100 puis 200 neurones et les différents réseaux PMC issus de l'article de Yann LeCun [Lecun, 1998a] dont la base MNIST n'a pas été modifiée. On remarque que la notion de poids partagés confère au réseau un taux de performances supérieur à celui d'un perceptron multi-couches appliqué sur des images avec une réduction considérable du nombre de paramètres libres (minimum de 50%).

Nous avons ensuite analysé l'impact de la représentation d'entrée par rapport à un perceptron travaillant non plus sur l'image mais sur des caractéristiques hors-ligne (précisément 172 dans le cadre de la thèse de Tay [Tay, 2002]). Même si le SDNN traite des données brutes (images)

de taille conséquente, 28×28=784 entrées, celui-ci offre une réduction de moitié du nombre de paramètres libres pour des performances voisines, 98,5% de reconnaissance pour le SDNN contre 98,6% dans le cas du perceptron proposé dans [Tay, 2002].

La deuxième partie du Tableau 16 référence des architectures gourmandes en mémoire telles que LeNet5, architecture SDNN complexe, ou encore les SVM, le SDNN proposé a des performances honorables comparées au nombre de paramètres libres.

Tableau 16. Taux de reconnaissance sur la base de généralisation IRONOFF

Réseau	Nombre de poids	Taux (%) de reconnaissance en 1ère position	
		Base d'apprentissage	Base de test
SDNN proposé (pixels)	**18 750**	**99,9**	**98,5**
PMC, classifieur linéaire (pixels)	7 850	93,5	92,4
PMC, 1 couche cachée (cc) de 100 neurones (pixels)	79 510	97,7	94,9
PMC, 1 couche cachée de 200 neurones (pixels)	159 010	99,4	98,2
PMC, 2 cc de 300 neurones (pixels, MSE) [LeCun, 1998]	328 810	-	95,30
PMC, 3 cc (500/500/150) (pixels) [LeCun, 1998]	719 660	-	97,05
PMC, 1 cc de 200 neurones, entrées : **172 Caractéristiques hors-ligne** [Tay, 2002]	36 610	99,2	98,6
SVM (Reduced Set, deg 5 polynomial, pixels) [LeCun, 1998]	-	-	99,00
k-ppv, (Euclidean, pixels) [LeCun, 1998]	-	-	95
SDNN LeNet5 (pixels) [LeCun, 1998]	60 000	-	99,05

De cette analyse, nous avons conclu qu'un SDNN était une approche intéressante en terme de résultats et coût mémoire pour apporter de l'information picturale à notre architecture initiale TDNN basée uniquement sur une représentation temporelle. Avant d'analyser les différentes techniques de couplage des réseaux à convolution, nous avons tout d'abord cherché à évaluer quels étaient les apports de chaque réseau, TDNN et SDNN, en analysant leurs performances croisées.

5.2 SDNN : reconnaisseur de caractères en-ligne

Dans cette démarche, il convient de synthétiser des images de caractères à partir des trajectoires disponibles originellement, sous forme de séquences de points (bases Unipen). Cette transformation est évidemment plus aisée que la transformation inverse [Lallican, 2000].

5.2.1 Passage d'un signal en ligne à une image hors-ligne.

La Figure 66 illustre les différentes étapes mises en œuvre pour cela. Nous pouvons dès lors utiliser les mêmes bases pour le TDNN et le SDNN.

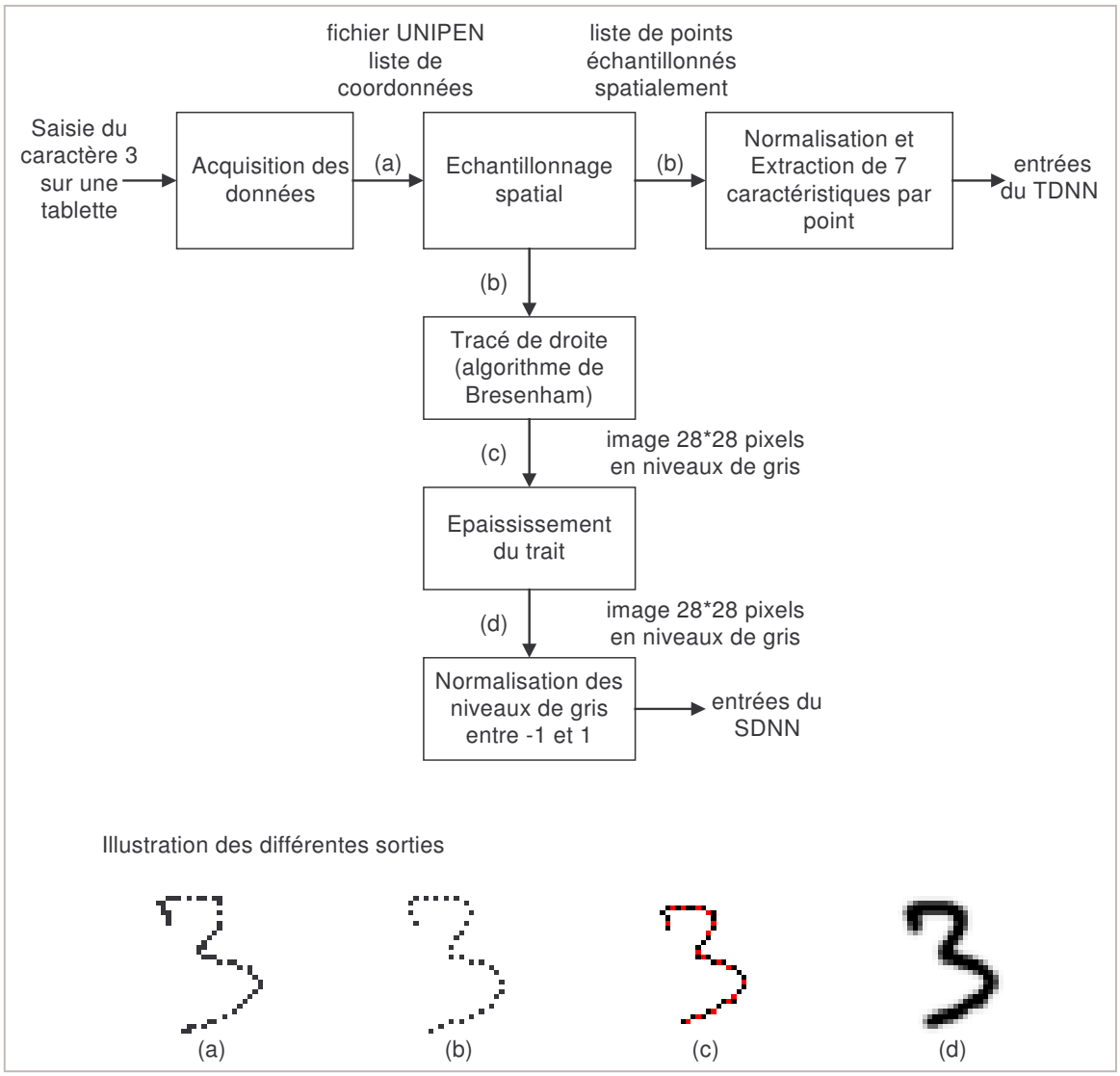


Figure 66. Normalisation et Extraction de l'information en-ligne et construction de l'information picturale à partir des données dynamiques normalisées.

L'entrée du TDNN résulte d'un échantillonnage spatial du signal de façon à obtenir des points équidistants et ainsi s'affranchir de la variabilité de la vitesse du tracé. Cet échantillonnage est suivi d'un recentrage des coordonnées par rapport au centre de masse du caractère suivi d'une extraction de sept caractéristiques normalisées entre [-1,1] : les coordonnées x et y, la direction ($\cos\theta$, $\sin\theta$) et la courbure ($\cos\Phi$, $\sin\Phi$) de la trajectoire et l'état

posé ou levé du stylet en ce point. Ces caractéristiques ont été détaillées dans le chapitre précédent.

Pour construire l'entrée du SDNN soit une image 28×28 en niveau de gris normalisé entre [–1,1], nous sommes partis du signal ré-échantillonné spatialement. Par un algorithme de Bresenham, nous avons tracé des chemins connexes entre chaque point du signal et puis nous avons appliqué un filtrage gaussien 5×5 de variance 0,6 afin d'épaissir le trait obtenu. La chaîne de transformation du signal dynamique en signal hors-ligne est illustrée à la Figure 66 sur un exemple de chiffre 3.

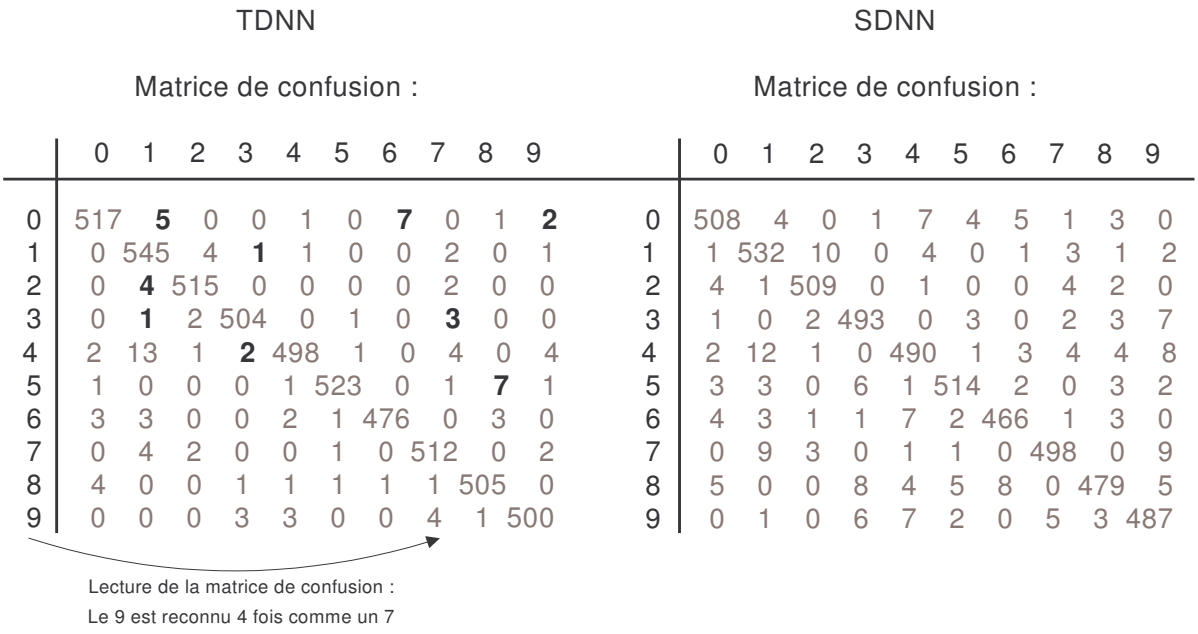
5.2.2 Performances du SDNN sur des données hors-ligne synthétisées

Nous avons testé le SDNN, en utilisant la même architecture que dans la section 5.1.2, sur les bases de chiffres, minuscules et majuscules en-ligne UNIPEN, décrites au chapitre précédent, qui ont été transformées en imagettes suivant le procédé exposé ci-dessus.

Tableau 17. Taux de reconnaissance du TDNN et SDNN sur les bases de test caractères UNIPEN

Bases Unipen	TDNN 7/20/20/5 PMC 1 cc 100 neurones		SDNN 12*12/20/6*6/2*2 4*4/20/6*6/2*2 PMC linéaire	
	Paramètres libres	Taux de Reconnaissance %	Paramètres libres	Taux de Reconnaissance %
Chiffres	17 930	97,9	18 750	95,5
Minuscules	19 546	92,8	23 886	86,6
Majuscules	19 546	93,6	23 886	89,5

Le SDNN, appliqué à la reconnaissance hors-ligne est naturellement moins performant que le TDNN adapté à traiter des données temporelles cependant les performances de reconnaissance sont tout de même acceptables. Le taux de reconnaissance n'est pas le facteur essentiel dans notre démarche. En effet si on analyse les matrices de confusion de chacun des deux classifieurs sur la base de chiffres en test, cf. Figure 67, on remarque que les confusions du TDNN et du SDNN ne sont pas identiques. On peut espérer récupérer des exemples mal reconnus par le TDNN en intégrant les connaissances du SDNN ; c'est en outre le cas des confusions matérialisées en gras dans la matrice de confusion du TDNN qui sont plus importantes que celles du SDNN pour le même label.



	0	1	2	3	4	5	6	7	8	9
0	517	**5**	0	0	1	0	**7**	0	1	**2**
1	0	545	4	**1**	1	0	0	2	0	1
2	0	**4**	515	0	0	0	0	2	0	0
3	0	**1**	2	504	0	1	0	**3**	0	0
4	2	13	1	**2**	498	1	0	4	0	4
5	1	0	0	0	1	523	0	1	**7**	1
6	3	3	0	0	2	1	476	0	3	0
7	0	4	2	0	0	1	0	512	0	2
8	4	0	0	1	1	1	1	1	505	0
9	0	0	0	3	3	0	0	4	1	500

	0	1	2	3	4	5	6	7	8	9
0	508	4	0	1	7	4	5	1	3	0
1	1	532	10	0	4	0	1	3	1	2
2	4	1	509	0	1	0	0	4	2	0
3	1	0	2	493	0	3	0	2	3	7
4	2	12	1	0	490	1	3	4	4	8
5	3	3	0	6	1	514	2	0	3	2
6	4	3	1	1	7	2	466	1	3	0
7	0	9	3	0	1	1	0	498	0	9
8	5	0	0	8	4	5	8	0	479	5
9	0	1	0	6	7	2	0	5	3	487

Figure 67. Matrice de confusion du TDNN et du SDNN sur la base des chiffres UNIPEN

Le TDNN et le SDNN offrent des performances de reconnaissance très intéressantes séparément, il est intéressant d'étudier le comportement respectif de l'un vis-à-vis de l'autre.

5.2.3 Résultats et Performances croisées TDNN/SDNN

Pour cela, nous avons analysé les exemples reconnus et non reconnus par chacun des classifieurs. Le Tableau 18 présente la répartition des échantillons de la base de test en sortie des deux reconnaisseurs suivant qu'ils soient bien reconnus (OK) ou non (KO). Nous constatons effectivement une complémentarité intéressante due à l'apport des deux types d'information, celle dynamique et celle spatiale.

Tableau 18. Performances croisées du TDNN et du SDNN, nombres d'exemples reconnus et non reconnus avec les pourcentages de reconnaissance associés sur la base des chiffres UNIPEN

		TDNN \ SDNN	OK	KO
Exemples non reconnus par le TDNN	KO	110 (2,1 %)	**38** (0,7 %)	72 (1,4 %)
Exemples reconnus par le TDNN	OK	5 102 (97,9 %)	4 937 (94,8 %)	**165** (3,1 %)
	Total : 5212 ex		4 975 (95,5 %)	237 (4,5 %)
		Total : 5212 ex	OK	KO
			Exemples reconnus par le SDNN	Exemples non reconnus par le SDNN

Cette première analyse montre que les deux systèmes n'ont pas le même comportement vis-à-vis de tous les exemples. Il y a par exemple, 38 échantillons (0.7%) qui sont bien reconnus dans le domaine statique alors qu'ils ne le sont pas en dynamique. Les Tableau 19 et Tableau 20 montrent que ce pourcentage est encore plus élevé dans le cas des majuscules (4.1 %) et des minuscules (3.3 %).

On peut donc espérer que la combinaison permettra de récupérer tout ou partie de ces exemples mal classifiés par le TDNN mais bien classifiés par le SDNN, voire même de récupérer des exemples qui ne seraient en première position ni dans le domaine statique (SDNN), ni dans le domaine dynamique (TDNN), mais tels que la combinaison leur soit favorable. L'inverse est aussi malheureusement possible, un exemple bien classé par l'un des deux classifieurs peut perdre sa première place après combinaison.

Tableau 19. Performances croisées du TDNN et du SDNN, nombres d'exemples reconnus et non reconnus avec les pourcentages de reconnaissance associés sur la base des minuscules UNIPEN

Exemples non reconnus par le TDNN	KO	1 255 (7,2 %)	**722** (4,1%)	533 (3,1 %)
Exemples reconnus par le TDNN	OK	16 168 (92,8 %)	14 367 (82,5%)	**1 801** (10,3 %)
	Total : 17 423 exemples	TDNN / SDNN	15 089 (86,6 %)	2 334 (13,4 %)
		Total : 17 423 exemples	OK	KO
			Exemples reconnus par le SDNN	Exemples non reconnus par le SDNN

Tableau 20. Performances croisées du TDNN et du SDNN, nombres d'exemples reconnus et non reconnus avec les pourcentages de reconnaissance associés sur la base des majuscules UNIPEN

Exemples non reconnus par le TDNN	KO	565 (6,4 %)	**289** (3,3%)	276 (3,1 %)
Exemples reconnus par le TDNN	OK	8304 (93,6 %)	7 648 (86,2%)	**656** (7,4 %)
	Total : 8 869 exemples	TDNN / SDNN	7 937 (89,5 %)	932 (10,5 %)
		Total : 8 869 exemples	OK	KO
			Exemples reconnus par le SDNN	Exemples non reconnus par le SDNN

Les performances hors-ligne sur la base MNIST sont dignes des résultats actuels collectés par Yann LeCun et disponibles sur le site de la base, sachant que la base n'a pas été modifiée (pas d'échantillons supplémentaires obtenus par déformation élastique, ni corrections de la base).

L'application sur des données en-ligne avait pour but de montrer l'intérêt de coupler des informations en-ligne et hors-ligne. L'analyse des performances croisées d'un TDNN et d'un SDNN sur la base UNIPEN a validé cette hypothèse. Le SDNN, qui analyse le signal dynamique transformé en une image de caractères, reconnaît un nombre non négligeable d'exemples non reconnus par le TDNN ce qui permettrait de réduire entre 30 et 50% l'erreur commise par le TDNN seul.

Nous allons donc maintenant analyser comment il est possible de combiner ces deux informations.

5.3 Combinaison des classifieurs

Pour intégrer les informations picturales et dynamiques plusieurs solutions sont envisageables. La première consiste à élargir le vecteur des caractéristiques présenté à l'entrée du système de sorte qu'il intègre à la fois des informations de nature statique et d'autres de type dynamique. C'est la solution qui est par exemple retenue lorsque l'on ajoute une imagette basse résolution représentant le contexte local associé à chaque point (REMUS, AMAP). Cette combinaison, illustrée à la Figure 68-a, a principalement une limitation : la complexité de la modélisation devient trop importante notamment sur des machines à ressources limitées.

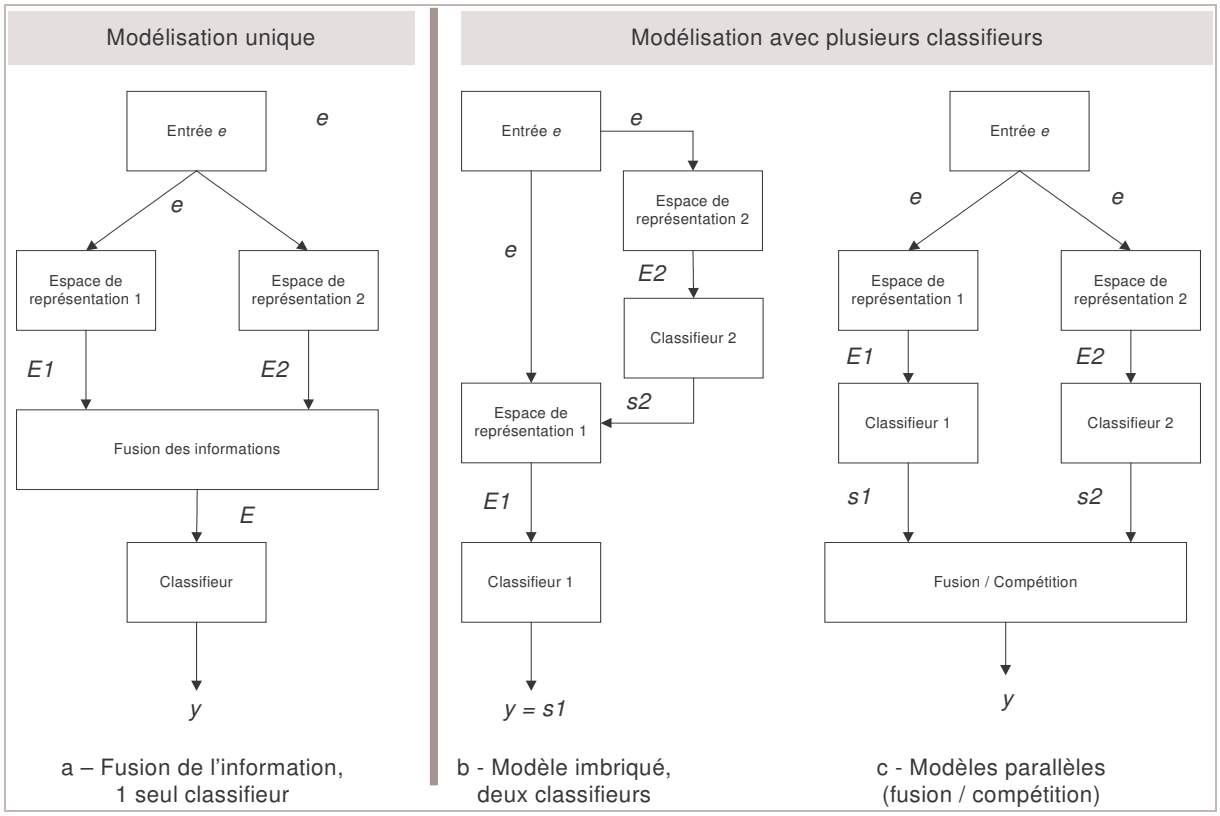


Figure 68. Schémas d'intégration d'une représentation statique dans une représentation dynamique.

Une autre solution consiste à traiter séparément les deux informations, grâce à deux classifieurs distincts, puis à combiner leurs résultats. La problématique de la combinaison de classifieurs est une thématique à part entière [Lam, 1995]. Plusieurs stratégies sont alors possibles : combinaison parallèle, combinaison cascade, schéma mixte. La Figure 68 présente certaines de ces solutions. D'autre part, la complémentarité peut porter soit sur la nature différente des données traitées, comme c'est le cas ici, soit sur la nature différente des classifieurs : (PMC/SVM [Bellili, 2002] ; MMC/PMC Prevost, 2003 ; k-ppv/PMC [Alpaydin, 2000]), l'idée principale étant de tirer parti au maximum de la complémentarité de différentes modélisations et de compenser leurs faiblesses propres.

Deux schémas pour le couplage TDNN/SDNN sont alors possibles. Le premier, cf. Figure 68-b est un modèle dit imbriqué, les sorties du SDNN sont ajoutées aux caractéristiques de haut niveau calculées par la partie extraction du TDNN qui sont présentées au perceptron de la partie classification du TDNN.

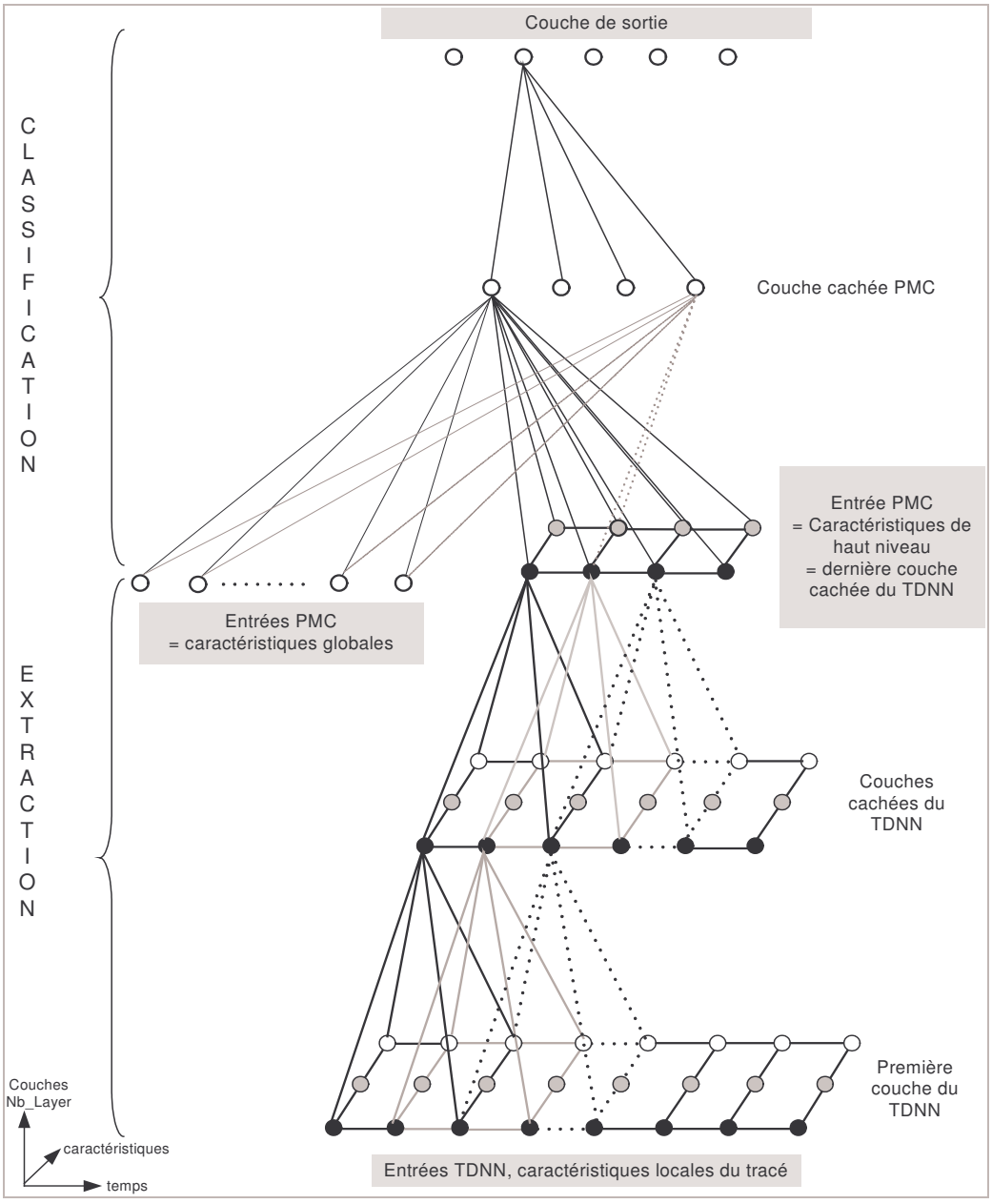


Figure 69. Architecture imbriquée

Les sorties du SDNN peuvent être ajoutées soit en entrée du PMC de la partie classification du TDNN, comme il est illustré à la Figure 69 soit directement connectées à la couche de sortie. Dans ce cas, l'imbrication des sorties SDNN dans le TDNN n'assure pas de conserver au minimum les performances croisées des deux reconnaisseurs.

Le deuxième schéma de couplage est basé sur une classification parallèle des deux représentations distinctes, l'une en-ligne et l'autre hors-ligne, les classifieurs 1 et 2 de la Figure 68-c correspondent aux TDNN et SDNN. Les sorties des deux classifieurs peuvent être soit mises en compétition soit fusionnées. C'est ce type de combinaison que nous avons développé et testé.

5.4 Architecture SDTDNN

La fusion des sorties des deux classifieurs a été réalisée selon deux principes, Figure 70. Tout d'abord nous avons employé un simple couplage en sortie des deux configurations précédemment définies en réalisant une moyenne géométrique des sorties. Il s'agit là d'une méthode classique de combinaison qui se justifie si l'on suppose l'indépendance des variables. Plus difficilement justifiable, des travaux ont également montré qu'une moyenne arithmétique permettait aussi d'améliorer les résultats de reconnaissance [Tax, 1999]. La deuxième configuration que nous avons développée repose sur une coopération via les couches cachées, nous l'avons appelée SDTDNN. Dans ce cas, l'apprentissage devra permettre d'équilibrer les poids respectifs des deux sous-systèmes.

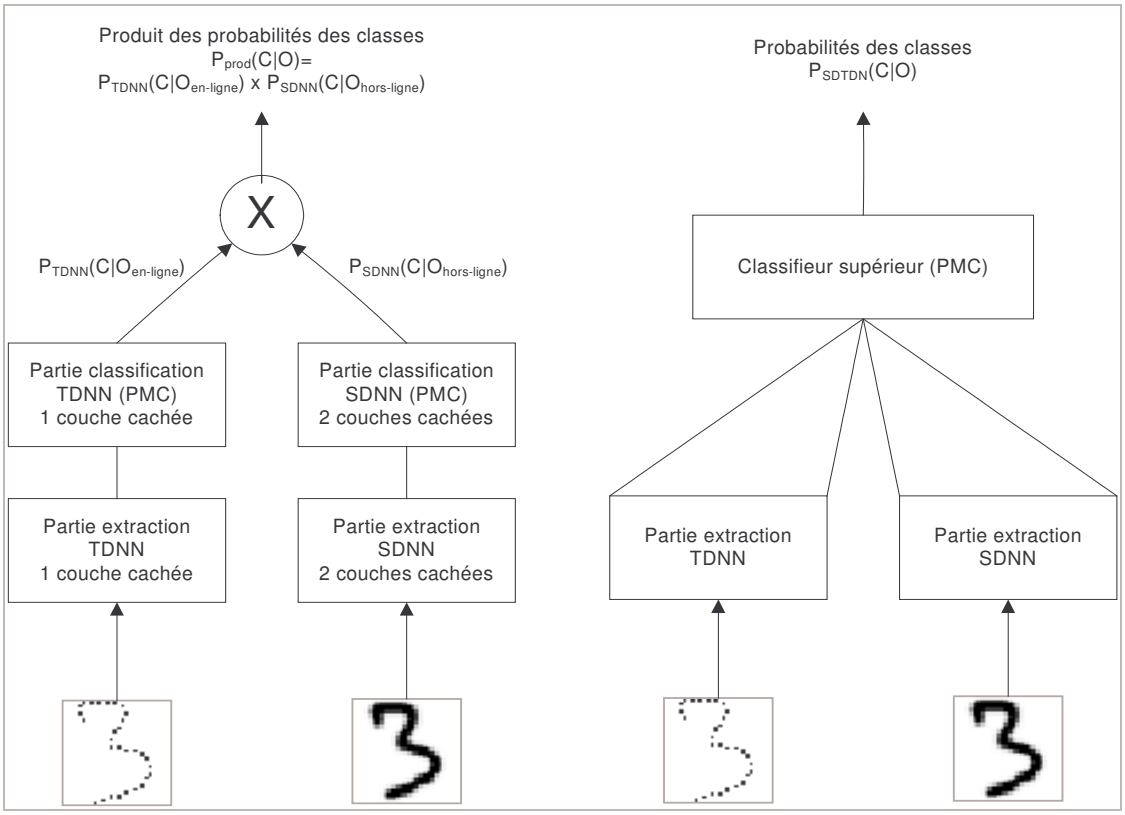


Figure 70. Couplages TDNN/SDNN

5.4.1 Combinaison produit

Cette configuration, appelée couplage produit, calcule le produit des sorties des réseaux TDNN et SDNN entraînés séparément. Les sorties de chaque réseau correspondent aux probabilités *a posteriori* des classes selon l'observation d'entrée *P(C|O),* obtenues par la fonction de transfert de type Softmax des neurones de la couche de sortie.

Pour analyser l'effet du couplage produit, nous avons conservé les architectures du TDNN et SDNN décrites dans la section 5.1. Le but est de vérifier si un tel couplage permet de récupérer les exemples non reconnus par le TDNN et reconnus par le SDNN mais aussi les exemples non reconnus par les deux réseaux. Cela est possible si les candidats classés avant la bonne solution par l'un se trouvent bien plus mal classés que la bonne solution dans l'autre. Par exemple, avec les scores ordonnés : TDNN : 'a' = 0.5 ; 't' = 0.4 ; …. ; 'u' = 0.01 et SDNN : 'u' = 0.6 ; 't' = 0.2 ; …. ; 'a' = 0.02, on obtient après produit : 't' = 0.08 ; 'a' = 0.01 ; 'u' = 0.006. Ce qui fait passer le caractère 't' en première position (en espérant qu'il s'agira du bon label).

Tableau 21. Effet du couplage produit sur les bases de test UNIPEN

	TDNN : OK SDNN : OK	TDNN :OK SDNN :KO	TDNN :KO SDNN :OK	TDNN :KO SDNN :KO	Total
Chiffres					
Produit : OK	4 937 (94,8%)	147 (2,8%)	29 (0,5%)	3 (0,1%)	5 116 (**98,2%**)
Produit : KO	0	18 (0,3%)	9 (0,2%)	69 (1,3%)	96 (1,8%)
Total	4937 (94,8%)	165 (3,1%)	38 (0,7%)	72 (1,4%)	5212
Minuscules					
Produit : OK	14 367 (82,5%)	1 451 (8,3%)	320 (1,8%)	65 (0,4%)	16 203 (**93,0%**)
Produit : KO	0	350 (2,0%)	402 (2,3%)	468 (2,7%)	1 220 (7,0%)
Total	14 367 (82,5%)	1 801 (10,3%)	722 (4,1%)	533 (3,1 %)	17 423
Majuscules					
Produit : OK	7 648 (86,2%)	549 (6,2%)	140 (1,6%)	41 (0,5%)	8 378 (**94,5%**)
Produit : KO	0	107 (1,2)	149 (1,7%)	235 (2,6%)	491 (5,5%)
Total	7 648 (86,2%)	**656** (7,4 %)	**289** (3,3%)	276 (3,1 %)	8 869

Le Tableau 21 montre l'effet du couplage produit du TDNN et SDNN, les résultats obtenus confirment le bénéfice de la combinaison. Prenons le cas de la base de test des chiffres, parmi les exemples reconnus par un seul des classifieurs TDNN (165) ou SDNN (38), la plupart des exemples (147+29) sont correctement classifiés par le couplage produit. Seuls 8 % des exemples (18 + 9 = 27) ne tirent pas profit de cette configuration à cause d'une estimation de probabilités trop déséquilibrée entre les deux classifieurs. C'est le cas par exemple des labels 4 et 9 donnés dans le Tableau 22, illustrant quelques cas non reconnus par le couplage produit

alors qu'ils l'étaient par l'un des classifieurs. Par ailleurs, le couplage permet de récupérer quelques exemples (0,1 %) reconnus ni par le TDNN ni par le SDNN.

Tableau 22. Exemples de chiffres non reconnus par le couplage produit

Fichier UNIPEN	Label	Couplage produit		TDNN	SDNN
		Label TOP1 (%)	Label TOP2 (%)	Label TOP1 (%)	Label TOP1 (%)
1A\TOS\T1\T1053065_0.unp	**1**	7 (0.29)	**1** (0.06)	7 (0,46)	7 (0,64)
1A\VAL\GRESAI38_67.unp	**8**	0 (0.92)	**8** (0.00)	0 (0.99)	0 (0.92)
1A\TOS\T3\T3010064_2.unp	**6**	3 (0.003)	8 (0.000)	3 (0.99)	8 (0.72)
1A\TOS\F3\F3017065_2.unp	**4**	9 (0.42)	**4** (0.04)	**4** (0.53)	9 (0.91)
1A\SYN\SYN1\JCP-DIG11_423.unp	**9**	7 (0.17)	**9** (0.00)	7 (0.94)	**9** (0.81)

Le couplage produit permet ainsi de réduire de près de 14% en moyenne le taux d'erreurs sur les bases de test chiffres, minuscules et majuscules Unipen par rapport au TDNN seul. Nous n'avons pas cherché ici à tirer profit de la supériorité intrinsèque du TDNN vis-à-vis du SDNN. On aurait pu le faire simplement avec une moyenne géométrique pondérée, en donnant un poids un peu supérieur aux sorties du TDNN ($\alpha > 0.5$).

$$P_{prod} = P_{TDNN}^{\alpha} \times P_{SDNN}^{1-\alpha} \qquad \text{avec } 0 \leq \alpha \leq 1$$

Il est important de noter que le nombre de paramètres libres du couplage est la somme des paramètres libres des deux réseaux. Même si cette somme est égale à peu près au double du nombre de poids du TDNN simple, elle reste largement inférieure au nombre de paramètres libres des classifieurs dominants (SVM, Mélidis) dans le Tableau 14 des références sur la base UNIPEN présenté au chapitre 4.

5.4.2 Combinaison SDTDNN

Pour que le couplage des deux systèmes bénéficie d'un apprentissage global unique, nous avons construit une architecture modulaire dite SDTDNN, Space Displacement and Temporal Delay Neural Network. Cette structure (Figure 70, droite) est composée de deux parties inférieures d'extraction de caractéristiques et d'une partie supérieure de classification unique donnant en sortie les probabilités *a posteriori* des classes. La première partie correspondant aux couches basses intègre deux modules : un module TDNN traitant les caractéristiques en-ligne et un module SDNN traitant l'image correspondante. La dernière partie combine les caractéristiques en-ligne et hors-ligne pour affecter à chaque classe une probabilité d'appartenance.

La définition de la topologie du SDTDNN a été déterminée pour répondre au critère du compromis taux de reconnaissance / nombres de paramètres libres. Nous avons retenu principalement deux architectures l'une avec un classifieur linéaire, l'autre avec un classifieur

avec une couche cachée de 50 neurones. Le Tableau 23 détaille ces architectures, leurs performances en terme de taux de reconnaissance et leurs nombres de paramètres libres (poids).

Tableau 23. Comparaison des couplages produit et SDTDNN

Base Unipen	TDNN 7/20/20/5 PMC 1cc 100		Couplage Produit		SDTDNN TDNN (21/20/10/2) SDNN (12*12/6/6*6/2*2-4*4/20/6*6/2*2) PMC linéaire)		SDTDNN TDNN (21/10/10/2-9/20/5/2) SDNN (12*12/6/6*6/2*2-4*4/10/6*6/2*2) PMC 1cc 50	
	Poids	Taux de reco %	Poids	Taux de reco %	Poids	Taux de reco %	Poids	Taux de reco %
Chiffres	17 930	97,9	36 300	98,2	13 382	97,9	21 672	98,1
Minuscules	19 546	92,8	43 052	93,0	25 222	92,8	22 472	92,8
Majuscules	19 546	93,5	43 052	94,5	25 222	93,6	22 472	94,2

Cette solution s'avère un peu décevante, elle ne fait guère mieux que le TDNN seul, avec une complexité légèrement supérieure. Seules les majuscules tirent un profit un peu plus marqué en passant de 93,5 à 94,2 %. Les résultats obtenus montrent que si l'on souhaite améliorer les performances de reconnaissance, c'est le couplage produit qui donne les meilleurs résultats. Cela se paye évidemment par une augmentation sensible du nombre des paramètres puisque les systèmes travaillent en parallèle.

5.5 Conclusion

Dans cette partie, nous nous sommes intéressés aux capacités des réseaux à convolution dans le cadre de la reconnaissance en-ligne et hors-ligne.

Notre étude a porté sur l'analyse des performances individuelles, puis couplées de ces réseaux – TDNN, SDNN. Ce dernier a été introduit dans le but d'inclure dans notre système l'information statique de l'écriture. Nous avons présenté une nouvelle architecture multi-modulaire appelée SDTDNN, basée sur la fusion des caractéristiques extraites d'un TDNN et d'un SDNN .

Nous avons démontré qu'il n'était pas rédhibitoire de construire un système à deux réseaux de neurones à convolution pour des applications à ressources limitées et que l'ajout d'une représentation hors-ligne permet d'accroître les taux de reconnaissances des caractères isolés.

Par ailleurs, en comparant les performances des classifieurs actuels sur la base UNIPEN avec celles des schémas que nous avons présentés (TDNN, couplage produit d'un TDNN et SDNN, architecture SDTDNN), nous pouvons conclure qu'un très bon compromis performance/ressources a été trouvé.

PARTIE III :
RECONNAISSANCE DE MOTS MANUSCRITS EN-LIGNE

6 SYSTEME DE RECONNAISSANCE TDNN MOT

Le passage d'un système de reconnaissance caractère à un système de reconnaissance mot est loin d'être trivial. Pour faciliter cette transition, on peut imposer certaines contraintes sur le style d'écriture, comme par exemple utiliser une écriture scripte [Pasquer, 2000 ; Oudot, 2003]. Dans les autres cas, la complexité du système devient rapidement très importante, à la fois dans son architecture et son mode d'apprentissage [LeCun, 2001 ; Tay 2002]. Ainsi, par exemple, le nombre de paramètres peut devenir rédhibitoire pour une application sur appareil mobile, c'est par exemple le cas des approches basées SVM [Ragot, 2003], ou bien encore la méthode d'apprentissage être très contraignante, avec des procédures en plusieurs étapes, éventuellement d'abord au niveau lettre ou graphème, puis au niveau mot et cela de façon itérative [Jaeger, 2000].

La solution que nous proposons cherche à satisfaire à la fois des contraintes de taille (coût mémoire) et de facilité d'apprentissage, de telle sorte que celui-ci se fasse globalement au niveau mot et que d'autre part il fasse intervenir une prise en compte des caractéristiques propres à chaque classe (connaissances intrinsèques : approche basée modèle) et également des caractéristiques discriminantes entre classe (approche basée frontière).

L'étude approfondie des réseaux de neurones à convolution pour la reconnaissance de caractères manuscrits isolés a montré que ce type de réseau était très adéquat par rapport à l'objectif visé : il apporte une bonne performance en terme de taux de reconnaissance avec une architecture de complexité raisonnable. De plus, on dispose d'un algorithme efficace (rétropropagation du gradient) pour entraîner une telle architecture, pour peu que l'on soit capable de ramener sur les sorties du réseau la fonction de coût définie globalement au niveau mot. Nous avons donc cherché à étendre cette architecture à la reconnaissance de mots manuscrits en-ligne. Dans ce chapitre, nous présentons une architecture neuro-markovienne couplant un réseau de neurones à délai (Time Delay Neural Network : TDNN) et des modèles de Markov Cachés (MMC/HMM).

6.1 Méthodologie : présentation générale du système

Ce système rentre dans la catégorie des systèmes de type analytique, avec segmentation implicite et apprentissage global au niveau mot. Le schéma général du système de reconnaissance que nous avons implémenté est présenté sur la Figure 72. C'est un système hybride qui fait coopérer un modèle de Markov caché et un réseau à convolution (RC). Cette coopération consiste à utiliser le MMC pour effectuer la tâche de reconnaissance au niveau mot en trouvant le meilleur alignement temporel (programmation dynamique) sur les différents modèles mots du lexique, à partir des probabilités d'observations estimées par le RC dans sa tâche de caractérisation des entités élémentaires -liées aux fenêtres de convolution, qu'il balaie régulièrement le long de la trajectoire.

Les données fournies en entrée du réseau à convolution proviennent directement du signal dynamique e(t) = (x(t), y(t)) obtenu après pré-traitement du signal acquis. Le réseau de neurones transforme la trame formée par les caractéristiques extraites en chaque point de la trajectoire (N points) en une autre trame (T observations) contenant les probabilités des observations selon les états du MMC. Il s'appuie pour cela sur une fenêtre glissante d'observations, ainsi que le montre la Figure 71. La dernière étape, à savoir l'étape de reconnaissance est quant à elle, basée

sur l'utilisation de modèles de Markov, elle fournit en sortie une liste des scores décroissants obtenus en fonction du dictionnaire utilisé.

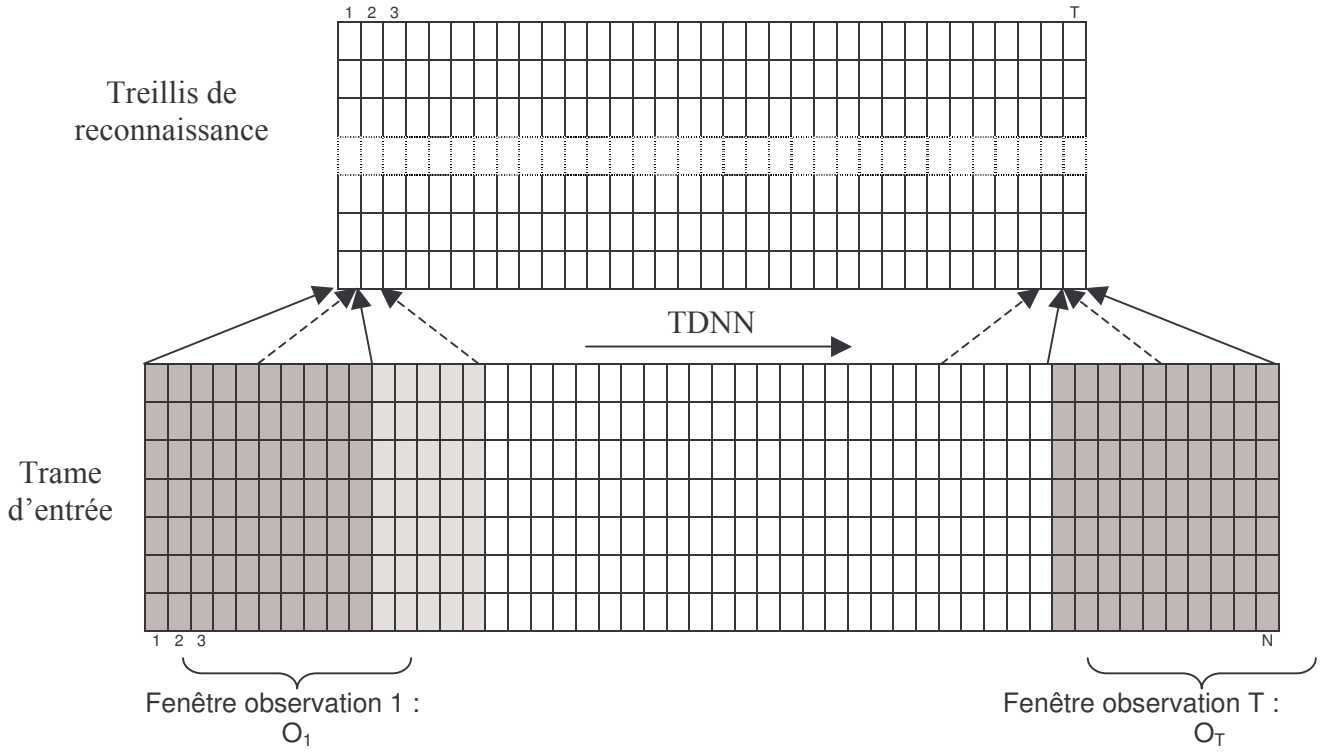


Figure 71. Principe du balayage temporel

Le signal d'entrée représente la trajectoire de l'outil d'écriture échantillonnée à intervalle de temps quasi-constant, typiquement 10 ms. Il sera disponible pour nous sous la forme d'un fichier au format Unipen [Guyon, 1994] pour chacun des mots des bases de données qui seront utilisées. Dans une première étape, ce signal subit une série de prétraitements afin de le normaliser. Cette étape est décrite dans la section 6.2, il en résulte un signal d'écriture indépendant du scripteur tant en vitesse, taille, orientation de l'écriture. Un ensemble de caractéristiques géométriques et structurelles est extrait de chaque point du signal pour former ainsi les données d'entrée du réseau de neurones à convolution. Le réseau effectue un balayage temporel du mot et transforme cette séquence de vecteurs de caractéristiques par point en une séquence de valeurs qui estiment pour chaque classe (chaque état) les densités de probabilités sur l'espace des observations. Durant la phase de reconnaissance, la vraisemblance de chaque mot issu d'un dictionnaire donné, est calculée en utilisant les modèles MMC appris. Ce calcul peut être réalisé en utilisant soit l'algorithme *forward*, soit l'approximation de Viterbi [Rabiner, 1989].

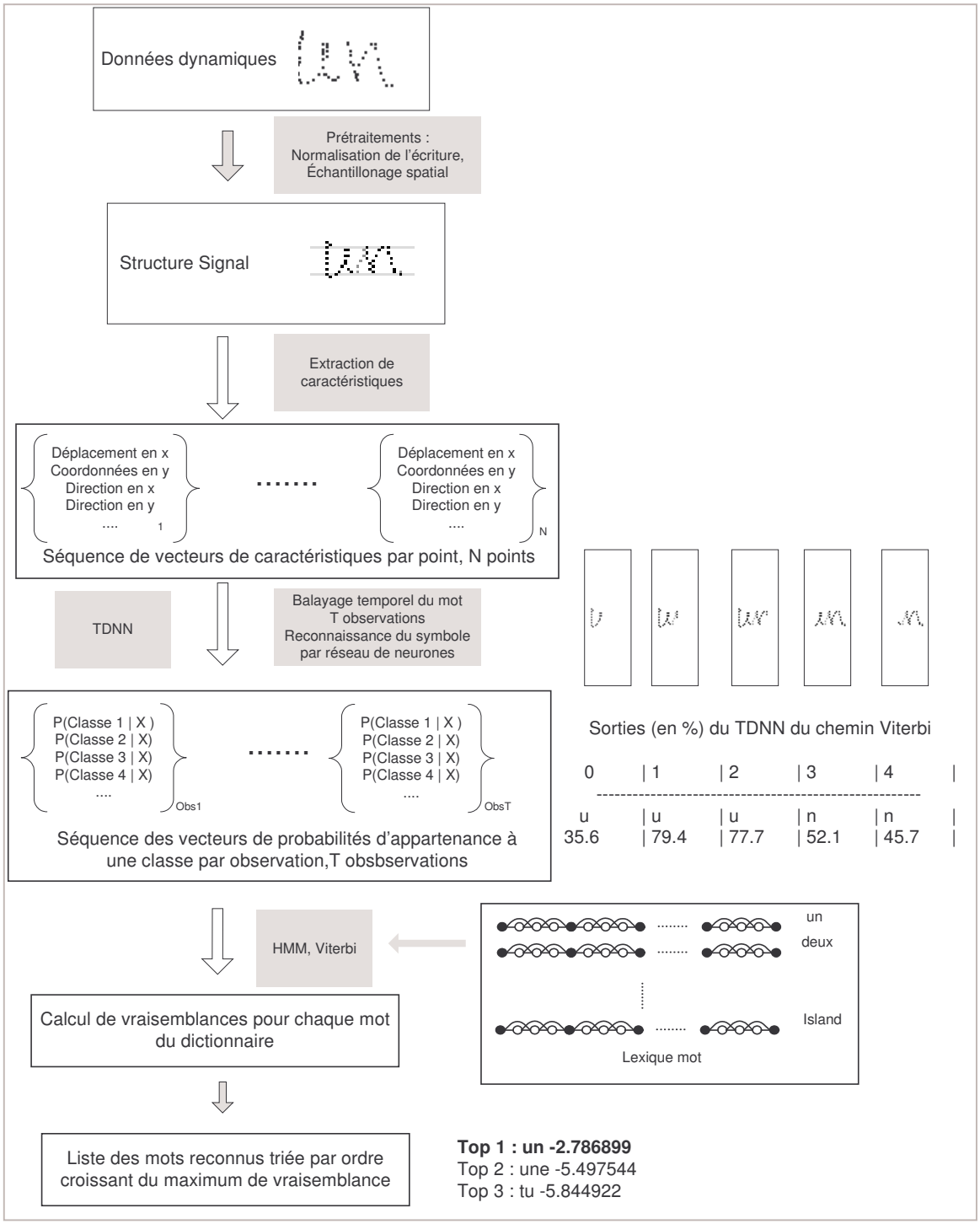


Figure 72. Schéma général du système de reconnaissance de mots saisis en-ligne

6.2 Prétraitements

La variété de l'écriture manuscrite dépend de différentes sources comme l'identité des auteurs (âge, état physique), l'environnement et la posture pendant l'écriture (à un bureau, sur le capot d'une voiture), mais aussi l'outil d'écriture (petit stylet, gros stylo digital) et les médias utilisés (papier, écran) sans compter l'objectif du message écrit. Dans un contexte omniscripteur, la plupart des variations sont non pertinentes, voire perturbatrices, il est donc

préférable de les éliminer le plus tôt possible pour soulager la difficile tâche de reconnaissance. C'est le but principal de l'étape de prétraitement du signal d'écriture dans notre système.

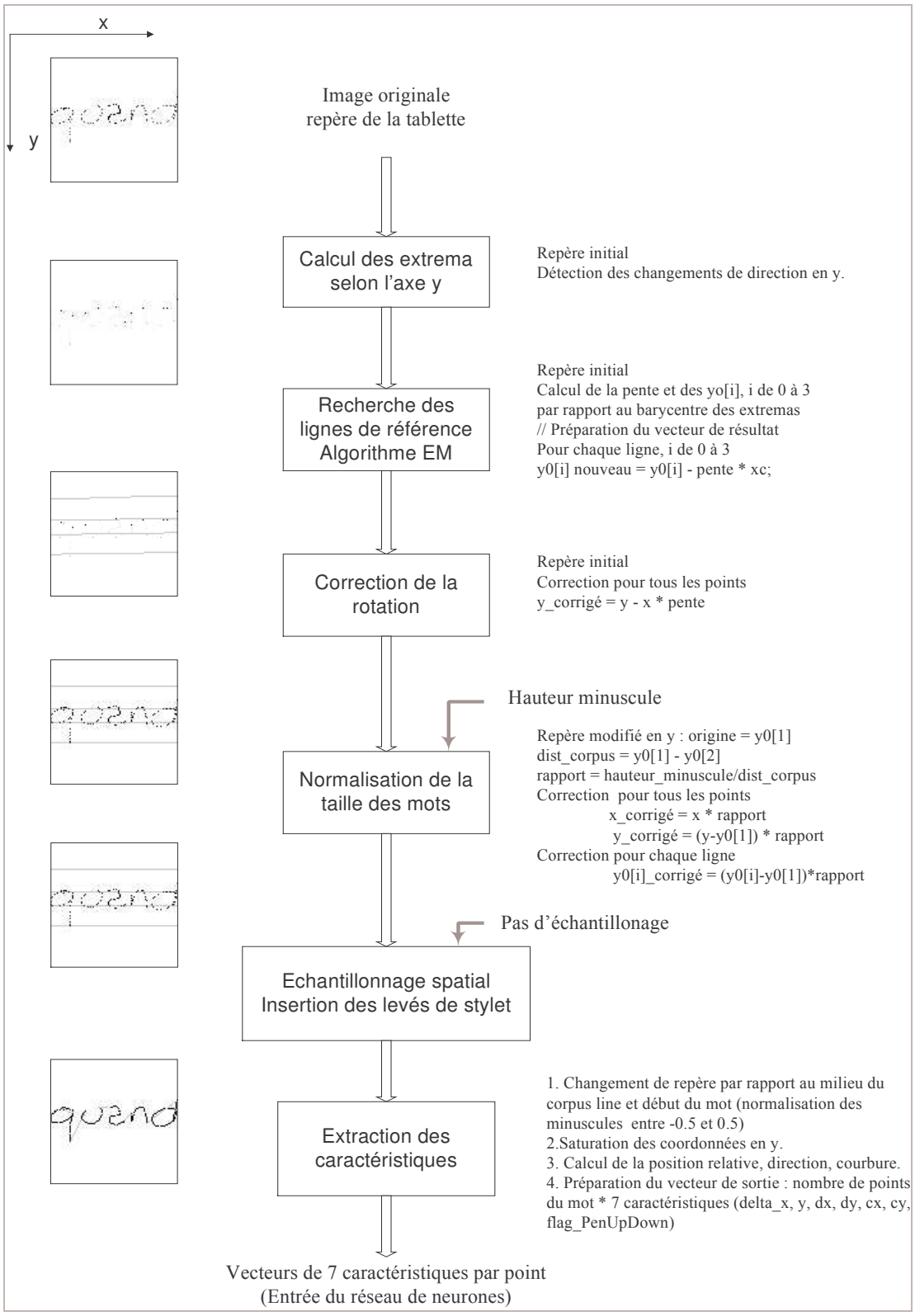


Figure 73. Visualisation des étapes de prétraitements sur le mot "quand"

La Figure 73 illustre les prétraitements effectués sur le mot « quand » et la phase d'extraction des caractéristiques. Les prétraitements se résument en quatre grandes phases décrites ci-après :

1. le calcul des lignes de références avec le calcul des extrema du signal,
2. la correction de la rotation du signal,
3. la correction de la taille du mot,
4. et l'échantillonnage spatial du signal d'écriture.

6.2.1 Détermination des lignes de références

Les lignes de référence (cf. chapitre 1) apportent une information importante pour des systèmes d'identification de l'écriture. Elles permettent de normaliser la taille de l'écriture et d'extraire des caractéristiques géométriques liées à la position et à l'orientation des lettres dans le mot. Le signal d'entrée fournit les coordonnées du stylet et ses levers ou posers sur le support graphique.

La détermination des lignes de références est basée sur une paramétrisation des lignes proposée dans [Bengio, 1994]. Nous avons simplifié la modélisation des lignes en les considérant droites (paraboliques dans la version originale), cela afin d'éviter une instabilité du modèle due à la prise en compte de la courbure du mot qui n'est pas toujours très stable. Le modèle est ainsi décrit par cinq paramètres optimisés par un algorithme stochastique EM (E pour Expectation et M Maximization): la pente et l'ordonnée des quatre lignes de référence, cf. Figure 74.

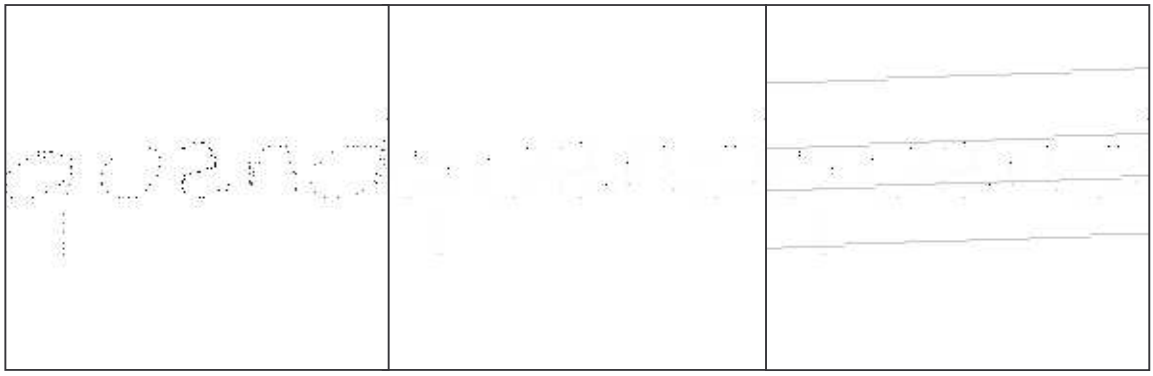


Figure 74. Visualisation de la détermination des lignes de référence du mot « quand ». Les vignettes dans l'ordre de gauche à droite correspondent à :1- le signal original, 2- détermination des extrema, 3- affichage des lignes de référence

Pour détecter les lignes de référence, nous extrayons d'abord les minima locaux et les maxima de l'écriture par une simple analyse des positions et changements de direction verticale. La Figure 74-2 illustre les minima et les maxima extraits sur un mot.

L'algorithme de recherche des points extrema et des lignes de référence peut être résumé de la manière suivante :

```
Détection des points Extrema (minima et maxima)
Calcul de l'écart type de la distribution en Y des extrema
Détermination des a priori sur les paramètres (inclinaison a priori,
attributs y0 a priori)
Initialisation des paramètres à estimer (pente, vecteur de
translation y0[i], i de 0 à 3)
Tant que le critère d'arrêt est non respecté : Analyse du changement
global des valeurs y0
```

```
Faire
    //PHASE Expectation (algo E-M)
    //Affectation des minima à l'une des lignes de référence
    Pour tous les minima
          Calcul des probabilités d'observation des lignes de
          référence
          Recherche de la probabilité d'observation maximale
          Comparaison entre la probabilité d'observation maximale et
          celle du bruit pour affecter ou non le minima
          Affectation ou non
    //Affectation des maxima à l'une des lignes de référence
    Pour tous les maxima
          Calcul des probabilités d'observation des lignes de
          référence
          Recherche de la probabilité d'observation maximale
          Comparaison entre la probabilité d'observation maximale et
          celle du bruit pour affecter ou non le maxima
          Affectation ou non
    //PHASE Maximisation (algo E-M)
    Calcul de la pente
    Calcul des ordonnées y0[i], i de 0 à 3
    Calcul du changement global des valeurs y0
    Ré-estimation des a priori de var y0 : lâcher les contraintes
    pour les deux lignes extérieures
Fin_Faire
```

Algorithme 7. Algorithme de détection des extrema et lignes de référence

6.2.2 Correction de l'écriture

Le lignes de référence vont permettre de corriger la taille et l'orientation de l'écriture. La première correction concerne l'annulation de la pente déterminée précédemment afin de ramener les lignes de référence horizontales. Une rotation d'angle $-\alpha$, avec pente = $tg(\alpha)$, permet d'obtenir le résultat. Il est ensuite important de normaliser la taille des caractères qui selon les supports -feuille blanche, papier quadrillé, papier Seyes, écran PDA … et les scripteurs peut varier dans de grandes proportions. Pour supprimer cette variabilité, on ramène la hauteur de l'œil du mot à une taille fixe choisie (taille unitaire) en appliquant une homothétie de même rapport selon les deux dimensions afin de ne pas déformer l'apparence du mot.

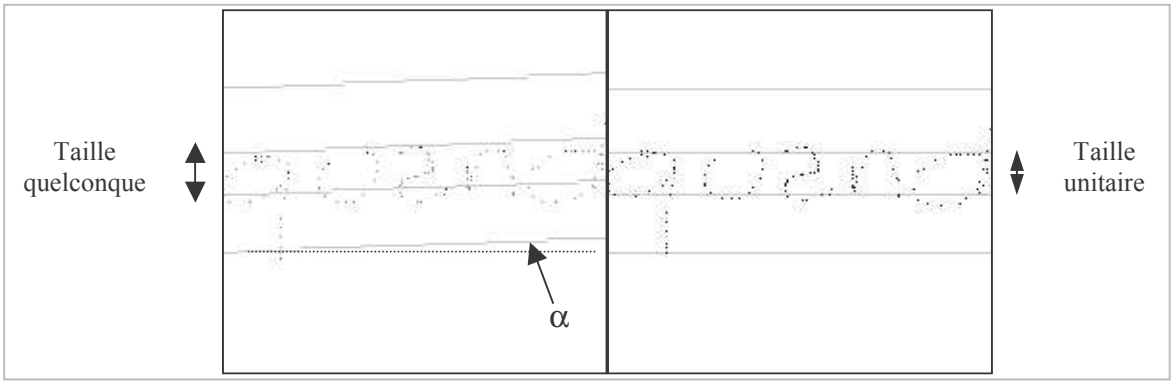


Figure 75. Visualisation de la correction de l'inclinaison du mot « quand ». Les vignettes dans l'ordre gauche droite correspondent à :
1- le signal original avec ses lignes de référence, 2 – le signal corrigé

La Figure 75 illustre ces corrections appliquées sur un exemple de mot.

Une autre correction est quelquefois appliquée à ce stade, elle concerne le redressement de l'inclinaison des lettres, voire cf. section 1.3.1.4 du chapitre 1. En ce qui nous concerne, nous n'avons pas appliqué ce type de correction. Nous comptons sur l'adaptabilité du modèle pour absorber ce type de variation. Il sera de la responsabilité des couches basses du TDNN de détecter la présence éventuelle de primitives graphiques inclinées dans certaines écritures.

6.2.3 Échantillonnage spatial du signal d'écriture

Les systèmes d'acquisition échantillonnent le signal d'entrée avec une fréquence qui peut varier d'un dispositif à l'autre, de l'ordre de 10 Hz à 100 Hz, par ailleurs chaque scripteur écrit avec une vitesse qui peut varier dans de grandes proportions. Il en résulte qu'un même mot pourra compter un très faible nombre de points, s'il est écrit rapidement et/ou échantillonné à faible cadence, ou un nombre beaucoup plus important de points, si inversement la vitesse d'écriture est réduite et/ou la fréquence d'échantillonnage élevée.

En conséquence, pour s'affranchir des variabilités des systèmes d'acquisition et de la vitesse intra et inter scripteur, il est nécessaire d'effectuer un ré-échantillonnage à pas spatial constant du signal. Ce pas est basé sur la hauteur normalisée de la ligne d'œil. Typiquement, cinq points sur cette hauteur donnent une définition correcte pour les caractères. Lors de ce ré-échantillonnage, des points additionnels, non présents dans le signal original, sont introduits entre les levés et les posés de stylos. En effet, les déplacements effectués entre deux traits (strokes) sont informants, ils correspondent « normalement » à des règles d'écriture bien précises. Pour cela, une simple interpolation linéaire est réalisée entre le levé et le posé de stylo, mais une marque distinctive est associée à ces points, elle sera utilisée dans le vecteur des caractéristiques associées à ce point. La Figure 76 illustre ce ré-échantillonnage spatial et affiche en gris clair les points correspondant à la trajectoire inter-traits interpolée.

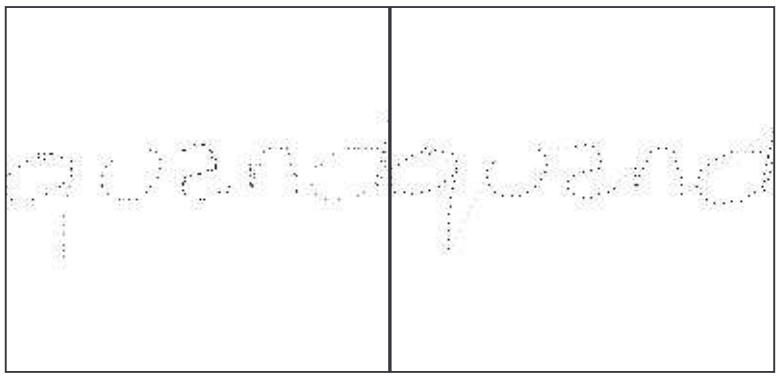


Figure 76. Visualisation de l'échantillonnage spatial et de la caractérisation des levés (en gris) et posés (en noir) de stylo. Les vignettes dans l'ordre gauche droite correspondent à : 1- le signal original, 2 – le signal corrigé et ré-échantillonné

6.2.4 Extraction des caractéristiques

L'objectif de l'étape d'extraction de caractéristiques est de capturer les informations les plus appropriées et discriminantes de l'objet à reconnaître. Une suite de vecteurs de sept composantes, appelées caractéristiques, est extraite du tracé point par point. Ces caractéristiques sont celles les plus standards : déplacement horizontal relatif, position verticale, directions horizontale et verticale et courbures horizontale et verticale du tracé, information de posé ou levé de stylet. Par rapport à la reconnaissance de caractères isolés où la notion de lignes de référence était absente, ici la normalisation des données est légèrement différente.

Les fonctions de transfert des neurones en entrée du réseau de neurones étant de type sigmoïde, les signaux en entrée ont été normalisés entre –1 et +1 pour travailler dans la partie linéaire de cette fonction. Ainsi l'ordonnée de la ligne de base a été fixée à une valeur égale à –0,5 et la ligne d'œil à +0,5. Les hampes sont alors représentées par des valeurs entre 0.5 et 1.0, avec un écrêtage à cette valeur, et les jambages entre –0.5 et -1.0. La même normalisation a été effectuée selon les abscisses pour conserver le ratio des formes.

Pour la première caractéristique, le déplacement horizontal $\Delta x(X_n)$ a été retenu, il correspond à la différence des abscisses du point considéré X_n avec le point précédent, soit : $\Delta x(X_n) = x(X_n) - x(X_n - 1) \quad \forall n > 1 \quad et \quad \Delta x(X_0) = 0$

Les autres caractéristiques sont calculées de la même façon que celles présentées au chapitre 4, cf. paragraphe 4.2.3.

Bien que les prétraitements et l'extraction de caractéristiques aient une place importante dans les performances d'un système de reconnaissance mot, nous avons opté pour des prétraitements et une extraction simples et rapides, compatibles avec une mise en œuvre dans une application temps réel. Nous analysons dans la suite le cœur du système, c'est-à-dire la partie reconnaissance proprement dite.

6.3 Caractérisation des modèles MMC

L'approche proposée étant de type analytique, deux niveaux de modèles sont à définir. Le premier définit l'élément de base retenu, celui-ci se situe au niveau lettre, il servira pour construire le modèle de niveau supérieur qui représentera le mot.

6.3.1 Modèles de lettres

Le modèle lettre le plus simple que l'on puisse envisager se limite à un état unique, Figure 77. Celui-ci devant pouvoir émettre plusieurs observations consécutives associées à la même lettre, il est nécessaire d'y faire figurer une transition intra-état (self).

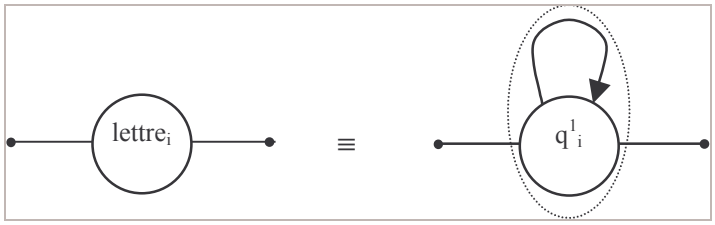


Figure 77. Modèle lettre de base

Dans ces conditions, le RN doit pouvoir estimer autant de densités de probabilités qu'il y a d'états-lettres, il comportera donc autant de sorties, soit dans notre cas 66, correspondant aux 66 symboles de notre alphabet.

En effet, la base de données de mots IRONOFF que nous utiliserons pour entraîner et tester le système, et qui est présentée plus en détails au chapitre suivant, est représentable par un alphabet de 66 caractères comportant : les 26 minuscules, les 26 majuscules, le c cédille : ç, les caractères accentués : à â é è ê ë î ï ô ù û, le trait d'union et l'apostrophe.

Si l'on souhaite donner plus de souplesse aux modèles lettres, il est possible de décomposer le tracé de chaque lettre en plusieurs graphèmes, et de faire prendre en charge la génération de chaque graphème par un état spécifique du modèle lettre. Ainsi, la Figure 78 montre un modèle

lettre avec deux états et la Figure 79 un modèle avec 3 états. Dans chacun de ces modèles, on note en plus des transitions intra-états et vers l'état suivant, des sauts d'états autorisant l'absence d'un, ou plusieurs, graphèmes au sein d'un caractère. Toutefois, au moins une observation doit être associée à un tel modèle lettre. De cette façon, un mot de quatre lettres par exemple, devra comporter au minimum quatre fenêtres d'observations pour être représenté par l'un de ces trois modèles.

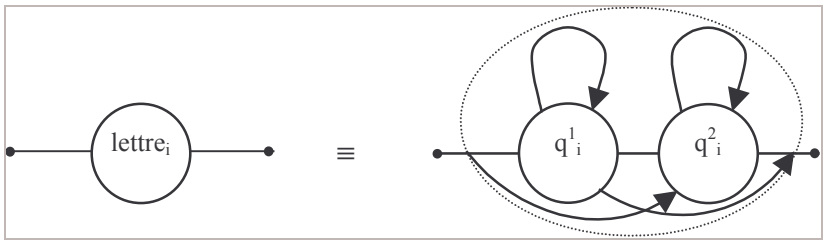


Figure 78. Modèle lettre deux états

De tels modèles gauche-droite sont très fréquemment utilisés pour modéliser l'écriture hors-ligne ou en-ligne [Rigoll, 1996 ; Wimmer 199 ;. Marti, 2000 ; Vincieralli, 2000 ;Viard-Gau, 2005]. L'intérêt d'une telle structure est qu'elle force les états du MMC à modéliser les différentes parties d'une lettre dans leur ordre d'apparition dans la séquence. Les variantes concernent le nombre d'états par lettre et la possibilité ou non de sauter des états. Généralement, ces choix sont faits de façon empirique à partir des résultats expérimentaux.

Le nombre d'états peut être :

- Identique pour chaque lettre,
- Variable et proportionnel au nombre moyen d'observations constituant cette lettre, ce nombre étant évalué sur une base d'apprentissage,
- Variable et déterminé par apprentissage pour optimiser les résultats de reconnaissance [Schamb., 2003].

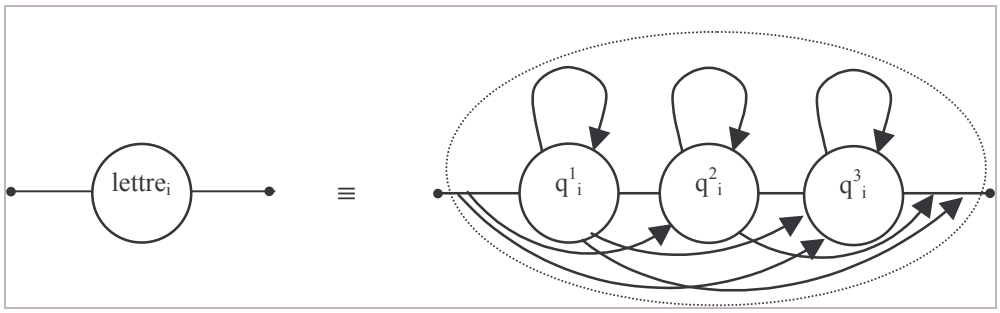


Figure 79. Modèle lettre trois états

Une extension à un nombre plus important d'états est envisageable. Toutefois, il peut être judicieux de limiter l'élasticité de tels modèles, en interdisant les sauts d'états (REMUS : modèle-lettre = 7 états, sans saut) ou en limitant leur portée, par exemple à 2 états sur la Figure 80.

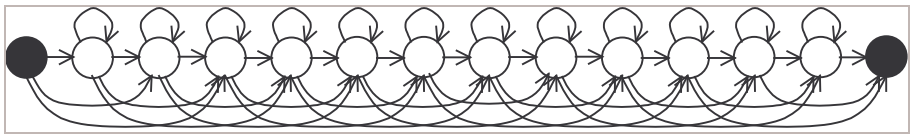


Figure 80. Modèle lettre N états avec 2 sauts

Toutefois, le nombre d'états-lettres augmentant, cela augmente d'autant le nombre de sorties du TDNN. Avec le modèle à trois états, cela représente 3 × 66 = 198 sorties pour ce réseau. Néanmoins, nous avons avec la solution retenue, un réseau de neurones unique qui partage ses poids sur les couches inférieures, à la différence, par exemple, de la solution proposée dans REMUS [Wimmer, 1999] où il y a autant de RN que d'états-lettres, chacun étant responsable de la prédiction des trames de l'état de la lettre auquel il est associé.

Dans ces modèles, les probabilités de transitions sont fixées unitaires de façon à ce qu'elles n'interviennent pas dans la détermination des coûts des chemins évalués. Autrement ces probabilités induiraient une loi de durée d'états décroissant exponentiellement ce qui ne correspond pas à un phénomène caractéristique de la production de l'écriture. Il est préférable de faire porter le résultat concernant la vraisemblance d'un mot uniquement sur les observations associées aux états.

6.3.2 Modèles de mots

Les modèles de mots correspondent à la concaténation des modèles de lettres qui les composent avec également des transitions unitaires pour le passage d'une lettre à la suivante. Aucune omission de lettre n'est autorisée, ce qui veut dire qu'au moins une observation devra être associée à chaque lettre. Cependant, dans le cas ou une lettre serait absente dans le signal écrit, compte tenu du fort taux de recouvrement entre les fenêtres d'observations, il ne serait pas trop pénalisant de dédier l'une des fenêtres pour absorber la lettre correspondante du modèle mot, et ainsi permettre la reconnaissance d'un mot mal orthographié.

La Figure 81 illustre la concaténation de trois mots « un », «quand » et « dépôt » avec un modèle lettre réduit à un seul état.

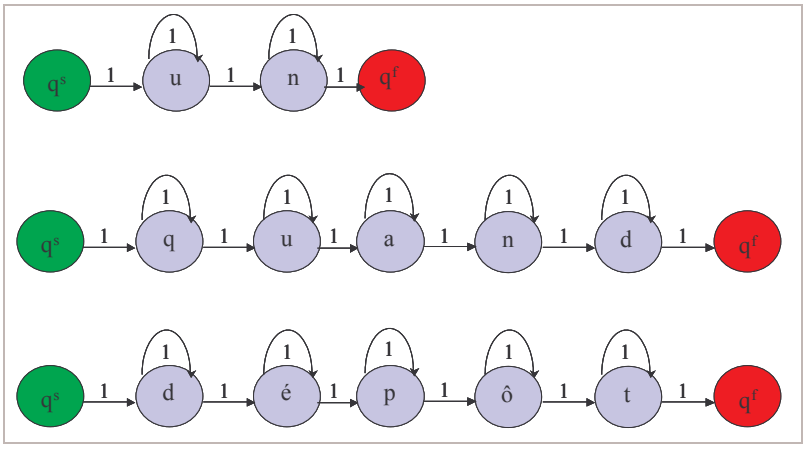


Figure 81. Représentation de modèles MMC mot des labels « un », « quand » et « dépôt » avec des modèles lettres fixés à un seul état

Plusieurs stratégies sont possibles pour traiter les caractères accentués et d'une manière générale les symboles diacritiques. Une solution consiste à détecter ces marques, à les retirer de la séquence, et à utiliser les caractères de base non accentués pour faire la reconnaissance, ceci

afin de s'affranchir de la variabilité temporelle de l'occurrence de ces marques [Marukatat, 2004]. Une autre solution serait d'associer un modèle spécifique aux marques diacritiques et de l'intercaler dans le modèle mot à chaque position possible d'une marque diacritique [Lallican, 1999]. Nous avons préféré considérer les caractères accentués comme étant des caractères spécifiques, modélisés indépendamment les uns des autres, c'est le cas dans l'exemple de Figure 81 des modèles lettre « é » et « ô » dans le mot dépôt. Cela explique les 66 éléments de notre ensemble de lettres. Une telle approche simplifie les prétraitements, et la construction des modèles mots mais est moins robuste à des marques décalées temporellement comme des accents tracés en fin de mot.

6.4 Processus de reconnaissance

Nous utilisons deux composants principaux dans notre système de reconnaissance : le réseau de neurones à convolution et la modélisation markovienne. Cependant l'outil essentiel est le TDNN, les MMC étant utilisés dans leur plus simple formalisme pour permettre d'effectuer simplement et efficacement l'alignement temporel par programmation dynamique.

La Figure 82 reprend le principe général du balayage temporel présenté à la Figure 71 et l'explicite sur l'exemple du mot « quand », en faisant apparaître notamment les sept composantes de la trame.

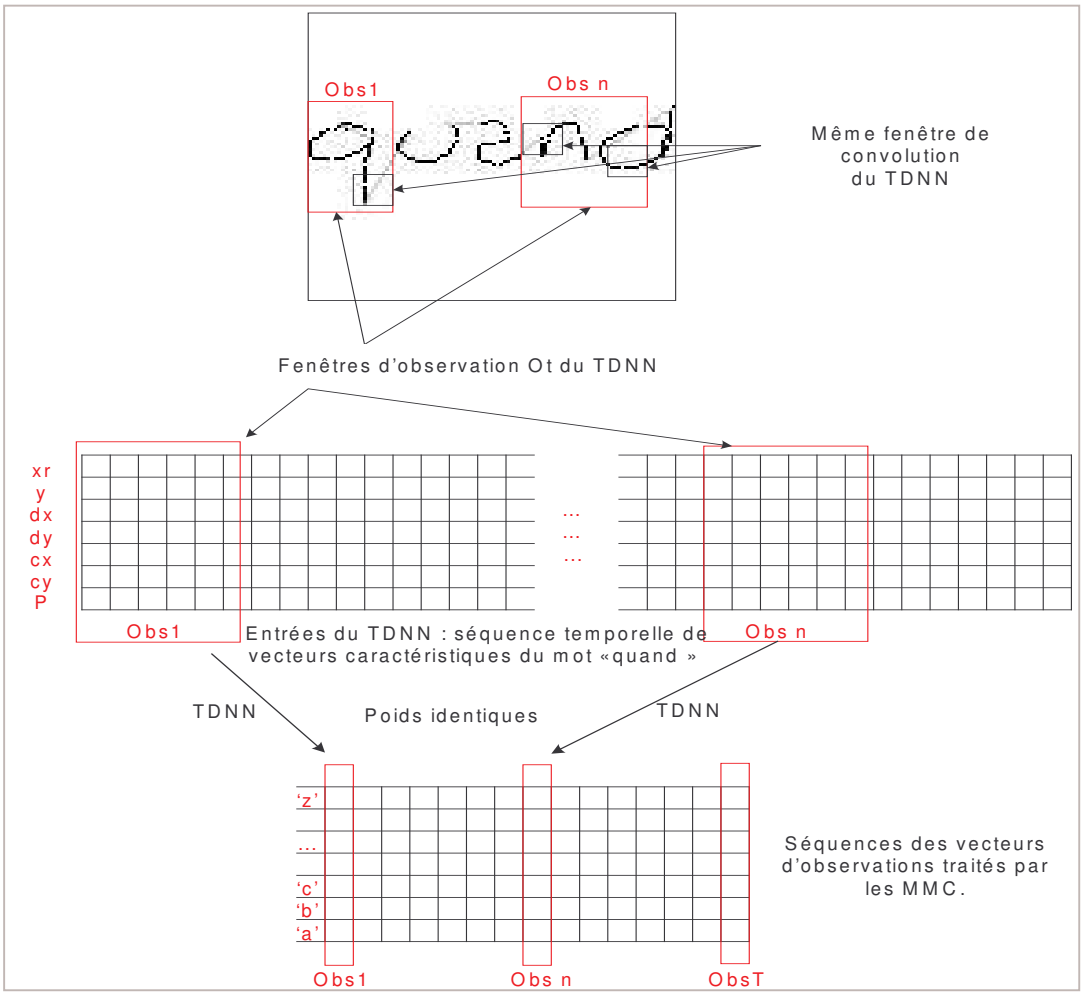


Figure 82. Balayage du TDNN par fenêtre d'observation de la trajectoire du mot « quand »

6.4.1 Balayage temporel

Le réseau est utilisé pour ses capacités d'approximateur universel [Bishop, 1995]. Il est entraîné par la méthode qui sera décrite au chapitre 7 pour estimer la densité de probabilité des observations dans chacun des états des MMC. En phase de reconnaissance, sur chaque position de la fenêtre glissante le long de la trajectoire, le TDNN fournira la vraisemblance d'une observation pour chacun des états-lettres.

La taille de la fenêtre d'observation est un paramètre important de ce système. Celle-ci doit contenir un nombre de points suffisant pour permettre d'estimer la probabilité qu'une lettre donnée ait généré, éventuellement partiellement, le contenu de cette fenêtre. Sa valeur a été fixée à partir de la distribution des longueurs de caractères après normalisation des mots en fonction de la hauteur d'oeil. La Figure 83 montre l'histogramme du nombre moyen de points par lettre. La longueur moyenne est de 32 points et l'écart type de 5.6 points. Nous avons fixé la taille de la fenêtre à 40 points afin d'englober dans la plupart des cas un caractère entier. Le décalage entre deux fenêtres consécutives doit correspondre à la taille du plus petit caractère, et par ailleurs doit être un multiple du délai des sous-fenêtres de convolution afin de pouvoir factoriser les calculs. Dans ces conditions, un pas de 10 points a été choisi entre deux observations.

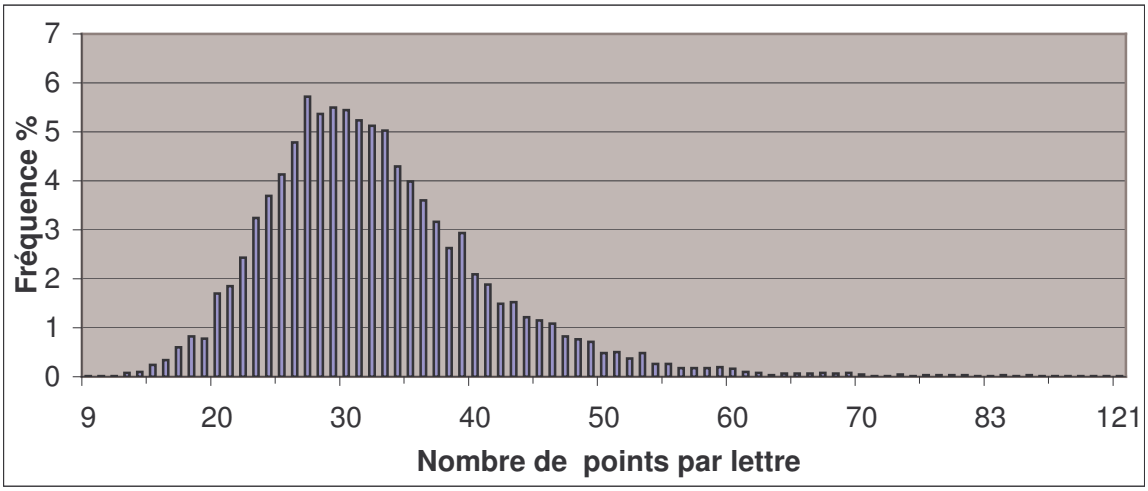


Figure 83. Histogramme des longueurs (en nombre de points) des caractères sur la base IRONOFF

Pour continuer les explications, nous allons choisir le mot « un » pour détailler la construction du treillis, ce choix est motivé par le nombre raisonnable d'observations à visualiser.

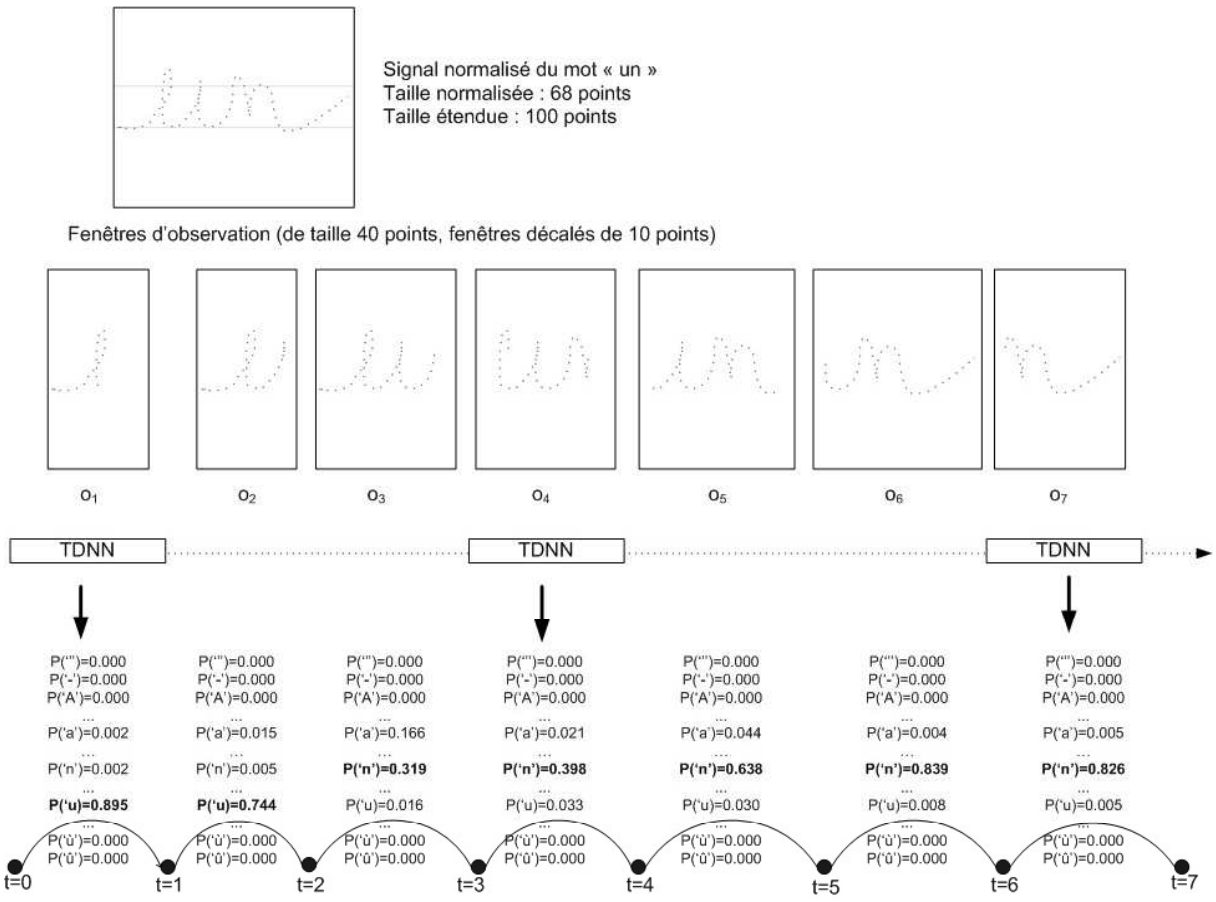


Figure 84. Propagation du mot « un » dans le TDNN. Le TDNN estime les probabilités d'observations pour chaque état

La Figure 84 illustre le signal normalisé d'une écriture du mot « un » à reconnaître. Ce signal a été échantillonné spatialement en 68 points, compte tenu de la hauteur des lignes de référence extraites, et est étendu en début et fin de trajectoire par simple duplication du point initial et du point final pour gérer les effets de bord. Ainsi on obtient sept fenêtres d'observations différentes vues par le TDNN. Pour faciliter la compréhension du système nous avons préféré afficher la trajectoire correspondante à l'observation au lieu de la séquence des 40 vecteurs de sept caractéristiques (cf. Figure 82) présentée au réseau. En tenant compte du recouvrement entre fenêtres d'observation et de la taille des sous-fenêtres de convolution, une optimisation de l'implantation permet de réduire la complexité de calcul en utilisant la redondance des informations (neurones) entre chaque fenêtre d'observation.

Ainsi, par exemple la topologie décrite dans l'encadré Figure 85 permet de s'affranchir de 1 320 propagations (par rapport à un nombre de 2 240 propagations sans factorisation) dans un neurone dans la partie extraction du réseau dans le cas de mot à 7 observations comme le mot « un ». Nous renvoyons à la partie reconnaissance de caractères isolés pour le détail de la propagation de l'entrée vers la couche de sortie d'un TDNN.

```
Réseau composé de :
   une partie extraction réalisée par un TDNN
   une partie classifieur réalisée par un MLP

Limites de l'architecture :
   Taille de mot maximale acceptée = 1000
   Nombre d'observations maximal du mot accepté = 100

   Taille de la fenêtre d'observation = 40
   Délai entre les fenêtres d'observations = 10

Architecture de la partie extraction réalisée par un TDNN
Extraction et traitement des caractéristiques en ligne
   nombre de couches du TDNN : 2
   nombre d'entrées selon t : 40
   nombre d'entrées selon f : 7
   soit un nombre total d'entrées : 280
   nombre de neurones de la couche cachée 1 selon t : 16
   nombre de neurones de la couche cachée (cc) 1 selon f : 20
   taille de la fenêtre selon t vus par les neurones de la cc 1 : 10
   délai selon t entre chaque fenêtre de la couche cachée 1 : 2

Réseau 2ème phase : classifieur par un MLP
   nombre de couches du MLP : 2
   nombre d'entrées de type en ligne : 320
   nombre de neurones en sortie du réseau (par observations): 66

Nombre de paramètres libres du réseau:
   MLP:    21186
   TDNN :  1420
Nombre total de paramètres libres: 22606
```

Figure 85. Caractéristiques de l'architecture typique

6.4.2 Alignement temporel

En sortie du TDNN on obtient un treillis de dimension : nombre de classes (état lettre) par nombre d'observations du signal d'entrée. Soit sur l'exemple de la Figure 86 : 66 × 7. Chaque élément correspond à la grandeur $b_j(o_t)$, c'est-à-dire la probabilité de l'observation o_t dans chacun des états. C'est à partir de ce graphe que sera calculée la vraisemblance de chaque mot du dictionnaire (contenant M mots différents, M=197 dans le cas retreint aux mots de la base IRONOFF). Pour cela, un modèle MMC-mot est construit de manière dynamique au moment de la reconnaissance par concaténation de modèles MMC-lettres.

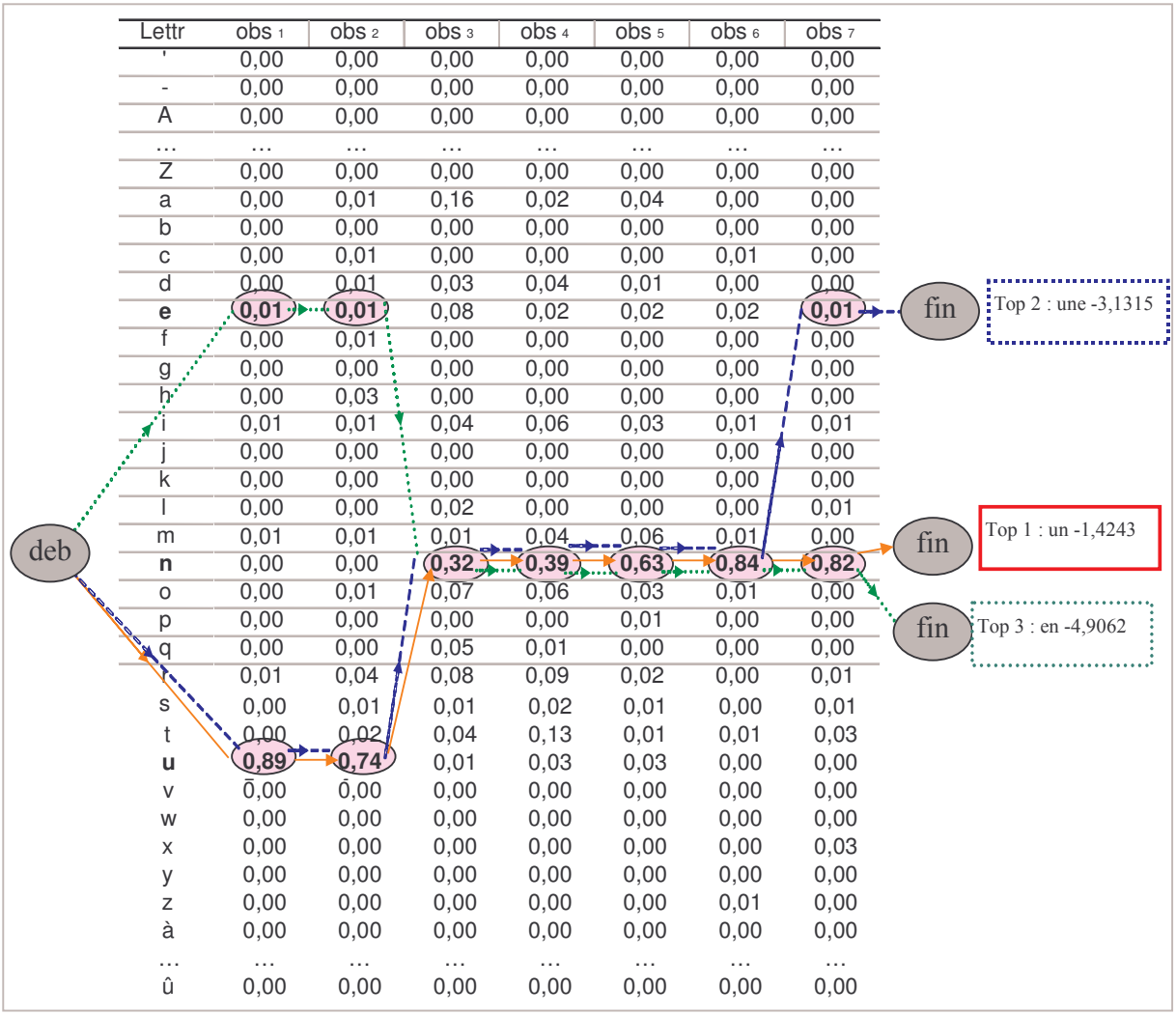


Figure 86. Graphe de reconnaissance en sortie du TDNN pour le signal du mot « un » où sont ajoutés les chemins Viterbi des trois mots de plus fortes vraisemblances (logarithme)

Il y a autant de classes en sorties du TDNN que le nombre d'états total représentant les MMC-lettres. Dans l'exemple présenté, un état unique est associé à chaque caractère, nous avons donc 66 sorties correspondant aux 66 caractères contenus dans la base IRONOFF.

6.4.3 Calcul de la vraisemblance

La vraisemblance $P(O \mid \lambda_i)$, de chaque MMC-mot λ_i sachant la séquence d'observations O, s'obtient en sommant les probabilités de toutes les séquences d'états qui permettent de générer O et présentes dans λ.

$$P(O \mid \lambda_i) = \sum_{\Gamma} \prod_{t=1}^{T} a_{q_{t-1}q_t} b_{q_t}(O_t)$$

Nous avons vu (chapitre 3, section 3.4.2) qu'elle se calculait de manière efficace par l'algorithme *Forward-Backward* ou encore par son approximation basée sur la conservation du seul chemin contribuant le plus à cette vraisemblance grâce à l'algorithme de *Viterbi*. Celui-ci permet en outre de fournir la séquence des états correspondant à ce chemin privilégié, et cela grâce à un simple *backtracking* additionnel.

Nous rappelons brièvement ici cet algorithme qui nous servira aussi lors de la phase d'apprentissage pour identifier les états affectés par le gradient de la fonction de coût.

Il nécessite le calcul en chaque nœud du treillis de deux variables : une variable δ qui correspond au plus haut score (plus forte probabilité) selon le chemin unique à l'instant *t* qui prend en compte les *t* premières observations et un état final *j* et une variable ψ qui mémorise l'index du meilleur état précédent et qui sera utilisée pour le *backtracking*.

$$\delta_t(j) = \max_{q_1, q_2, \cdots q_{t-1}} P(q_1 q_2 q_3 \cdots q_t = j, O_1 O_2 O_3 \cdots O_t \mid \lambda)$$

Par récursion on a $\delta_{t+1}(j) = [\max_i \delta_t(i) a_{ij}] \cdot b_j(O_{t+1})$.

L'algorithme de Viterbi suit la procédure donnée dans l'encadré suivant :

1. Initialisation

$$\delta_1(j) = \pi_j b_j(O_1), \quad 1 \le j \le N$$

$$\psi_1(j) = 0$$

2. Récursion

$$\delta_t(j) = \max_{1 \le i \le N} [\delta_{t-1}(i) a_{ij}] \cdot b_j(O_t), \quad 1 \le j \le N \text{ et } 2 \le t \le T$$

$$\psi_t(j) = \arg\max_{1 \le i \le N} [\delta_{t-1}(i) a_{ij}],$$

3. Terminaison

$$P^* = \max_{1 \le i \le N} [\delta_T(i)]$$

$$q_T = \arg\max_{1 \le i \le N} [\delta_T(i)]$$

3. Backtracking (calcul du meilleur chemin)

$$q_t = \psi_{t+1}(q_{t+1}) \qquad \text{pour } t = T-1, \ldots 1$$

Figure 87. Algorithme de Viterbi

Lors de la reconnaissance, ce module de programmation dynamique fournit en sortie une liste des candidats mots dans l'ordre décroissant du score obtenu. Le candidat reconnu en position 1 (noté Top1) est le mot avec le maximum de vraisemblance parmi les M modèles MMC_mot λ^*.

$$\lambda^* = \arg\max_i P(O \mid \lambda_i)$$

avec i=1,2,..., M le nombre total de mots du dictionnaire

La Figure 86 illustre les chemins Viterbi des trois mots reconnus de plus forte vraisemblance. Le mot « un » a été reconnu comme le mot le plus probable, ensuite le mot « une» puis en troisième position le mot « en », avec un score exprimé comme le logarithme de la vraisemblance respectivement TOP1 : -1.424378, TOP2 : -3.131501, TOP3 : -4.906297.

6.5 Conclusion

Dans ce chapitre, nous avons décrit le processus de reconnaissance. Le système est principalement décomposé en quatre étapes : la normalisation du signal, l'extraction de caractéristiques, le calcul des probabilités d'observations selon les états des MMC-lettres et l'alignement temporel pour arriver au niveau de reconnaissance mot.

Les étapes de prétraitement servent à éliminer les variations perturbatrices de l'écriture pour favoriser une reconnaissance omniscripteur. Pour cela, un algorithme de type EM (Expectation-Maximization) utilisant les extrema du contour du mot cherche à optimiser la position et l'orientation de quatre lignes parallèles. Ces lignes permettent de normaliser le mot en orientation, en taille et de caractériser implicitement les extensions hautes et basses de l'écriture selon les règles primaires de production de l'écriture. Ensuite une procédure de ré-échantillonnage spatial par interpolation linéaire du tracé est mise en œuvre pour s'affranchir des variabilités de vitesses d'écriture. Une suite de vecteurs de sept paramètres, appelés caractéristiques, est extraite du tracé point par point.

Un mot manuscrit est donc représenté par une séquence temporelle, de longueur variable en fonction de la taille du mot, chaque élément de la séquence étant un vecteur de sept composantes. Cette séquence va être balayée par le réseau de neurones avec une fenêtre de taille fixe, pour laquelle il calculera un vecteur $b_j(O_t)$ correspondant à la probabilité d'observer cette fenêtre dans le $j^{ème}$ état-lettre.

Nous avons choisi une structure particulière du classifieur : un réseau à délai. A l'intérieur de chaque fenêtre, des fonctions de convolution sur des sous-fenêtres glissantes à poids partagés sont opérées. Cette topologie permet de prendre en compte les aspects locaux de l'écriture et d'extraire des informations locales pour aller d'une vision locale vers une vision globale du signal. Le treillis temporel obtenu en sortie du TDNN est ensuite utilisé pour déterminer le mot de plus forte vraisemblance.

Nous avons souligné l'intérêt d'une telle structure hybride, TDNN suivi de MMC :

1. Réduction du nombre de paramètres libres par rapport à des approches de type perceptron multicouche classique ou machines à vastes marges.
2. Réduction de la complexité calculatoire en exploitant la redondance de la fenêtre de balayage du signal.
3. Segmentation implicite du mot en lettres, voire graphèmes grâce à la fenêtre d'observations glissante sur laquelle est calculée l'estimation des probabilités d'observations des entités élémentaires.
4. Modélisation markovienne réduite à la programmation dynamique.

Le point trois est un point important de la méthode proposée. Pour à la fois, simplifier le processus d'apprentissage, et améliorer les taux de reconnaissance au niveau mot, nous avons opté pour un apprentissage complètement global du système. De cette façon, il n'est pas nécessaire de disposer d'une base de mots étiquetée au niveau caractère. En effet, avec les systèmes hybrides classiques, il convient soit de disposer d'un système annexe de reconnaissance permettant d'initialiser le système de reconnaissance pour permettre la segmentation et l'étiquetage en caractères (post-segmentation par l'algorithme de Viterbi) soit

d'entraîner le réseau sur une base de caractères isolés qui ne peut être très représentative de l'écriture cursive. De plus, le comportement d'un réseau de neurones vis-à-vis des formes non apprises (hypothèse de segmentation ne correspondant pas à un vrai caractère) est toujours un problème délicat. Ici, il n'y a pas explicitement d'apprentissage au niveau caractère, en ce sens le réseau ne serait pas un bon reconnaisseur de caractères, il n'est pas supposé calculer les probabilités *a posteriori* des caractères, mais il a été entraîné pour satisfaire une fonction objectif définie au niveau global des mots. Nous allons détailler dans le chapitre suivant les fonctions de coût usuelles dans la littérature et présenter une fonction de coût générique qui permet d'effectuer un apprentissage global de ce système hybride neuro-markovien.

7 SCHEMAS D'APPRENTISSAGE

Dans ce chapitre, nous allons détailler le processus d'apprentissage unifié du système hybride neuro-markovien décrit dans le chapitre précédent. Nous présentons dans un premier temps le schéma général de l'apprentissage d'un tel système hybride. Nous définirons notre contribution : un critère générique d'optimisation des paramètres du réseau de neurones à délai sans initialisation préalable au niveau caractère. Nous déclinerons les paramètres de ce critère dans le but de comparer les méthodes classiques de type maximisation de vraisemblance (ML : Maximum Likelihood), maximum d'information mutuelle (MMI : Maximum Mutual Information) et la méthode proposée. Nous comparerons les propriétés et résultats des différentes méthodes d'apprentissage.

7.1 Schéma général de l'apprentissage unifié

Dans le chapitre 3, nous avons décrit les différentes possibilités de coupler des réseaux de neurones à des modèles de Markov. Nous avons argumenté le choix d'un apprentissage unifié d'un réseau de neurones, utilisé comme estimateur des densités de probabilités des observations pour les différents états-lettres, chacune étant fournie par une sortie du RN, suivi de modèles de Markov pour modéliser les séquences d'observations d'un signal mot.

L'utilisation des modèles de Markov étant restreinte à la programmation dynamique pour déterminer le mot le plus vraisemblable dans le lexique par rapport à la séquence d'observations en entrée du système, les transitions des modèles MMC-lettres sont considérées constantes. Il en est de même pour les modèles MMC-mots, chaque état final des MMC-lettres le composant est relié par une transition de poids unitaire à l'état initial du modèle MMC-lettre suivant. Entraîner le système revient à un apprentissage itératif du réseau de neurones selon un critère donné. A chaque itération, on traite tous les exemples mots de la base d'apprentissage.

La procédure d'apprentissage de base est résumée dans l'algorithme suivant.

```
Initialisation aléatoire (contrôlée) des poids du réseau
Tant que le critère d'arrêt n'est pas satisfait
  Pour chaque exemple de la base d'apprentissage e
    Soit e=(Mvrai,O) avec Mvrai le label mot correspondant et O le
    signal correspondant
    Calcul de T le nombre d'observations du signal selon la taille
    de la fenêtre de balayage du TDNN et du nombre de points du
    signal
    Normalisation et extraction des caractéristiques en entrée du
    réseau O=O1…OT,
    Propagation des entrées du réseau dans le TDNN
    Obtention des bj(Ot) : séquences des vecteurs de probabilités
    des observations
    Construire le MMC du mot Mvrai, soit MMC(Mvrai)
    Calcul de la segmentation optimale par Viterbi λMMCvrai
    Calcul du critère L et du gradient associé (∂L/∂W)
    Ré estimation des paramètres W du TDNN.
  Fin Pour
Fin Tant que.
```

Algorithme 8. Principes de base de l'apprentissage du système hybride

Pour chaque exemple mot de la base d'apprentissage, on normalise le signal O et on extrait une séquence de sept caractéristiques par point que balayera le TDNN. On construit le modèle du mot correspondant à son label vrai (connu car en mode apprentissage) en concaténant les modèles lettres le composant. Puis on détermine la segmentation optimale du signal dans ce modèle que l'on note $\lambda_{MMCvrai}$ à l'aide d'une programmation dynamique et des paramètres courants du système. Le critère d'erreur sera alors calculé, ainsi que le gradient de l'erreur par rapport aux poids du réseau. Pour un même exemple, la correction des poids peut être effectuée en une fois ou bien à chaque trame o_t, avec une nouvelle propagation des entrées et programmation dynamique selon les nouveaux paramètres du système. Cette dernière est coûteuse en calcul puisqu'elle redemande autant de rétropropagations et d'alignements temporels que de nombres d'observations T du signal. Nous avons vérifié par expérimentation qu'une rétropropagation globale avec un cumul des erreurs pour chaque élément de la trame o_t ne dégradait pas la convergence du système et évitait des perturbations trop importantes par rapport à des observations ne correspondant visuellement à aucun caractère particulier. Les équations du calcul de la matrice de gradient qui suivront tiendront compte de ce choix, et cela quel que soit le critère choisi. Nous allons maintenant énoncer le critère générique proposé et détailler le processus d'apprentissage associé.

7.2 Ecriture du critère

Rappelons un des objectifs essentiels de notre contribution : entraîner le réseau de neurones avec une fonction objectif définie globalement au niveau mot, et non pas caractère, en faisant coopérer une approche de type modèle générateur (maximisation de la vraisemblance du modèle mot correspondant) et une de type approche frontière (discrimination entre les classes). Pour cela, nous avons construit une fonction de coût paramétrique L_G au niveau mot ayant trois paramètres $\alpha, \beta,$ et ε combinant des critères de maximum de vraisemblance (ML) et de discrimination tels que le maximum d'information mutuelle (MMI) et le minimum d'erreur de classification (MCI).

Le critère proposé peut s'écrire de la manière suivante :

Soit O le signal mot en entrée balayé par le TDNN en T observations o_t

$$L_G = (1+\varepsilon)\log P(O|\lambda_{MMCvrai}) - \beta\left[(1-\alpha)\log P\left(O\middle|\lambda_{MMCreconnu}\right) + \alpha \log P\left(O\middle|\lambda_{TDNNreconnu}\right)\right]$$

$$avec\ MMCreconnu = \arg\max_i P(O|\lambda_i),\ \lambda_i \text{ modèle mot du lexique}$$

$$TDNNreconnu = \{Classe_{i,t} \mid i = \arg\max_j b_j(o_t)\}_{t=1\ à\ T}$$

Les paramètres $\alpha, \beta,$ et ε seront compris entre 0 et 1. Le Tableau 24 illustre les variantes possibles du critère que nous détaillerons et comparerons précisément dans la suite. Avec $\varepsilon = \beta = 0$, nous obtenons le critère classique du maximum de vraisemblance (MLE). Avec $\beta = 1$, on introduit un apprentissage discriminant (MMI simplifié) qui si $\alpha = 0$ prend en compte uniquement le mot reconnu en 1ère position par l'algorithme de Viterbi ($\lambda_{MMCreconnu}$) et, dans le cas où $\alpha = 1$ seulement les classes reconnues en 1ère position en sortie du TDNN ($\lambda_{TDNNreconnu}$). La valeur α permet de pondérer ces deux derniers types de discrimination.

Tableau 24. Paramètres en fonction des critères

Paramètres/Critères	ε	β	α
(1) MLE	0	0	0
(2) MMIs	0	1	0
(3) MLE + MMIs	1	1	0
(4) MLE + TDNN	1	1	1
(5) Mixte	0..1	0..1	0..1

7.3 Calcul de la matrice du gradient associé au critère

Nous allons décrire dans le cadre le plus générique l'étape de calcul du critère et de la matrice de gradient associée. La Figure 88 reprend le schéma général et montre le positionnement de la boucle d'apprentissage.

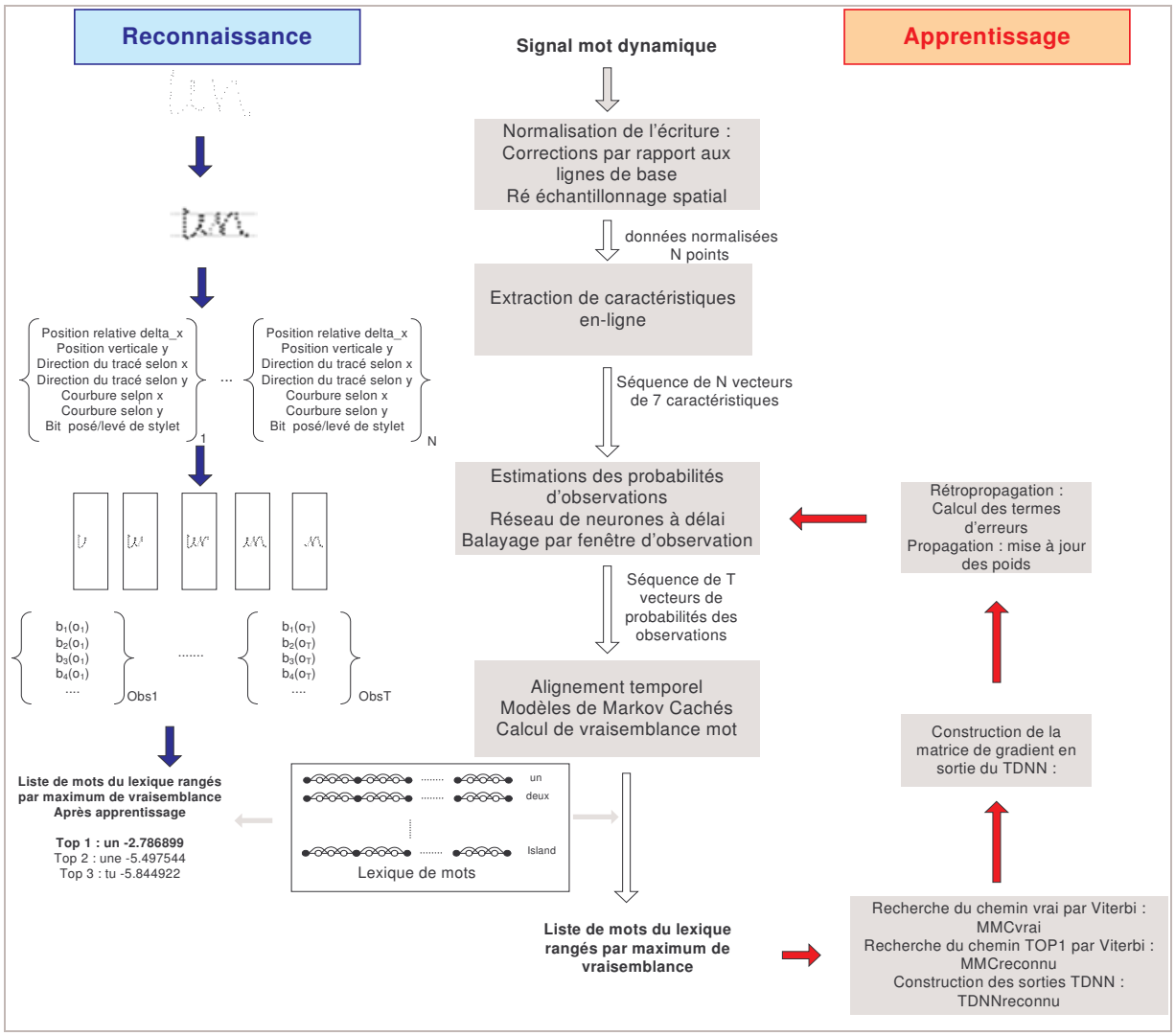


Figure 88. Mise en évidence du processus d'apprentissage en rouge

Détaillons les calculs correspondants à la ligne 11 de l'algorithme général (algorithme 8), soit le calcul du critère et du gradient associé. Pour chaque mot Mi du dictionnaire, on calcule par programmation dynamique son score de vraisemblance. On mémorise le chemin, obtenu par l'algorithme de Viterbi, du mot ayant le maximum de vraisemblance, noté MMCreconnu.

$$MMCreconnu = \arg\max_i P(O \mid \lambda_i),\ \lambda_i \text{ modèle mot du lexique}$$

On construit ensuite le chemin des sorties TDNN, noté TDNNreconnu qui correspond à la concaténation des classes de plus fortes probabilités d'appartenance.

$$TDNNreconnu = \{Classe_{i,t} \mid i = \arg\max_j b_j(o_t)\}_{t=1 \text{ à } T}$$

Ainsi, dans le cas de l'apprentissage du mot « un », le Tableau 25 donne un exemple des différents chemins considérés et les probabilités des observations des classes associées à une itération donnée. Le chemin du mot vrai « un » par Viterbi correspond à la séquence suivante « u u u n n », le mot du lexique de plus forte vraisemblance a pour label « en » et son chemin selon Viterbi est « e e n n n ». Les sorties TDNN donnent la séquence « i e n m n » qui ne correspond à aucun mot réel du dictionnaire.

Tableau 25. Détail des sorties du TDNN et états des mots avec et sans lexique

Chemin Vrai par Viterbi (mot vrai : un)	1	2	3	4	5
	u	u	u	n	n
	7.6	30.7	37.2	39.3	65.2
Chemin Reconnu par Viterbi (mot reconnu : en)	e	e	n	n	n
	24.1	49.4	57.7	39.3	65.2
Sorties Best-TDNN (« mot TDNN» : ieumn)	i	e	n	m	n
	52.1	49.4	57.7	46.1	65.2

A partir de ces chemins on peut construire la matrice de gradient de la fonction L_G par rapport aux poids du réseau. Elle peut être déterminée par le développement en chaîne suivant :

$$L_G = (1+\varepsilon)\log P(O|\lambda_{HMMvrai}) - \beta\left[(1-\alpha)\log P\left(O|\lambda_{HMMreconnu}\right) + \alpha \log P\left(O|\lambda_{TDNNreconnu}\right)\right]$$

Mise à jour des poids par descente de gradient

$$W_{ji}^{apres} = W_{ji}^{avant} + \mu * \frac{\partial L_G}{\partial W_{ji}} \qquad \text{avec } \mu \text{ pas de descente du gradient}$$

$$\frac{\partial L_G}{\partial W_{ji}} = \sum_t \frac{\partial L_G}{\partial v_j(O_t)} \cdot \frac{\partial v_j(O_t)}{\partial W_{ji}} \qquad v_{j,t} = v_j(O_t) \text{ et } x_{j,t} = x_j(O_t)$$

Où *j* est l'indice du neurone concerné et i celui du neurone associé à la couche inférieure, t l'indice temporel de l'observation et $v_j(O_t)$ le potentiel synaptique du neurone j pour l'observation t ; $x_j(O_t) = f(v_j(O_t))$ la sortie du neurone j.

Le second terme se calcule simplement par la formule suivante:

$$\frac{\partial v_{j,t}}{\partial w_{ji}} = \frac{\partial}{\partial w_{ji}}\left(\sum_k w_{jk}.x_{k,t}\right) = x_{i,t}$$

$$soit\ \frac{\partial Ls}{\partial w_{ji}} = \sum_t x_{i,t}.\frac{\partial Ls}{\partial v_{j,t}}$$

On introduit le terme d'erreur $\delta_{j,t}$ qui dépend de la couche du réseau concernée.

$$\delta_{j,t} = \frac{\partial Ls}{\partial v_{j,t}}$$

On calcule et mémorise les termes $\delta_{j,t}$ de la couche de sorties du TDNN. Les neurones de sorties sont liés par la fonction de Softmax, notée f_{soft}, qui permet d'obtenir des valeurs de probabilités (entre 0 et 1).

$$x_{j,t} = f_{soft}(v_{j,t}) = \frac{e^{v_{j,t}}}{\sum_k e^{v_{k,t}}}$$

$$\delta_{j,t} = \frac{\partial Ls}{\partial v_{j,t}} = \sum_k \frac{\partial Ls}{\partial x_{k,t}}.\frac{\partial x_{k,t}}{\partial v_{j,t}}$$

$$\delta_{j,t} = \frac{\partial Ls}{\partial x_{j,t}}.\frac{\partial x_{j,t}}{\partial v_{j,t}} + \sum_{k\neq j} \frac{\partial Ls}{\partial x_{k,t}}.\frac{\partial x_{k,t}}{\partial v_{j,t}}$$

La fonction Softmax a une dérivée intéressante qui s'écrit en fonction de celle-ci.

$$x_{j,t} = f_{soft}(v_{j,t}) = \frac{e^{v_{j,t}}}{\sum_k e^{v_{k,t}}} \rightarrow \frac{\partial x_{j,t}}{\partial v_{j,t}} = x_{j,t}\left(1 - x_{j,t}\right)$$

$$Et\ \frac{\partial x_{k,t}}{\partial v_{j,t}} = \frac{\partial}{\partial v_{j,t}}\left(\frac{e^{v_{k,t}}}{\sum_k e^{v_{i,t}}}\right) = \frac{1}{\left(\sum_i e^{v_{i,t}}\right)^2}\left[\frac{\partial e^{v_{k,t}}}{\partial v_{j,t}}.\sum_i e^{v_{i,t}} - e^{v_{k,t}}.\sum_i \frac{\partial e^{v_{i,t}}}{\partial v_{j,t}}\right]$$

$$or\ \frac{\partial e^{v_{k,t}}}{\partial v_{j,t}} = 0 \text{ et } \frac{\partial x_{k,t}}{\partial v_{j,t}} = \frac{1}{\left(\sum_i e^{v_{i,t}}\right)^2}\left(-e^{v_{k,t}}.e^{v_{j,t}}\right) = -x_{k,t}.x_{j,t}$$

D'où :

$$\delta_{j,t} = \frac{\partial Ls}{\partial x_{j,t}} . x_{j,t}\left(1 - x_{j,t}\right) - \sum_{k \neq j} x_{k,t} . x_{j,t} . \frac{\partial Ls}{\partial x_{k,t}}$$

$$\delta_{j,t} = x_{j,t}\left[\frac{\partial Ls}{\partial x_{j,t}} - \sum_{\forall k} x_{k,t} . \frac{\partial Ls}{\partial x_{k,t}}\right]$$

Reste à calculer le terme : $\frac{\partial Ls}{\partial x_{j,t}}$

$x_j(O_t)$ est obtenu en sortie du réseau de neurones et correspond à la probabilité de l'observation O_t dans l'état j à l'observation t. Cette grandeur est notée par $b_j(O_t)$ dans la notation MMC $\lambda(A, B, \pi)$.

$$\frac{\partial Ls}{\partial x_{j,t}} = \frac{\partial Ls}{\partial b_j(O_t)} = \frac{\partial}{\partial b_j(O_t)}\left[(1+\varepsilon)\log P(O|\lambda_{HMMvrai}) - \beta\left[(1-\alpha)\log P\left(O|\lambda_{HMMreconnu}\right) + \alpha \log P\left(O|\lambda_{TDNNreconnu}\right)\right]\right]$$

$$\frac{\partial Ls}{\partial x_{j,t}} = \frac{1+\varepsilon}{P(O|\lambda_{MMCvrai})} . \frac{\partial P(O|\lambda_{MMCvrai})}{\partial b_j(O_t)} - \frac{\beta(1-\alpha)}{P(O|\lambda_{MMCreconnu})} . \frac{\partial P(O|\lambda_{MMCreconnu})}{\partial b_j(O_t)} - \frac{\beta\alpha}{P(O|\lambda_{TDNNreconnu})} . \frac{\partial P(O|\lambda_{TDNNreconnu})}{\partial b_j(O_t)}$$

$$Or\ P(O|\lambda) = \sum_{\Gamma}\left(\prod_{t=1}^{T} a_{q_{t-1}} . b_{q_t}(O_t)\right)$$

avec Γ tous les chemins possibles du MMC mot $\lambda(A, B, \pi)$

qui se réduit en considérant des transitions unitaires et l'approximation de Viterbi à :

$$P(O|\lambda) = \prod_{t=1}^{T} b_{q_t}(O_t)$$

D'où :

$$\frac{\partial P(O|\lambda)}{\partial b_j(O_t)} = \frac{\partial}{\partial b_j(O_t)}\left(\prod_{t=1}^{T} b_{q_t}(O_t)\right)$$

$$\frac{\partial P(O|\lambda)}{\partial b_j(O_t)} = \begin{cases} 0 \text{ si } q_t \neq j \\ \prod_{t=1, t\neq j}^{T} b_{q_t}(O_t) \text{ sinon} \end{cases}$$

$$\frac{\partial P(O|\lambda)}{\partial b_j(O_t)} = \begin{cases} 0 \text{ si } q_t \neq j \\ \frac{\prod_{t=1}^{T} b_{q_t}(O_t)}{b_j(O_t)} = \frac{P(O, q_t = j|\lambda)}{b_j(O_t)} \end{cases}$$

Par conséquent :

$$\frac{\partial Ls}{\partial x_{j,t}} = \frac{1}{x_{j,t}}\left[(1+\varepsilon)\frac{P(O,q_t=j\mid\lambda_{MMCvrai})}{P(O\mid\lambda_{MMCvrai})} - \beta(1-\alpha)\frac{P(O,q_t=j\mid\lambda_{MMCreconnu})}{P(O\mid\lambda_{MMCreconnu})} - \beta\alpha\frac{P(O,q_t=j\mid\lambda_{TDNNreconnu})}{P(O\mid\lambda_{TDNNreconnu})}\right]$$

$$\text{En notant } \text{Grad}_{j,t} = (1+\varepsilon)\frac{P(O,q_t=j\mid\lambda_{MMCvrai})}{P(O\mid\lambda_{MMCvrai})} - \beta(1-\alpha)\frac{P(O,q_t=j\mid\lambda_{MMCreconnu})}{P(O\mid\lambda_{MMCreconnu})} - \beta\alpha\frac{P(O,q_t=j\mid\lambda_{TDNNreconnu})}{P(O\mid\lambda_{TDNNreconnu})}$$

On obtient : $\delta_{j,t} = Grad_{j,t} - x_{j,t}\sum_k Grad_{k,t}$

$P(O,q_t=j|\lambda)$ est calculé par programmation dynamique. Ainsi pour chaque observation O_t, un gradient positif est rétropropagé pour le vrai modèle MMC (dont une fraction ε correspond à un pur critère ML) et un gradient négatif pour le modèle MMC reconnu et/ou pour les classes reconnues par le TDNN.

Le Tableau 26 illustre pour différents critères, critères classiques - MLE et MMI simplifié, générique - les valeurs prises par la variable $Grad_{j,t}$ en fonction de la sortie du réseau *x(j,t)* et l'appartenance de son état correspondant *q(j,t)* aux différents chemins : faux (F) signifie que le chemin ne passe par l'état q_t = j du MMC en cette observation *t* et vrai (V) le contraire. Le MMC vrai correspond au chemin Viterbi du modèle vrai connu, le MMC reconnu au chemin du mot du lexique avec la plus grande vraisemblance et le TDNN reconnu aux sorties de plus forte probabilité. Une normalisation du gradient entre -1 et 1 est ensuite réalisée pour assurer la convergence du réseau.

Tableau 26. Calcul du gradient selon l'appartenance de l'état à un chemin spécifique (F=faux, V=vrai) et les différents critères

q(j,t) = MMC Vrai (j,t)	q(j,t) = MMC reconnu (j,t)	q(j,t) = TDNN reconnu (j,t)	$Grad_{j,t}$				
			MLE (1) (ε=0, β=0)	MMIs (2) (ε=0, β=1, α=0)	MLE+ MMIs (3) (ε=1, β=1 , α=0)	MLE+ TDNN (4) (ε=1, β=1, α=1)	Générique (5) (ε, β , α)
F	F	F	*0*	*0*	*0*	*0*	*0*
F	F	V	*0*	*0*	*0*	*–1*	*–βα*
F	V	F	*0*	*–1*	*–1*	*0*	*–β(1–α)*
F	V	V	*0*	*–1*	*–1*	*–1*	*–β*
V	F	F	*1*	*1*	*2*	*2*	*1+ε*
V	F	V	*1*	*1*	*2*	*1*	*1+ε–βα*
V	V	F	*1*	*0*	*1*	*2*	*1+ε–β(1–α)*
V	V	V	*1*	*0*	*1*	*1*	*1+ε–β*

Pour les couches cachées du réseau, on utilise aussi les dérivées en chaîne pour calculer $\delta_{j,t}$ On applique l'algorithme de rétro propagation classique, la seule difficulté réside dans la considération des neurones des fenêtres de convolution du TDNN.

$$\delta_{j,t} = \frac{\partial Ls}{\partial v_{j,t}} = \sum_k \frac{\partial Ls}{\partial v_{k,t}} \cdot \frac{\partial v_{k,t}}{\partial v_{j,t}} = \sum_k \delta_{k,t} \cdot \frac{\partial v_{k,t}}{\partial v_{j,t}}$$

Et $\frac{\partial v_{k,t}}{\partial v_{j,t}} = \frac{\partial v_{k,t}}{\partial x_{j,t}} \cdot \frac{\partial x_{j,t}}{\partial v_{j,t}} = \frac{\partial v_{k,t}}{\partial x_{j,t}} . \quad fsig'\left(v_j, t\right)$

avec $fsig$ la fonction sigmoïde et $fsig'$ sa dérivée

$$\frac{\partial v_{k,t}}{\partial x_{j,t}} = \frac{\partial}{\partial x_{j,t}} \left(\sum_l w_{kl} . x_{l,t} \right) = w_{kj}$$

$$\frac{\partial v_{k,t}}{\partial x_{j,t}} = w_{kj}$$

d'où $\delta_{j,t} = \sum_k \delta_{k,t} . w_{kj} . f'\left(v_{j,t}\right)$

Après avoir rétropropagé les termes d'erreurs de la couche de sortie vers la couche d'entrée du TDNN on peut corriger les poids du réseau dans le sens de la propagation.

Dans le cadre de l'exemple de l'apprentissage du mot « un », la matrice $Grad_{j,t}$ est construite de la façon suivante :

Tableau 27. Calcul de la matrice $Grad_{j,t}$ sur l'exemple présenté Tableau 25

Lettres	Obs 1	Obs 2	Obs 3	0bs 4	0bs 5
e	- β(1-α)	- β(1-α)-αβ	0	0	0
i	-βα	0	0	0	0
m	0	0	0	-βα	0
n	0	0	- β	1+ε - β(1-α)	1+ε-β(1-α)-αβ
u	1+ε	1+ε	1+ε	0	0
Autres états	0	0	0	0	0
Somme	**1+ε-β**	**1+ε-β**	**1+ε-β**	**1+ε-β**	**1+ε-β**

7.4 Déclinaison du critère

Dans cette section nous allons mettre en œuvre les approches d'apprentissage présentées ci avant, en commençant par la plus classique, pour aboutir sur le cas générique proposé, en montrant les intérêts et limitations de chacune. Nous argumenterons à chaque fois le fil conducteur du critère proposé.

7.4.1 Base d'expérimentation : base IRONOFF

Les taux de performance cités dans la suite de cette partie, sont obtenus à partir de la base de mots en-ligne IRONOFF. Elle comprend 31346 mots issus des différents formulaires B, C, E, F et G. Ceux-ci forment globalement un dictionnaire de 197 mots, listés dans le Tableau 27.

Tableau 28. Dictionnaire de mots d'IRONOFF

un	centimes	fjord	fredonner	Urgence	T-shirt	du	votre
deux	euros	dégâts	moissonner	Vacances	User	des	leur
trois	et	jazz	polygonale	Week-end	Voice	dans	entre
quatre	frs	buggy	père-noël	Xénophobie	Warehouse	en	on
cinq	cts	impôts	frapperions	Yaourt	X-ray	par	sur
six	repêché	conçu	Agglomération	Zénith	Yuppie	chez	sous
sept	blâmez	aïeux	Boîtier	Apple	Zero	pour	plus
huit	affût	fonça	Citoyen	Between	je	le	moins
neuf	l'élève	galette	Démocratie	Capability	tu	la	avec
dix	rugby	flûte	Encouragemen	Directory	il	les	ainsi
onze	jusque	accident	t	Earth	elle	ce	qui
douze	chômé	abbaye	Fréquence	Fuzzy	nous	cet	que
treize	vodka	éclabousser	Gymnase	Giving	vous	cette	quoi
quatorze	gîtes	déposerait	Hôpital	Hydrogen	mais	ces	quel
quinze	whisky	thermonucléai	Imperméable	Island	où	cela	quelle
seize	oeuvre	re	Journal	Job	donc	ceci	quand
vingt	voilà	sculpterai	Kiosque	Ku-Klux-	or	celle	tout
trente	zèbre	organisme	Littérature	Klan	ni	celui	tous
quarante	dépôt	secouraient	Maître	Liberty	car	mon	aussi
cinquante	quelqu'un	monétaires	Neptune	Money	puis	ton	dont
soixante	vêtir	malversation	Occident	North	ne	son	dès
cent	gâchez	pédalerions	Psychologue	Obvious	pas	si	autre
mille	figeront	compagnies	Quittance	Parking	à	une	
million	buvez	pivoteras	République	Quiz	au	même	
		surgelées	Société	Rabbit	de	notre	
francs	taxis	fréquemment		Smooth			
			Température				

Notons que ce dictionnaire fait intervenir des lettres minuscules et majuscules, ainsi que des apostrophes et des tirets. Les accents et signes diacritiques {´ , ` , ^} particuliers à la langue française, sont également bien représentés. Beaucoup de mots courts présentent un grand risque de confusion : « nous » et « vous », « donc » et « dont », « des » et « dès », « cette » et « celle », « tout » et « tous »…

Afin d'évaluer des taux de performances, la base est divisée en deux : une base d'apprentissage comprenant les deux tiers des exemples soit un total de 20 898 mots, et une base de test comprenant le reste (10 448 mots).

7.4.2 Critère de maximum de vraisemblance

Le critère d'apprentissage par maximum de vraisemblance est le plus utilisé en reconnaissance de mots en-ligne dans les systèmes hybrides et le plus simple à mettre en œuvre. L'algorithme associé de rétro propagation classique au niveau mot est le suivant :

```
Tant que le critère d'arrêt n'est pas satisfait
Nouvelle itération de la base d'apprentissage
    Mise à jour du pas de gradient
    Rangement aléatoire des exemples
Pour chaque exemple :
    Prétraitement de l'exemple
    Extraction des caractéristiques
    Propagation des caractéristiques dans le TDNN
    Recherche de l'indice du mot vrai dans la table des mots du
dictionnaire
Algorithme de Viterbi, Sauvegarde du chemin vrai λMMCvrai
```

```
Calcul du gradient correspondant à λMMCvrai
Rétro propagation de l'erreur
Mise à jour des poids du TDNN
Fin pour chaque exemple
Fin tant que
```

Algorithme 9. Algorithme de rétro propagation – Cas du Maximum de Vraisemblance

L'arrêt de l'apprentissage du système hybride est soit défini par un nombre fixé d'itérations soit par une mesure d'évolution des performances de la reconnaissance. L'obtention de la convergence et sa rapidité est liée au choix du pas de gradient des poids.

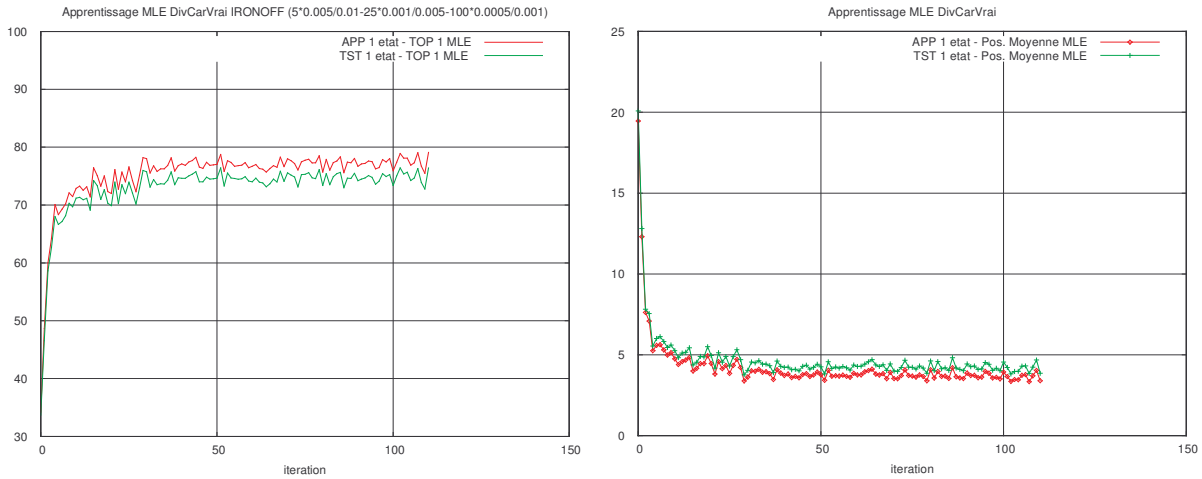


Figure 89. Performances de reconnaissance, taux de reconnaissance des mots en première position et mesure de la position moyenne du vrai mot, pour les bases d'apprentissage (APP) et de tests (TST) IRONOFF mot.

Les courbes présentées Figure 89 montrent l'évolution des performances sur les bases d'apprentissage et de tests IRONOFF en fonction du nombre d'itérations de la base d'apprentissage. Les pas ont été réglés de façon décroissante au fur et à mesure des itérations, et en différenciant les pas des parties extraction et classification du TDNN, voir Tableau 29.

Tableau 29. Evolution du pas du gradient en fonction des itérations

N°itération	Pas de la partie extraction du TDNN	Pas de la partie classification linéaire du TDNN
1 à 5	0,005	0,01
6 à 26	0,001	0,005
> 26	0,0005	0,001

La tendance des graphes montre que le reconnaisseur est capable de converger même si les poids du réseau ont été initialisés aléatoirement donc sans aucune connaissance au préalable au niveau caractère. Bien que les performances en test et apprentissage stagnent autour de 75 % avec une position moyenne du vrai mot de 3.88 sur un lexique de 197 mots, il n'y a pas de réels gains en continuant les itérations de la base d'apprentissage ni en modifiant les pas de descente du gradient.

Les performances limitées, notamment si on s'intéresse à la position moyenne peuvent s'expliquer par les raisons suivantes :

– l'absence de pouvoir de discrimination.

Comme nous l'avons défini au chapitre 3, un critère ML optimise la vraisemblance du vrai modèle indépendamment de tous les autres modèles du lexique proposé. La fonction de coût et le gradient associé s'obtiennent en annulant les paramètres (ε, β, α) du critère générique.

$$L_{MLE} = L_G \Big|_{(\varepsilon=0,\beta=0,\alpha=0)}$$
$$L_{MLE} = \log P\left(O \mid \lambda_{MMCvrai}\right)$$

Le gradient $Grad_{j,t}$ est donné par :

$$Grad_{j,t} = \begin{cases} +1 & \text{si } q_{j,t} \in \text{ chemin\_vrai (j,t)} \\ 0 & \text{sinon} \end{cases}$$

Il en ressort que :

$$\delta_{j,t} = Grad_{j,t} - x_{j,t} \sum_k Grad_{k,t}$$

$$\delta_{j,t} = \begin{cases} 1 - x_{j,t} & \text{si j est le numéro du label} \in \text{ Mot Vrai à la postion Ot} \\ 0 - x_{j,t} & \text{sinon} \end{cases}$$

et donc l'erreur qui sera rétropropagée dans le réseau correspond dans ce cas, à la différence entre une probabilité d'observation égale à 1 ou 0 (désirée) et la sortie réelle.

– La taille de la base d'apprentissage qui ne permet pas une modélisation précise de chaque mot.

Dans ces conditions, une première solution est de faire intervenir une compétition entre les classes, avec un critère de discrimination simple où une correction ne sera effectuée que si le modèle vrai ne correspond pas au modèle reconnu en première position, dans les autres cas, aucune correction ne sera effectuée. Cette proposition nécessite le balayage et la segmentation Viterbi de tous les mots du lexique pour obtenir celui avec le maximum de vraisemblance, mais en contrepartie, la convergence est plus rapide du fait de la correction d'un plus grand nombre de classes à chaque rétropropagation, celles-ci ne se limitant pas à celles du vrai modèle. C'est l'objectif du critère introduit ci-dessous.

7.4.3 Critère MMI simplifié

Un critère de type MMI simplifié (MMIs) permet d'annuler le gradient quand le label du mot à apprendre correspond au mot du label de plus forte vraisemblance. En effet, la fonction objectif définie par ce critère est donnée par l'équation suivante prenant ainsi en compte la différence des logarithmes des vraisemblances entre le chemin vrai $\lambda_{MMCvrai}$ et le chemin $\lambda_{MMCreconnu}$ de plus haut score issu du dictionnaire :

$$L_{MMIs} = L_G \Big|_{(\varepsilon=0,\beta=1,\alpha=0)}$$
$$L_{MMIs} = \log P(O \mid \lambda_{MMCvrai}) - \log(O \mid \lambda_{MMCreconnu})$$

De cette façon, si le chemin $\lambda_{MMCreconnu}$ se trouve être le chemin vrai alors le critère est nul et les coefficients du réseau de neurones ne seront pas affectés. Sinon, la rétropropagation du gradient de la fonction objectif est appliquée pour mettre à jour les poids W du réseau. Par ailleurs, il est à noter que ce critère ajoute une discrimination entre les classes du modèle vrai et celles du modèle reconnu. Ce qu'il est facile de comprendre dans l'écriture du gradient $Grad_{j,t}$.

$$Grad_{j,t} = \begin{cases} +1 & \text{si } q_{j,t} \in \text{ chemin\_vrai (j,t) et } q_{j,t} \notin \text{ chemin\_reconnu (j,t)} \\ -1 & \text{si } q_{j,t} \notin \text{ chemin\_vrai (j,t) et } q_{j,t} \in \text{ chemin\_reconnu (j,t)} \\ 0 & \text{sinon} \end{cases}$$

L'algorithme associé est alors le suivant :

```
Tant que le critère d'arrêt n'est pas satisfait
Nouvelle itération de la base d'apprentissage
    Mise à jour du pas de gradient
    Rangement aléatoire des exemples
Pour chaque exemple :
    Prétraitement de l'exemple
    Extraction des caractéristiques
    Propagation des caractéristiques dans le TDNN
    Pour chaque mot du dictionnaire
          Algorithme de Viterbi
          Si score reconnu > à tous les scores précédents
                Sauvegarde du chemin reconnu λMMCreconnu
                Sauvegarde de l'indice du mot reconnu i_reconnu
          Si label du mot = label vrai
                Sauvegarde du chemin vrai λMMCvrai
                Sauvegarde de l'indice du mot vrai i_vrai
    Fin pour chaque mot
    Si label_vrai != label_reconnu
Calcul du gradient correspondant à λMMCreconnu et λMMCvrai
          Rétro-propagation de l'erreur
    Mise à jour des poids du TDNN
Fin pour chaque exemple
Fin tant que
```

Algorithme 10. Algorithme de rétropropagation – Cas du Maximum d'information mutuelle simplifié

Le Tableau 30 montre l'accroissement des performances entre les critères ML (1) et MMIs (2) introduits dans le Tableau 26.

Tableau 30. Comparaison des performances d'un critère de type modèle générateur avec un critère d'information mutuelle simplifiée sur la base de test IRONOFF mot

Performances / Critères	Taux de reconnaissance TOP 1	Taux de reconnaissance TOP 2	Taux de reconnaissance TOP 3	Position moyenne du vrai mot (sur 197)
ML	77.43	83,46	86,47	3,88
MMIs	83,82	90,68	98,03	1,98

On remarque qu'avec ce critère de discrimination simple, les performances du système croissent significativement, elles dépassent le score de 98% de reconnaissance en considérant les trois premières propositions du système. La position du vrai label s'améliore également de façon importante, elle passe en moyenne au-dessus de la deuxième position (1.98).

Pour évaluer et tenter d'améliorer les performances du système, nous avons analysé les principales erreurs en sortie du système. Illustrées à la Figure 91, ces erreurs se caractérisent principalement par :

1. Une mauvaise détection des lignes de références
2. Une mauvaise discrimination des mots proches
3. La non-gestion des diacritiques
4. Mauvaise reconnaissance.

Le point 1 devrait être amélioré en amont du système de reconnaissance, nous nous contenterons ici d'en montrer ses effets. Pour cela, analysons les performances détaillées du système en fonction de la longueur des mots. Une des premières remises en cause est le même prétraitement pour des mots de longueur courte (< 5 lettres) ou longue (≥ 5 lettres).

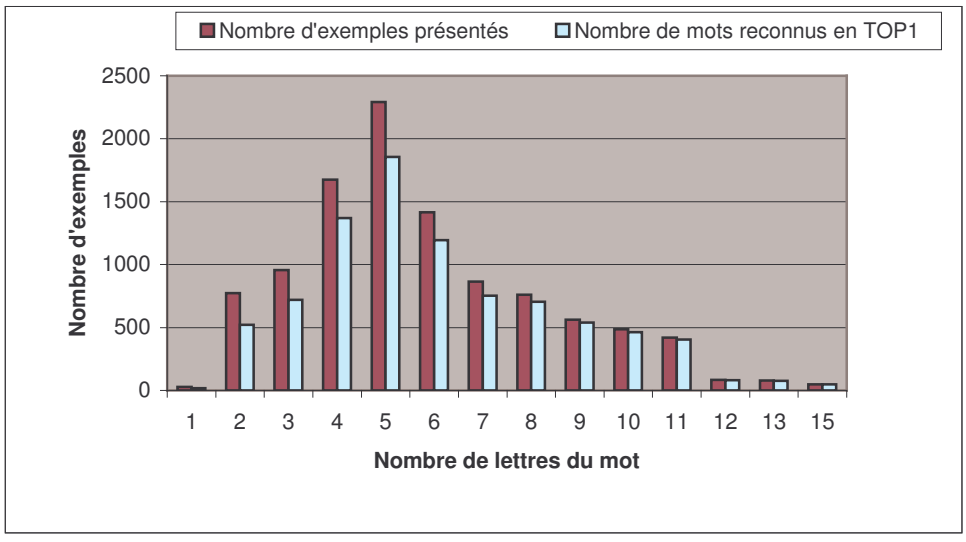
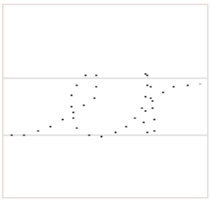


Figure 90. Performances du reconnaisseur mot selon la taille des mots.

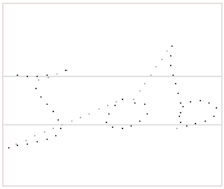
Ce sont principalement les erreurs causées sur les mots de taille inférieure à 6 lettres qui contribuent à dégrader les performances générales du système. Ceci est en partie dû à la même stratégie mise en œuvre pour la détermination des lignes de références quel que soit le nombre de lettres du mot. Or il est beaucoup plus difficile de poser les lignes de références avec un

nombre limité d'extrema locaux. L'écriture de marques diacritiques et de ratures est aussi sujette à entraîner ce type d'erreur.

Cas : Mauvaise détection des lignes de base

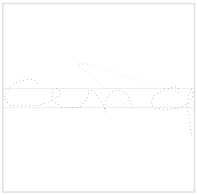


Label : et score -2.482145 position 2 (nb_obs = 4)
Données : Data\B318\B318.champs91.unp
Top 1 : en -2.384045
Top 2 : et -2.482145
Top 3 : ces -2.708121
Remarque : la deuxième lettre ne dépasse pas la ligne de base, elle est donc considérée comme une minuscule d'où le mot « en » en 1ère position



Label : Job score -9.470070 position 6 (nb_obs = 8)
Données : Data\F40\F40.champs10.unp
Top 1 : sous -5.699051
Top 2 : six -8.935712
Top 3 : sur -9.144403
Remarque : le jambage du J et la hampe du b ne sont pas assez prononcés donc un mot constitué de minuscules est plus probable

Cas : Perturbations des diacritiques et ratures



Label : cinq score -21.365084 position 7 (nb_obs = 16)
Données : Data\B74\B74.champs68.unp
Top 1 : centimes -14.598721
Top 2 : entre -14.921844
Top 3 : seize -18.550340
Remarque : le point du i a été fait à la fin du mot et n'est pas détecté d'où l'interprétation et l'influence du e en début de mot

Figure 91. Exemple et interprétation de mots non reconnus en première position

Pour considérer les erreurs liées à la reconnaissance (points 2 et 4), nous avons analysé le système au niveau caractère en observant les sorties du TDNN et la segmentation associée. Dans ce schéma, dès qu'un mot est bien reconnu, il ne participe plus à l'apprentissage, on peut penser que pour autant les estimations des probabilités des observations de ce mot ne sont pas nécessairement optimales : certaines sorties du réseau mériteraient encore d'évoluer pour permettre une meilleure généralisation sur d'autres exemples.

Par ailleurs, on constate qu'au vu du choix effectué pour le gradient, la somme des gradients par trame O_t est toujours nulle, ce qui a un impact important sur l'apprentissage. En effet, cette somme dans le cas général vaut :

$\sum_k Grad_{k,t} = 1 + \varepsilon - \beta$, relation qui peut être vérifiée sur le Tableau 27

ce qui donne ici avec $\varepsilon = 0$ *et* $\beta = 1$: $\sum_k Grad_{k,t} = 0$

dès lors, l'erreur rétro-propagée $\delta_{j,t} = Grad_{j,t} - x_{j,t} \sum_k Grad_{k,t}$ ne fait plus intervenir l'état de la sortie puisqu'elle devient : $\delta_{j,t} = Grad_{j,t}$.

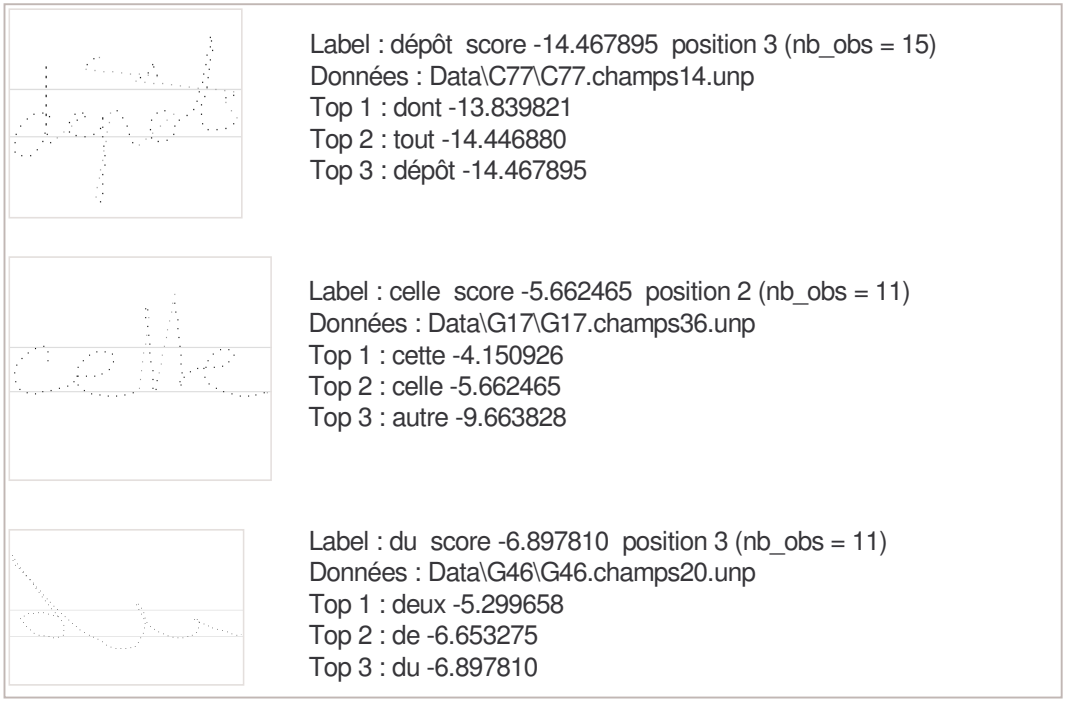


Figure 92. Exemples de mots non reconnus en première position due à une mauvaise discrimination de classes

Ce qui compte dans ce cas, c'est donc uniquement la présence d'un état dans un chemin pour influencer l'apprentissage, mais pas sa valeur en tant que telle.

Avec l'apprentissage MLE, on avait $\varepsilon = 0$ *et* $\beta = 0$, donc $\sum_k Grad_{k,t} = 1$, c'est-à-dire qu'alors :

$\delta_{j,t} = Grad_{j,t} - x_{j,t}$ ce qui correspond d'ailleurs exactement à l'erreur rétro-propagée quand on traite des caractères isolés (cf. section 2.2.2.3).

Il est possible de conserver cette formulation de la fonction d'erreur avec un autre choix sur nos paramètres α, β, et ε. En effet, une autre solution serait de fixer $\varepsilon = 1$ *et* $\beta = 1$ pour avoir $\sum_k Grad_{k,t} = 1$.

Cela correspond précisément au critère mixte MMIs-ML présenté ci-dessous.

7.4.4 Critère mixte MMIs-ML

Nous adaptons l'algorithme précédent pour faire systématiquement le calcul des gradients correspondants *à* $\lambda_{MMCreconnu}$ *et* $\lambda_{MMCvrai}$ et pour rétro-propager dans toutes les circonstances l'erreur $\delta_{j,t} = Grad_{j,t} - x_{j,t}$. L'algorithme est maintenant le suivant :

Algorithme de rétro propagation – Cas mixte MMIs-ML

```
Tant que le critère d'arrêt n'est pas satisfait
Nouvelle itération de la base d'apprentissage
    Mise à jour du pas de gradient
    Rangement aléatoire des exemples
Pour chaque exemple :
```

```
    Prétraitement de l'exemple
    Extraction des caractéristiques
    Propagation des caractéristiques dans le TDNN
    Pour chaque mot du dictionnaire
          Algorithme de Viterbi
          Si score reconnu > à tous les scores précédents
                Sauvegarde du chemin reconnu λ_MMCreconnu
                Sauvegarde de l'indice du mot reconnu i_reconnu
          Si label du mot = label vrai
                Sauvegarde du chemin vrai λ_MMCvrai
                Sauvegarde de l'indice du mot vrai i_vrai
    Fin pour chaque mot
    Calcul du gradient correspondant à λ_MMCreconnu et λ_MMCvrai
    Rétropropagation de l'erreur
          Mise à jour des poids du TDNN
Fin pour chaque exemple
Fin tant que
```

Algorithme 11. Algorithme de rétropropagation – Cas MLE+ MMIs

La déclinaison du critère en prenant $\varepsilon = 1$ *et* $\beta = 1$ est donné dans l'encadré suivant.

$$L_{MMIs-ML} = L_G \Big|_{(\varepsilon=1,\beta=1,\alpha=0)}$$

$$L_{MMIs-ML} = 2\log P(O|\lambda_{MMCvrai}) - \log(O|\lambda_{MMCreconnu})$$

$$Grad_{j,t} = \begin{cases} +2 & si \quad q(j,t) \in \text{chemin\_vrai(j,t)} \text{ et } q(j,t) \notin \text{chemin\_reconnu(j,t)} \\ -1 & si \quad q(j,t) \notin \text{chemin\_vrai(j,t)} \text{ et } q(j,t) \in \text{chemin\_reconnu(j,t)} \\ +1 & si \quad q(j,t) \in \text{chemin\_vrai}(j,t) = \text{chemin\_reconnu(j,t)} \\ 0 & \text{sinon} \end{cases}$$

Nous avons expérimenté l'incidence de ce critère avec les paramètres ε et β unitaires. Les résultats montrent une diminution de 15% du nombre d'erreurs entre un critère d'information mutuelle simplifié et un critère couplant discrimination et renforcement du label vrai mal reconnu.

Tableau 31. Performances du système hybride avec des apprentissages MMIs, ML et mixte MMIs-ML

Critères	ML	MMIs	MMIs-ML
Paramètres	*ε=0* *β=0* *α=0*	*ε=0* *β=1* *α=0*	*ε=1* *β=1* *α=0*
Taux	77,43	83,82	86,34

7.4.5 Critère hors lexique

L'emploi d'un critère MMI nécessite un apprentissage avec un lexique conséquent qui contient suffisamment de mots proches afin de générer de la confusion et de forcer le système à bien discriminer des formes proches. Il peut-être utile par exemple d'étendre le lexique de base lors de l'apprentissage pour rajouter des contre-exemples (ce, ajout : ces, cet…) pour obtenir une discrimination satisfaisante. Le critère MMIs-ML proposé permet de pallier cette contrainte. Cependant l'optimisation des paramètres du critère est un travail difficile demandant une série d'expérimentations très coûteuses en temps et mémoire mais aussi sur le choix des pas de descente. Une itération de la base d'apprentissage IRONOFF mot demande trois heures sur un Pentium IV à 1,8 GHz. Afin d'accroître les performances du système et surtout sa rapidité d'apprentissage nous nous sommes intéressés à un apprentissage complètement sans lexique guidé par les sorties du TDNN. Celui-ci consiste à fixer α = 1 dans la formule générale, on choisit aussi, comme précédemment, $\varepsilon = 1 \ et \ \beta = 1$.

$$L_{TDNN-ML} = L_G\Big|_{(\varepsilon,\beta,\alpha=1)}$$

$$L_{TDNN-ML} = 2\log P\left(O \mid \lambda_{MMCvrai}\right) - \log\left(O \mid \lambda_{TDNNCreconnu}\right)$$

$$Grad_{j,t} = \begin{cases} +2 \ \text{si} \quad q(j,t) \in \text{chemin\_vrai(j,t) et } q(j,t) \neq \text{argmax}_k\, b_k(O_t) \\ -1 \ \text{si} \quad q(j,t) \notin \text{chemin\_vrai(j,t) et } q(j,t) = \text{argmax}_k\, b_k(O_t) \\ +1 \ \text{si} \quad q(j,t) \in \text{chemin\_vrai(j,t) et } q(j,t) = \text{argmax}_k\, b_k(O_t) \\ 0 \ \text{sinon} \end{cases}$$

La discrimination ne porte plus sur les états correspondants à la segmentation Viterbi du mot de plus forte vraisemblance mais à la discrimination des classes de plus fortes probabilités en sortie du TDNN. Il n'y a donc pas d'exploration du lexique, seule une comparaison des sorties du TDNN pour chaque trame Ot est effectuée. Le mot reconnu, noté TDNNreconnu, peut donc n'avoir aucun sens (dans le cas de l'apprentissage du mot « un » la concaténation des classes de plus fortes probabilités donne la séquence «ieumn » qui ne correspond pas à un mot), puisque la reconnaissance est basée caractère.

Tableau 32. Performances d'un apprentissage avec ou sans compétition avec le lexique

Critères	ML	MMIs	ML-TDNN
Paramètres	$\varepsilon=0$ $\beta=0$ $\alpha=0$	$\varepsilon=0$ $\beta=1$ $\alpha=0$	$\varepsilon=1$ $\beta=1$ $\alpha=1$
Taux	77,43	83,82	82,26

Malgré son absence de discrimination au niveau mot, un apprentissage sans compétition avec le lexique permet de réduire le nombre de mots mal reconnus par rapport à un critère basé sur

les modèles générateurs mais n'est pas aussi efficace qu'un apprentissage basé sur un lexique de mots. Cette solution est tout de même intéressante pour initialiser le système et entraîner le réseau à discriminer des mots proches tels que « ces », « cet », « cette », « celle ».

7.5 Comparaison des critères d'apprentissage

Nous avons testé une dernière combinaison en intégrant à la fois une prise en compte des états du modèle MMC reconnu et des meilleurs états fournis par le TDNN, ceci est réalisé en fixant $\alpha = 0.5$. Le Tableau 33 illustre les taux de reconnaissance en première position de la base IRONOFF pour chaque critère et confirme l'intérêt d'introduire dans l'apprentissage une discrimination au niveau mot mais aussi de coupler une discrimination au niveau caractère.

Tableau 33. Taux de reconnaissance sur la base de généralisation (non apprise) IRONOFF

Critères	ML	MMIs	MMIs-ML	MMIs-ML-TDNN
Paramètres	$\varepsilon=0$ $\beta=0$ $\alpha=0$	$\varepsilon=0$ $\beta=1$ $\alpha=0$	$\varepsilon=1$ $\beta=1$ $\alpha=1$	$\varepsilon=1$ $\beta=1$ $\alpha=0,5$
Taux	77,43	83,82	86,34	87,09

Une discrimination au niveau caractère ($\alpha=0.5$) en plus d'une discrimination au niveau mot accroît les performances du reconnaisseur mot. Il est intéressant de souligner la convergence constatée expérimentalement de ce système d'apprentissage en dehors de toute initialisation spécifique du TDNN et de toute labellisation explicite au niveau caractère. Toutefois, mais c'est le cas assez systématiquement avec les approches neuronales, la valeur du paramètre de pas de gradient influence de manière importante la qualité de la convergence.

7.6 Conclusion

Ce chapitre a permis de vérifier la validité du système hybride proposé et du schéma global d'apprentissage proposé. Les résultats donnés sont à mettre en corrélation avec le nombre de classes à discriminer (66 classes) et la simplicité de la modélisation markovienne utilisée. Un mot est représenté comme la concaténation des modèles MMC-lettre qui le composent et chaque modèle MMC-lettre est identique. A savoir qu'une lettre ne possède qu'un seul état qui peut être répété. Ce modèle est peu flexible par rapport aux notions de durées inter et intra lettres. Dans le chapitre suivant nous allons affiner la représentation d'un MMC lettre et présenter une méthode d'intégration de la durée.

8 REPRESENTATION MULTI ETAT ET MODELISATION DE LA DUREE

Dans les deux chapitres précédents, reconnaissance et apprentissage mot, nous avons défini une structure de système hybride et proposé et analysé un critère d'apprentissage global. Les résultats des expérimentations montrent que ce critère permet effectivement l'apprentissage du système avec un niveau raisonnable de performance (87,09 % sur IRONOFF). Pour améliorer les performances de cette version de base, nous proposons d'analyser les limitations de la modélisation markovienne choisie et nous introduisons une modélisation lettre et mot plus adaptée à l'écriture naturelle.

8.1 Limitation du modèle 1 état

Les expériences menées précédemment étaient fondées sur une modélisation simplifiée : une lettre est représentée par un HMM à un état avec des probabilités de transitions constantes et un mot résulte de la concaténation de modèles HMM lettres. Le schéma suivant illustre la modélisation du mot « un » déjà présenté au chapitre précédent :

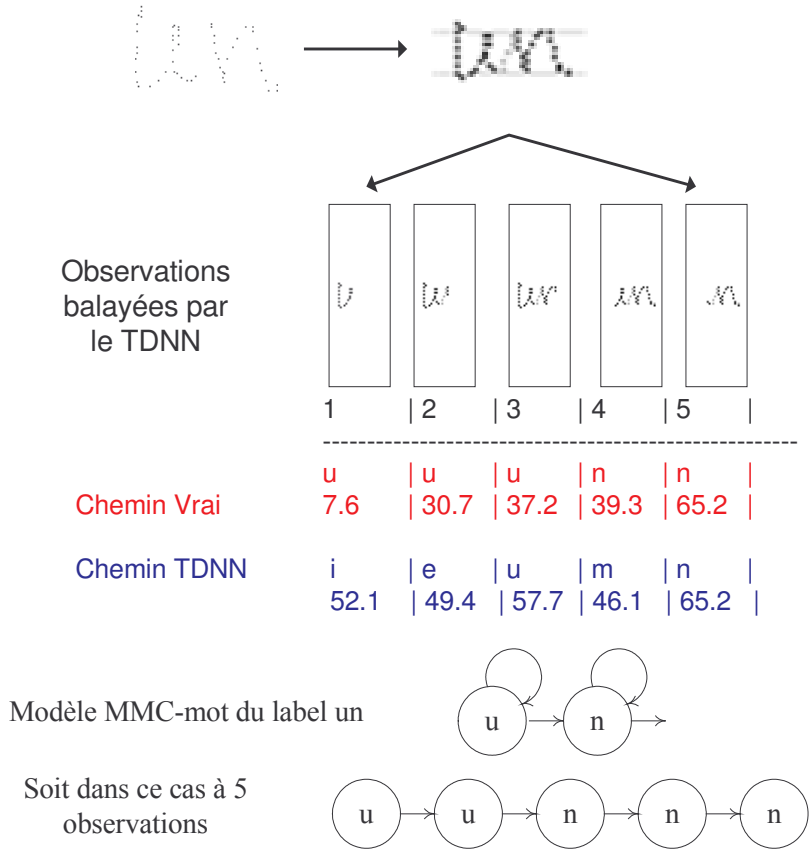


Figure 93. Modélisation markovienne du mot « un » avec 1 état par lettre

Le modèle MMC-mot du label « un » est un modèle gauche droite à deux états dont le premier caractérise la lettre 'u' et le suivant la lettre 'n'. La segmentation Viterbi sachant le label et les sorties du TDNN donne comme chemin trois passages dans l'état 'u' puis deux passages dans l'état 'n'.

Cette segmentation est déterminée par les probabilités d'observations calculées par la configuration des poids du réseau à cet instant de l'apprentissage. Dans le cas de cet exemple, nous aurons donc une segmentation parmi les cinq présentées à la Figure 94. Le même état assume donc une lourde tâche, il doit être capable de représenter un environnement très variable, suivant la position du caractère concerné. En particulier, lors de l'initialisation du réseau, chacune de ces configurations étant équiprobable, il se peut même que le caractère n'apparaisse pas encore, ou soit déjà parti, dans la fenêtre d'observation : le réseau devrait être capable de prédire l'arrivée d'un nouveau caractère ou se souvenir d'un élément hors de son de champ de vision. Par exemple, si l'on considère la première segmentation proposée à la Figure 94, la seconde observation est déjà associée à la lettre 'n', cet état serait donc, de la même façon que le dernier état 'n', chargé de détecter la présence d'un 'n'.

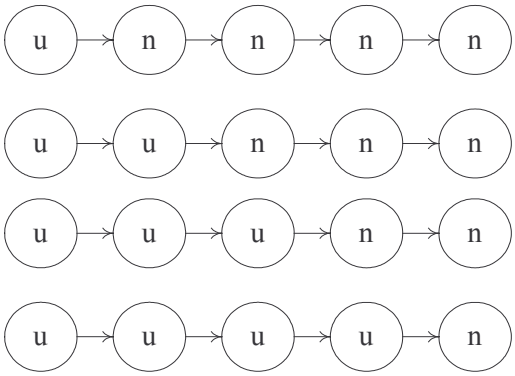


Figure 94. Chemins possibles du mot « un » pour un mot balayé en 5 observations

Posé ainsi, deux problèmes sont évoqués.

Premièrement, le modèle de lettre est trop grossier, avec un modèle plus fin, il sera plus aisé de caractériser les extrémités de caractères (début et fin). Avec un seul état, il est plus difficile de rendre compte des bruits générés par les lettres qui l'entourent et par les ligatures. Rappelons qu'avec le chevauchement du balayage (la fenêtre d'observation est égale à 4/3 de la longueur moyenne d'un caractère, elle est décalée d'une observation à l'autre de 1/3 de cette longueur) un même caractère est présent en moyenne dans six fenêtres consécutives, dont les deux centrales qui le contiennent entièrement.

Deuxièmement, l'étape d'initialisation de l'apprentissage est critique. A cet instant, le réseau ne connaissant rien, la segmentation est quasi-aléatoire, il va en résulter un défaut d'alignement préjudiciable à une convergence rapide. Ainsi, par exemple, les premières et dernières configurations de l'exemple précédent sont très pénalisantes et perturbatrices pour la correction des poids du réseau. Dans le cas d'un critère avec discrimination, au niveau mot et même caractère, la lettre 'n' dans le premier cas sera corrigée quatre fois plus que la lettre 'u' alors qu'elles ont à peu près le même poids dans le mot (deux lettres) et dans leur écriture (même longueur).

Cette modélisation du mot ne prend pas en compte :

- La représentativité des lettres dans le mot,

– Les variabilités de longueur des lettres.

Dans la section 8.2, nous présentons comment étendre le modèle de Markov avec plusieurs états afin d'absorber les différentes situations du caractère dans la fenêtre (début, milieu et fin de la fenêtre). La section 8.3 traitera les modèles de durée pour mieux contrôler la segmentation en phase initiale.

8.2 Représentation multi-états

Avec plusieurs états par lettre, il est plus facile d'absorber les différentes situations du caractère dans la fenêtre d'observation (début, milieu et fin de la fenêtre) et d'intégrer les variabilités de longueur intra et inter-lettres. Compte tenu du nombre variable de graphèmes dans une lettre, une lettre courte à un seul graphème (comme le « i », le « c ») pourra ainsi traverser le modèle par un seul état tandis qu'une lettre longue pourra s'étaler sur deux ou trois états (cas du « f », du « m », du « w »).

8.2.1 Modèle lettre multi-états

La Figure 95 illustre le cas général d'un modèle lettre à N états et les trois premières configurations : 1 état, 2 états, 3 états. Comme pour les mots, un modèle lettre est la concaténation ordonnée des états qui le composent. Les valuations des transitions n'ont pas été mentionnées sur le schéma car elles sont unitaires.

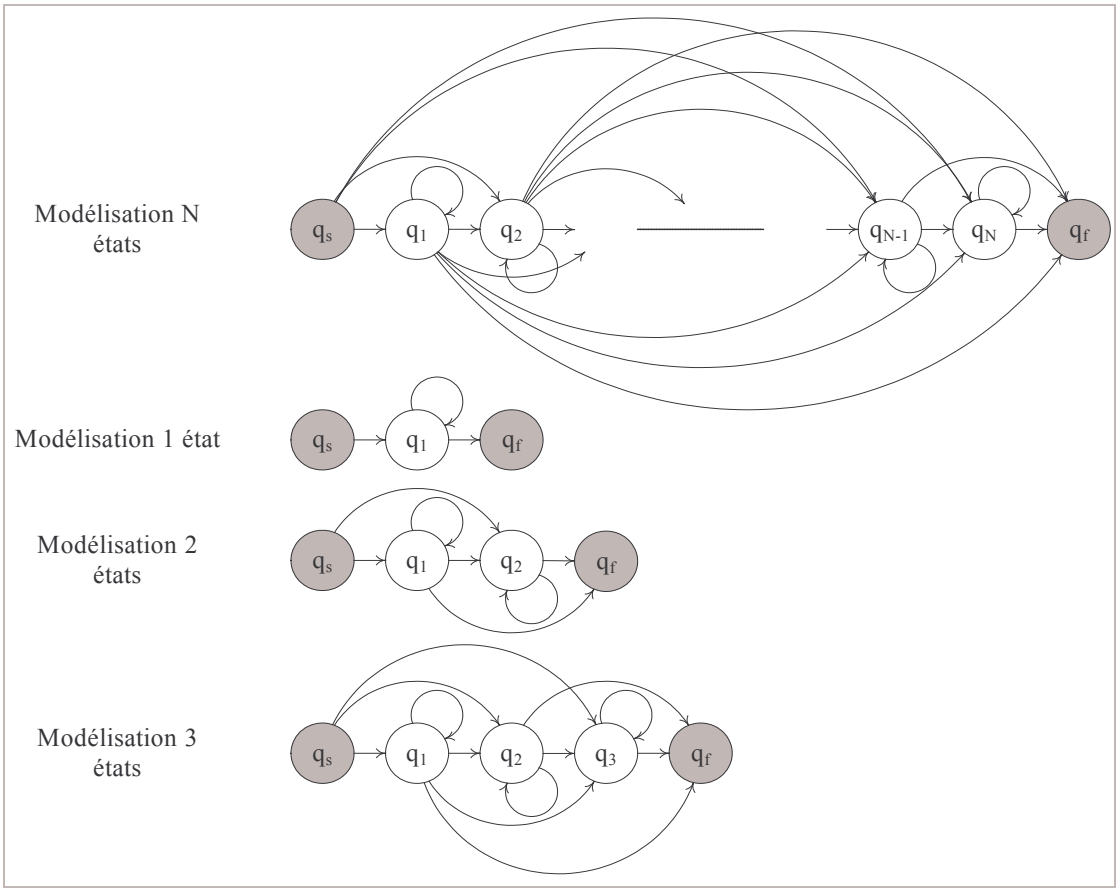


Figure 95. Représentation MMC lettre multi-état

Le nombre d'états peut être soit identique soit variable pour chaque lettre. Il est à noter que modifier le nombre d'états par lettre implique une modification de la topologie du TDNN. En effet, augmenter le nombre d'états revient à augmenter le nombre de sorties du TDNN qui estime les densités de probabilités de Ot dans chaque état. Ainsi, le TDNN possède autant de sorties que le nombre total d'états différents.

On a vu qu'en moyenne un caractère émargeait sur six fenêtres d'observations, il pourrait être intéressant d'associer un état à chacune de ces six positions de la fenêtre : présence de la fenêtre sur le premier tiers, sur les deux premiers tiers, …, sur les deux derniers tiers, sur le dernier tiers du caractère. Mais le nombre de sorties du classifieur serait alors très important et le nombre de paramètres libres à régler plus conséquent. Le Tableau 34 montre l'évolution du nombre de sorties du TDNN et du nombre de poids libre en fonction du nombre d'états pour la topologie à une couche cachée décrite dans le chapitre 6 (Reconnaissance Mot)..

Tableau 34. Impact du nombre d'états sur la topologie du TDNN à 1 couche cachée

Nombre d'états par lettre	1	2	3	4	5	6
Nombre de sorties du TDNN	66	132	198	264	330	396
Nombre de paramètres libres	22 606	43 792	64 978	86 164	107 350	128 536

Sans modifier la topologie du réseau et avec une base de données limitée, on constate qu'au delà de trois états, la taille du réseau devient relativement importante. Nous avons donc envisagé l'extension avec deux états et trois états, dans ce dernier cas, l'interprétation affichée est celle de vouloir modéliser le début, le corps et la fin de la lettre.

8.2.2 Algorithme d'adaptation entre TDNN et MMC

Nous détaillons ici une partie un peu plus technique permettant l'adaptation des sorties du TDNN avec les modèles lettres pour opérer la programmation dynamique dans l'algorithme général de reconnaissance du système hybride neuro-markovien. Pour exécuter une reconnaissance, les fichiers suivants sont nécessaires et stockés en mémoire :

- la topologie du réseau
- les poids du réseau
- la liste des modèles lettres
- le dictionnaire de mots
- l'exécutable.

Un tableau de structures comportant autant d'éléments que de labels lettres est créé. Dans chaque élément du tableau, on trouve : son label, sa matrice de transitions A (ici remplie de 1), une matrice B destinée à contenir les probabilités des observations associées à chacun des états de la lettre correspondante qui seront fournies pour chaque exemple par le TDNN, son vecteur d'initialisation Π (ici rempli de 1). Les sorties du TDNN sont ordonnées selon ce tableau avec prise en compte du nombre d'états par lettres, ainsi un état est représenté par son étiquette et son indice temporel dans le MMC lettre, par exemple, A1 correspond au 1er état de la lettre 'A'.

Tableau 35. Exemple de correspondance des sorties TDNN avec les états des modèles MMC lettre dans le cas d'un nombre fixe de trois états par lettre

N°TDNN	1	2	3	4	5	6	…	79	80	81	…	198
état	A1	A2	A3	B1	B2	B3	…	a1	a2	a3	…	ù3

L'algorithme d'adaptation est le suivant :

```
Algorithme d'adaptation TDNN -> HMM
Soit T le nombre d'observations du signal
Pour chaque observation t
  Pour chaque MMC lettre de base, MMC(e)
    Recherche de l'indice k de la lettre MMC(e) correspondant dans
le TDNN
    Tant que indice k non trouvé
      Balayage des sorties du TDNN
      Comparaison des labels MMC(e) et label_TDNN(k)
    Pour chaque état de la lettre considéré etat_hmm
          B[etat_hmm][t]=sorties_TDNN[[t][k+etat_hmm];
    Fin pour etat_hmm
  Fin pour MMC(e)
Fin pour l'observation t
```

Algorithme 12. Algorithme d'adaptation TDNN -> HMM

8.2.3 Expérimentations et résultats

Nous avons analysé l'impact d'un TDNN multi-états sur la base de mots IRONOFF. Nous avons conduit les mêmes tests que dans le chapitre précédent afin de vérifier l'intérêt du modèle de Markov étendu, et cela quel que soit le critère choisi.

Tableau 36. Taux de reconnaissance du système multi-états en généralisation

Critères	ML	MMIs	MMIs-ML	MMIs-ML-TDNN
Paramètres	$\varepsilon = 0$ $\beta = 0$ $\alpha = 0$	$\varepsilon = 0$ $\beta = 1$ $\alpha = 0$	$\varepsilon = 1$ $\beta = 1$ $\alpha = 0$	$\varepsilon = 1$ $\beta = 1$ $\alpha = 0,5$
1 état	77,43	83,82	86,34	87,09
2 états	80,87	87.46	88.22	90,51
3 états	84,69	90,57	92,01	92,78

Les résultats présentés dans ce tableau montrent tout l'intérêt de la modélisation multi-états. Les performances du reconnaisseur quel que soit la fonction de coût choisie augmentent de manière significative. En effet, en fonction du critère, cela se traduit par une diminution de taux d'erreurs de 32 % (critère ML) à 44 % (critère mixte). Les différentes versions de la fonction de

coût conservent leur rang, mais c'est le critère qui était le plus performant avec un seul état qui tire le plus parti de cette nouvelle modélisation. Ainsi, on obtient un taux de reconnaissance avec le critère mixte de 92.78 % sur la base IRONOFF.

8.3 Modélisation de la durée

Une représentation multi-états permet d'absorber les bruits liés aux ligatures mais ne prend pas en compte les effets de longueur des lettres dans un mot. Au niveau de l'apprentissage, il s'avère que le réseau est très sensible à l'influence du pas de gradient et diverge dans certains cas. Précisément dans le cas où la séquence Viterbi d'un mot-vrai entraîne une segmentation entre les caractères très déséquilibrée par rapport à la réalité.

Pour pallier ce défaut, il conviendrait, tant que le réseau n'a pas bien appris ses modèles, de guider la segmentation. Cette phase initiale va rajouter des contraintes pour déterminer le chemin optimal. Cette contrainte sera ensuite relâchée au fil des itérations jusqu'à disparaître. On trouve le même principe dans les schémas d'apprentissage tels que ceux exploités dans Remus [Garcia-Salicetti, 1996] ou encore Npen++ [Jaeger, 2000].

Analysons un exemple de séquence avec le mot « deux ». Dans cet exemple, le mot « deux » est balayé par le TDNN en 14 observations ; le mot deux contenant 4 lettres distinctes, une répartition équilibrée dans le chemin Viterbi serait donnée par 3 passages minimum dans chaque MMC lettre. Or, à un moment donné, le chemin observé est la séquence « d d e u x x x x x x x x x x». Celui-ci donne une importance considérable à la lettre 'x' par rapport aux autres lettres de même poids dans le mot.

Ce type de segmentation et la correction associée perturbe très nettement le système. La matrice du gradient considérant le chemin vrai est donnée par la table suivante. On remarque que la lettre 'x' est corrigée 5 fois plus que la lettre 'd' et dix fois plus que les lettres 'e' et 'u'. Expérimentalement, nous avons observé ce phénomène de répétition d'un état à différents instants, ils ne correspondent pas forcément à la fin du mot qui peut être souvent naturellement plus longue dans le processus d'écriture.

Tableau 37. Matrice du gradient dans le cas MLE

	Obs 1	Obs 2	Obs 3	Obs 4	**Obs 5**	**Obs 6**	**…**	**Obs 14**	**Somme**
d	1	1	0	0	**0**	**0**		**0**	**2**
e	0	0	1	0	**0**	**0**	**…**	**0**	**1**
u	0	0	0	1	**0**	**0**	**…**	**0**	**1**
x	0	0	0	0	**1**	**1**	**…**	**1**	**10**
Autres	0	0	0	0	**0**	**0**	**0**	**0**	**0**

Il serait en fait souhaitable d'entraîner le réseau pour qu'il favorise des séquences de longueurs équilibrées entre les différents états. Pour cela, il faudrait obtenir pour la distribution de la vraisemblance des différents chemins de segmentation possibles, une courbe telle que présentée sur la Figure 96. Avec l'exemple, on souhaite que la durée dans les états 'd' et 'e' soit de T/2, égale à 5 ici.. La séquence homogène au sens du nombre de lettres étant « dddddeeeee ».

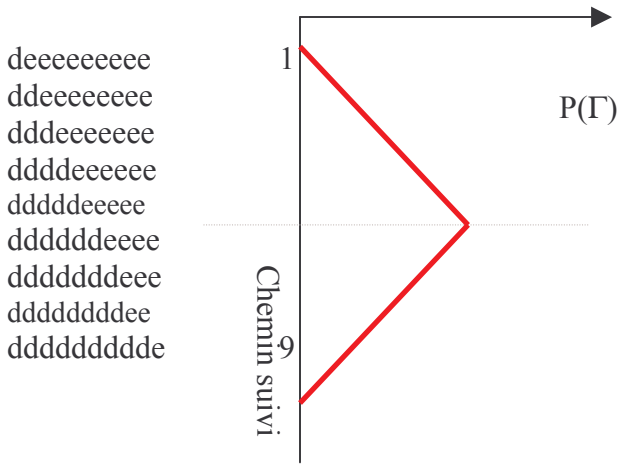


Figure 96. Répartition souhaitée de la vraisemblance du mot « d » à 10 observations

Nous allons discuter des possibilités de corriger ces bruits générés par la segmentation avec un modèle de durée et présenter deux méthodes :

- une modélisation de la durée classique se traduisant par une topologie spécifique du modèle de Markov.
- une seconde ne modifiant pas la structure du modèle de Markov mais intégrant un modèle de durée via les probabilités d'émission.

8.3.1 Modélisation classique de la durée

Schenkel et al. dans [Schenkel, 1995] propose un modèle correspondant à la topologie présentée Figure 97 où la modélisation de la durée se traduit par un calcul des probabilités de transitions. Chaque classe caractère noté *Ci* est associée à plusieurs états notés $S_{i,Ni}$ où *Ni* est le nombre d'états de la classe *Ci*. Les états associés à la même classe génèrent les observations avec la même probabilité *P(Ot|Ci)*. Les probabilités de transitions sont calculées de sorte à refléter la distribution de durée désirée, soit $\rho(d)$.

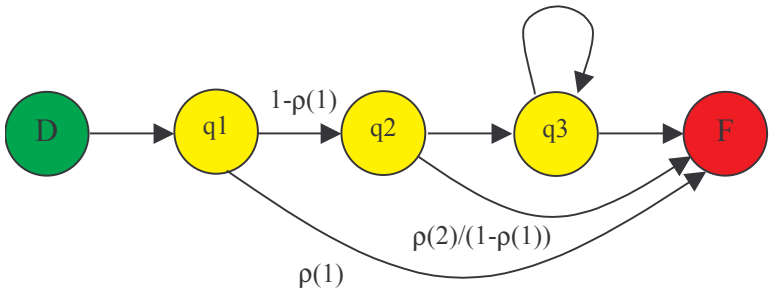


Figure 97. Modèle MMC lettre avec prise en compte de la durée

Ayant fixé $\rho(d)$, les probabilités de transitions sont alors données par la récursion suivante :

$$P(S_{i,2} \mid S_{i,1}) = 1 - \rho(1)$$

$$P(S_{i,d+1} \mid S_{i,d}) = 1 - \rho(d) / \prod_{t=1}^{d} P(S_{i,t} \mid S_{i,t-1})$$

Dans la pratique pour $\rho(d)$ on peut choisir une loi de Poisson, Figure 98, en limitant le nombre d'états d'un caractère à 3 et en ajoutant une transition-self sur le dernier état pour finir la distribution par une loi exponentielle. Les auteurs précisent qu'un réglage précis du paramètre λ de la distribution (moyenne = variance = λ) n'apporte pas d'amélioration significative en terme de reconnaissance. En fixant λ = 2, on favorise les durées égales à 1, 2 ou 3, les durées supérieures à d = 4 deviennent très peu probables.

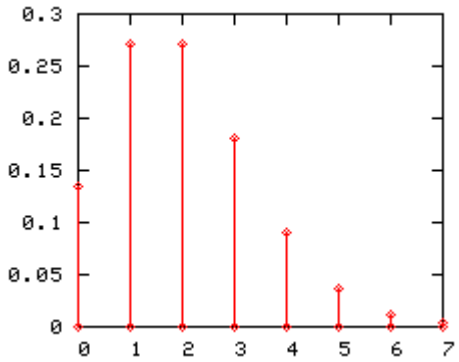


Figure 98. Loi de Poisson (λ = 2) $p(k) = e^{-\lambda}\frac{\lambda^k}{k!}$

La Figure 97 illustre les premières transitions de la structure d'un modèle lettre avec trois états.

Cette modélisation répond bien à la forme de la probabilité désirée le long des chemins illustrés à la Figure 96, cependant elle est sensible à la longueur de l'échantillon présenté. La Figure 99 montre les valeurs obtenues pour l'exemple « de » le long des 9 chemins de segmentations possibles (sans tenir compte des observations).

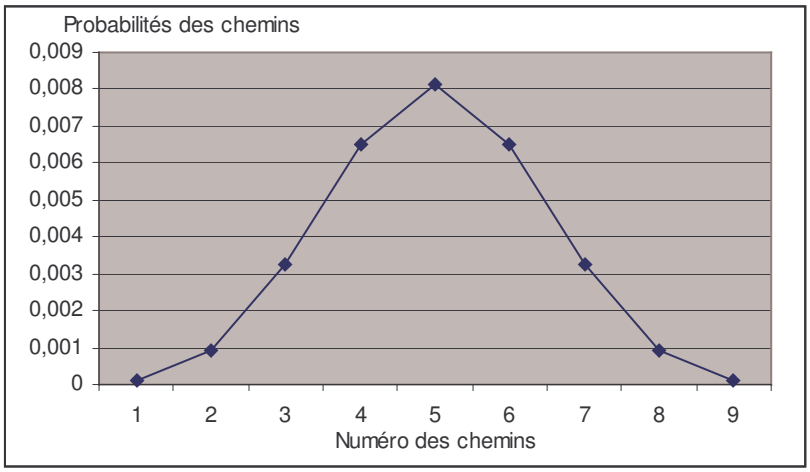


Figure 99. Le chemin numéro 5, segmentant le mot « de » en deux séquences de 5 états est le plus probable

S'il y a beaucoup d'observations pour peu de lettres, ce sera surtout le dernier état (troisième dans l'exemple) qui supportera la modélisation. En effet, avec cette topologie particulière, les premiers états ne sont que des états de passage sans possibilité de rester dans l'état. Inversement, il n'est pas possible de les sauter, ce qui oblige à avoir au minimum une séquence de longueur 3 pour traverser le modèle.

Plutôt que de modifier la topologie des modèles, une autre approche est proposée où le modèle de durée est cette fois absorbée dans les probabilités d'émissions.

8.3.2 Absorption d'un modèle de durée par les probabilités d'observations

Pour favoriser une segmentation correspondant au chemin optimal, nous ramenons les coefficients qui étaient précédemment associés aux transitions sur les probabilités d'observations. Cela revient à pondérer les sorties du TDNN avec une simple fonction triangle appliquée sur chaque lettre du label. Dans un cas multi-états tous les états sont pondérés par la même fonction. La pondération est suivie d'une normalisation des sorties entre 0 et 1 pour respecter l'échelle des probabilités. Dans le cas de l'exemple « de », le coefficient de pondération suit l'allure de la courbe suivante en fonction du numéro de l'observation considérée, le coefficient étant minoré par 1.

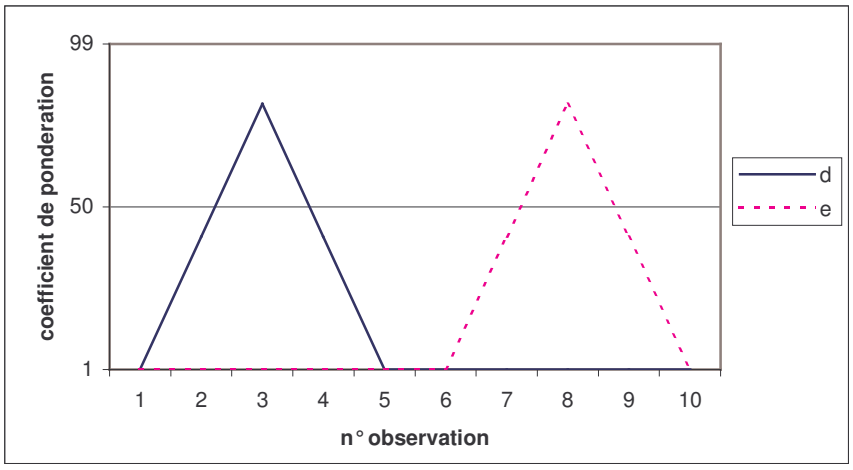


Figure 100. Allures de la pondération des sorties du TDNN pour le mot « de »

Seul le chemin de référence sera contraint, c'est-à-dire le chemin du label vrai en pondérant les probabilités d'observations correspondant à ce label. L'algorithme d'apprentissage avec un critère générique est alors le suivant :

Algorithme de rétro propagation – Cas générique

```
Tant que le critère d'arrêt n'est pas satisfait
Nouvelle itération de la base d'apprentissage
    Mise à jour du pas de gradient
    Rangement aléatoire des exemples
Pour chaque exemple :
    Prétraitement de l'exemple
    Extraction des caractéristiques
    Propagation des caractéristiques dans le TDNN
    Sauvegarde du chemin des sorties TDNN λTDNNreconnu
    Pour chaque mot du dictionnaire
          Algorithme de Viterbi
```

```
            Si score reconnu > à tous les scores précédents
                  Sauvegarde du chemin reconnu λMMCreconnu
                  Sauvegarde de l'indice du mot reconnu i_reconnu
            Si label du mot = label vrai
                  Si segmentation contrainte
                  Alors
                        Forçage des sorties du TDNN : modèle de durée
                        Algorithme de Viterbi
                  Sauvegarde du chemin vrai λMMCvrai
                  Sauvegarde de l'indice du mot vrai i_vrai
            Calcul du gradient correspondant à λMMCreconnu et λMMCvrai, et
λTDNNreconnu
      Fin pour chaque mot
      Rétropropagation de l'erreur
      Mise à jour des poids du TDNN
Fin pour chaque exemple
Fin tant que
```

Algorithme 13. Algorithme de rétro propagation – Cas générique

La condition de segmentation avec contrainte est levée soit :

- par un nombre d'itérations jugées suffisant par expérimentation
- soit par comparaison entre un seuil donné et le taux moyen de régularité de segmentation calculé sur une base de contrôle. Pour un mot, le taux de régularité de segmentation est défini par la moyenne géométrique des rapports du nombre d'observations pour un caractère sur le nombre d'observations désiré par caractère.

$$\text{taux de régularité moyen} = \frac{1}{nb\_exemples} \sum_{e=1}^{nb\_exemples} \left(\prod_{i=1}^{nb\_lettres(e)} \frac{nb\_obs(i)}{T / nb\_lettres(e)} \right)$$

avec

T : le nombre d'observations du signal balayées par le TDNN

Nb_exemples : le nombre d'exemples de la base d'apprentissage (ou Nombre limité)

nb_lettres(e) : le nombre de lettres du label de l'exemple considéré

nb_obs(i) : le nombre d'observations de la lettre i dans le chemin optimal selon Viterbi

Ce rapport est égal à un dans le cas d'une répartition du nombre d'observations homogène. Plus la segmentation du mot sera éloignée d'une répartition régulière, plus ce rapport moyen tendra vers 0. Si on reprend l'exemple du mot « de » à 10 observations, le Tableau 38 ci-dessous donne le taux de régularité de segmentation pour les différentes segmentations possibles de ce mot.

Tableau 38. Calcul du taux de régularité pour diverses segmentations du mot « de »

nb_obs(d)	1	2	3	4	5	6	7	8	9
nb_obs(e)	9	8	7	6	5	4	3	2	1
Taux de régularité	0,36	0,64	0,84	0,96	1	0,96	0,84	0,64	0,36

8.3.3 Protocoles et résultats

Pour analyser l'influence du modèle de durée, nous avons entraîné le réseau (1 état) en deux phases :

- 1ère phase : apprentissage avec un critère de maximum de vraisemblance contraint par un modèle de durée pendant 20 itérations.
- 2ème phase : apprentissage non contraint avec comme fonction de coût un critère de maximum d'information mutuelle simplifié (MMIs).

La Figure 101 illustre l'évolution des taux de reconnaissance en test pour la configuration citée ci-dessus intégrant un modèle de durée et celles correspondant aux critères ML et MMI simplifié sans modèle de durée.

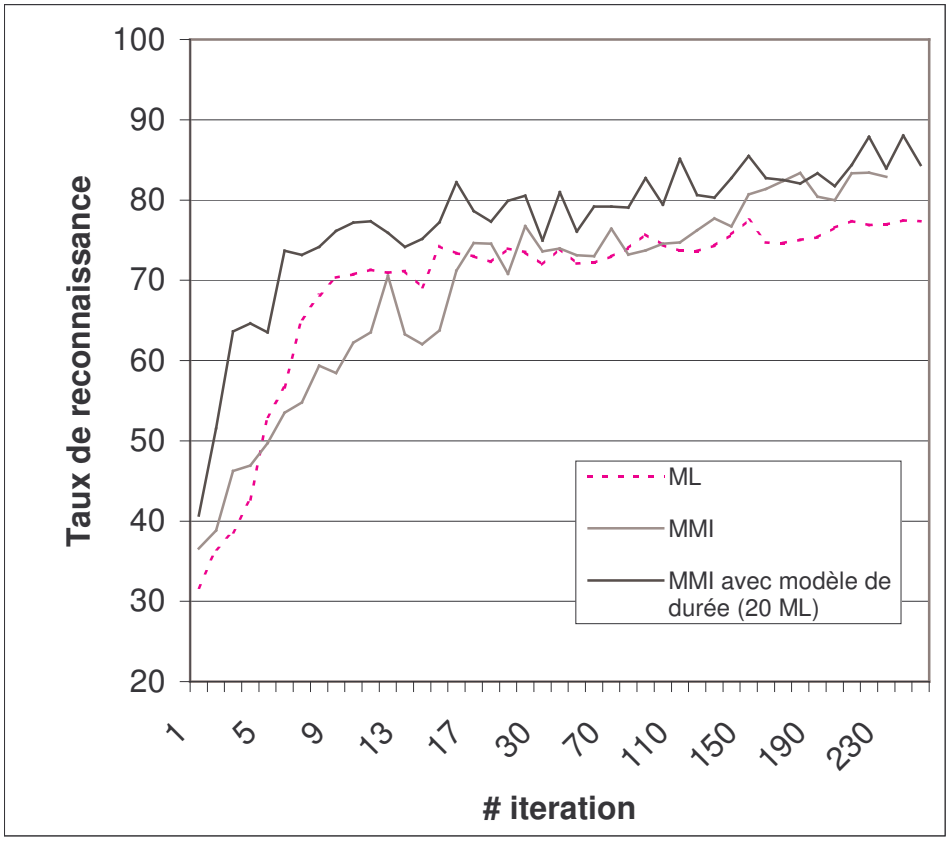


Figure 101. Intégration d'un modèle de durée sur un modèle 1 état, base IRONOFF mot

On remarque que le modèle de durée permet d'accélérer l'apprentissage du réseau et apportent un petit gain en performances. Le taux de reconnaissance dépasse dans ce cas légèrement 88 % contre 83,8 % avec un critère MMIs sans modèle de durée.

Le Tableau 39 illustre l'influence d'une initialisation de l'apprentissage par un critère ML avec modèle de durée appliqué sur 20 itérations pour des critères de Maximum de Vraisemblance, de Maximum d'Information Mutuelle et mixte. On constate que les performances avec le critère ML restent quasi-identiques, tandis qu'elles progressent significativement pour les deux autres critères (diminution du taux d'erreur de l'ordre de 20 %).

Tableau 39. Influence d'une initialisation du système par un modèle de durée, taux de reconnaissance (%) du système 1 état sur la base de test IRONOFF

Taux de reconnaissance pour chaque critère (TDNN 1 état)	ML $\varepsilon = 0$ $\beta = 0$ $\alpha = 0$	MMIs $\varepsilon = 0$ $\beta = 1$ $\alpha = 0$	MMIs-ML $\varepsilon = 1$ $\beta = 1$ $\alpha = 0$
Sans modèle de durée	77,43	83,82	86,34
Avec modèle de durée	77,61	88,02	89,04

Il serait intéressant d'analyser ces différences de comportement entre ces trois critères en les rapprochant des taux de régularité de segmentation, donnés au Tableau 40. On constate que le taux de régularité reste faible avec le critère de maximum de vraisemblance même lorsque le modèle de durée est appliqué durant la phase d'initialisation. Par contre, ce taux progresse de manière importante entre les versions, sans et avec modèle de durée, pour les deux critères discriminants.

Comme on pouvait s'y attendre, la segmentation a un impact non négligeable sur les performances de reconnaissance.

Tableau 40. Taux de régularité moyen sur la base d'apprentissage IRONOFF

Taux de régularité pour chaque critère (TDNN 1 état)	ML $\varepsilon = 0$ $\beta = 0$ $\alpha = 0$	MMIs $\varepsilon = 0$ $\beta = 1$ $\alpha = 0$	MMIs-ML $\varepsilon = 1$ $\beta = 1$ $\alpha = 0$
Sans modèle de durée	0,260	0,290	0,306
Avec modèle de durée	0,273	0,663	0,503

8.4 Conclusion

Dans ce chapitre nous avons proposé deux voies d'exploration pour optimiser les performances de reconnaissance.

Le point le plus important, et qui a apporté le plus en terme d'amélioration repose sur le passage à la topologie multi-états pour les modèles lettres. L'architecture TDNN multi-états vise à rendre le système plus robuste à la variabilité inter et intra lettre. Les résultats obtenus sur la base IRONOFF montrent qu'avec 3 états, on diminue le taux d'erreur de reconnaissance de 44 % par rapport au modèle 1 état avec le critère d'apprentissage mixte défini globalement au niveau mot. Il serait intéressant de vérifier si ce nombre de 3 états constitue une limite. On peut penser qu'avec une base plus importante, on serait capable de mieux couvrir l'espace des observations et donc d'apprendre le réseau plus complexe correspondant à 4, voire 5 états. Mais

alors, les temps d'apprentissage risquent de devenir prohibitifs puisque nous sommes déjà sur des durées de quelques jours pour la base mot IRONOFF.

La deuxième voie a été de construire un modèle de durée intervenant directement au niveau des sorties du réseau de neurones afin de mieux intégrer des connaissances *a priori* et donc de favoriser l'apprentissage du réseau. On constate une légère amélioration à la fois dans la vitesse d'apprentissage et le niveau de performances atteint. Davantages de tests seraient à mener dans ce cadre, en particulier, sa généralisation avec le modèle multi-état.

Par ailleurs, nous avons vérifié l'effet du modèle de durée sur la qualité de la segmentation grâce à la mesure du taux moyen de régularité de segmentation, il serait intéressant d'intégrer cet indicateur dans le contrôle de l'apprentissage.

CONCLUSION & PERSPECTIVES

Le fil conducteur de cette thèse a été de rechercher des techniques de reconnaissance de l'écriture naturelle en ligne offrant les meilleurs compromis performances / coût mémoire pour des interfaces de saisie embarquées dans des appareils à capacités limitées.

Nous sommes alors partis de la problématique de la reconnaissance de caractères manuscrits isolés pour la généraliser à la reconnaissance de mots cursifs en-ligne sans contrainte.

Dans ce contexte nous avons opté pour des classifieurs particuliers les réseaux de neurones à convolution. Nous avons démontré que de telles architectures (TDNN, réseau de neurones à décalage temporel ou son dual, le SDNN, réseau à décalage spatial) étaient adéquates et performantes pour des applications portables.

La reconnaissance d'unités ou de caractères isolés est une tâche complexe et encore d'actualité malgré un passé riche en travaux sur ce sujet. Nous avons étudié différentes techniques de classification basées sur les réseaux de neurones à la fois dans un cadre de reconnaissance de caractères en-ligne et hors-ligne. Les expérimentations réalisées sur des benchmarks de référence, bases UNIPEN et IRONOFF pour le cadre en-ligne et la base MNIST pour le cadre statique, montrent que les systèmes que nous avons implémentés ont des performances compétitives et architectures très intéressantes par rapport aux systèmes actuels. Dans le cas du TDNN, les architectures présentées permettent un gain de performances et d'encombrement notables par rapport à un classique réseau à couches (PMC) et rivalisent avec les performances des reconnaisseurs de type systèmes à vastes marges (SVM) avec une réduction d'un facteur de plus 50 en encombrement.

Dans le but d'accroître les taux de reconnaissance, nous avons cherché comment il était possible d'apporter au système initial de l'information complémentaire. Nous avons démontré par l'analyse des performances croisées de deux réseaux, un TDNN et un SDNN, utilisant chacun des données de nature différente (dynamique versus statique) mais provenant du même tracé que ces représentations de l'écriture étaient complémentaires.

Nous avons ensuite proposé deux méthodes pour faire coopérer ces deux classifieurs. La première par simple produit des sorties des deux classifieurs, présente l'avantage de ne pas redemander d'apprentissage mais conduit à cumuler l'encombrement des deux architectures initiales. La seconde conduit à redéfinir une architecture, appelée SDTDNN, qui comporte une couche de sortie unique, reliée à la fois aux neurones des couches supérieures des sous-parties TDNN et à ceux du SDNN.

Nous avons démontré qu'il n'était pas rédhibitoire de construire un système à deux réseaux de neurones à convolution pour des applications à ressources limitées et que l'ajout d'une représentation hors-ligne permet de diminuer entre 10 et 15% les erreurs faites en reconnaissance de caractères isolés.

La deuxième phase de nos travaux a porté sur la reconnaissance de mots manuscrits saisis en-ligne sans contrainte. Nous proposons un système hybride TDNN-MMC avec une architecture simplifiée de type reconnaissance analytique avec segmentation implicite et apprentissage global au niveau mot. La coopération TDNN-MMC consiste à utiliser le MMC pour effectuer la tâche de reconnaissance au niveau mot, en trouvant le meilleur alignement

temporel (programmation dynamique) sur les différents modèles mots du lexique à partir des probabilités d'observations estimées par le TDNN dans sa tâche de classification des entités élémentaires –liées aux fenêtres de convolution, qu'il balaie régulièrement le long de la trajectoire. Ce premier système a été développé avec une topologie HMM réduite, une importante phase d'expérimentation a montré la convergence du système.

Ce système hybride TDNN-MMC de reconnaissance original a été développé toujours dans une finalité de bon compromis performances/mémoire. Le classifieur TDNN permet notamment une réduction notoire du nombre de paramètres libres (22 606) par rapport à des approches de type perceptron multicouche classique ou machines à vastes marges mais aussi une réduction de la complexité calculatoire en exploitant la redondance de la fenêtre de balayage du signal.

L'apprentissage des systèmes hybrides combinant un réseau de neurones en amont d'une modélisation markovienne était le plus souvent basé sur une optimisation d'un critère de type modèles générateurs cherchant à maximiser la vraisemblance du vrai modèle (critère de maximum de vraisemblance MLE). A partir d'une étude plus large des systèmes hybrides, principalement en reconnaissance de la parole et de quelques systèmes récents en reconnaissance de l'écriture en-ligne, nous nous sommes intéressés à d'autres modes d'apprentissage par maximum d'information mutuelle (MMI) et minimum d'erreur de classification (MCE).

Dans une logique descendante, nous avons implémenté et comparé ces différentes méthodes d'entraînement du système. Les résultats obtenus avec un critère MMI, (83,8% de reconnaissance des mots de la base IRONOFF en première position contre 77,4% avec un critère MLE) montrent l'intérêt d'introduire un apprentissage discriminant au niveau mot. Cependant, par une analyse fine des exemples non reconnus, on constate que ce critère n'est pas complètement satisfaisant : le système ne modélise pas correctement les variabilités inter et intra lettres. En réponse, nous avons construit une fonction de coût générique combinant à la fois un critère MLE et un critère MMI permettant de discriminer le modèle mot HMM reconnu à partir d'un lexique et/ou des sorties du TDNN mal classées. Cette fonction est paramétrée par trois variables qui permettent de la spécialiser sur un critère particulier ou de combiner et pondérer certains critères.

Les approches neuro-markoviennes permettent d'introduire un apprentissage discriminant local et apportent une première réponse intéressante au problème de la discrimination (MMI, combinaison MMI-MLE). Nous avons montré que l'introduction d'un critère plus local tel qu'une discrimination des sorties du TDNN par rapport au modèle vrai donné par Viterbi permet de renforcer l'apprentissage, tant dans sa phase d'initialisation qu'en terme de reconnaissance. Avec une pondération égale entre une discrimination au niveau mot et au niveau des classes caractères, l'erreur de classification des mots est réduite d'environ 40% par rapport à des critères classiques de maximum de vraisemblance. Différentes améliorations du système ont été présentées notamment l'adaptation de la modélisation markovienne : extension à un TDNN multi-états, modélisation de la durée. Les différents tests effectués ont montré l'intérêt d'une structuration TDNN-MMC du système de reconnaissance de mots en-ligne et du schéma d'apprentissage final proposé. Les différentes optimisations en termes de fonction d'entraînement, topologie, modélisation markovienne ont permis d'améliorer les performances de l'ensemble de façon notable.

Perspectives

Avant d'aborder les perspectives nouvelles, on peut déjà souligner les travaux en cours. Notamment, les résultats présentés au chapitre 7 sur la base IRONOFF incluant les différents critères d'apprentissage du système de reconnaissance mot sont étendus à la base de mots UNIPEN. Cette base nécessite un temps d'apprentissage beaucoup plus long. En effet elle contient environ le double d'exemples et surtout un dictionnaire de 11 957 mots soit 60 fois plus que celui de la base IRONOFF. Une seconde étude, menée en parallèle, met en oeuvre un critère d'apprentissage du réseau plus large qu'une simple discrimination par rapport au mot de plus forte vraisemblance. Pour cela, nous avons construit une fonction de coût permettant la discrimination de tout ou partie des mots du lexique de vraisemblance supérieure à celle du mot vrai.

Au-delà, un certain nombre d'améliorations et de perspectives de travaux se dégagent de ce système hybride de reconnaissance de mots cursifs en-ligne.

La première à réaliser est la structuration du dictionnaire. En effet, dans le système actuel, le dictionnaire est parcouru séquentiellement avec un balayage complet. Une telle stratégie n'est plus envisageable dès lors que la taille du dictionnaire dépasse quelques milliers de mots. Il faut alors s'orienter vers des techniques combinant élagage, sur des critères morphologiques, représentation par *Trie* (arbre lexical avec factorisation des préfixes et/ou des suffixes) et algorithmes de type N-meilleurs (*N-best*) pour n'explorer qu'un sous-ensemble des solutions de taille constante et être indépendant de la taille du lexique. L'effet serait bénéfique, non seulement en phase de reconnaissance, mais aussi en apprentissage où de très nombreuses itérations de la base d'apprentissage sont nécessaires. Pour entraîner notre système de reconnaissance de mots sur la base IRONOFF (20 898 exemples en apprentissage avec un lexique de 197 mots), les temps de simulation sont de l'ordre d'une semaine sur un Pentium IV à 2,7 GHZ.

Nous avons énuméré un certain nombre de points faibles dans le schéma complet, comme l'absence de gestion des marques diacritiques qui font exploser le nombre de sorties du TDNN et bruitent à la fois la reconnaissance des unités élémentaires et celles des mots. Nous avons aussi souligné qu'un bon nombre d'exemples était perdu à cause d'une détection des lignes de référence non propice aux petits mots. Le module de prétraitement reste à améliorer.

Pour l'apprentissage du système hybride TDNN/MMC, nous avons défini une fonction de coût générique basé sur trois paramètres α, β, et ε combinant des critères de maximum de vraisemblance (MLE) et de discrimination, tels que le maximum d'information mutuelle (MMI) et le minimum d'erreur de classification (MCI). Seulement un petit nombre de combinaison a été testé : un critère de maximum de vraisemblance (ε=0, β=0, α=0), un critère MMI simplifié (ε=0, β=1, α=0), un critère mixte ML-MMI (ε=1, β=1, α=0), un critère de discrimination au niveau caractère (ε=1, β=1, α=1) $\varepsilon\tau$ un critère plus général équilibrant la pondération entre la discrimination avec ou sans lexique (ε=1, β=1, α=0.5). Il serait intéressant d'analyser l'impact de chaque paramètre par une série d'expérimentations manuelles voire élaborer un algorithme d'optimisation des paramètres.

Une autre d'optimisation, actuellement en cours d'expérimentation, concerne un apprentissage du réseau plus large qu'une simple discrimination par rapport au mot de plus forte vraisemblance. Pour cela, nous avons construit une fonction de coût permettant la

discrimination de tout ou partie des mots du lexique de vraisemblance supérieure à celle du mot vrai.

Il pourrait être envisagé d'une part d'optimiser le nombre d'états lettre par lettre, de travailler sur les modèles de durée, et cela de façon dynamique au cours de l'apprentissage afin d'introduire d'abord un *a-priori* assez fort. D'autre part, il serait intéressant de faire évoluer la fonction de coût générique, en jouant au fur et à mesure des itérations sur les paramètres de contrôle.

La combinaison TDNN/SDNN étudiée en reconnaissance de caractères isolés a montré que son architecture était adéquate pour des applications portables, et qu'elle permettait de corriger certaines confusions du TDNN, il serait alors intéressant d'insérer cette combinaison de classifieurs dans notre schéma de reconnaissance mot.

D'un point de vue formel, les systèmes hybrides classiques s'appuient sur un premier étage, le réseau de neurones (RN) qui calcule les probabilités *a posteriori* des classes $P(C|O)$ et qui sont ensuite converties en vraisemblances normalisées $b_j(O) = \hat{P}(O|C)$ pour finaliser le calcul au niveau MMC. Dans l'approche proposée ici, nous n'assignons pas au TDNN le même rôle, son travail consiste davantage à modéliser les densités de probabilités (*pdf*) dans l'espace des observations. On ne peut donc le considérer comme un reconnaisseur de caractères. D'ailleurs, la séquence des étiquettes les mieux classées par le TDNN n'est pas toujours très explicite par rapport au résultat de la reconnaissance mot. D'un point de vue macroscopique, nous avons en fait conçu un système pour lequel nous avons spécifié une fonction de coût dont la minimisation agit sur les poids du TDNN. Les grandeurs de sortie du TDNN peuvent être considérées comme des variables internes qui n'ont pas forcément une interprétation explicite évidente. D'une manière plus générale, l'hypothèse que le TDNN calcule une densité de probabilité peut être oubliée, il suffit de préserver une gradation bornée des valeurs. Ceci est obtenu grâce à la topologie contrainte du TDNN et la normalisation des sorties par la fonction *SoftMax*. Si l'on était intéressé à mieux relier ces variables internes aux densités de probabilités des classes, on pourrait envisager des expériences sur de petits exemples académiques, partant de *pdf* connues, pour mieux établir le lien entre ces variables internes et les vraies *pdf*.

Nous souhaitons par la suite coupler un modèle de langage pour étendre notre système à la reconnaissance au niveau phrase.

BIBLIOGRAPHIE

[Ahmad, 2004] A.R. Abdul, M. Khalid**,** E. Poisson and C. Viard-Gaudin. "Online Handwriting Recognition using Support Vector Machine". *IEEE TENCON Analog and Digital Techniques for Electrical Engineering*. Thailand, Novembre 2004.

[Alimoglu, 1997] F. Alimoglu, E. Alpaydin. "Combining Multiple Representations and Classifers for Pen-based Handwritten Digit Recognition", In proceedings of *International Conference on Document Analysis and Recognition (ICDAR'97)*, pages 637-660, Ulm, Août 1997.

[Alleau, 2003] F. Alleau, E. Poisson, C. Viard-Gaudin et P. Le Callet. "TDNN with Masked Inputs", in *Proc. of the* 4th *International Conference on Information, Communications & Signal Processing and IEEE Pacific-Rim Conference on Multimedia (ICICS-PCM 2003),* Nanyang Technological University*,* Singapour, December 2003.

[Alpaydin, 2000] E Alpaydin, C Kaynak, F Alimoglu. "Cascading Multiple Classifiers and Representations for Optical and Pen-Based Handwritten Digit Recognition", in *International Workshop on Frontiers in Handwriting Recognition* (*IWFHR'00*), Amsterdam, The Netherlands, September 2000.

[Amari, 1998] S. Amari : "Natural gradient works efficiently in learning.", *Neural Computation*, volume 10, issue 2, pages 251276, 1998.

[Anquetil, 1997] E. Anquetil. « *Modélisation et reconnaissance par logique floue : Application à la lecture en-ligne de l'écriture manuscrite omni-scripteur »*. Thèse, Université de Rennes I, 1997.

[Artieres, 2002] T. Artieres, P. Gallinari. "Stroke Level HMMs for On-Line Handwriting Recognition," *International Workshop on Frontiers in Handwriting Recognition*, *IWFHR'2002*, pages 227. 2002.

[Bahl, 1986] L.R. Bahl, P.F. Brown, P.V. de Souza, and R. L. Mercer. "Maximum mutual information estimation of hidden Markov model parameters for speech recognition". In Proceedings *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 49-52, Tokyo, 1986.

[Bahlmann, 2001] C. Bahlmann and H. Burkhardt. "Measuring HMM Similarity with the Bayes Probability of Error and its Application to Online Handwriting Recognition" In proceedings of *International Conference on Document Analysis and Recognition (ICDAR'01)*, Seattle, WA, September 2001.

[Bandi, 2005] K. Bandi and S. N. Srihari, "Writer Demographic Identification using Bagging and Boosting," *Proc. International Graphonomics Society Conference (IGS)*, pages 133-137, Salerno, Italy, June 2005.

[Belaïd, 1994] A. Belaïd et J. C. Anigbogu. « Mise à contribution de plusieurs classifieurs pour la reconnaissance de textes multifontes. » Revue *Traitement du signal*, volume 11, n°2, 1994.

[Bellili, 2002] A. Bellili, M. Gilloux, P. Gallinari , « Combinaison hybride MLP-SVM pour la reconnaissance de chiffres manuscrits », Actes du *13ème Congrès Francophone AFRIF-AFIA de Reconnaissance des Formes et Intelligence Artificielle*, 2002.

[Bengio, 1993] Y. Bengio. "A Connectionist Approach to Speech Recognition". *International Journal on Pattern Recognition and Artificial Intelligence*, volume 7, issue 4, pages 647-668, 1993. (disponible sur la toile : http://www.iro.umontreal.ca/~lisa/bib/pub_subject/speech/)

[Bengio, 1994] Y. Bengio and Y. Le Cun. "Word normalization for on-line handwritten word recognition". In Proceedings of *International Conference on Pattern Recognition*, pages 409-413, Jerusalem, 1994.

[Bengio, 1995] Y. Bengio, Y. LeCun, C. Nohl, and C. Burges, "LeRec: A NN/HMM Hybrid for On-Line Handwriting Recognition," *Neural Computation*, vol. 7, no. 6, pages 1289-1303, Nov. 1995. [14 pages]

[Bensefia, 2003] A. Bensefia, T. Paquet, L. Heutte. "Information retrieval based writer identification". 11th Conference of the *International Graphonomics Society*, *IGS'2003*, Scottsdale, Arizona, pages 320-323, 2003.

[Bensefia, 2004] A. Bensefia, T. Paquet, L. Heutte. „Handwriting analysis for writer verification". *9th IAPR International Workshop on Frontiers in Handwriting Recognition, IWFHR'2004*, Tokyo, Japan, IEEE Proceedings, pages 196-201, 2004.

[Bahl, 1983] L.R. Balh, F. Jelinek, L. Mercer. *"A Maximum likelihood approach to continuous speech recognition"*, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume PAMI-5, pages 179-190, mars 1983.

[Bishop, 1995] C. M. Bishop, *Neural Networks for Pattern Recognition*, Clarendon Press, Oxford, 1995.

[Boiteau, 1993] D. Boiteau, P. Haffner, "Connectionist segmental post-processing of the n-best solutions in isolated and connected word recognition task", In proc. of the *European Conference on Speech Communication and Technology (EUROSPEECH'93)* , pages 1933-1936. Berlin, September 22-25, 1993.

[Bottou, 1991] Léon Bottou, "*Une Approche théorique de l'Apprentissage Connexionniste: Applications à la Reconnaissance de la Parole*", Thèse, Université de Paris XI, Orsay, 1991.

(disponible sur la toile : http://leon.bottou.com/publications/)

[Bourlard, 1995] H. Bourlard, Y. Konig and N. Morgan, "REMAP : Recursive Estimation and Maximization of A Posteriori Probabilities in connectionist speech recognition", In proceddings of the *European Conference on Speech Communication and Technology (EUROSPEECH '95),* Madrid, Sept. 1995. (disponible sur la toile : http://citeseer.ist.psu.edu/bourlard95remap.html)

[Bourlard, 1998] H. Bourlard and N. Morgan. "Hybrid HMM/ANN Systems for Speech Recognition: Overview and New Research Directions," in *Adaptive Processing of Sequences and Data Structures*, C.L. Giles and M. Gori (Eds.), Lecture Notes in Artificial Intelligence (1387), Springer Verlag (ISBN 3-540-64341-9), pages 389-417. 1998.

[Bourlard, 1989] H. Bourlard, C.J. Wellekens**, "**Speech Dynamics and Recurrent Neural Networks", *IEEE Conference on Acoustics, Speech & Signal Processing ICASSP-89*, volume 1, pages 33-36, Glasgow (UK). 1989.

[Breiman, 1996] L. Breiman. "Bagging predictors". *Machine Learning*, Volume 24, n°2, pages 123-140, 1996 (disponible sur la toile : http://citeseer.ist.psu.edu/breiman96bagging.html).

[Bridle, 1989] J. Bridle. "Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition". In F. Fogelman-Soulie & J. Herault (Eds.), *Neurocomputing: algorithm, architectures, and applications*, pages 227-236. New York: Springer. 1989.

[Bridle, 1990] J.S. Bridle "Training Stochastic Model Recognition Algorithms as Networks Can Lead to Maximum Mutual Information Estimation of Parameters", in *Advances in Neural Information Processing Systems 2*, D.S. Touretzky, ed., pages 211-217, 1990. (Morgan Kaufman).

[Brown, 2001] G Hinton and AD Brown. "Training Many Small Hidden Markov Models*"*. In proceedings of *Institute of Acoustics Workshop on Innovation in Speech Processing*, pages 1-17, Stratford-upon-Avon, 2001. (disponible sur la toile : http://www.cs.toronto.edu/~hinton/chronological.html)

[Brown, 2002] A. Brown, G. Hinton. "Relative Density Nets: A New Way to Combine Backpropagation with HMM's". Advances in *Neural Information Processing Systems*, 14, MIT Press, Cambridge, MA, 2002. (disponible sur la toile : http://www.cs.toronto.edu/~hinton/chronological.html)

[Bouché, 2000] R. Bouché, H. Emptoz, F. LeBourgeois , « DEBORA -Digital accEs to Books of RenaissAnce », Présentation de DEBORA à *Colloque International Francophone sur l'Ecrit et le Document* (CIFED'00). Lyon (France), juin 2000. (Ouvrage en ligne de 167 pages disponible sur la toile : rfv6.insa-lyon.fr/debora)

[Burges, 1998] C. J. C. Burges. "A Tutorial on Support Vector Machines for Pattern Recognition". *Knowledge Discovery and Data Mining*, volume 2 (2), 1998.

[Caillault, 2005] E. Caillault, C. Viard-Gaudin and P.M. Lallican. "Training of hybrid ANN/HMM systems for on-line handwriting word recognition", *Proceedings of 12th Conference of International Graphonomics Society (IGS 2005)*, Salerno, Italie, juin 2005.

[Caillault, 2005b] E. Caillault, C. Viard-Gaudin, A. Ahmad, "MS-TDNN with Global Discriminant Trainings Recognition", *International Conference on Document Analysis and Recognition, ICDAR 2005*, Volume II, pages 856-860, Seoul, Korea, Aug 2005.

[Carbonnel, 2004] S. Carbonnel, E. Anquetil, "Lexicon Organization and String Edit Distance Learning for Lexical Post-Processing in Handwriting Recognition"**,** *in 9th International Workshop on Frontiers in Handwriting Recognition (IWFHR 9),* Pages 462-467, Octobre 2004.

[Chen, 2000] W.T. Chen, P. Gader, "Word level discriminative training recognition for handwritten word recognition", *in 7th International Workshop on Frontiers in Handwriting Recognition*, IWFHR'00, ISBN 90-076942-01-3, 2000, pages 393-402.

[Choisy, 2002a] C. Choisy et A. Belaid. « Couplage d'une vision locale par HMM et globale par RN pour la reconnaissance de mots manuscrits ». Actes de la *Conférence Internationale Francophone sur l'Ecrit et le Document - CIFED'02*. (Hammamet, Tunisie). 2002. 10 pages.

[Choisy, 2002b] C. Choisy et A. Belaid. « Normalisation par modèles locaux et reconnaissance par modèles globaux pour la reconnaissance de l'écriture manuscrit ». In *13e Congrès Francophone AFRIF-AFIA de Reconnaissance des Formes et Intelligence Artificielle - RFIA 2002*. (Angers, France). 2002. 8 pages.

[Choisy, 2003] C. Choisy and A. Belaïd. "Coupling of a local vision by markov field and a global vision by neural network for the recognition of handwritten words". *International Conference on Document Analysis and Recognition, ICDAR'03*, volume 2, pages 849–853, 2003.

[Connell, 2001] S.D. Connell, A. K. Jain. "Template-based online character recognition" *Pattern Recognition* volume 34 issue 1, pages 1-14 (2001).

[Côté, 1996] M. Côté, M. Cheriet, E. Lecolinet, C.Y. Suen. « Détection des lignes de base de mots cursifs à l'aide de l'entropie », *Actes du congrès de l'Association canadienne-française pour l'avancement de la science*, pages 183-193, Montréal, Canada, mai 1996.

[Côté, 1997] M. Côté, « *Utilisation d'un modèle d'accès lexical et de concepts perceptifs pour la reconnaissance d'images de mots cursifs* », Thèse de l'Ecole Nationale Supérieure des Télécommunications (ENST), Juin 1997.

[Coüasnon, 2002] B. Coüasnon, J. Camillerapp, « DMOS. une méthode générique de reconnaissance de documents : évaluation sur 60 000 formulaires du XIXe siècle »**,** in *Actes du Colloque International Francophone sur l'Écrit et le Document (CIFED'02),* Hammamet, Octobre 2002.

[Egmont-Peterson, 1994] M. Egmont-Peterson, J.L. Talmon, J. Brender, P. NcNair : "On the quality of neural net classifiers," *Artificial Intelligence in Medecine*, Volume 6, N°5 pages 359-381, 1994.

[Ephraim,1990] Y. Ephraim and L. R. Rabiner, "On the relations between modeling approaches for speech recognition," *IEEE Trans. Inform. Theory*, vol. IT-36, pp. 372-380, Mar. 1990. (disponible sur la toile : http://ece.gmu.edu/~yephraim/ephraim.html)

[Flake, 1998] G. W. Flake, "Square Unit Augmented, Radially Extended, Multilayer Perceptrons", *Neural Networks: Tricks of the Trade*, book of 1996 NIPS workshop, pages 145-163, January 1998.

[Freund, 1996] Y. Freund & R.E. Schapire (1996), "Experiments with a New Boosting Algorithm", in Proceedings of the Thirteenth International Conference on Machine Learning, pages 148-156, Morgan Kaufmann. 1996. (disponible sur la toile : http://citeseer.ist.psu.edu/freund96experiments.html)

[Fujisaki, 1971] Fujisaki, H., Nagai, S. and Hidaka, N. "On-line recognition of hand-written numerals", *Annual Report of the Engineering Research Institute*, Faculty of Engineering, University of Tokyo, Japan, Vol 30, pages 103-110 August 1971. (cité dans [Tappert, 1990]).

[Garcia-Salicetti, 1996] S. Garcia-Salicetti. « *Une approche neuronale prédictive pour la reconnaissance en-ligne de l'écriture cursive* », thèse de l'Université Paris 6.

[Gentric, 1998] S. Gentric et M. Milgram. « Reconnaissance de mots cursifs et génération non supervisée de sous-classes », *Actes du 1er Colloque International Francophone sur l'Ecrit et le Document (CIFED'98)*, mai 1998.

[Gentric, 2000] S. Gentric, L. Prevost, M. Milgram. « Un classifieur neuronal évolutif dédié à la génération automatique de sous-classes ». *Application à la reconnaissance d'écriture dynamique. CAP' 2000 (Colloque Francophone sur l'Apprentissage Automatique)*, pages 147-156, Grenoble, 2000.

[Gilloux, 1996] M. Gilloux, « Reconnaissance de chiffres manuscrits par modèle de Markov pseudo-2D », *Traitement du Signal*, 1996.

[Gilloux, 1998] M. Gilloux, « Réduction dynamique du lexique par la méthode Tabou », *Actes du 1er Colloque International Francophone sur l'Ecrit et le Document (CIFED'98)*, pages 24-31, mai 1998.

[Gorski, 2001] N. Gorski, V. Anisimoy, E. Augustin, O. Baret, S. Maximov. "Industrial bank check processing: the A2iA CheckReader^{TM}". *IJDAR* volume 3 issue 4: pages 196-206, 2001.

[Guillevic, 1997] D. Guillevic and C. Y. Suen. "HMM word recognition engine". In *Proceedings Int. Conf. on Document Analysis and Recognition*, pages 544-547, 1997.

[Günter, 2003] S. Günter, H. Bunke. "Fast Feature Selection in an HMM-based Multiple Classifier System for Handwriting Recognition", in B. Michaelis, G. Krell : *Pattern Recognition*, Proceedings of the 25th DAGM Symposium, LNCS 2781, Springer, pages 289-296, 2003

[Guo, 1993] J. Guo and H.C. Lui, "A Multilayer Perceptron Postprocessor to Hidden Markov Modeling for Speech recognition", *Proc. of IEEE Conference on Acoustics, Speech and Signal Processing (ICASSP-93)*, volume II, pages 263-266. Minneapolis, April 27-30, 1993.

[Guyon, 1991] 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*, volume 24, issue 2, pages 105-119, 1991.

[Guyon, 1994] I. Guyon, L. Schomaker, R. Plamondon, M. Liberman, S. Janet. "UNIPEN project of on-line data exchange and recognizer benchmarks", *Proceedings of the 12th International Conference on Pattern Recognition, ICPR'94*, pages 29-33, Jerusalem, Israel, October 1994.

[Guyon, 1995] I. Guyon, J. Bromley, N. Matic , M. Schenkel, and H. Weissman. "Penacee: A neural network system for recognizing on-line handwriting". In E. Domany, J. L. van Hemmen, and K. Schulten, editors, *Models of Neural Networks*, volume 3, pages 255-279, Springer. 1995

[Hérault, 1994] J. Hérault et Ch. Jutten, « *Réseaux neuronaux et traitement du signal* » Hermes, 1994.

[Héroux, 1998] P. Héroux, S. Diana, A. Ribert et E. Trupin. « Étude de méthodes de classification pour l'identification automatique de classes de formulaires ». *Proceedings de la 1ère conférence international sur l'Écrit et le Document (CIFED 1998)*. Pages 463-472. Canada, Québec. 11-13 mai, 1998.

[Hochberg, 1995] M.M. Hochberg, G.D. Cook, S.J. Renals, A.J. Robinson, R.S. Schechtman. "The 1994 ABBOT Hybrid Connectionist-HMM Large-Vocabulary Recognition System". In *Spoken Language Systems Technology Workshop*, pages 170-176, ARPA, January 1995. (disponible sur la toile : http://citeseer.ist.psu.edu/hochberg95abbot.html).

[Ho, 1998] T.K. Ho. "The random subspace method for constructing decision forests." *IEEE Trans. on PAMI*, volume 20, n° 8, pages 832-844, 1998.

[Hu, 1996] J. Hu, M.K. Brown, W. Turin. "HMM Based On-Line Handwriting Recognition". *IEEE Trans. Pattern Analysis and Machine Intelligence*. Volume 18 n°10, pages 1039-1045, 1996.

[Hu, 1997] J. Hu , A.S. Rosenthal , M.K. Brown. "Combining High-Level Features with Sequential Local Features for On-Line Handwriting Recognition". *Proceedings of the 9th International Conference on Image Analysis and Processing*-Volume II, pages 647-654, September 17-19, 1997

[Hu, 2000] J. Hu, S.G. Lim and M.K. Brown, "Writer independent on-line handwriting recognition using an HMM approach" *Pattern Recognition*, January 2000.

[Huteau, 2004] M. Huteau. « Écriture et personnalité », *Approche critique de la graphologie*. Dunod Collection, Psycho Sup, ISBN : 2100485474, 272 pages. 2004.

[Jaeger, 2000] S. Jaeger, S. Manke, J. Reichert, A. Waibel, "On-Line Handwriting Recognition: The NPen++ Recognizer", In *International Journal on Document Analysis and Recognition (IJDAR'00)*, volume 3, pages 169-180, 2000.

[Jarousse, 1998] C. Jarousse, C. Viard-Gaudin,. « Localisation du code postal par réseau de neurones sur bloc adresse manuscrit non contraint », *1ère conférence international sur l'Écrit et le Document (CIFED'98)*. Pages 72-81. Canada, Québec. 11-13 mai, 1998.

[Knerr, 1998] S. Knerr, E. Augustin, "A Neural Network-Hidden Markov Model Hybrid for cursive Word Recognition", *ICPR*, 1998.

[Koerich, 2003] A.L. Koerich, R.Sabourin, C.Y. Suen, "Large vocabulary off-line handwriting recognition : A survey". *Pattern Analysis and Applications*, volume 6, pages 97–121, 2003.

[Lallican, 1999] P.-M. Lallican, « *Reconnaissance de l'Écriture Manuscrite Hors-ligne : Utilisation de la Chronologie Restaurée du Tracé* ». Thèse soutenue le 24 novembre 1999. Université de Nantes.

[Lallican, 2000] P-M. Lallican, C. Viard-Gaudin, S. Knerr, « From Off-line to On-line Handwriting Recognition », *IWFHR'2000*, Amsterdam, Netherlands, pages 303-312, Sept. 11-13, 2000.

[Le Callet, 2005] P. Le Callet, C. Viard-Gaudin, D. Barba. "Continuous quality assessment of MPEG2 video with reduced reference", *Workshop on Video Processing and*

Quality Metrics for Consumer Electronics, VPQM 2005, Scottdale, Arizona, Jan. 2005.

[Le Cerf, 1994a] P. Le Cerf, W. Ma and D. Van Compernolle. "Multilayer Perceptrons as Labelers for Hidden Markov Models". *IEEE Transactions on Speech and Audio Processing*, volume 2, No. 1, pages 185-193, January 1994. (Special Issue on Neural Networks for Speech Processing).

[Le Cerf, 1994b] P. Le Cerf and D. Van Compernolle. "Using MLPs as Probability Generators vs. as Labelers: A Comparative Study". In *Proc. EUSIPCO*, pages 1629--1632, Edinburgh, U.K., September 1994. (Disponible sur la toile : http://www.esat.kuleuven.ac.be/~spch/)

[LeCun, 1987] Y. LeCun, « *Modeles connexionnistes de l'apprentissage (connectionist learning models)* », [277 pages] June 1987.

[LeCun , 1990] Y. LeCun, 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 Lisboa P.G.J., editor, *Neural Netwotks, current applications*. Chappman and Hall, 1992.

[LeCun, 1993] Y. LeCun, P. Simard, and B. Pearlmutter : "Automatic learning rate maximization by on-line estimation of the hessian's eigenvectors". In S. Hanson, J. Cowan, and L. Giles, editors, *Advances in Neural Information Processing Systems*, volume 5. Morgan Kaufmann Publishers, San Mateo, CA, 1993.

[LeCun, 1995] Y. LeCun and Y. Bengio : "Convolutional networks for images, speech, and time-series". In M. A. Arbib, editor, *The Handbook of Brain Theory and Neural Networks*. MIT Press, 1995.

[LeCun, 1998a] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-Based Learning Applied to Document Recognition", *Proceedings of the IEEE*, vol. 86, no. 11, pages 2278-2324, Nov. 1998. (disponible sur la toile : http://yann.lecun.com/exdb/publis/#lecun-98).

[LeCun, 1998b] [LeCun, 2001] Y. LeCun, L. Bottou, G. Orr, and K. Muller, "Efficient BackProp", in *Neural Networks: Tricks of the trade*, (G. Orr and Muller K., eds.), [44 pages] 1998.

[Lee, 2001] J.J. Lee, J. Kim J. and J.H. Kim. "Data-driven design of HMM topology for on-line handwriting recognition", *International Journal of Pattern Recognition and Artificial Intelligence*, Volume 15, n° 1, pages 107-121.

[Leroux, 1991] M. Leroux. « *Reconnaissance de textes manuscrits à vocabulaire limité avec application à la lecture automatique des chèques* ». Thèse de doctorat de l'université de Rouen, 1991.

[Liwicki, 2005] M. Liwicki and H. Bunke. "Handwriting Recognition of Whiteboard Notes". *Proceedings of the 12th Biennal Conference of the International Graphonomics Society (IGS 2005)*. ISBN 88 89702 13 3. Edited by A. Marcelli and C. De Stefano. University of Salerno, Italy, pages 118-122, June 26-29, 2005.

[Manke, 1995] S. Manke , M. Finke , A. Waibel. “Npen++: a writer independent, large vocabulary on-line cursive handwriting recognition system”, *Proceedings of the Third International Conference on Document Analysis and Recognition* (Volume 1), pages 403, August 14-15, 1995.

[Lam, 1995] L. Lam, C. Y Suen, “Optimal combination of pattern classifiers”, *Pattern Recognition Letters*, 16, p. 945-954, 1995.

[Lyon, 1996] R. F. Lyon and L. S. Yaeger. “On-Line Hand-Printing Recognition with Neural Networks”. *Proceedings of the Fifth International Conference on Microelectronics for Neural Networks and Fuzzy Systems*, pages 201-212, Lausanne, Switzerland, February 12-14, 1996.

[Mari, 1994a] J.F. Mari . “Hidden Markov Models and Selectively Trained Neural Networks for Connected Confusable Word Recognition”, *Proc. ICSLP-94*, Yokohama, pages 1519-1522, 1994.

[Mari, 1994b] J.F. Mari and J.P. Haton. “Automatic Word Recognition Based on Second-order Hidden Markov Models”, *Proc. ICSLP-94*, Yokohama, pages 247-250, 1994.

[Marti, 2000] U. Marte, H. Bunke. “Handwritten Sentence Recognition”. *Intern. Conf. On Pattern Recognition (ICPR)*, volume 3, pages 467-470, Barcelona, Spain, 2000.

[Marukatat, 2004] S ; Marukatat. « *Une approche générique pour la reconnaissance de signaux écrits en ligne* ». Thèse de doctorat de l’université de l’Université Paris 6, 2004.

[McClelland, 1986] J.L. McClelland and D.E. Rumelhart. “Parallel Distributed Processing. Explorations in the Microstructure of Cognition”, volume 2: *Psychological and Biological Models*. Cambridge, MIT Press. 1986.

[McCulloch, 1943] W.S. McCulloch, W. A. Pitts, “A logical calculus of the ideas immanent in nervous activity”, *Bull. Math. Biophys*. Volume 5, pages 115-133, 1943.

[Minoux, 1983] M. Minoux : Livre «Programmation Mathématique, Théorie et algorithmes » tome 1, pages 95-168 – *Collection Technique et Scientifique des Télécommunications* – ENST, édition DUNOD 1983.

[Minski, 1969] M. Minski, S. Papert. “*Perceptrons*”, MIT Press, Cambridge (MA) , 1969.

[Nathan, 1995] K.S. Nathan, H.S.M. Beigi, H. Subrahmonia, G.J. Clare, and H. Maruyama, “Real-Time OnLine Unconstrained Handwriting Recognition Using Statistical Methods”, *Proc. Int’l Conf. Acoustics, Speech and Signal Processing (ICASSP '95)*, volume 4, pages 2619-2623, 1995.

[Nicolas, 2003] S. Nicolas, T. Paquet, L. Heutte. “Digitizing cultural heritage manuscripts: the Bovary project”. *ACM Symposium on Document Engineering, ACM Doc Eng 2003*, Grenoble, France, pages 55-57, 2003. (disponible sur la toile : http://www.univ-rouen.fr/psi/heutte/reference.html)

[Nosary, 2002] A. Nosary, L. Heutte, T. Paquet. « Reconnaissance de mots manuscrits par segmentation-reconnaissance: apports d'une reconnaissance lettres par niveau avec rejet »; *Colloque International Francophone sur l'Ecrit et le Document (CIFED'02)* ; pages 355-364, Hammamet, Tunisie, 20-23 Octobre 2002.

[Opitz, 1999] D. Opitz , R. Maclin. “Popular Ensemble Methods: An Empirical Study”, in *Journal of Artificial Intelligence Research,* volume 11, pages 169-198, 1999. disponible sur la toile : http://www.jair.org/contents/v11.html

[Oudot, 2003] L. Oudot. « *Fusion d’informations et adaptation pour la reconnaissance de textes manuscrits dynamiques* », Thèse de doctorat, Université Pierre et Marie Curie, Paris, 2003.

[Oudot, 2004] L. Oudot, L. Prevost and M. Milgram, “An activation-verification model for on-line handwriting texts recognition”, in *Proc. of International Workshop on Frontiers for Handwriting Recognition (IWFHR’9)*, Tokyo, Japan, 2004.

[Oudot, 2004b] L. Oudot & L. Prevost, « *Techniques de coopération pour la reconnaissance d’écriture en contexte* », RIA'04 (Revue Intelligence Artificielle), 2004. (disponible sur la toile : http://loic.oudot.free.fr/Publi.php).

[Parizeau, 2001] Marc Parizeau, Alexandre Lemieux, Christian Gagné: “Character Recognition Experiments Using Unipen Data”. pages 481-485. *ICDAR 2001*.

[Pasquer, 2000] L. Pasquer. « *Conception d'un modèle d'interprétation multi-contextuelle, application à la reconnaissance en-ligne d'écriture manuscrite* ». Thèse de doctorat. Université de Rennes 1, Janvier 2000.

[Perraud, 2003] F. Perraud, E. Morin, C. Viard-Gaudin et P.M. Lallican. « Modèles n-grammes et n-classes pour la reconnaissance de l'écriture manuscrite en ligne ». *Traitement Automatique des Langues (TAL) / Modélisation probabiliste du Langage naturel*,*vol. 44*,(1).2003.

[Perraud, 2005] F. Perraud, C. Viard-Gaudin, E. Morin, P.-M. Lallican, “Statistical Language Models for On-Line Handwriting Recognition” *IEICE Transactions on Information and Systems/Document Image Understanding and Digital Document,* Vol.E88-D No.8 pages 1807-1814 2005/8.

[Plamondon, 2000] R. Plamondon, S.N. Srihari. “On-Line and Off-Line Handwriting Recognition: A Comprehensive Survey”. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 22, issue 1, pages: 63-84. ISSN:0162-8828, Janvier 2000.

[Poisson, 2001] E. Poisson et C. Viard-Gaudin. « Réseaux de neurones à convolution et reconnaissance de l'écriture manuscrite non contrainte », *Revue VALGO*, In *Valgo numéro 01-02, La revue en ligne de l'Association des Connexionnistes en THèse,* Association des Connexionnistes en Thèse (ACTH), ISSN 1625-9661, pages 726-730, *Montélimar, France*, Octobre 2001.

[Poisson, 2002] E. Poisson, C. Viard-Gaudin et P.M. Lallican. « Combinaison et analyse de réseaux de neurones à convolution pour la reconnaissance de caractères manuscrits en-ligne », *CIFED'02 - Colloque International Francophone sur l'Ecrit et le Document*, Hammamet, Tunisie, Octobre 2002.

[Poisson, 2002b] E. Poisson, C. Viard-Gaudin et P.M. Lallican. “Multi-modular architecture based on convolutional neural networks for online handwritten character recognition”, in *Proc. of ICONIP'02 - 9th International Conference on Neural Information Processing*, IEEE Neural Network Society, ISBN 981-04-7536-5, Singapour, Novembre 2002.

[Poisson, 2004] **E. Poisson**, C. Viard-Gaudin et P.M. Lallican. « Système TDNN/HMM de reconnaissance de mots cursifs en ligne à apprentissage simplifié », *CIFED'04 - Colloque International Francophone sur l'Ecrit et le Document*, La Rochelle, France, Juin 2004.

[Prevost, 2003] L. Prevost, C. Michel-Sendis, A. Moises, L. Oudot & M. Milgram, "Combining model-based and discriminant classifiers : application to handwritten character recognition", *International Conference on Document Analysis and Recognition (ICDAR'03)*, Edimbourg, Ecosse, 2003.

[Rabiner, 1989] L. Rabiner, B.H. Juang, "Tutorial on hidden Markov models and selected applications in speech recognition", *Proceedings of the IEEE*, vol. 77, no. 2, pages 257-285, 1989.

[Rabiner, 1993] L. Rabiner, B.H. Juang, "*Fundamentals of Speech Recognition*", édition Prentice Hall PTR, ISBN 0-130-15157-2, 1993.

[Ragot, 2003] Nicolas Ragot, « *MÉLIDIS : Reconnaissance de formes par modélisation mixte intrinsèque/discriminante à base de systèmes d'inférence floue hiérarchisés* », Thèse de l'Université de Rennes 1, Octobre 2003.

[Ragot, 2003b] N. Ragot, E. Anquetil. "A generic hybrid classifier based on hierarchical fuzzy modeling experiments on on-line handwritten character recognition", IEEE Proceedings, *Actes des Seventh International Conference on Document Analysis and Recognition, ICDAR'2003*, Edinburgh, UK, vol. 2, pages 963-967, 2003.

[Ratzlaff, 2003] E. H. Ratzlaff. "Methods, Report and Survey for the Comparison of Diverse Isolated Character Recognition Results on the UNIPEN Database". *7th International Conference on Document Analysis and Recognition (ICDAR 2003)*, Volume 1, pages 623-628 UK. IEEE Computer Society 2003, ISBN 0-7695-1960-1. Edinburgh, Scotland, August 2003

[Ridder, 1996] D. de Ridder . "*Shared weights neural networks in image analysis*", M.Sc. Thesis, Delft, March 1996.

[Rigoll, 1994] G. Rigoll. "Maximum Mutual Information Neural Networks for Hybrid Connectionist-HMM Speech Recognition Systems", *IEEE Trans. on Speech and Audio Processing*, Vol. 2, N° 1, pages 175-184, 1994.

[Rigoll, 1996] G. Rigoll, A. Kosmala, J. Rottland, C. Neukirchen. "A Comparison Between Continuous and Discrete Density Hidden Markov Models for Cursive Handwriting Recognition". *Intern. Conf. On Pattern Recognition (ICPR)*, vol. 2, pages 205-209, Vienna, 1996.

[Rigoll, 1998] G. Rigoll, A. Kosmala, D. Willett. "A new hybrid approach to large vocabulary cursive handwriting recognition". In proceedings of the *Fourteenth International Conference on Pattern Recognition*, Volume 2, pages 1512–1514. Aug 1998.

[Rojas, 1996] R. Rojas. "Neural Networks - A Systematic Introduction". Springer-Verlag, Berlin, New-York, 1996 (502 pages, 350 illustrations). (Chapitre 7 concerné, disponible sur la toile : http://page.mi.fu-berlin.de/~rojas/neural/)

[Rosenblatt, 1962] F. Rosenblatt. "*Principles of Neurodynamics*". Spartan Books, New York. 1962.

[Rossi, 1996] Fabrice Rossi. “Second Differentials in Arbitrary Feed-Forward Neural Networks”. *Technical report, ThomsonCSF /SDC*, October 1995. disponible sur la toile : http://citeseer.ist.psu.edu/article/rossi96second.html

[Rousseau, 2004] L. Rousseau, E. Anquetil, J. Camillerapp. « Reconstitution du parcours du tracé manuscrit hors-ligne de caractères isolés », *actes du 8ème Colloque International Francophone sur l'écrit et le Document, (CIFED'04)*, Pages 123-127, La Rochelle, France, Juin 2004.

[Saon, 1997] G. Saon and A. Belad. “Off-line Handwritten Word Recognition Using A Mixed HMMMRF Approach”. In *Fourth International Conference on Document Analysis and Recognition (ICDAR'97)*, volume 1, pages 118-122, Ulm, Germany, Aug. 1997. (disponible sur la toile http://citeseer.ist.psu.edu/saon97offline.html

[Sarle, 1997] W.S. Sarle, ed. (1997), *Neural Network FAQ*, part 2 : Part 2: “Learning, What are batch, incremental, on-line, off-line, deterministic, stochastic, adaptive, instantaneous, pattern, constructive, and sequential learning?”, periodic posting to the Usenet newsgroup comp.ai.neural-nets, (disponible sur la toile : ftp://ftp.sas.com/pub/neural/FAQ.html)

[Sayre, 1973] K.M. sayre, “*Machine Recognition of Handwrittten Words : A Project Report, Pattern Recognition*” volume 5, pages 213-228, 1973.

[Schamb., 2003] M.P. Schambach, “Model Length Adaptation of an HMM based Cursive Word Recognition System”, *ICDAR 2003*, Edinburgh, pages 109-113, 2003.

[Schenkel, 1995] M. Schenkel, I. Guyon, D. Henderson. “On-line cursive script recognition using Time Delay Neural Networks and Hidden Markov Models”. *Machine Vision and Applications*, special issue on Cursive Script Recognition, volume 8, pages 215-223, 1995.

[Schwenk , 1996] H. Schwenk and M. Milgram. “Constraint tangent distance for online character recognition”. In *Proceedings of International Conference on Pattern Recognition*, pages 520–524, 1996.

[Schwenk, 1997] H. Schwenk and Y. Bengio, “AdaBoosting Neural Networks: Application to on-line Character Recognition”, In *International Conference on Artificial Neural Networks,* pages 967-972. Springer-Verlag, 1997. (disponible sur la toile : http://www.limsi.fr/Individu/schwenk/Papers.A4/icann97_ftp.ps.gz)

[Schwenk, 1998] H. Schwenk. “*The diabolo classifier, Neural Computation*”, volume 10, issue 8, pages 2175-2200, 1998.

[Seni, 1996] G. Seni, R.K. Srihari , N. Nasrabadi. “Large Vocabulary Recognition of On-Line Handwritten Cursive Words”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 18 n.7, pages 757-762, July 1996

[Senior, 1998] A.W.Senior and A.J.Robinson, “An Off-line Cursive Handwriting Recognition System”. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Volume 20 No. 3, pages 309-321, March 1998.

[Shawe-Taylor, 1994] J. Shawe-Taylor. “Introducing invariance: a principled approach to weight sharing”, in *Proceedings of the IEEE International Conference on Neural Networks, Volume I, IEEE World Congress on Computational Intelligence,* Orlando, pages 345-349, 1994.

[Singer, 1999] Y. Singer and R. Schapire. "Improved boosting algorithms using confidence-rated predictions". *Machine Learning*, volume 37, n°3, pages 297-337, 1999. (http://citeseer.ist.psu.edu/schapire99improved.html).

[Sutton, 1998] Richard S. Sutton and Andrew G. Barto, "Reinforcement Learning: An Introduction", *MIT Press*, Cambridge, MA, A Bradford Book, 1998. (disponible sur la toile : http://www-anw.cs.umass.edu/~rich/book/the-book.html)

[Swett, 2005] B. Swett, A. Braun, J.L. Contreras-Vidal. "Examination of sequence learning using a novel, graphomotor task in Experts, Controls, and Parkinson's disease patients : a behavioral & fMRI study". Proceedings of the *12th Biennal Conference of the International Graphonomics Society (IGS 2005)*. ISBN 88 89702 13 3. pages 227-231. Edited by A. Marcelli and C. De Stefano. University of Salerno, Italy, June 26-29, 2005.

[Tappert, 1990] C.C. Tappert, C.Y. Suen and T. Wakahara. "The State of the Art in Online Handwriting Recognition". *IEEE Transactions on Pattern Analysis and Machine Intelligence*. ISSN:0162-8828. Volume 12, Issue 8, pages 787 - 808. 1990.

[Tax, 2000] D.M.J. Tax, M. Van Breukelen, R.P.W. Duin, J. Kittler, "Combining multiple classifiers by averaging or multiplying?", *Pattern Recognition*, 33, pages 1475-1485, 2000.

[Tay, 2001] Y.H. Tay, P.M. Lallican, et al., "An Analytical Handwritten Word Recognition System with Word-Level Discriminative Training", Proceedings of *Sixth International Conference on Document Analysis and Recognition (ICDAR'01)*, pages 726-730, Seattle, Sept. 2001.

[Tay, 2002] Y.H. Tay, "*Off-line Handwriting Recognition using artificial Neural Network and Hidden Markov Model*"- PhD University of Nantes and University Technologi Malaisia, 2002.

[Teulings, 2005] H.L. Teulings, H.T. Czerepacha, D.H. Romero, R. Subramanian. "Handwriting Teaching Tool for Children". Proceedings of the *12th Biennal Conference of the International Graphonomics Society (IGS 2005)*. ISBN 88 89702 13 3. pages 95-99. Edited by A. Marcelli and C. De Stefano. University of Salerno, Italy, June 26-29, 2005.

[Torres-Moreno, 1997] J.M. Torres-Moreno, Juan Manuel, « *Apprentissage et généralisation par des réseaux de neurones: étude des nouveaux algorithmes constructifs* », Thèse Institut National Polytechnique de Grenoble, France, 1997. disponible sur la toile : http://www.lia.univ-avignon.fr/fich_art/459-t224_1.pdf

[Vapnik, 1995] V. Vapnik. "*The Nature of Statistical Learning Theory*", Springer-Verlag, 1995.

[Velek, 2003] O. Velek, S. Jaeger and M. Nakagawa: "Accumulated-Recognition-Rate Normalization for Combining Multiple On/Off-Line Japanese Character Classifiers Tested on a Large Database," Proc. 4th International Workshop on Multiple Classifier Systems (MCS), Gildford, UK, pages196-205, Juin 2003.

[Viard-Gaudin, 1999] C. Viard-Gaudin, P.M. Lallican, S. Knerr, P. Binter, "The IRESTE ON-OFF (IRONOFF) Handwritten Image Database", ICDAR'99, *International Conference on Document Analysis and Recognition*, pages 455-458, Bangalore, 1999.

[Viard-Gau., 2005] C. Viard-Gaudin, P.-M Lallican, S. Knerr. "Recognition-directed recovering of temporal information from handwriting images". *Pattern Recognition Letters*, Article in press. 2005.

[Vincieralli, 2000] A. Vincieralli,. J. Luettin. "Off-line cursive script recognition based on continuous density hmm". In Proceedings of the *seventh international workshop on frontiers in handwriting recognition (IWFHR)*, Amsterdam, pages 493-498, 2000.

[Vinciarelli, 2002] A. Vinciarelli. "*A survey on Off-Line Cursive Script Recognition Pattern Recognition*", Volume 35, no. 7, pages 1433-1446, June 2002.

[Vuurpijl, 2000] L. Vuurpijl, L. Schomaker, "Two-Stage Character Classification : A Combined Approach of Cluste ring and support Vector Classifiers", Proceedings of *International Workshop on Frontiers in Handwriting Recognition (IWFHR)*, Amsterdam, 2000, pages 423-432.

[Waibel, 1988] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, K. Lang. "*Phoneme Recognition Using Time-Delay Neural Networks*". ATR Interpreting Telephony Research Laboratories. 30 Octobre 1987.

[Wimmer, 1999] Z. Wimmer, S. Garcia-Salicetti, A. Lifchitz, B. Dorizzi, P. Gallinari, T. Artières, « REMUS (Reconnaissance de l'Ecriture ManUScrite) ». Présentation du système sur la toile : http://www-connex.lip6.fr/~lifchitz/Remus/, 1999.

PUBLICATIONS DE L'AUTEUR

Revues

E. Poisson et C. Viard-Gaudin. « Réseaux de neurones à convolution et reconnaissance de l'écriture manuscrite non contrainte », *Revue VALGO*, In Valgo numéro 01-02, La revue en ligne de l'Association des Connexionnistes en THèse, Association des Connexionnistes en Thèse (ACTH), ISSN 1625-9661, pages 726-730, Montélimar, France, Octobre2001. (Prix AFIA pour le meilleur article soumis.)

P. LeCallet, C. Viard-Gaudin, S. Pechard and **E. Caillault**. "No reference and reduced Reference video quality metrics for end to end QoS monitoring". in *IEICE Transactions on Communications*, special issue on Multimedia QoS Evaluation and Management Technologies. Volume E85. Article in Press, February 2006.

Conférences internationales avec comité de lectures et actes

E. Poisson, C. Viard-Gaudin et P.M. Lallican. « Combinaison et analyse de réseaux de neurones à convolution pour la reconnaissance de caractères manuscrits en-ligne », *CIFED'02 - Colloque International Francophone sur l'Ecrit et le Document*, Hammamet, Tunisie, Octobre 2002.

E. Poisson, C. Viard-Gaudin and P.M. Lallican. "Multi-modular architecture based on convolutional neural networks for online handwritten character recognition", in *Proc. of ICONIP'02 - 9th International Conference on Neural Information Processing*, IEEE Neural Network Society, ISBN 981-04-7536-5, Singapour, Novembre 2002.

F. Alleau, **E. Poisson**, C. Viard-Gaudin and P. Le Callet. "TDNN with Masked Inputs", ICICS 2003, in *Proc. of the* 4th *International Conference on Information, Communications & Signal Processing and IEEE Pacific-Rim Conference on Multimedia (ICICS-PCM 2003),* Nanyang Technological University, Singapour, Decembre 2003.

E. Poisson, C. Viard-Gaudin et P.M. Lallican. « Système TDNN/HMM de reconnaissance de mots cursifs en ligne à apprentissage simplifié », *CIFED'04 - Colloque International Francophone sur l'Ecrit et le Document*, La Rochelle, France, Juin 2004.

A.R. Ahmad, M. Khalid**,** C. Viard-Gaudin and **E. Poisson**. "Online Handwriting Recognition using Support Vector Machine". *IEEE TENCON Analog and Digital Techniques for Electrical Engineering*. Thailand, November 2004.

E. Caillault, C. Viard-Gaudin and P.M. Lallican. "Training of hybrid ANN/HMM systems for on-line handwriting word recognition", *Proceedings of 12th Conference of International Graphonomics Society (IGS 2005)*, Salerno, Italy, june 2005.

E. Caillault, C. Viard-Gaudin and A.R. Ahmad. "MS-TDNN with Global Discriminant Trainings", *in Proceedings of 8th International Conference on Document Analysis and Recognition (ICDAR 2005)*, volume II, pages 856-860, Seoul, Corea, august 2005.

E. Caillault, C. Viard-Gaudin. « Apprentissages discriminants en reconnaissance de mots cursifs en-ligne ». *GRETSI 2005*. Belgique, Septembre 2005.

Résumé

Les travaux présentés dans le cadre de cette thèse portent sur l'étude, la conception, le développement et le test d'un système de reconnaissance de mots manuscrits non contraints en-ligne pour une application omni-scripteurs. Le système proposé repose sur une architecture hybride neuro-markovienne comportant d'une part, un réseau de neurones à convolution (TDNN et/ou SDNN), et d'autre part des modèles de Markov à états cachés (MMC). Le réseau de neurones a une vision globale et travaille au niveau caractère, tandis que le MMC s'appuie sur une description plus locale et permet le passage du caractère au niveau mot. Nous avons d'abord étudié le système de reconnaissance au niveau caractère isolé (digits, majuscules, minuscules) et optimisé les architectures des réseaux en termes de performances et de taille. La seconde partie du travail a porté sur le passage au niveau mot. Ici, l'effort a consisté avant tout à la définition d'un schéma d'apprentissage global au niveau mot qui permet d'assurer la convergence globale du système, en définissant une fonction d'objectif qui mixe des critères basés modèle générateur (typiquement par maximum de vraisemblance) et des critères discriminants (de type maximum d'information mutuelle). Les différentes résultats présentés (sur les bases MNIST, IRONOFF, UNIPEN) montrent l'influence des principaux paramètres du système, soit en termes de topologie, de sources d'information, de modèles d'apprentissage (nombre d'états, pondération des critères, durée).

Mots-clés : Reconnaissance de l'écriture, réseaux de neurones à convolution, modèles de Markov, programmation dynamique, système hybride, maximum de vraisemblance, critère d'information mutuelle.

Abstract

Architecture and Training of a Hybrid Neuro-Markovian System
for On-Line Handwriting Recognition

This thesis deals with the study, the conception, the development and the test of an online unconstrained handwriting word recognition system for an omni-writer application. The proposed system is based on a hybrid architecture including on the one hand, a neural convolutional network (TDNN and/or SDNN), and on the other hand Hidden Markov Models (HMM). The neural network has a global vision and works at the character level, while the HMM works on a more local description and allows the extension from the character level to the word level. The system was first dedicated for processing isolated characters (digits, lowercase letters, uppercase letters). This architecture has been optimized in terms of performances and size. The second part of this work concerns the extension to the word level. In this case, we have defined a global training scheme directly at the word level. It allows to insure the global convergence of the system. It relies on an objective function that combines two main criteria: one based on generative models (typically by maximum likelihood estimation) and the second one based on discriminant criteria (maximum mutual information). Several results are presented on MNIST, IRONOFF and UNIPEN databases. They show the influence of the main parameters of the system, either in terms of topologies, information sources, and training models (number of states, criteria weighting, duration).

Keywords : handwriting recognition, convolutional neural networks, Markov models, dynamic programming, hybrid system, maximum of likelihood, maximum mutual information critria.