5. Classification phonétique

Le chapitre 5 concerne les expériences menées au début de cette thèse pour étudier les capacités de différents modèles connexionnistes (LVQ2, Features Map, MLP et TDNN) pour la classification phonétique. Ces travaux ont fait l'objet d'une présentation au workshop "*Neurospeech*" en 1989 et d'un article au JEP90 [Devillers 90]. Ces expériences nous ont permis de selectionner le modèle le plus adapté (en terme de temps d'apprentissage, de complexité algorithmique et de performance) à notre tâche de classification phonétique. Elles ont été menées sur une base de données de taille réduite en français, décrite dans l'annexe 1. Ces données nous ont servi de base de développement avant de tester une base de donnée de taille plus importante DARPA RM1 dépendante du locuteur (chapitre 6).

La première partie de ce chapitre relate nos premières expériences de classification sur des formes statiques.

Deux des algorithmes les plus performants à l'heure actuelle: l'algorithme de LVQ2 et l'algorithme de rétro-propagation ont été testés. Nous présentons tout d'abord des résultats de classification sans prendre en compte l'aspect temporel des formes avec des réseaux LVQ2 et MLP qui permettent de comparer les qualités de ces classifieurs à un modèle classique de kNN (avec k=1) utilisant une quantification vectorielle.

La deuxième partie porte sur la gestion temporelle des réseaux.

L'aspect séquentiel de la parole est fondamental, la prise en compte du contexte permet d'obtenir de meilleures performances de classification. Malgré la gestion d'une certaine dynamique temporelle, les réseaux testés ne sont pas

capables de gérer des distorsions temporelles non apprises et n'ont pas de mécanisme permettant de traiter directement la parole continue.

5.1 Classification de formes statiques

Nous abordons un problème de classification de 31 phonèmes provenant d'une base de parole continue en français pour un locuteur, cf annexe 1. Les données sont des vecteurs de 17 coefficients FFT correspondant à 12.8 ms de parole. Chaque segment phonétique comporte un nombre variable de vecteurs. Les données ont été segmentées manuellement sur les instants stables, c'est-à-dire, par exemple pour les plosives, sur le silence précédent l'explosion. Les premiers tests ont été menés sans prendre en compte le contexte des trames. Pour les phonèmes les plus instables, des macro-classes sont utilisées regroupant sous la même étiquette P (/P/, /T/, /K/, /silence/) (notations phonétiques du LIMSI répertoriées dans l'annexe 1), B pour (/B/, /D/, /G/) et L pour (/L/, /R/). Le nombre de classes est finalement de 25.

5.1.1 Cartes topologiques et LVQ2

Les paramètres déterminés a priori pour la carte topologique intègrent notre connaissance du problème à traiter. Ces choix, cf §3.4 sont les suivants :

- la carte topologique permettant de coder les vecteurs a une couche de sortie plane,

- la carte a pour dimension 15 vecteurs sur 15 vecteurs, chaque vecteur de cette carte a lui-même une dimension de 17,

- la forme du voisinage des vecteurs est carrée,

- la taille du voisinage est au maximum de 6 degrés de profondeur (elle décroît lors de l'apprentissage),

- la métrique utilisée est la distance euclidienne,

- enfin, la fonction de modification des coefficients est une fonction linéaire décroissante.

Les formes à classifier sont présentées de façon alternée. L'apprentissage nécessite la présentation d'un grand nombre de données, il faut environ 2000 cycles d'apprentissage dans cette expérience pour que la carte converge vers un état stable,

1 cycle correspondant à la présentation d'une occurrence des 25 phonèmes à classifier.

Lorsque la carte a atteint un état stable, elle est étiquetée. Un vecteur reçoit l'étiquette majoritairement reconnue par tous les exemples de la base de données qui sont les plus proches de ce vecteur.

La carte topologique contient 15x15 références v. Soit X l'ensemble des données x, soit d la distance euclidienne, pour tous les échantillons x de classe C_x, on cherche la référence de la carte R(x) la plus proche:

$$R(x) = v_i \text{ tel que } v_i = \min_{c=1,..,V} d(x,v_c)$$

On étiquette un vecteur de la carte avec la classe majoritairement reconnue par tous les exemples (figure 5.1). Après étiquetage de la carte topologique, des partitions de références émergent. L'algorithme LVQ2 est utilisé pour affiner les décisions aux frontières entre les classes. La carte obtenue ici est le résultat d'un apprentissage non supervisé FM de 2000 cycles et d'un apprentissage supervisé LVQ2 de 700 cycles.

*	*	*	A	A	A	+	+	+	/	/	/	L	/	O
*	*	*	A	A	A	E	+	+	/	/	/	/	O	O
L	*	<	<	A	A	+	+	E	/	/	/	O	W	W
L	V	<	<	<	A	A	E	E	+	L	O	W	W	W
V	V	L	<	<	(	(	E	E	E	O	L	W	W	W
B	B	W	L	<	(	(	(	E	=	=	L		%	%
B	B	B	V	L	(	(	(	E	=	=	=	L	%	%
B	B	B	V	V	(	(	)	)	=	=	=	U	%	M
B	B	V	Z	Z	)	)	)	)	L	=	U	L	M	M
V	Z	V	Z	I	I	)	)	)	U	U	N	N	N	M
S	S	Z	Z	I	I	I	)	)	U	U	%	%	N	N
S	S	S	Z	I	I	I	U	U	U	U	M	N	N	M
S	S	S	J	I	I	I	U	U	U	V	V	Z	V	L
X	X	X	J	J	J	J	Z	Z	V	F	F	F	P	P
X	X	X	J	J	J	J	J	F	F	F	F	F	P	P

*Fig. 5.1 : **Carte topologique FM-LVQ2** (15x15) de 25 classes (les vecteurs pouvant être rejetés sont indiqués en relief).*

La surface occupée par les vecteurs d'une même étiquette est une **approximation de la fonction de densité d'un phonème.** La topologie des phonèmes est respectée (figure 5.1), il existe cependant des points de rupture illustrés par exemple sur la figure par la séparation entre les nuages de voyelles et de consonnes. Il est impossible d'éviter ce phénomène en 2 dimensions.

Environ 10% des vecteurs peuvent être éliminés afin d'améliorer le pouvoir de généralisation de la carte. Ce problème est dû au nombre fini de données d'apprentissage. Il s'agit de vecteurs atypiques ou ambigus. Un vecteur atypique est un vecteur représentant peu de données, on ne peut alors avoir confiance en sa représentation. Un vecteur ambigü se situe en bordure de plusieurs partitions de classes phonétiques et ne représente pas de façon décisive l'une ou l'autre des formes. Nous indiquons quelques cas atypiques sur la carte de la figure 5.1 avec une fonte en relief.

5.1.2 Comparaison entre les modèles connexionnistes FM, FM-LVQ2 et une quantification vectorielle

L'algorithme de quantification des boules optimisées, variante de l'algorithme des k-moyennes décrit au §2.4.3, est utilisé avec en moyenne 7 représentants par phonème. Un classifieur du type plus proche voisin appliqué aux vecteurs de référence issus de l'algorithme de boules optimisées et de la carte topologique permet d'obtenir des scores de classification. Les phonèmes appris sont les 25 phonèmes correspondant aux phonèmes les plus stables. Les modèles connexionnistes FM et FM-LVQ2 sont décrits au §5.1.1. Les performances des différents classifieurs sont décrites dans le tableau 5.1.

%Reconnaissance	*Apprentissage*		*Test*	
	Trames	*Segments*	*Trames*	*Segments*
FM	92.5	97.4	64.8 ± 1.4	70.6 ± 2.0
FM + LVQ2	94.0	97.6	68 ± 1.4	75.0 ± 1.9
QV	93.0	97.5	63.9 ± 1.4	74.9 ± 1.9

Tab. 5.1 : ***Classication trame à trame avec des méthodes connexionnistes non supervisées et supervisées (FM, FM+LVQ2) ainsi que classique (QV).***

Les pourcentages de reconnaissance sur les trames ont été obtenus en comparant les vecteurs de la base de données avec les vecteurs de la carte. Un segment est bien

reconnu lorsque la moitié au moins de ses trames ont été correctement classifiées. Les résultats sont donnés pour le corpus d'apprentissage et de test.

Le nombre de vecteurs de référence est important pour juger des performances obtenues. Le nombre de vecteurs est à peu près le même pour les deux approches. Les boules optimisées utilisent environ 7 x 25 = 175 vecteurs de référence tandis que la carte contient 225 vecteurs dont 10% sont éliminés, soit environ 200 vecteurs utilisés.

Les capacités des cartes topologiques se sont révélées être celles d'une quantification vectorielle non supervisée. On obtient une bonne visualisation, ici en deux dimensions, de l'espace des données compressées. Comme pour une quantification classique, ce système n'est pas limité à un nombre réduit de formes à classifier. Les cartes topologiques éliminent le problème de l'initialisation qui était un désavantage de l'algorithme des k-moyennes (cf §2.4.3) mais le nombre de vecteurs reste choisi a priori. Une bonne connaissance du problème à résoudre est nécessaire pour obtenir de bons résultats.

Les scores du modèle FM+LVQ2 montrent un meilleur pouvoir de généralisation que la QV puisque le pourcentage de trames reconnues en test augmente de 6.5%, cette amélioration est due à l'algorithme de LVQ2. LVQ2 détermine de façon indirecte des frontières linéaires autour des classes. Elle calcule le risque moyen de classification en modifiant localement les vecteurs proches des frontières.

5.1.3 Comparaison entre deux méthodes connexionnistes MLP et FM-LVQ2

Ces tests ont été ménés sur les mêmes données. Les performances des réseaux MLP et FM-LVQ2 sont décrites dans le tableau 5.2.

%Reconnaissance	*Apprentissage*		*Test*	
	Trames	*Segments*	*Trames*	*Segments*
FM + LVQ2	94.0	97.6	68.0 ± 1.4	75.0 ± 1.9
MLP	94.2	97.4	66.8 ± 1.4	75.3 ± 1.9

Tab. 5.2 : ***Classication trame à trame avec deux méthodes connexionnistes à apprentissage supervisé (FM+LVQ2, MLP).***

Les pourcentages de reconnaissance sur les trames ont été obtenus en comparant les vecteurs de la base de données avec les vecteurs de la carte. Un segment est bien

reconnu lorsque la moitié au moins de ses trames ont été correctement classifiées . Les résultats sont donnés pour le corpus d'apprentissage et de test.

Les méthodes de classification de type perceptron multi-couches ou LVQ2 améliorent très peu les performances de la quantification vectorielle. On observe cependant un meilleur taux de généralisation des modèles neuronaux sur les trames. Tous ces modèles ont été utilisés sans contexte.

Afin d'obtenir une classification sur les phonèmes non stables comme les plosives et de faciliter la classification de manière générale, nous avons étudié la gestion du contexte avec des cartes topologiques spécialisées et avec des réseaux mutlicouches, MLPs et TDNNs. Les tests sont décrits dans le §5.2.

5.1.4 Comparaison LVQ2 / MLP

Les méthodes à apprentissage supervisé, FM-LVQ2 et MLP avec la rétropropagation du gradient, sont des méthodes très différentes. Les performances obtenues par chacune d'elle sont pourtant très proches sur cette tâche de classification.

Les différences entre ces deux méthodes sont de plusieurs types, nous en faisons un résumé.

- **apprentissage global/local**

LVQ2 fait une minimisation du critère d'erreur de classification en modifiant **localement** les vecteurs références w_i. On modifie les vecteurs proches des frontières de classification. Elle fournit une décision directement sur la classe optimale, c'est à dire celle qui présente le moins de risque en moyenne. En effet, lorsque une entrée est mal classée, on approche le risque moyen au voisinage de la frontière de classification proportionnellement à l'écart relatif entre les carrés des distances de l'entrée et d'une part le vecteur de référence le plus proche de bonne classe, d'autre part avec le vecteur de référence le plus proche de mauvaise classe. Les autres formes ne sont pas prises en compte.

La **rétro-propagation du gradient d'erreur** dans un MLP utilise le critère de coût des moindres carrés (MSE). Le MLP fait une adaptation des poids qui correspond à un apprentissage **global** de l'ensemble des paramètres afin que toutes les entrées soient bien classées.

- **notion de distance**

Les performances de **LVQ2** sont très **dépendantes du choix de la distance** et sont mauvaises sur des données bruitées.

La notion de distance existe dans le **MLP** au niveau des sorties du réseau, elle **intervient indirectement** sur la classification des entrées. Celles-ci sont projetées sur des espaces intermédiaires ayant d'autres dimensions afin de les rendre plus facilement séparables au niveau des sorties.

- initialisation

LVQ2 est une méthode efficace si elle intervient après une bonne initialisation obtenue avec un algorithme de k-moyennes ou de cartes topologiques.

Le **MLP** s'affranchit facilement de cette dépendance, cf §6.4.

- nombre de références/phonème

LVQ2 modifie les vecteurs de références pour chaque classe directement dans l'espace de données. Le nombre de références est fixé par la méthode qui lui sert de modèle initial. Elle s'affranchit d'une distribution multimodale en ayant **plusieurs références par forme à classifier**. Le nombre de vecteurs choisis est donc très important.

Le **MLP** utilise des projections des données dans des espaces de dimensions différentes pour essayer de regrouper chaque classe dans un nuage de données compact bien séparé des autres classes. La classification est faite sur la dernière couche. Chaque cellule de sortie du MLP peut être comparée à une référence de LVQ2. Le réseau n'a qu'**une référence par classe**.

- temps d'apprentissage

LVQ2 est plus simple à mettre en oeuvre et donc plus rapide.

Le **MLP** est plus coûteux en temps d'apprentissage

En conclusion, la LVQ2 donne des résultats meilleurs [Mc Dermott 92] pour des tâches où une distance bien choisie suffit pour classifier les formes. LVQ2 est plus rapide à mettre en oeuvre que le MLP mais est très dépendante d'une bonne initialisation. Le MLP est une méthode plus élaborée et plus robuste que LVQ2 [De Bollivier,Gallinari 90], notamment pour classifier des formes bruitées ou mal segmentées et pour gérer des vecteurs d'entrée de grandes dimensions.

5.2 Gestion temporelle

La gestion temporelle intervient à deux niveaux :

- elle implique la prise en compte de l'aspect séquentiel des formes, c'est-à-dire simplement la gestion d'un contexte temporel,

- elle doit permettre le traitement de parole continue, c'est à dire l'intégration de séquences de parole en mots puis en phrase.

Pour la prise en compte de l'aspect temporel aux deux niveaux, il est intéressant d'avoir un mécanisme de gestion des distorsions temporelles.

5.2.1 Codage d'un contexte temporel

Chaque phonème possède ses caractéristiques propres mais aussi les effets de coarticulation de son contexte qu'il est important de prendre en compte pour les classifier. Le codage d'un contexte temporel est particulièrement important pour les phonèmes très peu stables.

5.2.1.1 Sous-cartes topologiques-LVQ2

Dans les expériences précédentes, nous avions cherché à classifier les parties "stables" du signal de parole. La dynamique du signal, l'aspect séquentiel des trames d'entrée n'était pas pris en compte. Quelques tests ont été menés sur le codage des phonèmes très peu "stables" comme les plosives /P/, /T/, /K/ et /B/, /D/, /G/. Les plosives non voisées sont composées en général d'une partie "stable" appelée silence et d'une partie "instable" qui correspond à la barre d'explosion. Pour les plosives non voisées, la barre d'explosion est souvent inexistante. Nous avons décidé de prendre a priori comme partie "instable" pour toutes les plosives, les trois trames suivant la partie stable. Ces trames correspondent à la barre d'explosion de la plosive ou au début du phonème suivant, ici une voyelle (le corpus en français ne contient en fait que des contextes consonnes/voyelles). Nous avons construit des cartes topologiques caractérisant ces parties "non stables", l'entrée est alors un "super-vecteur" composé de trois trames. Les dimensions des sous-cartes, fig. 5.2 et fig 5.3, ont été ajustées après expérience. Les dimensions des cartes sont d'environ 40 formes pour coder tous les contextes de transition de plosive à voyelle.

Nous avons utilisé ces sous-cartes avec la carte apprise précédemment: durant la phase de reconnaissance, une plosive est détectée par la carte topologique grâce à sa

partie stable qui correspond à une macro-classe, silence ou silence voisé. Les trames de la partie non stable sont ensuite comparées à la sous-carte correspondante, plosive voisée ou non voisée, qui permet de discriminer entre les phonèmes /P/, /T/, /K/ ou /B/, /D/ et /G/. Les scores issus des sous-cartes sont intégrés aux scores des macro-classes par une simple multiplication.

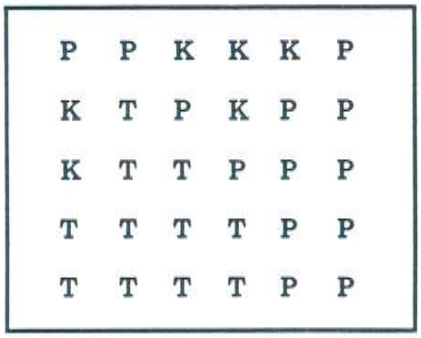

*Fig. 5.2 : **Carte représentant les phonèmes /P/, /T/, /K/** (454 trames / 100 segments) de dimension 6 * 5 = 30 références.*

Les résultats de classification sur la macro-classe P de la carte globale tenant compte de la classification issue de la sous-carte représentant les phonèmes /P/, /T/, /K/ sont donnés par le tableau 5.3.

Test	P		T		K	
	Tr.	Seg.	Tr.	Seg.	Tr.	Seg.
Macro P	88.3	90.6	88.3	90.6	88.3	90.6
Macro P + PTK	66.6	75.5	64.5	75.4	65.9	66.8

*Tab. 5.3 : **Reconnaissance des phones /P/, /T/, /K/** (454 trames,100 segments). Les résultats donnés sont des pourcentages de reconnaissance, Les scores de la macro-classe P sont issus de la carte topologique Fig. 5.1. Les scores de Macro P + PTK correspondent à la multiplication des scores de la macroclasse avec le score issu du sous-réseau. La précision des performances sur les trames est de* ±0.6 et sur les segments de ±1.2. *L'abréviation Tr. indique une évaluation sur les trames, l'abréviation Seg. indique une évaluation sur les segments phonétiques. Un segment est considéré comme reconnu si la moitié au moins des trames qui le composent ont été reconnues.*

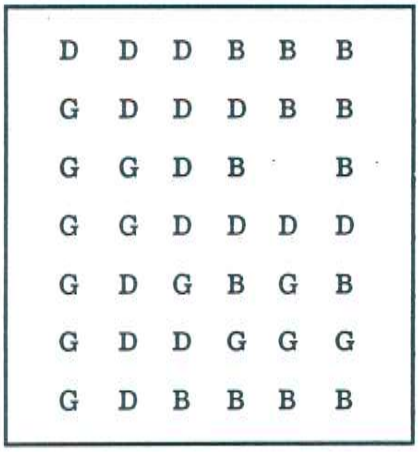

*Fig. 5.3 : **Carte de BDG** (456 trames / 99 segments) de dimension 7 * 6 = 42 références.*

Les résultats de classification sur la macro-classe P de la carte globale tenant compte de la classification issue de la sous-carte représentant les phones /P/, /T/, /K/ sont donnés par le tableau 5.4.

Test	B		D		G	
	Tr.	Seg.	Tr.	Seg.	Tr.	Seg.
Macro B	86.6	85.1	86.6	85.1	86.6	85.1
Macro B + BDG	65.2	68.3	66.2	73.3	65.8	75.5

*Tab. 5.4 : **Reconnaissance des phones /B/, /D/, /G/** (456 trames, 99 segments). Les résultats donnés sont des pourcentages de reconnaissance, Les scores de la macro-classe B sont issus de la carte topologique Fig. 5.1. Les scores de Macro B + BDG correpondent à la multiplication des scores de la macroclasse avec le score issu du sous-réseau. La précision des performances sur les trames est de* ±0.7 et sur les segments de ±1.5. *L'abréviation Tr. indique une évaluation sur les trames, l'abréviation Seg. indique une évaluation sur les segments phonétiques. Un segment est considéré comme reconnu si la moitié au moins des trames qui le composent ont été reconnues.*

Les plosives sont assez mal reconnues, les transitions sont très instables et ne permettent pas de généraliser facilement. Le contexte de trois trames est insuffisant. Le codage d'un contexte plus large semble indispensable.

Le codage de "super-vecteurs" correspondant à la concaténation de plusieurs trames de contexte dans une carte topologique devient rapidement lourd et les calculs de

distance avec toutes les références fastidieux. Cependant, des tests ont été menés avec la S-LVQ2 utilisant 7 trames de contexte par [McDermott 89] avec des performances analogues aux TDNNs codant le même contexte.

Un MLP à contexte ou un TDNN semblent malgré tout plus appropriés pour le traitement d'un contexte temporel. Ces réseaux utilisent un apprentissage global, on calcule une matrice de connexions entre chaque couche qui sert à classifier toutes les formes.

5.2.1.2 TDNNs et MLPs à contexte

La différence entre un MLP codant un contexte et un TDNN réside dans la gestion des matrices de connexions. Dans le cas du TDNN, les poids sont partagés entre plusieurs cellules, le codage est alors plus compact. Nous comparons les résultats de classification des 31 unités phonétiques sur la base en français CVCV avec des réseaux MLPs à contexte et des TDNNs.

Pour le MLP, la taille du contexte est codée directement dans la première couche.

Pour le TDNN, la taille du contexte de la couche d'entrée permet de discriminer les transitions rapides, les variations plus lentes étant prises en compte par le contexte de la couche cachée. L'intégration totale sur la couche de sortie doit être suffisamment grande pour discriminer un phonème. La taille moyenne des unités phonétiques a été utilisé pour estimer la taille des contextes. Plus la taille de la fenêtre d'analyse est grande, plus on apprend des séquences différentes pour une même étiquette, le réseau est donc plus robuste vis à vis des mauvais alignements des segments. La taille moyenne des segments du corpus étant de 6 trames, l'entrée utilisée est une séquence de taille fixe de 10 trames centrée sur la partie stable du segment phonétique, chaque trame correspondant à 12.8ms de parole. Une fenêtre de taille fixe permet de coder directement le même nombre de positions de la fenêtre pour chaque phonème.

Les variations des fenêtres contextuelles testées sont schématisées dans la figure 5.4 . Le tableau 5.5 récapitule les performances des différents modèles.

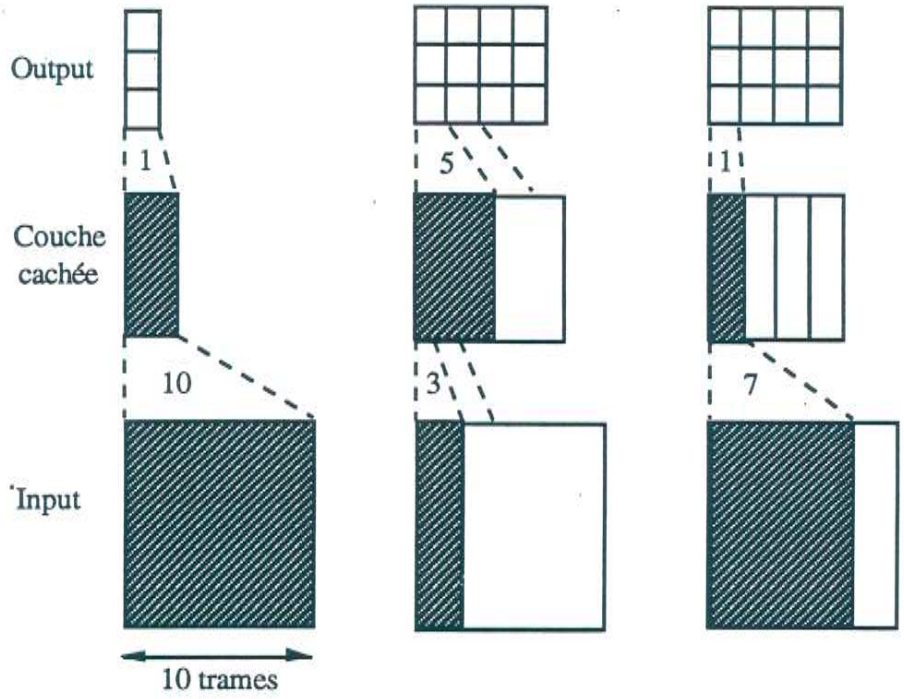

Fig 5.4 : ***Classification de segments de taille fixe (10 trames) avec un MLP complètement connecté, un TDNN (3-5) et un TDNN (7-1).***

Modèles et nb classes	*trames de contexte*	*fenêtre d'analyse nb trames*	*nb de connexions*	*Apprentissage Tr / Seg*		*Test Tr / Seg*	
MLP 25	1	1	1070	94.2	97.4	66.8± 1.4	75.3±1.9
MLP 31	3	3	2521	82.8	87.3	75.8± 1.2	79.6± 1.7
TDNN 31	10	7 (7-1)*	4561	92.7	95	85.1± 1	89± 1.3
TDNN 31	10	7 (3-5)*	6241	91.5	94.8	83.2± 1.1	88.3± 1.3
MLP 31	10	10	10291	94.1	96.2	80.8±01.1	86.2± 1.5

Tab. 5.5 : ***Variation des fenêtres contextuelles pour les réseaux MLP et TDNN.*** *Nous précisons le taux de reconnaissance sur les segments (Seg.) et sur les trames de sorties (Tr.) pour le corpus d'apprentissage et de test.*

**Les chiffres entre parenthèses dans la colonne "fenêtre d'analyse" correspondent aux fenêtres d'intégration temporelle sur les couches des structures TDNN, cf fig 5.4. Le nombre de classes est indiqué après le nom du modèle, par exemple MLP 25.*

Par exemple pour le TDNN, si la fenêtre d'intégration de la première couche est de 3 trames, de 5 trames pour la couche cachée, on obtient une décision phonétique, c'est-à-dire une étiquette phonétique pour 7 trames d'entrée du signal de parole soit 7

*12.8ms = 90ms de parole. Pour un segment de taille fixe de 10 trames en entrée, le pas de translation étant de 1 sur chacune des couches, on obtient en sortie du TDNN, 4 vecteurs auxquels on associe 4 étiquettes phonétiques ou "quasi-phonèmes". Un segment phonétique est considéré comme bien reconnu si au moins la moitié des vecteurs de sortie associés à ce segment ont la bonne étiquette phonétique. Un conflit entre deux étiquettes, chacune ayant été reconnue deux fois par les vecteurs de sortie, est résolu en privilégiant les meilleurs scores obtenus pour décider de l'étiquette "gagnante". Pour le MLP avec un contexte de 10 trames, on prend une seule décision phonétique.

On obtient de bien meilleurs résultats en codant un contexte temporel, le MLP sans contexte reconnait seulement les unités phonétiques les plus stables avec un score de 15% inférieur qu'un MLP codant un contexte de 10 trames. La fenêtre d'intégration de 10 trames du MLP est visiblement trop grande pour bien généraliser ces données, c'est le phénomène de sur-apprentissage: les performances d'apprentissage ont été améliorées tandis que le pouvoir de généralisation a diminué.

L'architecture TDNN est intéressante pour son codage compact et a un plus grand potentiel de généralisation. Elle offre un niveau d'abstraction supérieur au MLP grâce au codage du contexte sur plusieurs couches. Cette structure est aussi souvent moins onéreuse en nombre de paramètres qu'un MLP codant tout le contexte sur la première couche.

5.2.2 TDNN : Invariance en translation et distorsion temporelle

L'algorithme de Viterbi utilisé avec les modèles de Markov cachés (HMMs) ou la programmation dynamique (DTW) compense efficacement les distorsions temporelles, nous avons étudié ce phénomène de compensation dans les réseaux TDNNs.

Le TDNN intègre les fenêtres temporelles de la même couche en utilisant les mêmes poids de connexions ce qui permet de reconnaître les événements dans le temps. Le pas de translation des fenêtres des deux couches est de 1. Chaque trame apparaît dans toutes les positions possibles dans la fenêtre d'analyse du signal. On parle alors d'invariance en translation.

Le test suivant porte sur des données particulières pour tester la robustesse du TDNN aux distorsions temporelles. Les données d'apprentissage sont des segments

de plosives segmentés manuellement. On essaye dans cette expérience de reconnaître des plosives prononcées à un débit rapide alors que l'apprentissage a été mené avec un débit lent dans le même contexte de voyelle/consonne/voyelle. On effectue un apprentissage avec un TDNN intégrant 3 trames sur la première couche et 5 sur la deuxième, on ne code donc qu'une position du phonème correspondant à un contexte de 7 trames. Le tableau 5.6 compare les deux tests à débit lent et rapide avec un apprentissage à débit lent.

% des trames reconnues	*Apprentissage* *Débit lent*	*Test 1* *Débit lent*	*Test 2* *Débit rapide*
Reseau B D G	96.2	89.2± 3.1	72.5 ± 4.7

Tab. 5.6 : ***Robustesse du réseau TDNN aux distorsions temporelles.***

Les résultats montrent pour ce test que le réseau est incapable de gérer cette distorsion temporelle non présente dans les données d'apprentissage. Le réseau ne connaît que les séquences de trames qu'il a apprises, la robustesse à la distorsion temporelle est due à l'apprentissage de plusieurs positions du phonème dans la fenêtre d'analyse. Plus la base de données est importante, plus le TDNN est robuste aux distorsions.

Le TDNN code des formes statiques. Avec des délais temporels de longueur fixe, deux problèmes simples se posent: le délai n'est pas forcément bien aligné avec les données d'entrée comme dans l'expérience décrite ci-dessus et la taille du contexte temporel n'est pas approprié pour prendre en compte les informations pertinentes de tous les phonèmes. Une gestion dynamique des délais [Rander, Unnikrishnan 92], [Bodenhausen, Waibel 91] apporte une solution à ces deux problèmes. Rander modélise les délais temporels par des gaussiennes, l'algorithme d'apprentissage proposé doit alors estimer en plus des poids de connexion, la moyenne et la variance de la fonction pour caractériser les délais associés aux poids.

5.2.3 Intégration des sorties des TDNNs

Le réseau TDNN proposé par A. Waibel intègre de façon linéaire les activités de l'avant-dernière couche dans la dernière couche. Le nombre de vecteurs de l'avant-

dernière couche est figé, on ne peut gérer aucune variabilité temporelle. Pour la reconnaissance de mots isolés, le problème a été résolu en prenant une fenêtre d'entrée suffisamment large pour contenir le mot le plus long du corpus. Les mots courts sont appris en contexte de silence. Pour les phonèmes en contexte, il est difficile d'utiliser la même stratégie, on prend alors une fenêtre temporelle correspondant à la taille moyenne des phonèmes. Pour, les phonèmes longs, on apprend plusieurs positions de la fenêtre d'entrée. Les sorties en continu du réseau correspondent alors à une analyse centisecondes (1 vecteur de sortie tous les 10ms). Le meilleur score d'un vecteur de sortie correspond à une étiquette phonétique. La classification obtenue en sortie de ces réseaux est un treillis d'étiquettes phonétiques qu'il faut ensuite associer entre elles pour déterminer des phrases.

Lorsqu'une phrase inconnue est traitée, le réseau obtient en sortie une suite d'étiquettes phonétiques associées aux vecteurs de sortie sans pouvoir associer ces séquences d'unités en phonèmes, puis mots. Pour intégrer des séquences de parole continue, il est nécessaire d'utiliser un mécanisme d'alignement temporel dans les réseaux.

Pour obtenir les taux de classification phonétique au niveau des réseaux, les sorties sont intégrées de façon linéaire ; on reconnaît un segment phonétique lorsque la moitié au moins de ses trames a été bien reconnue. Le conflit entre deux phonèmes ayant le même nombre de trames reconnues est résolu en privilégiant le phonème ayant les meilleurs scores de sortie.

5.3 Conclusion

Les réseaux multicouches entraînés avec la rétropropagation du gradient et les réseaux LVQ2 permettent d'obtenir de meilleurs résultats de classification que les méthodes classiques de quantification vectorielle. Le MLP est cependant plus efficace que la LVQ2 pour traiter des données mal segmentées, bruitées ou de grandes dimensions. Cependant, le traitement "trame à trame" effectué dans les premières expériences ne permet pas d'obtenir une classification robuste. Il est d'autant plus critique pour les phonèmes non stables comme les plosives. Le codage statique n'est de toute façon pas bien adapté à la parole qui est par essence dynamique et nécessite un traitement temporel.

Le traitement temporel se situe à deux niveaux; au niveau contextuel et au niveau global d'une phrase. La gestion du contexte temporel des trames apporte plus de

robustesse pour reconnaître des segments phonétiques. L'intégration temporelle de des séquences d'unités phonétiques permet de reconnaître une phrase. Pour ces deux niveaux de traitement, il est intéressant d'avoir une gestion temporelle dynamique.

Pour coder un contexte temporel avec un algorithme de quantification ou de LVQ2, on utilise une représentation matricielle au lieu de vectorielle. Avec un MLP codant un contexte, on utilise un codage plus compact, une matrice de poids par couche pour tous les phonèmes, ce qui offre généralement de meilleurs performances de généralisation. Les TDNNs sont des dérivés des MLPs qui ont été développés pour prendre en compte des séquences temporelles et sont très bien adaptés à la classification phonétique.

Cependant, que ce soit les TDNNs ou les MLPs à contexte, le contexte temporel reste toujours fixe. Ces réseaux n'ont pas de gestion temporelle dynamique permettant de gérer des distorsions temporelles. Les réseaux multi-couches récurrents permettraient théoriquement cette intégration temporelle mais ils ne sont pas faciles à mettre en oeuvre. Les comparaisons entre des réseaux récurrents et les TDNNs ou MLPs à contexte [Franzini 89], [Bengio 91] sur des problèmes de classification phonétique montrent que les réseaux récurrents sont un peu plus performants. Cependant, ils sont beaucoup plus coûteux en temps calcul et de plus il n'existe pas à l'heure actuelle d'algorithmes d'apprentissage très performants pour les entraîner. Pour ces raisons, ils n'ont pas été abordés dans cette étude. Le type de réseau utilisé dans la suite de notre travail est le TDNN qui permet de prendre en compte les distorsions temporelles présentes dans les données et obtient de bonnes performances en classification. Une étude des capacités de ces réseaux, notamment pour résoudre des tâches complexes, fait l'objet du chapitre 6.

La reconnaissance de la parole continue nécessite un mécanisme d'intégration de séquences que n'ont pas ces réseaux. Les modèles de Markov cachés (HMMs) ou la programmation dynamique (DTW) gèrent des séquences de parole et compensent les distorsions temporelles. Pour gérer les distorsions et traiter de la parole continue avec des réseaux, il est nécessaire de leur intégrer un mécanisme d'alignement temporel ou d'utiliser un système ayant cette propriété en coopération avec le réseau. Ceci fait l'objet de la partie "systèmes hybrides" de cette thèse.

6. Apprentissage de TDNNs pour une tâche complexe

Le chapitre 5 relatait des expériences de classification phonétique pour des problèmes relativement simples. La base de données, "*DARPA Ressource Management 1*", testée dans la suite de cette étude, est d'une taille beaucoup plus importante, elle contient un vocabulaire de 1000 mots. La tâche est alors beaucoup plus complexe que dans nos premières expériences. La complexité ici réside dans la difficulté de généralisation des données d'apprentissage. Une base de données de taille importante n'implique pas forcement une complexité supérieure, la mesure donnée par le nombre de mots de vocabulaire est surement plus fiable pour évaluer la difficulté de la tâche. Cette base de données est largement utilisée comme base de comparaison au niveau international (cf chapitre 4 sur l'état de l'art des systèmes hybrides). Nous avons mené nos expériences au début sur un locuteur de cette base, JWS0 puis nous les avons validées sur deux autres locuteurs, CMR0 et BEF0. Le choix de cette base a été fait par l'ensemble des partenaires du contrat ESPRIT "Polyglot" et était motivé par l'existence de nombreux résultats au niveau international permettant de situer les performances obtenues.

Ce chapitre s'articule en deux parties :

La première partie porte sur l'étude des capacités des TDNNs pour une tâche complexe de généralisation. L'adéquation de la tâche à l'architecture pose des problèmes. Pour l'apprentissage de grandes bases de données, on construit de grands réseaux nécessitant l'évaluation de nombreux paramètres avec des temps d'apprentissage très longs. Les résultats obtenus avec des réseaux de plus petite taille en séparant la tâche globale en sous-tâches montrent tout l'intérêt des

architectures modulaires. Nous étudierons aussi les relations entre les prétraitements et les performances de classification.

La deuxième partie porte sur l'optimisation des algorithmes d'apprentissage. L'algorithme de rétropropagation du gradient est un algorithme non optimal. Il est d'autant plus difficile à optimiser que la tâche est complexe. Plus la base de données est grande, meilleure est la généralisation obtenue après apprentissage mais plus un mimimum global est difficile à trouver. Afin de vérifier la stabilité de l'algorithme programmé et les heuristiques utilisées pour le rendre plus optimal, nous avons testé systématiquement des jeux de paramètres différents. Certains de ces paramètres permettent de régler la structure du réseau, d'autres servent aux stratégies d'apprentissage.

La troisième partie résume les performances obtenues pour des sous-réseaux dédiés à la classification de sous-ensembles phonétiques. Les tests ont été menés pour trois locuteurs de la base DARPA RM1. Ces sous-réseaux ont été ensuite réutilisés dans des architectures modulaires afin de revenir à un critère global de classification. Les architectures modulaires font l'objet du chapitre 7.

6.1 Classification phonétique avec des TDNNs

6.1.1 Capacités des TDNNs pour une tâche complexe

Pour la reconnaissance de phonèmes prononcés par un seul locuteur, provenant d'un petit vocabulaire avec un bruit de fond réduit, des réseaux performants sont facilement construits. En éliminant une de ces contraintes, la tâche devient plus complexe. Nous avons éliminé la contrainte d'utiliser un petit vocabulaire. La base de données d'apprentissage DARPA RM1 contient 600 phrases pour un locuteur, le vobulaire utilisé est de mille mots. Nous avons étudié la facultée d'apprentissage d'un réseau pour un locuteur de cette base comparée à la base en français testée dans le chapitre 5.

La classification de phonèmes issus d'une grande base de données est bien évidemment plus difficile. La complexité de la tâche vient du nombre de formes à classifier et de la variabilité de ces formes. Le nombre de cellules cachées est nécessairement plus grand pour coder une tâche plus difficile et le temps de calcul nécessaire pour estimer tous les paramètres du réseau plus important.

Nous comparons les performances d'un réseau construit pour classifier les plosives voisées, puis tous les phonèmes, pour deux bases de données de tailles différentes : la base de données en français est décrite à l'annexe 1 et la base DARPA en américain à l'annexe 2.

Base de données en français

Pour la base de données en français d'un vocabulaire de 247 mots, le réseau global doit apprendre à classifier 31 phonèmes à partir de peu d'exemples, 2173 segments phonétiques.

Pour un réseau classifiant 31 formes avec 30 cellules dans la couche cachée, le nombre de paramètres, c'est-à-dire de poids associés aux connexions, s'élève à 6241. L'algorithme converge lentement et a plus de difficultés à trouver un optimum global qu'un petit réseau classifiant peu de formes. Pour classifier les phonèmes B D G, un réseau d'environ 500 paramètres avec 8 cellules en couche cachée obtient une performance supérieure à celle obtenue avec un grand réseau.

La fenêtre d'analyse est de 7 trames, pour un segment de 10 trames en entrée, le pas de translation étant de 1 sur chacune des couches, on obtient en sortie du TDNN, 4 vecteurs auxquels on associe 4 quasi-phonèmes, on reconnait un segment si au moins la moitié des quasi-phonèmes ont la bonne étiquette phonétique. Si parmi ces quatres vecteurs, la moitié est étiquetée d'un certain phonème et l'autre moitié par un autre phonème, on ne peut décider quel est des deux phonèmes celui qui est reconnu. On utilise alors les scores obtenus par les différents quasi-phonèmes pour décider du "gagnant".

Base DARPA RM, locuteur JWS0_4

Pour reconnaître un vocabulaire de 1000 mots, le réseau global doit apprendre à classifier 47 phonèmes à partir de 23000 segments phonétiques issus des 600 phrases d'apprentissage de DARPA. Pour classifier les formes /b/, /d/, /g/, /dd/, /dx/, le nombre de segments de la base d'apprentissage est de 1817 et de 241 sur le test.

La tâche de généralisation est beaucoup plus difficile pour cette base de donnée; la base est plus grande, le nombre de phonèmes est plus important, la segmentation utilisée est obtenue automatiquement à partir de HMMs. De plus, tous les contextes phonétiques sont présents (la base en français ne contient que des suites consonne-voyelle et voyelle-consonne)...

Les segments phonétiques ont une longueur moyenne de 7 trames par segment mais une variance plus importante que dans la base en français puisque la longueur varie entre 1 trame et 30 trames pour 1 segment. Nous avons utilisé des segments de taille variable en entrée du réseau, avec une taille minimale de 7 trames. Dans les premiers tests, les parties transitoires entre phonèmes n'ont pas été utilisées comme données d'apprentissage avec l'idée a propri que l'on ajouterait beaucoup trop de variabilité aux formes si on les apprenait. Le même nombre de coefficients est utilisé en entrée des réseaux, 16 coefficients FFT toutes les 12.8ms pour le corpus en français (échantillonnage à 10kHz) et toutes les 10ms pour DARPA (échantillonnage à 16kHz), à chaque coefficient, on soustrait l'energie moyenne de la trame qui est ajoutée en 17ième coefficient. Ces deux tests ne sont pas directement comparables mais donnent une idée des différences de performances entre les deux bases de données pour une tâche du même ordre de grandeur (4 ou 5 phonèmes).

Trouver les structures adéquates des classifieurs neuronaux nécessite de bien connaître les données utilisées. La tâche d'une méthode de classification est de décrire les frontières de décision entre les différentes classes dans un espace de formes aux dimensions souvent très grandes. Une solution optimale est particulièrement difficile à trouver pour les tâches complexes. On peut l'approcher en utilisant des connaissances a priori sur la base de données pour définir la struture du réseau. Par exemple, le nombre de cellules en couche cachée doit être adapté en fonction de la complexité de la tâche. Le tableau 6.1 montre les différences de performance pour plusieurs tâches de classification.

TDNN		*Base de données*	*Cellules*	*Apprentissage Trames*	*Test Trames*
b d g	3	Française 100 seg.	10	96.4	92.1± 2.6
tous les phonèmes	31	Française 2173 seg.	30	91.3	83.2±1
b d g dd dx	5	DARPA 1817 seg.	15	84.6	65.4±3.3
tous les phonèmes	47	DARPA 23000 seg.	50	57.3	45.8±1

Tab. 6.1 : ***Résultats obtenus pour différentes tâches de classification en fonction du nombre de formes à classifier et de la taille de la base de***

***données**. Les pourcentages donnés sont des taux de reconnaissance sur les trames de sortie du réseau.*

Les tests que nous avons effectués sur la base de donnée DARPA avec un réseau de 50 cellules dans la couche cachée, ont donné des scores très mauvais. Le nombre de cellules cachées est insuffisant pour la taille de la base de données. Le temps nécessaire pour un apprentissage avec un nombre de cellules suffisant pose le problème du nombre de paramètres à évaluer et est beaucoup trop long pour être envisageable sur une station de travail SPARC1.

H. Bourlard, sur cette base de données, utilise 512 cellules en couche cachée. Une structure "*hardware*" utilisant 5 RAPs "*Ring Array Processors*" [Bourlard 91] a été conçue pour entraîner des réseaux de très grandes dimensions. Par exemple pour la base DARPA RM indépendant du locuteur [Bourlard 92], une classification de 69 phones avec une structure de 512 cellules dans la couche cachée, une intégration temporelle de 9 trames sur la couche d'entrée et des vecteurs en entrée de dimension 26, nécessite ((26 x 9) +1) x 512 + ((512 x 1) +1) x 69 paramètres, c'est à dire environ 150 000 poids de connexions à estimer. Le temps d'apprentissage d'un tel réseau sur une machine mono-processeur comme un sparc est très long. Pour l'apprentissage de grands réseaux, des architectures modulaires sont étudiées, cf le chapitre 7.

Les TDNNs sont très performants pour des tâches de classification concernant peu de formes. Nous avons évalué la classification d'un sous-ensemble phonétique, les plosives voisées, dans le paragraphe suivant, nous présentons les scores de sous-réseaux permettant de décrire les 47 unités phonétiques de la base. Ces sous-réseaux seront ensuite "unifier" dans une architecture globale.

6.1.2 Sous-réseaux

Nous présentons les performances de sous-réseaux entraînés sur des tâches de moindre complexité correspondant à la classification de sous-ensembles phonétiques. Les partitions phonétiques ont été obtenues de façon empirique à partir des matrices de confusion de différents réseaux de macro-classes et de connaissances sur les ambiguités entre les classes phonétiques.

Ces sous-réseaux ont été optimisés. Un apprentissage des données "en continu" et non pas "segment par segment" a permis de prendre en compte les parties transitoires entre phonèmes. On a alors obtenu de meilleurs scores de généralisation à l'encontre de nos premières intuitions. Différents prétraitements

des données ont aussi été testés et ont amélioré les performances des sous-réseaux. Nous décrirons ces expériences dans les paragraphes suivants.

6.1.2.1 Apprentissage sur les segments

Pour l'apprentissage des TDNNs sur la base en français ou sur DARPA, la fenêtre d'analyse est de 7 trames, ce qui correspond à générer une sortie pour 7 trames en entrée. La fenêtre est déplacée d'une trame sur le segment. Dans cette expérience, les transitions ne sont pas apprises, l'évaluation des pourcentages de reconnaissance sur les segments est fonction de l'étiquetage, cette évaluation ne reflète pas la performance réelle des réseaux sur les phrases en continu. C'est pourtant souvent l'évaluation donnée sur les réseaux, elle n'est alors comparable avec aucune autre méthode. Les réseaux comportent une seule couche cachée.

Classification phonétique avec des sous-réseaux en français

Le tableau suivant (6.2) contient les résultats de classification pour des sous-ensembles de phonèmes de même taille que les plosives voisées pour la base de données en français. Les segments ont une taille fixe de 10 trames (la fenêtre est centrée sur le segment initial). Les contextes du TDNN sont respectivement de 3 et 5 trames sur les couches. Seul, le nombre de cellules dans la couche cachée varie en fonction des sous-réseaux. Cette partition phonétique a été obtenue à partir de la matrice de confusion du réseau global de 31 unités phonétiques et de connaissances phonétiques . Le tableau 6.2 regroupe les performances des sous-réseaux.

La différence entre la meilleure et la plus basse performance des sous-réseaux sur le test est de 18.6%. Globalement les scores sont assez bons (91.9%) et peu de différences entre les scores d'apprentissage et de test (4% en moyenne). Certains sous-ensembles phonétiques sont cependant plus difficiles à classifier que d'autres, les plosives non voisées, certaines voyelles (**peu, peur, peau,...**).

Sous-réseaux	*Couche cachée*	*Apprentissage Trames*	*Test Trames*
p t k #	15	93.2	84.1
b d g	10	96.4	92.1
f s ʃ	10	100	99.4
v z ʒ	10	100	97.8
m n ɲ	10	92.9	83.9
l r	8	98.3	97.3
a e ɛ	10	95.2	89.8
ã ɛ̃ ɔ̃	10	99.1	96.9
œ ø o ɔ	15	86.1	80.8
u y i	10	98.7	97.3
Moyenne	-	95.9	91.9± 2.6

Tab. 6.2 : ***Classification phonétique avec des sous-réseaux de dimensions réduites sur la base de données en français.*** *Le nombre total de segments de test pour 31 phonèmes est de 1964, soit environ 200 segments pour chaque sous-réseau.*

Classification phonétique avec des sous-réseaux sur DARPA

Les résultats sur DARPA sont beaucoup moins bons qu'en français. Nous présentons des résultats sur une partition de classes déterminée a priori à partir de la matrice de confusion du réseau global et de connaissances sur les ambiguités entre les classes phonétiques. Le tableau 6.3 regroupe les performances des sous-réseaux entraînés sur DARPA pour le locuteur JWS0.

Sous-réseaux DARPA	*Couche cachée*	*Apprentissage Trames*	*Test Trames*
p t k pt td kd si	25	82.5	68.9
b d g dd dx	15	84.6	65.4
m n ng en	12	84.5	76.1
l r y hh w	15	83.5	76.9
f s th sh	12	95.3	91.5
v z jh dh ch	15	92.6	83.9
aa ao ah eh ae	15	86.0	78.4
iy ix ey ih ay oy	20	87.6	78.8
ax aw uh uw ow er	20	95.6	88.0
Moyenne		87.8	78.3± 2.8

Tab. 6.3 : ***Classification avec des sous-réseaux de dimensions réduites sur la base de données DARPA.***

Le nombre total de segments phonétiques de test correspondant à 100 phrases étiquetées avec 47 phonèmes est de 3698 soit environ 400 segments phonétiques testés pour chaque sous-réseau.

La différence entre la meilleure et la plus basse performance des sous-réseaux sur le test est de 26.1%. Globalement les scores sont beaucoup moins bons qu'en français (78.3% en moyenne au lieu de 91.9) et il y a plus de différences entre les scores d'apprentissage et de test (9.5% en moyenne). Certains sous-ensembles phonétiques sont plus difficiles à classifier que d'autres, les plosives, les nasales, les liquides et certaines voyelles.

Variation de la taille de la fenêtre d'analyse du TDNN pour les voyelles en américain

Les voyelles posent un réel problème en américain, leur taille est extrêmement variable. Il peut paraître judicieux de les regrouper selon ce critère, 3 classes ont été déterminées: les phonèmes très courts /ax/ et /ix/, les phonèmes de taille moyenne (/ae/, /eh/, /ih/, /iy/, /uh/, /ah/, /aa/) et les diphtongues qui sont très longues (/ao/, /uw/, /aw/, /ay/, /ey/, /ow/, /oy/). le tableau 6.4 présente les taux de classification avec différentes tailles de contexte pour des sous-ensembles de voyelles.

Réseaux	*Fenêtre d'analyse*	*Apprentissage Trames*	*Test Trames*
ax ix	5	79.3	74.7
ax ix	7	77.2	67.4
phonèmes	7	83.8	77.8
diphtongues	7	90.5	84.9
diphtongues	9	91.2	85.7

*Tab. 6.4 : **Variation de la taille de la fenêtre d'analyse pour les voyelles courtes et les diphtongues.***

Ces expériences sur de petits réseaux montrent l'intérêt de faire varier la taille de fenêtre d'analyse notamment pour les voyelles très courtes [Hataoka, Waibel 89]. Ces résultats montrent que la taille de la fenêtre d'analyse de 7 trames est beaucoup trop grande pour les phonèmes ax et ix mais convient "relativement bien" aux autres voyelles. Le biais de cette classification est par exemple que le réseau classifiant les phonèmes courts ne contient que deux formes. Une petite fenêtre d'analyse peut aussi avoir des conséquences négatives si les formes ne sont pas bien segmentées. Quelques approches de gestion dynamique des délais sont à l'heure actuelle proposées [Rander, Unnikrishnan 92] [Bodenhausen,Waibel 91].

Nous avons utilisé pour les expériences sur DARPA, une fenêtre d'analyse toujours fixée à 7 trames et des tailles de fenêtre d'analyse contextuelle respectivement de 3 et 5 trames dans la première et la deuxième couche.

6.1.2.2 Apprentissage en continu

Les transitions sont souvent omises dans l'apprentissage des réseaux avec l'hypothèse tacite qu'une connaissance doit spontanément émerger pour les transitions dans le corpus de test.

Les transitions entre phonèmes n'ont pas été apprises dans les expériences décrites jusqu'ici. Plusieurs raisons justifient ce choix :

- Il est difficile d'entraîner un réseau ayant plus de 100 étiquettes de sortie pour un problème de temps calcul et le nombre de transitions différentes est de plus de 1000.

- On est donc obligé de classer avec la même étiquette la partie stable du phonème et les parties de transitions, ce qui a pour conséquence de rajouter à chaque classe une grande variabilité.

Les motivations pour modifier ce choix sont venues des résultats obtenus avec les méthodes hybrides. La reconnaissance des phonèmes après alignement temporel sur les scores de sorties a montré de nombreuses insertions entre les phonèmes dues au non-apprentissage des transitions. Nous avons donc appris les transitions entre les phonèmes. De plus, cette évaluation en continu reflète la performance réelle des réseaux sur les phrases, elle est alors comparable avec d'autres méthodes.

La fenêtre d'analyse glisse en continu sur le signal, cf fig. 6.1, et le réseau apprend toutes les positions de cette fenêtre. Soit P et A deux phonèmes adjacents, les fenêtres d'analyse sur la transition entre P et A reçoivent pour moitié l'étiquette P et l'étiquette A. L'exemple de la figure 6.1 montre plus clairement ce procédé.

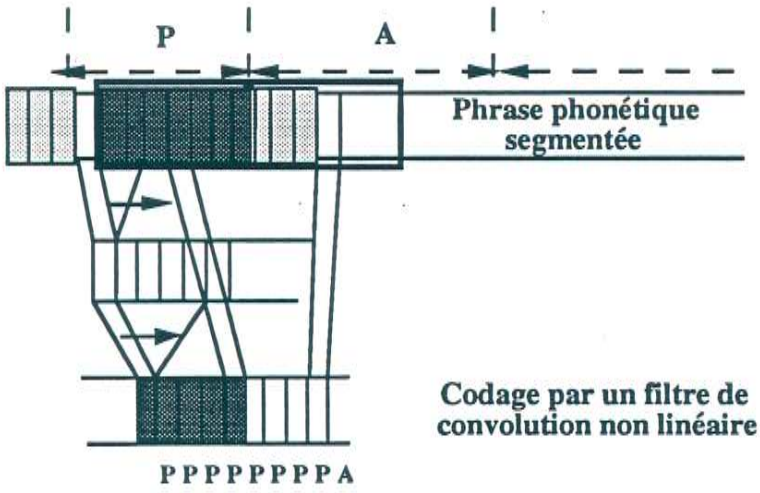

*Fig. 6.1 : **TDNN en continu aussi appelé S-TDNN pour SHIFT-TDNN.** En gris foncé sont indiquées les positions des fenêtres d'entrée apprises jusqu'à présent pour un phonème P, en gris pâle les positions de la fenêtre apprises sur la transition avec le phonèmes suivant. La fenêtre glisse sur les vecteurs d'entrée en continu. Les transitions sont apprises de moitié par le segment précédent et suivant.*

Sous-réseaux	*Couche cachée*	*Apprentissage Trames*	*Test Trames*
p t k pt td kd si	28	75.4	74.0
b d g dd dx	15	84.3	77.2
m n ng en	12	74.4	75.5
l r y hh w	15	91.9	87.8
f s th sh	15	90.1	86.9
v z jh dh ch	15	88.4	84.1
aa ao ah eh ae	15	83.3	79.2
iy ix ey ih ay oy	18	84.1	80.9
ax aw uh uw ow er	21	86.7	84.6
Moyenne	-	84.3	80.9± 1.4

*Tab. 6.5 : **Classification avec des sous-réseaux de dimensions réduites sur la base de données DARPA avec un apprentissage en continu.** Le nombre total de segments phonétiques de test pour 47 phonèmes est de 3698 soit environ 400 segments phonétiques testés pour chaque sous-réseau. Dans le cas continu, on apprend 6 fois plus de positions de la fenêtre d'analyse pour un phonème.*

On apprend toutes les positions de la fenêtre d'analyse en continu sur la phrase d'apprentissage dans cette expérience. La fenêtre d'analyse comprend 7 trames, c'est-à-dire qu'une trame de sortie du réseau correspond à 7 trames d'entrée. On obtient de moins bons résultats sur l'apprentissage qui est plus difficile, par contre les taux de généralisation sont globalement meilleurs puisqu'on a appris plus de

variabilité pour chaque forme et représentent réellement les performances des sous-réseaux sur les phrases. On ne peut d'ailleurs comparer les deux scores 78.3 et 80.9 obtenus sur le corpus de test pour deux apprentissages différents, sur les segments et en continu. L'apprentissage sur les segments donne un score qui ne correspond pas à une évaluation en continu. Il serait beaucoup plus bas, n'ayant jamais appris les transitions, si on étiquetait avec le réseau toute la phrase en continu.

6.1.3 Différents prétraitements du signal

Un avantage des réseaux MLPs par rapport aux méthodes utilisant directement une distance, est de pouvoir tester, en entrée du modèle, différents types de coefficients numériques sans définir une métrique a priori. Ils ont d'autres avantages indéniables comme ceux d'ignorer les détails non pertinents des formes d'entrée. On peut donc ajouter des connaissances de divers types au niveau des entrées des réseaux. Nous présentons quelques codages différents des entrées d'un réseau. Tous les tests ont été menés avec un apprentissage en continu des sous-réseaux.

Dans toutes les expériences présentées précédemment, on utilise une FFT classique. Pour la base de données DARPA RM1, le signal est échantillonné à une fréquence de 16Khz. Il est pré-accentué avec un filtre de 1 - 0.97(1/z), puis passé par une Fenêtre de Hamming de largeur 30ms. 256 coefficients FFT sont calculés tous les 10ms et convertis en 16 coefficients sur une échelle de Bark (100-8000hz). On prend ensuite le Log des energies (quasi-log). La moyenne des énergies est soustraite à chaque coefficient et rajoutée en 17ième coefficient (on note cette analyse FFT1 dans le tableau Ces valeurs sont normalisées entre -1 et 1 pour être utilisées en entrée du réseau.

6.1.3.1 Comparaison entre coefficients FFT et coefficients cepstraux

L'utilisation de coefficients cepstraux permet de décorréler les coefficients FFT. Nous présentons une comparaison entre des coefficients corrélés et décorrélés pour un réseau classifiant les plosives voisées.

Utiliser des coefficients décorrélés est important pour les modélisations gaussiennes de HMMs continus ou de réseaux à cellules gaussiennes. En effet, les gaussiennes utilisées dans la majorité des cas sont à matrices de covariance diagonales pour limiter le nombre de paramètres libres et le temps de calcul, ce choix n'est théoriquement consistant que si les coefficients d'entrée sont non

corrélés. Des études, [Bengio 91], [Boiteau 92], ont montré que les réseaux à unités sigmoïdes offrent de moins bons résultats pour des paramètres décorrélés en entrée. Pour ce type de données les plans séparateurs interclasses sont orthogonaux aux axes des paramètres d'entrée, plans séparateurs qu'un réseau a du mal à définir. Les résultats obtenus (tableau 6.6) confirment cette constatation.

Analyses	*Nb de coefficients*	*Apprentissage Trames*	*Test Trames*
FFT	16	79.1	72.2 ± 2
FFT + E	17	84.3	77.2 ± 2
CEPSTRES	16	85.8	73.5 ± 2
CEPSTRES	10	84	74.5 ± 2

*Tab. 6.6 : **Différentes analyses** Réseau TDNN(3,5) pour la classification de B D G DD DX avec une couche cachée de 15 cellules. L'apprentissage est fait en continu.*

Le coefficient d'énergie moyenne soustrait à chaque paramètre permet de normaliser le vecteur et donne une meilleure description des vecteurs. Ces expériences montrent que la normalisation des trames est très importante. Les coefficients décorrélés améliorent l'apprentissage mais n'apportent pas de meilleurs scores de généralisation.

6.1.3.2 Ajout de coefficients FFT et des dérivées premières et secondes

L'analyse du signal utilisé par le laboratoire de Philips à Aix-la-chapelle pour entraîner les CHMMs (HMMs continus) est aussi une analyse FFT. Cette analyse est utilisée ici pour entraîner les sous-réseaux afin d'avoir un système complètement comparable. Toutes les 10ms de parole, 30 coefficients sont calculés sur une fenêtre de 25 ms de parole. L'energie moyenne d'une trame est soustraite aux coefficients et ajoutée au vecteur comme 31ième valeur (FFT 2 dans le tableau de résultats). 15 coefficients correspondant aux dérivées premières et 15 coefficients correspondant aux dérivées secondes obtenus à partir des dérivées premières sont calculés sur 70 ms de parole (30ms de contexte gauche et 30 ms à droite) et ajouté au vecteur, on ajoute la dérivée première et seconde de l'energie. Les vecteurs d'entrée sont donc de dimension 63 (FFT 3 dans le tableau de résultats).

De meilleurs scores avec les HMMs ont été obtenus avec 33 coefficients. Ces 33 coefficients correspondent à 15 coefficients FFT calculés en sautant un canal sur deux sur les 30 coefficients FFT (cf FFT2), plus la derivée première sur les 15 coefficients ainsi que l'energie moyenne, sa dérivée première et seconde.

Les résultats de ce tableau 6.7 correspondent à la moyenne des scores pour tous les sous-réseaux.

Analyses	*Nb de coefficients*	*Apprentissage Trames*	*Test Trames*	*Diminution d'erreur / FFT(1)*
FFT (1) LIMSI	17	84.3	80.9± 1.4	-
FFT (2) Philips	31	85.5	83.2± 1.3	-12,0%
FFT (3) + d' + d" Philips	63	88.9	85.8± 1.2	-25.7%
FFT (4) + d' + d" Philips	33	85.6	84± 1.3	-16.2%

*Tab. 6.7 : **Ajout de coefficients, FFT, dérivées premières et secondes.** Performances moyennes obtenues pour tous les sous-réseaux.*

Ce tableau montre clairement que de meilleurs résultats sont obtenus avec plus de coefficients. Ces meilleurs taux de classification dus à la précision des vecteurs sont acquis aux dépens du temps calcul. La comparaison entre des vecteurs de 31 coefficients et de 63 coefficients composés des 31 précédents paramètres auquels sont adjoints les dérivées premières et secondes tend à montrer que le TDNN n'est pas capable d'extraire tout seul toute l'information temporelle grâce à la fenêtre d'analyse du réseau puisqu'on obtient de meilleures performances avec les informations de la dérivée première et seconde. Cependant, cette idée s'avère fausse car on peut aussi expliquer cette différence avec le raisonnement suivant, le TDNN est capable d'extraire des informations de type dérivées en intégrant un contexte, c'est ce qu'il fait pour les deux approches avec 31 et 63 coefficients. On peut alors imaginer qu'il calcule des dérivées de degrés 3 ou 4 à partir d'une fenêtre d'intégration sur des dérivées premières ou secondes ce qui permet d'obtenir de meilleurs résultats.

Quelques remarques concluent ce paragraphe sur les prétraitements. Le prétraitement (FFT, cepstre...), la précision utilisée (nombre de coefficients) et le codage des données d'entrée (energie, dérivées...) sont très importants pour les réseaux . Pour la reconnaissance de la parole, les résultats obtenus indiquent que le choix des paramètres influence beaucoup les performances du système de reconnaissance. Le calcul du cepstre ne permet pas d'obtenir de meilleurs résultats pour ces tests de classification (tableau 6.6). Par contre, l'ajout du coefficient d'energie est important (tableau 6.6). Avec plus de paramètres FFT (tableau 6.7), on obtient de meilleurs taux d'apprentissage pour cependant un coup de calcul plus important, on obtient 3% de mieux entre 17 et 31 coefficients. La prise en compte des dérivées est intéressante, les pourcentages gagnent 2%. Globalement, le codage des entrées (FFT (32) + Dérivées1 et 2 + Energie + Dérivée de l'energie) a permis d'extraire des indices plus robustes et d'améliorer le pouvoir de généralisation du réseau puisque le gain est de 5% de reconnaissance sur le test.

6.2 Problèmes d'apprentissage

Les performances de l'apprentissage sont données par le pouvoir de généralisation du réseau à d'autres données que celles qu'il a apprises. L'apprentissage est sujet à un compromis entre fiabilité et coût. Plus la base de données est grande, plus la tâche est complexe et plus le coût pour trouver un optimum global est important. Mais plus on apprend de données et plus l'apprentissage est fiable. En effet, l'apprentissage d'une base de données de taille réduite ne produit pas une généralisation robuste à des données inconnues. Mis à part ce problème de généralisation dû à la taille de base utilisée, on cherche à optimiser l'apprentissage et à réduire le coût de calcul.

Le choix des paramètres d'apprentissage permettant d'atteindre un optimum global satisfaisant est très important pour éviter des temps d'apprentissage trop longs. L'apprentissage d'un réseau nécessite donc une bonne connaissance de l'influence de tous les paramètres mis en cause et une bonne stratégie d'apprentissage. Les expériences décrites ont toutes été menées sur la base de données DARPA RM pour un locuteur, JWS0_4. Les meilleures achitectures et choix des paramètres sont testées pour deux autres locuteurs BEF0 et CMR0.

6.2.1 *Bon conditionnement* de l'apprentissage

L' algorithme de rétro-propagation du gradient d'erreur est non optimal :

- Il peut facilement être piégé dans un minimum local,

- Plusieurs configurations de poids peuvent être solutions pour un minimum satisfaisant,

- La configuration initiale, l'ordre de présentation des données influent sur la solution obtenue.

La stratégie d'apprentissage et la structure du réseau peuvent tenter d'éviter les minima locaux. Pour optimiser l'apprentissage du réseau, des tests ont été menés sur la structure :

- sur l'initialisation,

- sur le nombre de couches,

- sur le nombre de cellules des couches cachées,

et sur la stratégie d'apprentissage:

- validation croisée,

- mis en attente de données,

- apprentissage automatique,

- choix de la sigmoïde et sorties désirées,

- gestion du gain de modification des poids EPS.

Nous présentons les différents tests que nous avons menés dans les paragraphes suivants.

6.2.2 Variations de la structure

Initialisation

On initialise les poids de chaque couche en tirant aléatoirement des nombres dans un intervalle qui tient compte des connexions arrivant sur les cellules des couches (ou fan-in). Cet intervalle est du type [-Kte / Racine(fan-in) , +Kte / Racine(fan-in)] où la constante Kte est déterminée de façon heuristique en fonction de la sigmoïde. L'activité est donc normalisée, quelque soit la dimension de l'entrée ou des couches

cachées, la fonction sigmoïde est la même. Le but est de rendre l'algorithme transparent à l'initialisation choisie.

Différentes initialisations impliquent différentes évolutions de la surface d'erreur. Une bonne initialisation ne doit pas beaucoup influer sur le pouvoir de généralisation du réseau. Effectivement, le pouvoir de généralisation est le même ici malgré les différentes initialisations (tableau 6.8).

Initialisation	*Nb. de cellules*	*Appentissage Trames*	*Test Trames*
INIT1	15	83.5	77.0
INIT2	15	84.9	76.4
INIT3	15	85.3	76.8
INIT4	15	84.3	77.2
INIT5	15	86.4	76.3

*Tab. 6.8 : **Différentes initialisations du réseau TDNN pour la classification des plosives voisées /b/, / d/, /g/, /dx/, /dd/.** La validité statistique sur le test est de ± 1.9.*

Nombre de couches

On a émis l'hypothèse que les phonèmes pouvaient être séparés à l'aide d'une couche cachée. Voici un test avec deux couches cachées qui valide ce choix. Le tableau 6.9 montre la différence entre un apprentissage avec une et deux couches cachées.

Réseau **B D G DD DX**	*Nb. de connexions*	*Apprentissage. Trames*	*Test Trames*
TDNN(3,5) 15	775	84.3	77.2
TDNN(3,4,2) 15-10	1495	90.1	77.8

*Tab. 6.9 : **Variation du nombre de couches cachées pour un réseau TDNN permettant de classifier les plosives voisées.** La validité statistique sur le test est de ± 1.9.*

On apprend mieux les données d'apprentissage avec un réseau à deux couches cachées. Le réseau code plus précisément les données initiales, grâce à une projection sur une couche en plus mais il ne généralise pas mieux. Apprendre avec

une couche en plus (3 couches cachées) serait encore plus long et risquerait de simplifier trop les données sur la dernière couche et d'accentuer le phénomène déjà remarqué avec deux couches cachées. En conclusion, un réseau trop spécialisé sur les données d'apprentissage risque de moins bien généraliser à des données inconnues.

Nombre de cellules des couches cachées

Pour améliorer le codage, l'intuition est d'ajouter des cellules dans la couche cachée pour avoir des frontières plus fines entre les données. Le risque est d'avoir le même phénomène de sur-apprentissage d'une base sans augmentation du pouvoir de généralisation.

Le nombre de cellules dans la couche cachée influe sur le pouvoir de généralisation. Nous avons fait varier le nombre de cellules de 5 à 40 (tableau 6.10).

Nb. de connexions	*Nb de cellules*	*Apprentissage Trames*	*Test Trames*
390	5	75.9	70.1
775	10	82.8	74.3
1160	15	84.3	77.2
1545	20	84.8	77.0
3085	40	87.4	76.2

*Tab. 6.10 : **Variation du nombre de cellules dans la couche cachée pour la classification des plosives voisées.** La validité statistique sur le test est de ± 1.9.*

Il est remarquable de voir que 15 cellules suffisent à obtenir un bon taux de généralisation. Les performances ne sont pas meilleures avec plus de 40 cellules. Lorsqu'il y a trop de cellules dans la couche cachée, l'apprentissage avec le même nombre de données devient trop lié à ces données et risque d'avoir moins de pouvoir de généralisation. Plus le nombre de paramètres est élevé, plus le temps d'apprentissage est long, on choisira toujours le plus petit nombre satisfaisant de cellules dans la couche cachée.

Optimisation avec plusieurs réseaux

L'algorithme étant non optimal, on peut imaginer que des réseaux de configurations différentes mais ayant appris les mêmes tâches pourront avoir des performances globales similaires avec des répartitions des scores différentes par

classe . Si les performances des réseaux sont suffisamment élevées, la coopération des réseaux en parallèle ne peut qu'améliorer la performance globale.

L'architecture de ce réseau modulaire est très simple, on propage l'activité en parallèle dans tous les réseaux, la somme (SOM.) ou le maximum (MAX.) des activités de leurs dernières couches est effectué. L'utilisation de l'opérateur MAX sur les sous-réseaux signifie que l'on prend en compte les scores le plus élevés de tous les réseaux pour prendre une décision. Les sous-réseaux mis en parallèle ont différentes initialisations ou différents nombres de cellules dans la couche cachée. En moyenne, ils ont tous des performances de généralisation d'environ 76% (cf tableau 6.11).

Architecture	*nb de réseaux*	*Apprentissage Trames*	*Test Trames*
réseau	1	84.3	77.2
réseaux MAX.	4	86.6	79.9
réseaux SOM.	4	86.5	79.8
réseaux MAX.	7	88.5	80.5
réseaux SOM.	7	87.9	80.6

Tab. 6.11 : ***Coopération de plusieurs réseaux BDG en parallèle pour obtenir une solution plus optimale.*** *La validité statistique sur le test est de ± 1.9.*

Avec 4 sous-réseaux en parallèle, on améliore les score du test d'environ 2.5%. Avec 7 sous-réseaux, on améliore le score de l'apprentissage sans produire d'effet sur le score du test qui est stable. Cette stabilité n'est pas une preuve certaine d'optimisation. Les 3 sous-réseaux ajoutés à l'architecture peuvent ne pas contenir d'informations complémentaires.

L'utilisation de sous-réseaux en parallèle montre peu d'amélioration par rapport au réseau seul pour un coût certain. De plus, pour valider réellement cette idée, il faudrait utiliser beaucoup de réseaux pour éliminer l'a priori du choix des réseaux et ne pas être dépendant de la redondance qu'ils contiennent.

6.2.3 Stratégies d'apprentissage

Ce paragraphe regroupe des idées pouvant optimiser et accélérer l'apprentissage. Ces deux actions ne vont pas souvent de pair. En effet, l'accélération de l'apprentissage se fait souvent aux dépens de la fiabilité. On cherchera le meilleur compromis entre les deux.

Validation croisée

Le mécanisme de validation croisée est utilisé pour améliorer le pouvoir de généralisation de l'apprentissage. La base de donnée est en général divisée en trois parties dont deux parties servent à l'apprentissage :

- une servant à la modification des poids,

- une pour tester l'erreur et le pouvoir de généralisation lors de l'apprentissage, ce corpus à valider l'apprentissage.

Il permet d'éviter le phénomène de sur-apprentissage de la base de données qui fait en général baisser les taux de reconnaissance sur la base de test.

Ce problème de sur-apprentissage est inhérent à une base de données de taille limitée. Au cours de l'apprentissage, une partie des données sert exclusivement à tester la performance de généralisation. En pratique, par exemple pour la base de données DARPA, on apprend avec 500 phrases et l'erreur globale est calculée sur les 100 phrases restantes du corpus d'apprentissage qui ne servent pas à modifier les poids.

Apprentissage automatique

L'apprentissage permet d'ajuster automatiquement les paramètres. La stratégie d'apprentissage dans nos expériences est décrite avec un interpréteur de commandes qui est capable de gérer des boucles de traitements, des tests d'arrêt, de créer des variables et de les tester. L'interpréteur de commandes contient toutes les fonctions du programme; initialisation, création de réseaux, apprentissage, test d'erreur, affichage des performances du réseau, affichage des activités dans les couches, affichage des matrices de connexions, mise en attente des données, différents prétraitements sur les données... plus des commandes de plus hauts niveaux qui décrivent la stratégie globale.

Description simplifiée de la stratégie utilisée:

1- **Description des entrées et des sorties**

2- **Description et initialisation du réseau**

3- **Initialisation des paramètres** servant à l'apprentissage: le pas de modification **EPS**, le seuil de tolérance de l'erreur, le seuil d'arrêt, l'erreur initiale....

4- Mémorisation des commandes suivantes permettant de faire des sauts en avant ou en arrière sur les lignes de commandes.

5- Sauvegarde du réseau

6- **Apprentissage de N cycles** (N étant égal au nombre moyen d'occurrences par forme). On utilise un gradient stochastique; A chaque cycle, c'est à dire après chaque présentation d'un échantillon de toutes les formes à classifier, les poids des connexions sont modifiés. Cette commande équivaut à N modifications des poids.

7- **Test d'erreur** sur le corpus de validation

8- **Heuristiques sur l'erreur.** On étudie l'évolution de l'erreur. Le seuil de tolérance d'erreur diminue en fonction du nombre de cycles d'apprentissage. Pour simplifier, on peut retenir les trois cas suivants.

- si la différence entre l'erreur obtenue lors de ce test et le précédent est inférieure au seuil de tolérance d'erreur, on recommence à l'étape 5 sans modifier EPS.

- si la différence a augmenté au dessus du seuil de tolérance donné, on reutilise la dernière sauvegarde du réseau et on modifie le gain EPS, si le gain est à zéro on sort de la boucle de commande étape 9 sinon on recommence à l'étape 5.

- si l'erreur est inférieure au seuil d'arrêt on sort de la boucle de commande étape 9.

9- fin de la boucle de commande

Le contrôle de l'évolution de l'apprentissage utilisé est automatique mais il est très dépendant de seuils et de la connaissance a priori du comportement du réseau. Les paramètres importants sont le nombre de cycles d'apprentissage, EPS et le seuil de tolérance d'erreur.

- Le nombre de cycles choisi correspond en moyenne au nombre de segments par classe, c'est-à-dire pratiquement à toute la base d'apprentissage.

- Le seuil de tolérance d'erreur est négatif en début d'apprentissage, ce qui permet de sortir de minima locaux. Cette méthode s'apparente à la technique de recuit simulé. Au cours de l'apprentissage, ce seuil devient moins permissif.

La stratégie pour le gain EPS

Dans nos expériences, le gain décroît de façon linéaire d'un pas faible en fonction de l'erreur globale sur le corpus servant à la validation croisée. Il est choisi initialement à une valeur relativement petite. On utilise un gain spécifique pour chaque unité simplement en divisant le gain EPS par la racine du nombre de connexions incidentes à une unité ce qui permet de normaliser l'erreur rétropagée.

la sigmoïde et les sorties désirées

Le choix de la sigmoïde permet souvent de régler un bon apprentissage. Il est intéressant d'utiliser une fonction n'ayant pas de courbures très fortes vers 1 et -1 sinon la dérivée est rapidement proche de 0. En effet, lorsque l'activité est proche de 1, les poids ont tendance à ne plus bouger si la courbure est trop forte. Utiliser des sorties désirées à -0,9 et 0.9 au lieu de -1 et 1 a le même effet et permet d'obtenir de meilleurs apprentissages.

La sigmoïde choisie est une fonction symétrique : $\mathrm{Tanh}(\frac{kx}{2})$

Le paramètre k sert à régler la sigmoïde. On choisit l'activité x correspondant à une sortie à 0,9 en modifiant k.

$$k = \frac{\ln 19}{x}$$

Quelque soit le nombre de poids du réseau et le nombre de cellules en couche cachée, la fonction est la même. Tous les poids initiaux et les modifications sur les poids sont normalisés par la racine du fan-in, le fan-in étant le nombre de connexions incidentes d'une cellule.

Le schéma suivant Fig 6.2 montre deux courbes sigmoïdes très proches Arctan() et Tanh().

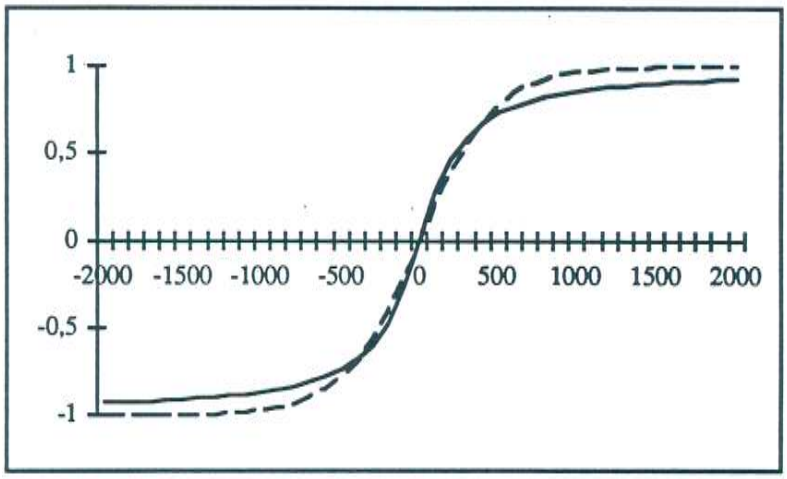

Fig 6.2 : La courbe en pointillé représente la fonction ***Tanh(kx/2)*** *pour k= LN(19)/700, la sortie de cette fonction est de 0.9 pour une activité de 700. La courbe en trait plein représente* ***Arctan(kx/2)*** *pour k= Tan(0.9 pi/2)/700. L'axe des abcisses x correspond aux activités incidentes d'une cellule. L'axe des ordonnées y correspond aux sorties des cellules, les sorties sont comprises entre -1 et 1.*

La courbe ARCTAN pourrait être un meilleur choix pour les raisons invoquées plus haut. Plutôt que de modifier la fonction sigmoïde TANH, nous avons utilisé des sorties désirées ayant pour valeur 0.9 et -0.9.

Les résultats du tableau suivant (tableau 6.12) correspondent à la moyenne des scores pour tous les sous-réseaux.

Sorties désirées	*Apprentissage Trames*	*Test Trames*	*Diminution de l'erreur*
-1 , 1	88.9	85.8 ± 1.2	-
-0.9 et 0.9	93.8	90.1 ± 1	- 30.3%

Tab. 6.12 : ***Moyenne des scores pour tous les sous-réseaux, on utilise des sorties désirées à 0,9 et -0,9 au lieu de 1 et -1.***

Les sous-réseaux ont tous été entraînés dans les mêmes conditions, seules les sorties désirées ont été modifiées. Le nombre de cycles d'apprentissage effectués est cependant beaucoup plus long dans le cas des sorties à 0.9. En effet, les poids ne sont pas bloqués à 1 ou -1. Les scores sont bien meilleurs (environ +4.5%) aux dépens du temps d'apprentissage qui est de 2 à 3 fois plus long en moyenne sur les sous-réseaux. Par exemple, pour l'apprentissage d'un réseau classifiant les plosives voisées, le nombre de cycles passe de 87400 à 178200. Ce réseau contient 3230 poids à évaluer. Avec une machine à 24 MIPS, 200 cycles d'apprentissage équivaut à 1mn

30 de TCPU. Soit pour l'apprentissage des plosives voisées 24 heures dans le cas des sorties à -1 et 1 et un peu plus du double pour les sorties à -0.9 et 0.9.

6.3 Performances des sous-réseaux pour trois locuteurs

Les sous-réseaux ont tous été entraînés dans les mêmes conditions. Le tableau 6.13 regroupe la moyenne des performances des sous-réseaux pour trois des locuteurs de la base DARPA RM1.

Locuteurs	*Apprentissage Trames*	*Test Trames*
JWS0	93.8	90.1± 1
CMR0	90.7	85.2± 1.2
BEF0	91.5	86.3± 1.2

*Tab. 6.13 : **Moyenne des scores de tous les sous-réseaux pour trois locuteurs de DARPA.***

Les entrées sont des vecteurs de 63 coefficients. On a utilisé les meilleurs choix pour la stratégie d'apprentissage et l'architecture de réseau obtenus pour le locuteur JWS0 pour entraîner les sous-réseaux de deux autres locuteurs, dont l'un est une femme et l'autre est assez difficile à reconnaître d'après les résultats obtenus avec des HMMs. On obtient des performances tout à fait correcte avec ces deux autres locuteurs, ce qui valide d'une certaine manière les heuristiques trouvées et les paramètres d'apprentissage utilisés. Chacun des réseaux nécessite 2 à 3 jours de TCPU sur une machine mono-processeur d'environ 20 MIPS (200 cycles d'apprentissage pour de tels réseaux prend environ 1 mn 30 de TCPU).

6.4 Conclusion

Nous avons résumé dans ce chapitre une série d'expériences visant à tester les capacités des réseaux, le problème de l'adéquation de l'architecture du réseau à la tâche, les relations entre prétraitements et performances de classification et l'optimisation des algorithmes d'apprentissage.

Pour des tâches complexes, une structure globale est difficilement réalisable sans utiliser une machine spécialisée en traitement parallèle. Le problème d'optimisation de l'apprentissage devient aussi beaucoup plus critique avec de grands réseaux. L'algorithme de descente de gradient est un mécanisme global qui converge avec difficulté lorsque les dimensions des réseaux augmentent. Il serait intéressant d'avoir des règles d'apprentissage globales et locales dans les réseaux permettant de gérer des tâches complexes. Plus la tâche est simple, plus les perceptrons sont performants. Pour des classifications de sous-tâches, l'apprentissage utilisé converge assez facilement vers des optimums satisfaisants.

Différentes expériences ont été menées avec des sous-réseaux. Elles ont conduit à adopter certaines stratégies afin d'obtenir de bonnes performances en classification phonétique. Par exemple, le codage des entrées (FFT (32) + Dérivées1 et 2 + Energie + Dérivée de l'energie) a permis d'extraire des indices plus robustes et d'améliorer le pouvoir de généralisation du réseau puisque le gain est de 5% de reconnaissance sur le test, pour le locuteur JWS0. Un autre exemple est celui de l'utilisation de sorties à 0.9 et -0.9 qui a permis d'optimiser l'apprentissage et d'obtenir de meilleurs performances, soit environ 4.5% de mieux sur le test. Un autre aspect important des réseaux est l'apprentissage des transitions, l'amélioration du pourcentage de reconnaissance n'est pas notable pour l'évaluation des sorties des réseaux mais il le devient sur les mots lorsqu'on utilise une intégration temporelle de type algorithme de Viterbi sur les sorties du réseau (cf chapitre 8).

Les scores de classification sur les vecteurs de sortie des sous-réseaux pour trois locuteurs (JWS0, CMR0, BEF0) de la base DARPA sont en moyenne de 87%. On constate pour les autres locuteurs les mêmes améliorations dues au prétraitement, à l'optimisation de l'algorithme d'apprentissage que celles obtenues avec le locuteur JWS0. Les heuristiques trouvées et les paramètres d'apprentissage utilisés sont donc stables puisqu'on obtient d'aussi bonnes performances avec d'autres locuteurs.

Des architectures modulaires réutilisant les sous-réseaux entraînés séparément permettent de gérer une tâche plus difficile. La modularité apporte d'une certaine façon un aspect d'apprentissage local qui manque aux réseaux multicouches. Différentes architectures modulaires sont proposées dans le chapitre 7.

7. Architectures modulaires

On présente dans ce chapitre une solution à l'apprentissage sur de grandes bases de données: la modularité. La notion de modularité permet également de concevoir des systèmes correspondant à la décomposition d'une tâche complexe en sous-tâches enchainées séquentiellement ou réalisées en parallèle. Elle introduit ainsi dans l'architecture les connaissances de base que l'on possède sur le problème. En procédant de la sorte, la taille de l'espace de paramètres du système reste relativement faible et la recherche d'une solution lors de l'apprentissage est ainsi considérablement allégée. En effet, la capacité d'un réseau à réaliser une tâche n'offre d'intérêt que si le temps de calcul nécessaire à son élaboration est suffisamment court et si les coefficients numériques qui lui sont attachés sont suffisamment petits pour le rendre opérationnel (cf chapitre 6). Travaillant avec une puissance de calcul limitée à un processeur, il est impossible d'élaborer un réseau de très grande taille. Ce problème a été résolu par H. Bourlard puisqu'il utilise une machine hautement parallèle comprenant 5 "*Ring Array Processor*" pour construire un réseau de plus de cinq cents cellules dans la couche cachée. Il est possible cependant pour traiter ce problème avec des moyens plus simples de décomposer la tâche en sous-tâches. Des réseaux plus petits sont alors construits pour classifier des phonèmes particuliers. L'apprentissage devient local à des sous-ensembles de l'espace. Ensuite, la construction d'une architecture modulaire à partir des sous-réseaux déjà entraînés a pour but de revenir à un critère global de classification.

Dans le but d'apprendre à un réseau une tâche complexe de classification phonétique, nous avons étudié différentes architectures de réseaux modulaires. Une fois la décomposition de la tâche effectuée, il faut construire une architecture

modulaire et gérer les échanges entre modules. Nous proposons plusieurs solutions d'intégration des sous-réseaux dans une architecture globale. La base de données DARPA RM1 "speaker dependent" a permis de vérifier l'intérêt de ces architectures modulaires. Pour un locuteur, on a utilisé un corpus d'pprentissage de 600 phrases soit 160000 trames étiquetées parmi 48 phonèmes.

La première partie de ce chapitre est dédiée aux architectures utilisant un réapprentissage neuronal global, on les notera G-TDNN (G pour global). L'architecture modulaire présentée est inspirée des travaux d'Alex Waibel [Waibel 88]. Nous proposons aussi une nouvelle architecture éliminant l'a priori de choisir une partition particulière de phonèmes, appelée G-FE-TDNN (FE-TDNN pour "*Features Extracted TDNN*").

La deuxième partie est dédiée aux architectures hiérarchiques. Nous proposons des architectures hiérarchiques de sous-réseaux appelée H-TDNN pour "*Hierachical TDNN*" ou HT-TDNN "*Hierachical TDNN with Traps*" ne nécessitant pas de réapprentissage global.

7.1 Capacités des réseaux modulaires

Une architecture modulaire permet une souplesse d'utilisation intéressante. Elle scinde le problème en sous-problèmes beaucoup plus simples à résoudre, où un minimum satisfaisant est plus facile à trouver. Une caractéristique des réseaux est l'élaboration de représentations internes. Les informations codées par de petits réseaux peuvent être réutilisées ensuite dans une architecture globale.

Cet aspect modulaire permet donc d'une part de réduire le temps de calcul et d'autre part d'aider l'apprentissage en alliant connaissances a priori et connaissances déduites des données. Au même titre que certaines méthodes telles que le recuit simulé qui permet d'éviter d'être pris dans un minimum local au cours de l'apprentissage, l'utilisation de structures modulaires est un moyen de guider l'apprentissage en utilisant nos connaissances des données à traiter.

Dans les expériences menées, les connaissances introduites dans les réseaux concernent les ambiguïtés acoustiques. Ces connaissances vérifient la proximité spatiale des formes. Le but de la structure modulaire est de dédier certaines unités à à la résolution de sous-problèmes tels que la séparation de phonèmes acoustiquement proches comme /b/, /d/, /g/ ou /p/, /t/, /k/... que l'on sait délicate.

Partition de l'ensemble des classes

Dans toutes les architectures modulaires présentées, on a émis l'hypothèse de disposer de l'expertise suffisante pour décomposer la tâche de façon a priori satisfaisante. Un sous-ensemble de l'ensemble des classes sera défini comme une macro-classe. Nous avons utilisé des connaissances phonétiques pour définir a priori une partition de l'ensemble des classes et la matrice de confusion obtenue avec un réseau classifiant ces macro-classes a été utilisé pour confirmer ce choix.

Partitionner l'espace a pour conséquence de concentrer l'apprentissage sur certaines ambiguïtés. Tenter de partitionner automatiquement l'ensemble des classes a pour effet d'avoir une répartition optimale pour le critère utilisé mais a la même conséquence.

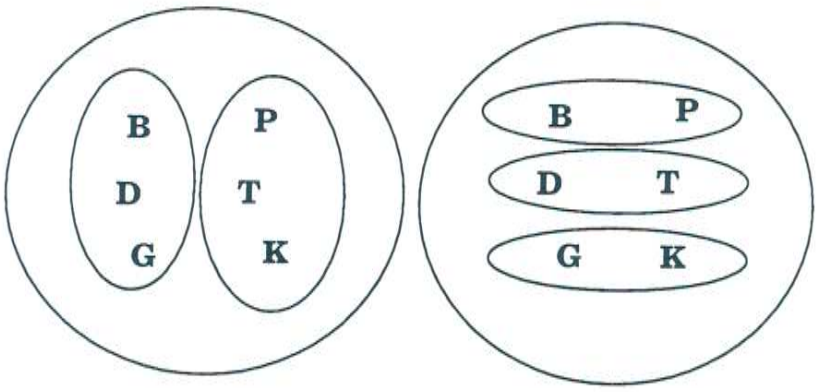

Fig 7.1 : ***Exemples de 2 partitions de l'ensemble des classes phonétiques.***

Si sur la figure 7.1 la représentation optimale correspond à celle de gauche, la deuxième représentation montre d'autres ambiguïtés phonétiques moins fréquentes mais tout aussi délicates à résoudre qui ne seront pas prises en compte par un sous-réseau.

On voit qu'il est difficile de faire un choix a priori. Il est alors intéressant d'utiliser une structure basée sur différentes partitions définies à partir de plusieurs critères. Nous proposons une architecture de ce type appelée FE-TDNN "*Features Extracted TDNN*", cette structure est décrite un peu plus loin.

Recherche automatique d'une structure modulaire

Il n'est toutefois pas toujours possible de disposer de l'expertise suffisante pour décomposer une tâche. Il serait intéressant de trouver une méthode qui effectue cette décomposition automatiquement.

Un algorithme hybride, de rétropropagation et d'apprentissage compétitif, a été récemment proposé afin de déterminer automatiquement les modules [Jacobs et al., 1991], [Jacobs 90]. La solution globale est alors optimale. Cette approche est assez complexe et n'a pour l'instant été appliquée qu'à des problèmes d'ordre restreint. L'architecture décrite contient des modules experts et un module superviseur qui va diriger la répartition des formes dans les réseaux experts. Il s'agit d'apprendre tout en décomposant la tâche. Cette méthode est une voie de recherche prometteuse mais est inutilisable sur de véritables problèmes actuellement.

7.2 Architecture modulaire avec apprentissage global

Cette partie est dédiée aux architectures modulaires utilisant un réapprentissage global. Ces architectures consistent à réutiliser de façon globale les poids de sous-réseaux ayant été entraînés séparemment. Le réapprentissage consiste à adapter les poids des sous-réseaux en fonction des données qu'ils n'ont jamais apprises. En résumé, chacun des sous-réseaux est adapté afin de reconnaître les formes qu'il a apprises non plus par rapport à un sous-ensemble mais par rapport à l'ensemble des formes à classifier. Nous nous sommes inspirés d'une architecture modulaire développée par A. Waibel que nous appellerons G-TDNN (G pour l'aspect apprentissage global) où des réseaux ayant appris à classifier des sous-ensembles de l'ensemble des classes participent à un réapprentissage neuronal qui permet de combiner leurs connaissances. Nous avons utilisé une structure modulaire très semblable avec des sous-réseaux détectant des indices phonétiques plutôt que des phonèmes et correspondant à différentes partitions de l'espace des données. Cette architecture est appelée FE-TDNN pour "*Features Extracted TDNN*".

7.2.1 Architecture G-TDNN

Les réseaux dédiés à des sous-tâches de classification ont été entraînés séparément. On utilisera le terme de sous-réseaux pour les définir. Les matrices de connexions de ces sous-réseaux contiennent des poids permettant une séparation spatiale de phonèmes particuliers. Pour revenir à un critère global, Il suffit d'apprendre à chacun des sous-réseaux à rejeter tous les phonèmes qui ne sont pas dans sa classe. Ce réapprentissage est réalisé dans une structure neuronale globale. On propage l'activité en parallèle dans tous les sous-réseaux, les poids des connexions sont modifiés en tenant compte de toutes les classes. La structure globale est composée

des sous-réseaux et d'un réseau de macro-classes qui facilite le réapprentissage. Un réseau apprenant à classifier des macro-classes a simplement des sorties associées aux étiquettes des macro-classes. Ce réseau ajoute un critère de confiance dans la détection du phonème. Le score de la macro-classe sert à pondérer les scores de tous les phonèmes de cette classe.

Réapprentissage

Chacun des sous-réseaux a appris à reconnaître séparément un sous-ensemble de formes. Le but du réapprentissage est d'adapter les paramètres de tous les sous-réseaux pour reconnaître les formes qu'il a apprises non plus par rapport à un sous-ensemble mais par rapport à l'ensemble des formes à classifier. Les activités sont propagées dans chacun des sous-réseaux en parallèle dans l'architecture globale, enfin la dernière couche sert à unifier les sorties des sous-réseaux. Pendant la phase de réapprentissage, certaines matrices de connexions sont figées et d'autres participent à nouveau à un apprentissage, cf fig. 7.1. Le réseau de macro-classes permet de "coller" les connaissances des sous-réseaux. Ce réseau est quelquefois un réseau vide appelé "*glue connectionnist*" qui apprend durant ce réapprentissage à peu près les connaissances d'un réseau de macro-classes. Il est donc plus rapide d'initialiser la "*glue connectionnist*" directement avec un réseau de macro-classes. C'est le seul réseau de l'architecture à avoir connaissance de toute la base de données au début du réapprentissage.

On ne réinitialise pas les matrices de connexions de la couche cachée vers la dernière couche mais elles sont pondérées afin de désaturer les poids des connexions et de permettre un nouvel apprentissage. La somme pondérée des activités est ramenée le plus près possible de la partie linéaire de la sigmoïde. Le paramètre EPS, le pas de modification des poids, doit être choisi avec prudence pour ce réapprentissage. Lorsque l'algorithme converge à nouveau, il peut être utile quoique délicat de libérer les poids des premières matrices (poids indiqués comme fixes dans le schéma de la figure 7.1) pour optimiser l'ensemble de la structure.

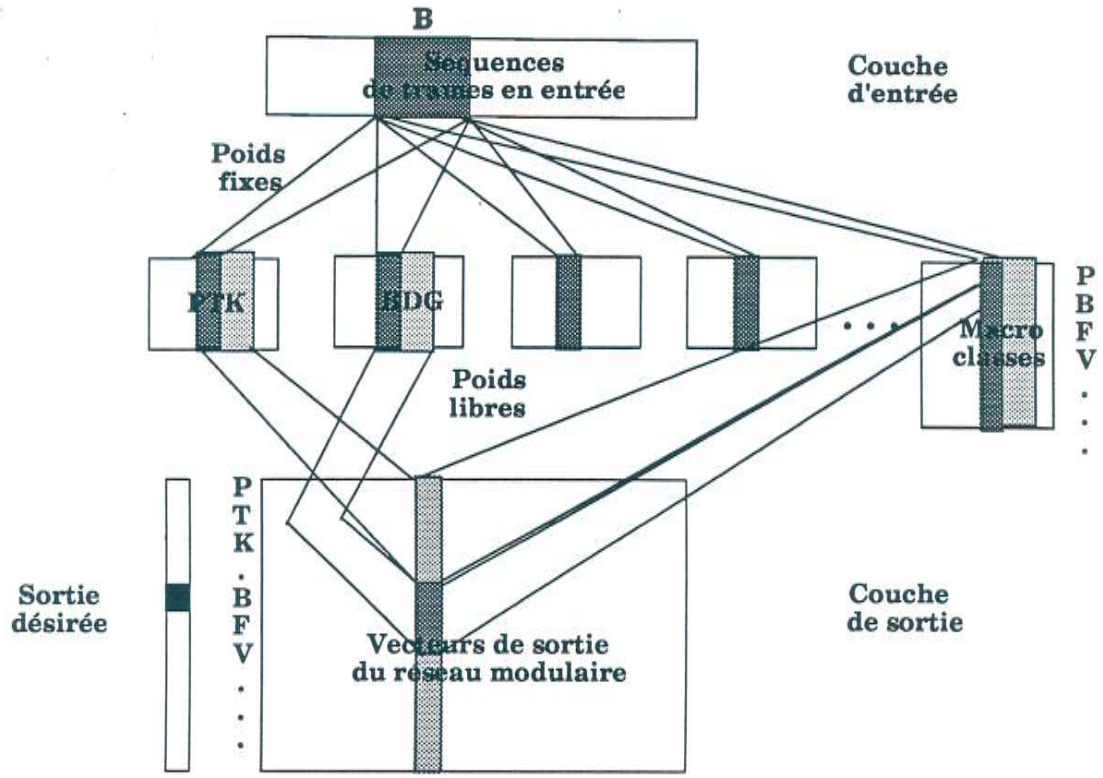

*Fig 7.2 : **Architecture modulaire G-TDNN.***

Le principal défaut de ce réapprentissage neuronal est de perdre une certaine partie de l'information contenue dans les sous-réseaux en modifiant après coup les poids des connexions. De plus, cette structure est limitée à un petit nombre de sous-réseaux si on veut réaliser un réapprentissage en un temps raisonnable.

Tests avec une architecture modulaire G-TDNN

Nous présentons des résultats de classification phonétique avec des architectures modulaires de sous-réseaux ayant appris des classes de phonèmes acoustiquement proches. Les partitions des phonèmes utilisées sont issues de l'expertise phonétique et sont validées par les résultats de classification obtenus avec ces macro-classes. Un exemple de matrice de confusion des vecteurs de sortie d'un réseau de macro-classes est donné pour l'américain dans le tableau 7.2.

Partition phonétique

Les architectures modulaires de TDNNs utilisées pour l'américain et le français contiennent les sous-réseaux correspondants aux classes décrites dans le tableau suivant (Tab 7.1).

Partition en américain	Partition en français
p t k pd td kd si	p t k #
b d g dd dx	b d g
m n ng en	m n %
l r y w hh	l r
f s th sh	f s x
v z jh dh ch	v z j
aa ao ah ae eh	a ( )
iy ix ey ih ay oy	* < /
ax uh er uw aw ow	e = o +
	u w i

Tab. 7.1 : ***Partition phonétique en français et en américain.*** *Les symboles utilisés sont décrits dans les annexes 1 et 2.*

Le tableau suivant (Tab. 7.2) représente la matrice de confusion d'un réseau de macro-classes en américain obtenue sur les 100 phrases de test de DARPA. Les scores en gras qui ne sont pas en italique, sont les confusions les plus graves avec les autres macro-classes de phonèmes. Certains phonèmes sont souvent mal reconnus comme ax, ix, td, l, r... Ils sont en effet très variables et de courte durée ce qui les rend difficile à reconnaître. Cette matrice de confusion (tab 7.2) a un taux global de confusion d'environ 30%. Ce taux est malgré tout assez élevé. Ce résultat amène à la reflexion suivante : ne faut-il pas faire coopérer plusieurs réseaux de macro-classes utilisant des partitions différentes des phonèmes pour réduire le taux global de confusion?

	Matrice de confusion des vecteurs de sortie du réseau de macro-classes (en américain)								
	p	b	f	v	m	l	aa	iy	ow
p	437	21	20	20	.	19	7	1	3
t	374	24	124	40	4	18	4	2	2
k	618	15	56	13	14	15	.	4	9
pd	38	1	.	.	.	.	1	.	8
td	174	82	46	43	10	12	33	36	25
kd	146	.	.	.	.	1	1	.	.
si	2001	31	98	13	30	121	6	2	10
b	47	228	12	16	14	10	6	3	5
d	14	271	16	9	14	7	6	11	13
g	8	171	5	4	13	13	.	8	6
dd	16	82	14	2	37	15	2	13	8
dx	17	159	6	14	.	10	21	35	5
f	13	1	559	23	2	3	8	.	6
s	32	2	1556	80	9	.	7	15	1
th	11	2	94	28	.	2	2	1	2
sh	3	1	494	42	1	6	.	6	12
v	28	26	51	171	13	10	51	18	34
z	10	5	361	487	8	1	2	10	8
jh	1	1	20	117	.	.	.	4	.
ch	.	.	26	102	.	.	.	.	.
dh	.	19	54	254	31	20	7	17	30
m	4	22	1	8	467	17	4	8	8
n	20	92	29	16	874	51	41	117	1
ng	4	3	3	.	167	1	2	4	2
en	.	.	.	.	22	.	.	.	.
l	20	17	21	81	24	474	125	53	239
r	20	22	20	25	15	681	164	60	396
w	33	3	4	8	16	452	35	7	14
y	1	3	3	3	2	138	.	14	3
hh	19	.	4	7	11	179	14	3	19
aa	8	17	3	6	14	34	480	43	72
ae	21	.	15	17	2	9	555	81	48
ah	12	2	.	3	9	8	213	35	9
eh	15	20	23	44	60	35	593	280	33
ao	.	4	1	5	2	27	344	7	36
ix	29	20	52	76	105	24	73	192	59
ih	23	33	15	42	21	11	63	403	78
iy	65	58	32	29	47	129	20	972	66
ay	.	2	1	6	9	5	18	322	8
oy	.	1	.	.	.	.	.	13	.
ey	24	3	3	3	29	23	111	559	6
uh	.	1	.	1	.	6	.	1	24
er	35	7	16	16	16	121	38	9	515
ax	95	33	72	57	109	79	86	106	188
ow	11	2	7	10	14	26	94	19	313
uw	17	7	13	20	25	52	18	34	359
aw	.	.	.	6	9	2	36	12	114

Tab. 7.2 : ***Exemple d'une matrice de confusion en américain sur 100 phrases de test de DARPA.*** *Le réseau utilisé a une couche cachée de 50*

cellules, les transitions ont été apprises (cf § 6.1.2.2). Les symboles sont décrits dans l'annexe 2. Les sorties sont les 9 macro-classes.

Comparaison d'une architecture modulaire avec un réseau global sur la base de données française

Base en français de 256 mots contexte CVCV	*Apprentissage Trames*	*Test Trames*
moyenne des sous-réseaux (cf Tab 6.2)	95.9	91.9 ±2.6
macro-classes	91.5	89 ±0.9
réseau global	91.2	85 ±1
G-TDNN (10) avant réapprentissage	70.2	50.9 ± 1.5
G-TDNN (10)	90.8	86 ±1

Tab. 7.3 : ***Comparaison entre une architecture globale et modulaire pour la base de données en français.*** *Les réseaux ont été entraînés sur des entrées de dimension 17(16 coefficients FFT + Energie) sans prendre en compte les transitions.*

Le tableau (Tab. 7.3) compare les résultats obtenus avec une architecture modulaire et globale sur la base de données française. En moyenne, les sous-réseaux obtiennent de très bons scores sur leurs sous-tâches. Il en va de même pour le réseau de macro-classes. L'architecture G-TDNN sans réapprentissage global obtient toutefois des taux de classification assez bas. Cette mauvaise performance est facilement explicable. Chacun des sous-réseaux a appris une petite partie de l'espace d'entrée. En conséquence, pour des entrées inconnues, il génère une sortie indéfinie qui peut être la même que celle des autres sous-réseaux et gêner la reconnaissance. La phase de réapprentissage permet de gagner plus de 30% sur les performances obtenues avec le corpus de test en donnant à chacun des sous-réseaux une connaissance globale des autres classes.

Les scores du réseau global et du réseau modulaire avec réapprentissage sont sensiblement les mêmes. Le réseau global atteint une performance de classification

de 85% sur les trames et de 90.2% sur les segments au lieu de 86% et 90.8% avec le réseau modulaire G-TDNN (10). Pour une base de données de cette taille, il est plus intéressant de construire un réseau global permettant de classifier en même temps tous les phonèmes. Toutefois, l'architecture modulaire permet une économie de temps et une souplesse d'utilisation que n'a pas l'architecture globale. L'aspect temporel devient un facteur très important en faveur d'une architecture modulaire lorsque la tâche de classification est plus importante, par exemple pour la base DARPA. Ce facteur pour la base de données en français n'est pas significatif.

Résultats sur la base DARPA RM1

DARPA 1000 mots tous contextes phonétiques	***Apprentissage Trames***	***Test Trames***
moyenne des sous-réseaux (cf Tab 6.3)	87.8	78.3 ±2.8
macro-classes	76	65.4 ±1
G-TDNN (9)	62	50.4 ±1.1
global	57.3	45.8 ±1.1

Tab. 7.4 : ***Comparaison entre une architecture globale et modulaire pour la base de données en américain.*** *Les réseaux ont été entraînés sur des entrées de dimension 17(16 coefficients FFT + Energie) sans prendre en compte les transitions.*

Les performances du réseau global sont nettement moins bonnes sur cette base de données (cf Tab. 7.4). L'architecture G-TDNN permet d'obtenir de meilleurs résultats qu'un réseau global. La tâche de classification est trop difficile pour être apprise par un réseau global avec une architecture de 50 cellules dans la couche cachée et prendrait beaucoup trop de temps avec plus de cellules. L'architecture modulaire est nécessaire pour cette tâche avec une puissance de calcul monoprocesseur. Cependant, les scores de classification obtenus ne sont pas très élevés. Nous proposons d'autres architectures modulaires n'utilisant pas de réapprentissage global (H-TDNN, HT-TDNN) dans le paragraphe suivant qui permettent d'améliorer les taux de classification.

Structures globales et modulaires

TEST *% trames*	*Nb cells Global*	*G-TDNN*	*Global*
Base en français 31 phonèmes CVCV	30	86 ±1	85 ±1
DARPA plosives + fricatives 21 phonèmes	40	73.7 ±1.4	73 ±1.4
DARPA 47 phonèmes	50	50.4 ±1.1	45.8 ±1.1

Tab. 7.5 : ***Comparaison entre structures globales et modulaires pour des tâches de différentes difficultés.*** *Les réseaux ont été entraînés sur des entrées de dimension 17 (16 coefficients FFT + Energie) sans prendre en compte les transitions.*

Pour une tâche de classification de peu de phonèmes ou pour une base de données de taille réduite, une architecture globale est plus simple à utiliser et offre de bonnes performances. L'architecture modulaire apporte de meilleures performances lorsque la tâche devient plus complexe et nécessiterait un réseau unique de très grande taille.

7.1.2 Architecture de type FE-TDNN

L'idée est d'utiliser plusieurs partitions phonétiques afin d'éliminer l'a priori d'une partition particulière de phonèmes. La partition utilisée pour entraîner le réseau de macro-classes et les sous-réseaux utilisés dans les architectures modulaires décrites jusqu'à présent pour DARPA correspond pour les consonnes à un regroupement des classes par mode d'articulation et voisement (plosive sourde, plosive sonore, fricative sourde, fricative sonore, nasale, liquide et semi-voyelle) et pour les voyelles à un regroupement par rapport aux positions des deux premiers formants F1 et F2 dans le triangle vocalique (voyelle /aa/, /iy/ et /uw/). Nous décrivons une expérience menée avec plusieurs partitions pour la classification d'un sous-ensemble des phonèmes de DARPA : les plosives et les fricatives, sourdes et sonores. Une architecture utilisant plusieurs partitions phonétiques sera comparée à des architectures utilisant une seule partition.

Extraction d'indices

La plupart des réseaux utilisent en sortie une étiquette phonétique par classe. Nous avons exploré des réseaux dont les sorties sont des étiquettes de macro-classes correspondant à des caractéristiques phonétiques. Chaque sous-réseau détecte différents indices comme le voisement, le point d'articulation (labial, dental...), le mode d'articulation (plosive,fricative...)...

Prenons l'exemple d'une architecture avec deux sorties, l'une apprend à reconnaître la macro-classe des plosives voisées (/b/, /d/, /g/) et l'autre, les plosives non voisées (/p/, /t/, /k/), on espère coder par le réseau si les formes sont bien segmentées des incides permettant de discriminer entre ces deux formes comme par exemple la présence ou non de la barre de voisement.

Nous présentons des expériences avec différentes partitions sur les plosives et les fricatives issues du corpus DARPA qui représentent 21 phonèmes, soit un ensemble d'apprentissage de 9775 segments dont 1650 segments servent pour la validation croisée et de 1487 segments pour le corpus de test. La décomposition que nous avons utilisée est la suivante : la place de l'articulation, le mode d'articulation et le voisement. Les différentes partitions de macro-classes sont décrites dans les tableaux suivants (Tab 7.6, Tab 7.7 et Tab 7.8) :

Partition: ***Mode d'articulation (MAC4)***	
p t k pd td kd si	P
b d g dd dx	B
f s th sh	F
v z jh dh ch	V

Tab. 7.6 : ***Partition de l'ensemble des plosives et fricatives en fonction du mode d'articulation.***

Partition: ***Place de l'articulation (MAC5)***	
p pd b f v	labial
th dh	dental
t d td dd z s	alvéo-dental
k g kd	vélaire
sh jh ch	alvéo-palatal

Tab. 7.7 : ***Partition de l'ensemble des plosives et fricatives en fonction de la place de l'articulation.***

Partition: *Phonème Voisé / Non voisé (MAC2)*	
p t k dx sil pd td kd f s th sh	NVoise
b d g dd v z jh ch dh	Voisé

Tab. 7.8 : ***Partition de l'ensemble des plosives et fricatives en fonction du voisement.***

Les taux de classification pour chacun de ces réseaux de macro-classes sont donnés dans le tableau suivant (Tab 7.9).

% reconnaissance	*Nb cell. cachées*	*Apprentissage Trames*	*Test Trames*
MAC4	30	85.1	82.9 ±1.2
MAC2	15	88.5	87.1 ±1
MAC5	40	84.6	82.4 ±1.2

Tab. 7.9 : ***Taux de reconnaissance pour les différentes macro-classes.*** *L'apprentissage a été fait en tenant compte des transitions. Les données en entrée sont des vecteurs de dimension 17(16FFT + Energie).*

Les sous-réseaux de macro-classes que nous avons définis apprennent des connaissances croisées. En effet, si on combine ces réseaux classifiant des indices phonétiques, on espère classifier les 21 phonèmes. Le tableau 7.6 montre les différents indices utilisés pour cette expérience.

Indices	*p*	*t*	*k*	*pd*	*td*	*kd*	*si*	*b*	*d*	*g*	*dd*	*dx*	*f*	*s*	*sh*	*th*	*v*	*z*	*jh*	*dh*	*ch*
Non Voisé	*	*	*	*	*	*	*						*	*	*	*					
Voisé								*	*	*	*	*					*	*	*	*	*
Labial	*			*				*					*				*				
Dental																*				*	
Alvéo-Dental		*			*				*		*			*				*			
Vélaire			*			*				*											
Alvéo-Palatal															*				*		*
Plosive NV	*	*	*	*	*	*	*														
Plosive V								*	*	*	*	*									
Fricative NV													*	*	*	*					
Fricative V																		*	*	*	*

Tab. 7.10 : ***Macro-classes croisées***

Les indices utilisés ne suffisent pas pour discriminer les classes /p/ de /pd/, /t/ de /td/, /k/ de /kd/ et /d/ de /dd/ mais ils permettent de distinguer les 17 autres phonèmes. Les différenciations entre /p/ et /pd/, /t/ et /td/... sont très difficiles à apprendre. Des classes spéciales ont été créées pour les phonèmes /p/ , /t/ ... dont la représentation spectrographique est très différente du phonème. Les phonèmes (/pd/, /td/, /kd/, /dd/) représentent des /p/, /t/, /k/, /d/ qui peuvent être élidés en position terminale d'un mot.

Nous avons construit à partir de ces sous-réseaux une architecture globale. La figure 7.3 représente cette structure notée FE-TDNN. La phase de réapprentissage dans cette structure consiste à associer des pondérations aux indices extraits par les sous-réseaux de façon à reconnaître les phonèmes. Ces informations sont codées par la dernière couche des sous-réseaux et par un réseau vide dont tous les poids sont libres.

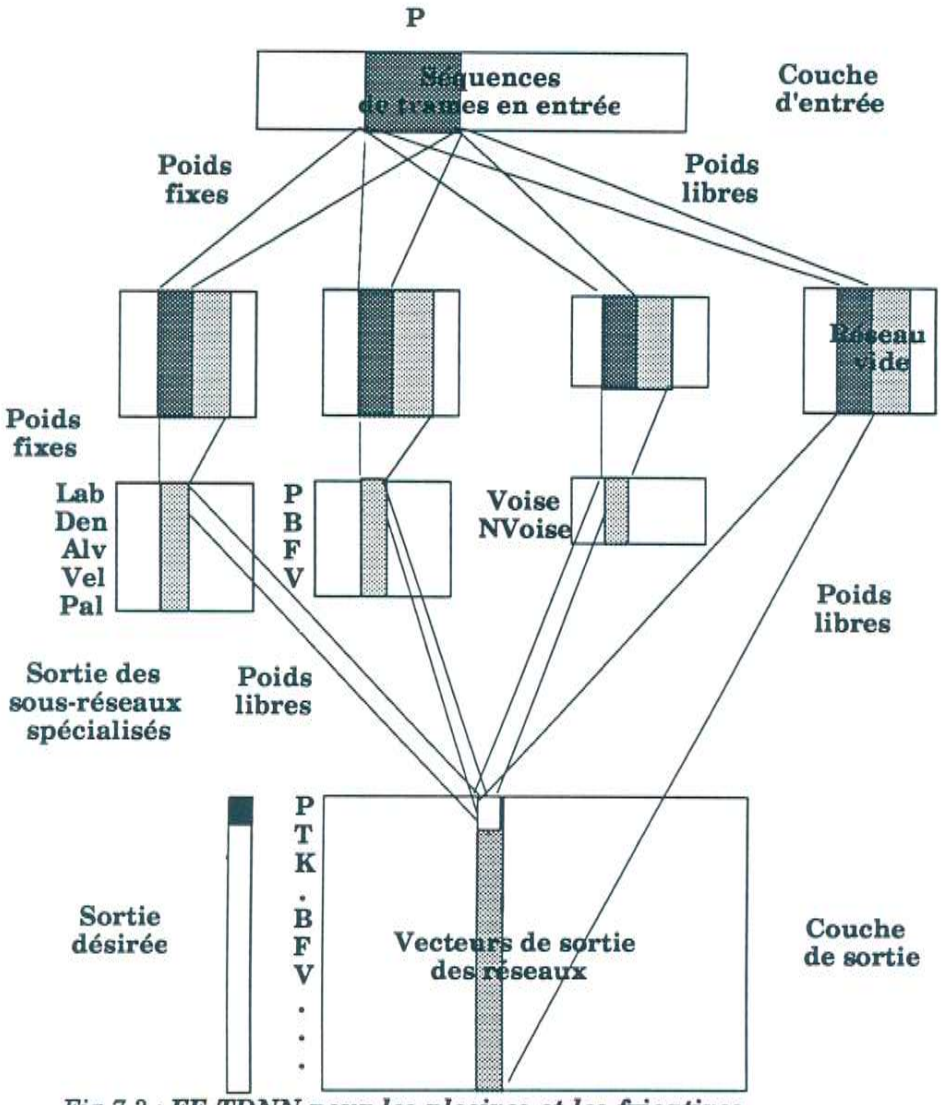

*Fig 7.3 : **FE-TDNN pour les plosives et les fricatives.***

Nous avons confronté plusieurs architectures modulaires : deux architectures utilisant une partition différente de l'espace des données (MAC4 + 4NN), (MAC5 + 5NN) et une architecture FE-TDNN utilisant plusieurs partitions (MAC4, MAC2, MAC5).

% reconnaissance	*nb poids*	*Apprentissage Trames*	*Test Trames*
global	6301	76.4	73
MAC4 + 4 NN	7186	80.2	73.7
MAC5 + 5 NN	7987	78.1	73.3
FE-TDNN	6177	75.4	70.9

Tab. 7.11 : ***Taux de reconnaissance pour les différentes macro-classes.*** *L'apprentissage a été fait en tenant compte des transitions. La validité statistique sur le test est de* ± 1.4.

Dans cette expérience, les deux architectures MAC4 + 4NN et MAC5 + 5NN, obtiennent avec des partitions différentes des résultats identiques à ceux de l'architecture globale pour pratiquement le même nombre de poids de connexions. Les taux de reconnaissance sur le corpus apprentissage sont meilleurs qu'avec l'apprentissage global mais ne se répercutent pas sur le test.

Dans le cas de la structure FE-TDNN, le nombre de poids donnés dans le tableau correspond à la somme des poids des matrices de connexions issues des réseaux MAC2, MAC4 et MAC5. Le réapprentissage de cette structure globale est assez difficile en partie à cause du réseau vide qui a du mal à associer les indices des réseaux spécialisés. La structure FE-TDNN donne de bons résultats sur 17 phonèmes mais elle n'a pas de connaissances suffisantes pour différencier /p/ de /pd/, /t/ de /td/... et globalement le taux de reconnaissance est moins bon qu'avec les autres architectures. Le problème réside dans le manque de connaissances codées par les sous-réseaux et soulève le problème de l'expertise nécessaire à la décomposition de la tâche et du choix des indices à extraire pour une tâche complexe. De plus, cette approche nécessite une segmentation très fiable de la base de données.

Ce type d'architecture n'a pas été plus approfondi à cause du problème d'expertise qui rend la structure trop ad-hoc. Cependant, l'idée d'utiliser des réseaux extracteurs d'indices phonétiques n'est pas forcément à rejeter. On pourrait par

exemple utiliser ce type de réseaux pour faire une sorte de "Phonetic-Features Spotting", c'est-à-dire ajouter de manière locale à une classification phonétique des indices phonétiques comme par exemple le voisement ou la nasalité... Une autre solution serait d'ajouter les sorties des réseaux de macro-classes directement aux vecteurs d'entrée d'un réseau multicouche. Le vecteur initial qui a ici une dimension de 17 coefficients deviendrait un vecteur de 17 + 4 + 5 + 2 + 5, c'est-à-dire de 33 coefficients. Des prétraitements différents peuvent aussi être envisagés pour faciliter l'apprentissage de certains indices dans les sous-réseaux.

Des recherches ont été menées avec ce type de réseaux extrayant des caractéristiques phonétiques par [Bengio 91], [Bimbot 91]. Pour faire collaborer les connaissances de réseaux spécialisés afin de classifier 8 formes (des plosives issues de la base Timit), Bengio propose un réseau linéaire qui calcule les 8 composantes principales des vecteurs de sorties concaténés des réseaux spécialisés. Les deux réseaux spécialisés utilisent des prétraitements différents et correspondent l'un à l'extraction de 16 caractéristiques phonétiques (*"silence, nasal, fricative, liquid, back, front, flap+front, flap+back, voiced, unvoiced, velar+front, velar+back, alveolar+front, alveolar+back, labial+back, labial+front"*), l'autre de 5 indices sur le mode d'articulation (*"sonorant, nasale, fricative, plosive, silence"*). Ce type d'architecture nécessite un étiquetage et une segmentation précise des données. Les résultats obtenus sont très intéressants sur des petites tâches comme la classification des plosives [Bengio 91].

Conclusion

En conclusion, l'ajout d'information au niveau de la structure est surtout important pour rendre abordable un problème de dimensions trop grandes. L'utilisation d'une partition a priori n'est pas si gênante, en effet les ambiguïtés non traitées directement par des sous-réseaux le sont indirectement au cours du réapprentissage global. Les inconvénients de ce réapprentissage global sont de faire baisser les performances des sous-réseaux pris séparément et de nécessiter un temps de réapprentissage relativement long puisqu'on utilise à nouveau toutes les formes à classifier en même temps. Ce type d'architecture pour les mêmes raisons de temps d'apprentissage est limité à un nombre raisonnable (une dizaine) de sous-réseaux .

Pour l'architecture FE-TDNN décrite, utilisant un réapprentissage global, nous nous heurtons à la même limite: la définition du nombre de sous-réseaux de la structure modulaire. De plus, cette approche soulève le problème de l'expertise

nécessaire à la décomposition et du choix des indices à extraire pour une tâche complexe.

7.3 Architectures hiérarchiques

Pour éviter de perdre les informations locales codées par les sous-réseaux lors du réapprentissage global, l'idée majeure est d'intégrer directement au cours de l'apprentissage de ces sous-réseaux une connaissance globale sous la forme d'une sortie de rejet des autres phonèmes (appelée trappe). La structure hiérarchique n'est en fait qu'une image permettant de comprendre l'intégration des sorties des sous-réseaux sachant que chaque sous-réseau est entraîné séparément. Cette architecture permet de pallier la limitation du nombre de sous-réseaux utilisés et évite un réapprentissage global long et insatisfaisant.

Nous ne donnerons pas d'évaluation des taux de classification phonétique avec les architectures hiérarchiques dans ce chapitre. Elles sont utilisées dans des systèmes hybrides TDNN-HMM que nous décrirons dans le chapitre 8 et les performances obtenues pour différentes strutures G-TDNN et hiérarchiques H-TDNN "*Hierachical TDNN*" et HT-TDNN "*Hierachical TDNN with Traps*" seront alors comparées.

Les architectures modulaires H-TDNN et HT-TDNN sont composées de sous-réseaux dédiés à des tâches particulières. Pour ce type d'architecture, il n'existe pas d'apprentissage global. Chaque sous-réseau a été entraîné de façon indépendante. Le problème est de faire coopérer les connaissances des sous-réseaux dédiés à des sous-tâches de façon à obtenir une connaissance globale pour toutes les classes. Les architectures hiérarchiques associent de façon empirique les sorties de tous les sous-réseaux au moment de la reconnaissance.

Les scores de sortie de chaque réseau pour une observation x sont utilisés afin d'obtenir un score global. Le score global est ensuite tranformé en probabilité a posteriori d'obtenir le phonème P parmi l'ensemble des phonèmes pour l'observation x. Le but de ce paragraphe est d'expliquer les règles d'intégration des scores de sortie des sous-réseaux pour les 2 architectures H-TDNN et HT-TDNN.

7.3.1 Architecture de type H-TDNN

L'architecture modulaire H-TDNN "*Hierarchical TDNN*" utilise les sous-réseaux et le réseau de macro-classes de la structure modulaire G-TDNN (cf §7.2.1) sans réapprentissage, c'est-à-dire sans ajout de connaissances globales.

Le réseau de macro-classes est le seul à avoir une connaissance globale des formes. On cherche à estimer pour toutes les classes p, P(p/x), la probabilité a posteriori de la classe p sachant x.

Pour cette architecture, le réseau de macro-classes sélectionne le sous-réseau à prendre en compte. On fait donc l'hypothèse que le réseau de macro-classes ne fait pas d'erreur. Les performances d'une telle architecture sont alors totalement dépendantes des performances du réseau de macro-classes.

On notera S(p/x) le score du phonème p pour l'observation x qui résulte de l'intégration des scores des différents sous-réseaux, K_j la macro-classe contenant p, $S(K_j/x)$ le score de K_j dans le réseau de macro-classes et $S(p/K_j,x)$ le score de p dans le réseau classifiant les phonèmes de la macro-classe K_j,

$$S(p/x) = S(p/K_j,x)S(K_j/x)$$

Soit P(p/x) l'approximation de la probabilité réalisée par rapport à cette règle d'intégration :

$$P(p/x) = \frac{S(p/x)}{\sum_{p} S(p/x)}$$

L'architecture (fig. 7.4) du réseau simule l'intégration des sorties. Cette architecture est totalement dépéndante des performances du réseau de macro-classes.

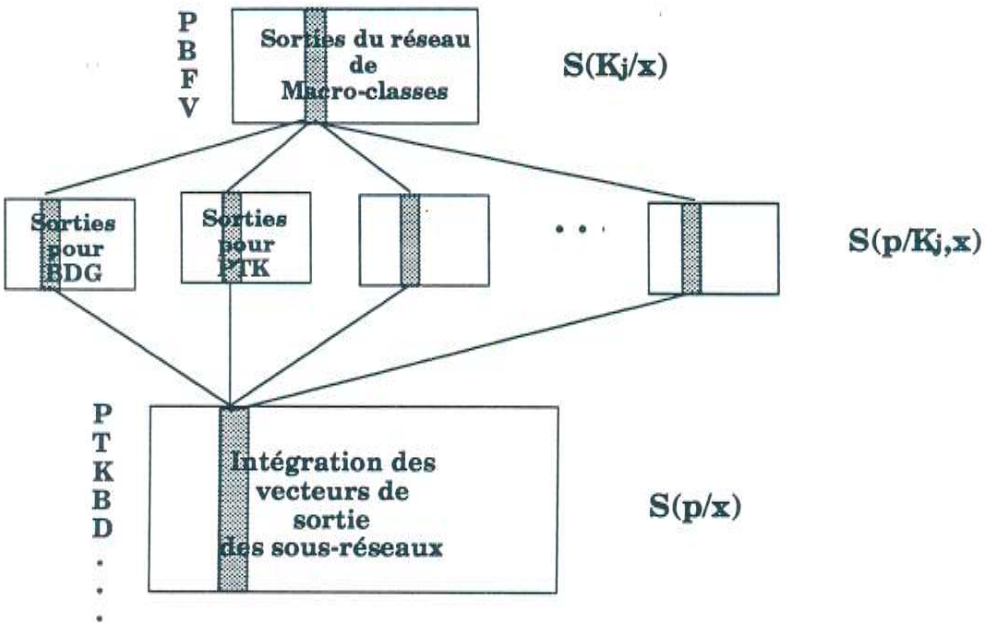

*Fig 7.4 : **Architecture hiérarchique des sorties des réseaux.***

7.3.2 Réseaux avec trappes et architecture HT-TDNN

Dans le but d'ajouter des connaissances globales directement dans la structure de chaque sous-réseau, une sortie appelée trappe est jointe aux sorties désirées. Cette sortie étiquettera les formes qui ne correspondent pas aux autres sorties du réseau. Ces formes pourront être introduites durant l'apprentissage ce qui fournira au réseau une information globale sur la tâche à réaliser.

Par exemple, si le réseau est dédié à la tâche B D G, la trappe correspondra à tous les phonèmes non B D et G dans l'ensemble des données, c'est-à-dire à tous les autres phonèmes P, T, K, M, N, A, I, ...

En pratique, il s'est avéré délicat d'apprendre toutes les autres formes dans une seule trappe. Il est en effet très simple de satisfaire cette sortie trappe par rapport aux autres sorties qui cherchent à extraire des indices précis permettant de distinguer par exemple un B d'un D.

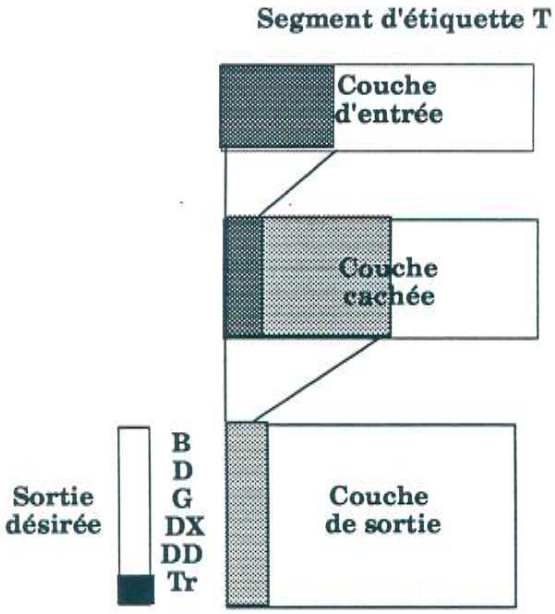

Fig 7.5 : ***Réseau BDG avec une trappe rejetant les autres phonèmes.***

Supposons que l'on ait non plus 1 trappe mais k-1 trappes permettant de rejeter les k - 1 autres macro-classes de l'ensemble de données. Les sorties d'un réseau à k-1 trappes correspondent alors aux sorties des classes qui sont éléments d'une macro-classe K, plus 1 sortie pour chaque autre macro-classe. Cette solution apporte une information globale à chaque sous-réseau. La règle donnée ci-dessus pour l'architecture H-TDNN peut encore être appliquée si on néglige les trappes. On peut cependant lui ajouter d'autres connaissances issues des trappes permettant d'être moins dépendant des performances du réseau de macro-classes.

Soit S(p/x) le score de p pour l'observation x qui résulte de l'intégration des scores des différents sous-réseaux, K_j la macro-classe contenant p, $S(K_j/x)$ le score de K_j dans le réseau de macro-classes et $S(p/K_j,x)$ le score de p dans le réseau classifiant les phonèmes de la macro-classe K_j et les autres macro-classes K_i pour tout i différent de j sous forme de trappes, la règle préalablement donnée est similaire si on néglige l'information donnée par les trappes :

$$S(p/x) = S(p/K_j,x)S(K_j/x)$$

Si on tient compte de l'information donnée par les trappes, on apporte des corrections au premier système en ajoutant une information négative si les sous-systèmes spécialisés ne reconnaissent pas la classe K_j donnée par la classification

du réseau de macro-classes. Soit K_1, K_2, ...,K_k les différentes macro-classes, soit $S(K_j/K_i,x)$ le score de la macro-classe K_j donné par une trappe du sous-réseau entraîné plus spécifiquement sur les classes de K_i, on peut utiliser une règle telle que :

$$S(p/x) = S(p/K_j,x)S(K_j/x) + \sum_{i \neq j} S(K_i/x)S(K_j/K_i,x)$$

On ajoute à la règle initiale une connaissance issue des autres sous-réseaux. La figure 7.6 ci-après schématise l'arbre des sous-réseaux et montre clairement les connaissances ajoutées par les autres réseaux sur un exemple avec deux macro-classes K_1 et K_2, K1 = (P,K), K2 = (B,D). Pour calculer le score d'un phonème P, élément de K_1, dans l'arbre des sorties des sous-réseaux, on tient compte de la trappe K_1 dans l'autre sous-réseau.

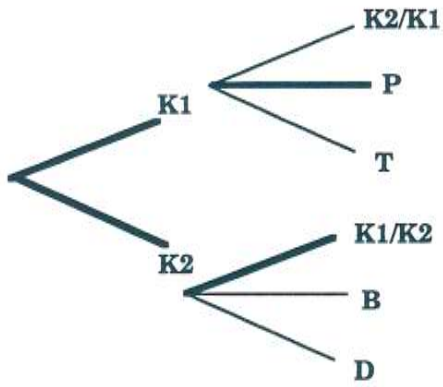

Fig. 7.6 : ***L'intégration des scores dans l'arbre des sorties des réseaux est repérée en gras sur le schéma.***

En pratique, si le score K_j est très elévé, le deuxième terme de la règle énoncée ci-dessus, qui est pondéré par le score des autres macro-classes, n'apporte pas d'informations supplémentaires. Par contre, si le score K_j n'émerge pas clairement alors le deuxième terme de l'équation sert à valider ou invalider le choix de la macro-classe. Les informations codées par les trappes des autres sous-réseaux interviennent donc pour corriger les erreurs du réseau initial de macro-classes..Tous les sous-réseaux, différents de celui qui classifie les phonèmes de K_i, ont codé dans une trappe un score correspondant à la macro-classe K_i. On tient compte de ces informations en pondérant le score de la macro-classe K_i par le score de K_j. Soit P(p/x) l'approximation de la probabilité réalisée par rapport à cette règle d'intégration :

$$P(p/x) = \frac{S(p/x)}{\sum_{p} S(p/x)}$$

Nous avons expérimenté une solution très proche de celle-là qui consiste à apprendre k-1 sous-réseaux possédant chacun 1 trappe différente, k étant le nombre de macro-classes définies. La raison en est simple, les sous-réseaux comportant les k-1 trappes en même temps nécessitent trop de temps d'apprentissage. Si le nombre de macro-classes est de 9, le nombre de sous-réseaux devient 9*8 soit 72 sous-réseaux. Nous avons appelé cette architecture HT-TDNN, elle est schématisée dans la figure 7.7.

L'architecture générale de ce réseau modulaire est un arbre neuronal hierarchique, du sous-réseau le plus général au plus spécifique. Le noeud principal correspond à un réseau de k macro-classes. Chaque sous-noeud reçoit k-1 sous-réseaux, n étant le nombre de macro-classes, il intègre donc la connaissance de toutes les autres classes.

Soit S(p/x) le score de p pour l'observation x qui résulte de l'intégration des scores des différents sous-réseaux, K_j la macro-classe contenant p, $S(K_j/x)$ le score de K_j dans le réseau de macro-classes et $S(p/K_{jk},x)$ le score de p dans le réseau k classifiant les phonèmes de K_j et une trappe pour la macro-classe K_k, la règle préalablement donnée devient si on néglige l'information donnée par les trappes (cf figure 7.8) :

$$S(p/x) = S(K_j/x) \sum_{k \neq j} \lambda_k \, S(p/K_{jk},x)$$

$$\text{avec} \sum_{k \neq j} \lambda_k = 1$$

Soit $S(K_j/K_i,x)$ le score de la trappe K_j dans le réseau classifiant les phonèmes de la classe K_i et la macro-classe K_j,

soit $S(p/K_j,x) \approx \sum_{k \neq j} \lambda_k S(p/K_{jk},x)$ le score du phonème p dans la classe K_j avec

$\sum_{k \neq j} \lambda_k = 1$.

La règle est alors la même que précédemment :

$$S(p/x) = S(p/K_j,x)S(K_j/x) + \sum_{i \neq j} S(K_i/x)S(K_j/K_i,x)$$

Soit P(p/x) l'approximation de la probabilité réalisée par rapport à cette règle d'intégration :

$$P(p/x) = \frac{S(p/x)}{\sum_{p} S(p/x)}$$

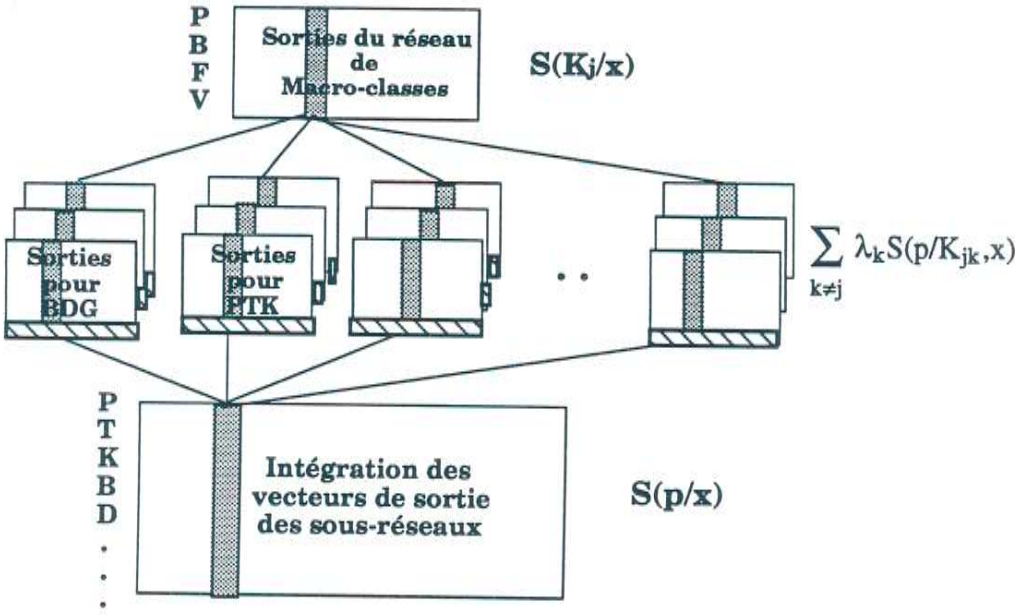

Fig 7.7 : ***Architecture hiérarchique des sorties des réseaux avec trappes.***

L'architecture HT-TDNN utilisant la règle d'intégration simple sans prendre en compte l'information codée dans les trappes dépend beaucoup de la performance du réseau de macro-classes. Si on utilise la règle intégrant les connaissances des trappes des autres sous-réseaux, le réseau de macro-classes n'est plus indispensable. Il est cependant toujours utile de garder le plus d'informations possibles pour prendre une décision.

La figure 7.8 présente l'architecture HT-TDNN utilisée dans les tests du chapitre 8. Le nombre de macro-classes pour les expériences menées estde 9, le nombre total de sous-réseaux est de 73 (9*8 sous-réseaux + 1 réseau de macro-classes).

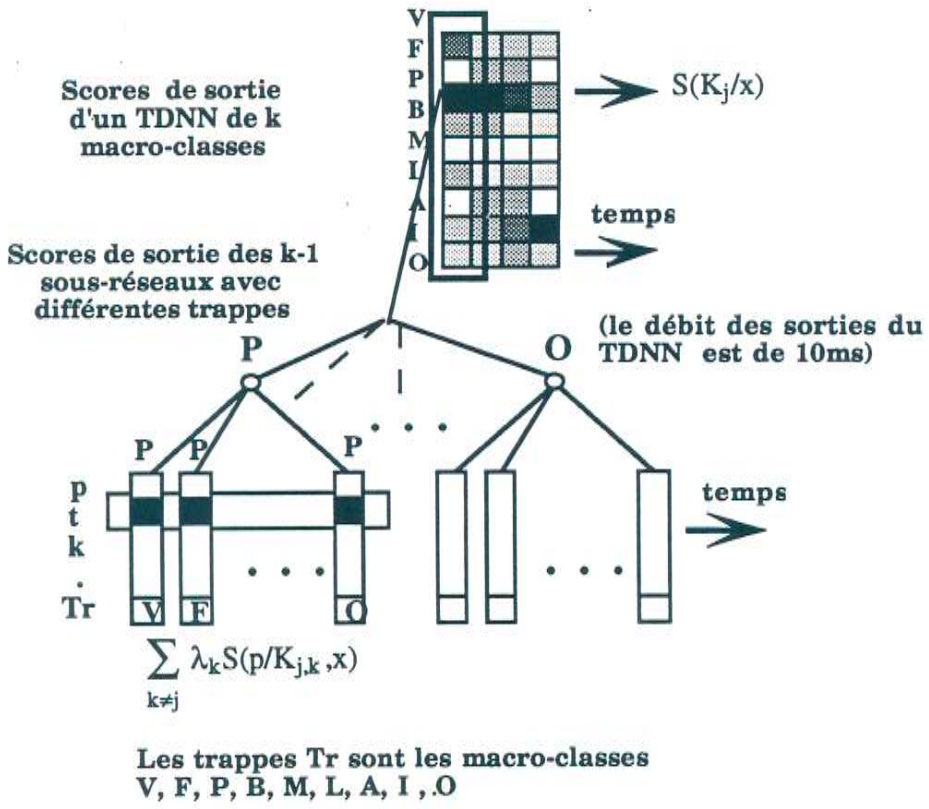

Fig 7.8 : ***Architecture hiérarchique des sorties des réseaux avec trappes.*** *Dans cette intégration, on n'utilise pas l'information contenue dans les trappes et les* λ_k *sont définis a priori.*

La construction modulaire hiérarchique permet une souplesse d'utilisation très appréciable. On peut imaginer d'autres architectures de type HT-TDNN. Des réseaux de phonèmes en contexte par exemple peuvent être intégrés très facilement en rajoutant un niveau dans l'arbre hiérarchique. De même des réseaux codant des indices phonétiques (cf 7.1.2) peuvent être ajoutés à la structure. Une hiérarchie avec des réseaux de macro-classes binaires au lieu d'un seul réseau de macro-classes, peut être aussi envisagée [Sawai, Waibel 89].

L'architecture hiérarchique avec trappes a permis d'obtenir de meilleures performances que les architectures G-TDNN et H-TDNN. Les résultats sont décrits dans le chapitre 8. La règle d'intégration utilisant les informations codées dans les trappes des sous-réseaux n'a pas encore été testée.

7.4 Conclusion

L'ajout de connaissances au niveau de la structure du réseau donne la possibilité de construire une architecture modulaire qui facilite l'apprentissage sur de grandes bases de données. La notion de modularité permet de concevoir des systèmes correspondant à la décomposition d'une tâche complexe en sous-tâches. En procédant de la sorte, la taille de l'espace de paramètres du système reste relativement faible et la recherche d'une solution lors de l'apprentissage est ainsi considérablement facilitée. Nous avons conçu plusieurs architectures modulaires pour réutiliser au mieux les informations codées dans les sous-réseaux.

Nous avons proposé une architecture originale : un arbre hiérarchique de sous-réseaux éliminant le problème du réapprentissage global et la limitation du nombre de sous-réseaux. Dans le but d'ajouter des connaissances globales directement dans la structure de chaque sous-réseau, une sortie appelée trappe est jointe aux sorties désirées. Cette sortie etiquette les formes qui ne correspondent pas aux autres sorties du réseau. Ces formes sont introduites durant l'apprentissage, ce qui fournit au réseau une information globale sur la tâche à réaliser. L'architecture hiérarchique utilise des règles d'intégration des scores des sous-réseaux lors de la reconnaissance. Elle permet une souplesse d'utilisation très intéressante et a obtenu de très bonnes performances lors de la reconnaissance dans un système hybride TDNN-HMM (cf chapitre 8).

Nous avons présenté quelques idées de découpage de la tâche en sous-tâches et d'intégration des sous-tâches. Les résultats obtenus sont favorables à ces méthodes modulaires. En effet, les tests menés ont montré qu'on ne perd pas de connaissance par rapport à un réseau global et de plus on rend gérable l'apprentissage des réseaux avec de grandes bases de données. Dans l'avenir, il semble cependant indispensable d'étudier plus avant des méthodes automatiques de découpage de la tâche et d'apprentissage coopératif dans les réseaux modulaires.

Partie 3

Reconnaissance de parole continue et Systèmes hybrides

Cette troisième partie regroupe les expériences de reconnaissance de parole continue menées avec des systèmes hybrides. Afin de pallier le problème d'intégration temporelle dans les réseaux, des systèmes hybrides NN-HMM ont été développés. L'intérêt est en fait double, il s'agit d'une part de rendre les modèles markoviens discriminants et d'autre part de gérer des problèmes de parole continue avec un réseau. Le but recherché est d'allier les avantages de différentes méthodes afin de concevoir un système plus performant que chacune des méthodes prises séparément. On distingue plusieurs types de coopérations entre modèles markoviens et réseaux, le ***chapitre 8*** *décrit nos expériences avec différentes coopérations et l'apport des systèmes hybrides par rapport aux HMMs utilisés seuls.*

8. Systèmes hybrides TDNN-HMM

La notion de système hybride fait référence à des systèmes combinant plusieurs techniques, si possible complémentaires. Le but recherché est d'allier les avantages de différentes méthodes afin de concevoir un système plus performant que chacune des méthodes prises séparément. Il existe de nombreuses "hybridations" : MLP-HMM [Bourlard 91], TDNN-HMM [Franzini,Waibel 91], TDNN-HMM [Haffner 92], RNN-HMM [Robinson,Fallside 91], TDNN-DTW [Bottou 91], TDNN-DTW-LVQ2 [Driancourt 92],... Nous avons étudié des systèmes hybrides neuronaux et markoviens TDNN-HMM afin d'utiliser les qualités de discrimination des réseaux et les mécanismes temporels ainsi que la gestion intégrée des modèles markoviens.

Ces systèmes peuvent être séparés en quatre classes en fonction de leur coopération. Nous les avons introduits dans le chapitre 4 "Etat de l'art des systèmes hybrides". Nous nous sommes intéressés plus particulièrement à deux de ces classes où :

- le modèle neuronal sert de "pré-traitement" aux modèles markoviens,

- les deux modèles coopèrent lors de la reconnaissance. Nous avons utilisé la même topologie de modèles et une architecture globale identique afin de comparer au mieux les différentes intégrations TDNN-HMM avec des modèles markoviens continus.

8.1 Architecture globale des HMMs et des systèmes hybrides

Les systèmes utilisant les modèles de Markov décrits dans cette étude ont été développés par le laboratoire de Philips à Aix-la-Chapelle. Ils ont servi de comparaison à tous les systèmes hybrides développés. L'algorithme d'apprentissage utilisé est l'algorithme de Viterbi avec le critère de MLE, cf chapitre 2.

On illustrera la description des méthodes utilisées par des exemples sur la base Darpa RM1, décrite en annexe 2.

8.1.1 Topologie des modèles

La topologie des modèles est la même pour toutes les approches HMMs et hybrides TDNN-HMMs. Les modèles sont indépendants du contexte. Chaque phonème est modélisé par un modèle de Markov caché de type Bakis cf Fig. 8.1.

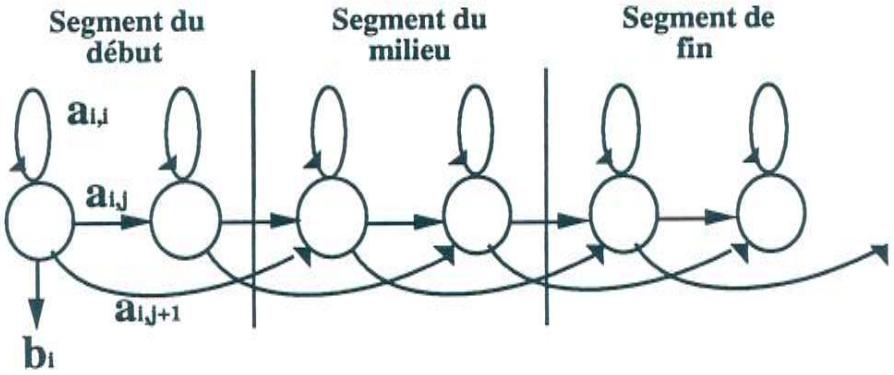

Fig. 8.1 : ***Modèle phonétique de Markov de type modèle de Bakis.*** *Un modèle de Bakis est un modèle de Markov à trois états comprenant les transitions suivantes : une boucle sur l'état courant, un lien vers l'état suivant et un saut d'un état. Dans notre cas, c'est un modèle à 3 segments qui représentent le début, milieu et fin du phonème, chaque segment est modélisé par deux états. La séparation en segment est indiquée par des barres verticales.*

Un phonème consiste en trois segments. Le premier et le dernier modélisent implicitement les effets de coarticulation entre les phonèmes, le segment central modélise la partie du phonème la moins sensible aux effets de voisinage. Une densité de probabilité ou une probabilité est associée à chaque segment. Pour mieux modéliser l'aspect temporel, un segment comprend un ou deux états. Les probabilités de transitions entre états $a_{j',j}$ (entre un état j' et j) sont fixées a priori et

sont les mêmes pour tous les modèles. Elles sont déterminées de façon empirique pour chaque approche HMM et TDNN-HMM. L'ordre du coût des transitions est en général le suivant :le lien avec l'état suivant coûte le moins cher, puis ensuite la boucle sur un état et enfin le saut d'un état.

8.1.2 Recherche intégrée pour la reconnaissance de parole continue

L'architecture globale du système est la même pour toutes les approches HMMs et hybrides. La reconnaissance est basée sur la règle de décision du maximum de vraisemblance (MLE) et intègre de façon hiérarchique différentes sources de connaissance : la liste des phonèmes, le dictionnaire phonétique des mots du vocabulaire et le modèle de langage. Pour la reconnaissance phonétique, ces sources de connaissance sont réduites à la liste des phonèmes. L'intégration des différents modules est gérée par l'algorithme de décision. La recherche intégrée est basée sur une recherche en faisceau de gauche à droite temporellement synchrone. Elle permet de déterminer une séquence de mots ou de phonèmes. On peut utiliser cette recherche intégrée avec ou sans modèle de langage. Etant donnée une entrée acoustique, la séquence de mots ou de phonèmes est déterminée par la séquence d'états la plus probable dans l'espace de recherche défini par le vocabulaire de mots ou de phonèmes. On utilise un algorithme d'alignement temporel de type Viterbi, le "*One Stage Dynamic Programming*" [Ney 84] pour optimiser en même temps la segmentation et la reconnaissance des mots d'une phrase. Soit les vecteurs observés x d'indice i=1,...,I (notés sur le schéma 8.2 x_1 x_2 x_3 x_4... pour symboliser une séquence de vecteurs) la figure 8.2 schématise la correspondance entre des entrées acoustiques et les mots du vocabulaire.

Pour déterminer la séquence de mots (ou phonèmes) inconnus de longueur inconnue w[1:N] qui est la plus probable sachant l'entrée X[1:I], il faut trouver la séquence de mots (ou phonèmes) qui maximise le produit:

$$\max_{w} P(w)\, P(x/w)$$

Les probabilités P(w) (issues du modèle de langage) et P(x/w) (issues du modèle acoustique) peuvent être estimées séparément. La règle de décision gère l'interaction entre les observations et les différentes sources de connaissances qui sont de trois sortes :

-le modèle de langage qui donne la probabilité P(w),

- la liste des phonèmes,

- les mots du dictionnaire traduits phonétiquement qui en fonction de x et des modèles appris donnent la probabilité P(x/w).

Le résultat doit être le meilleur compromis entre les observations et les connaissances a priori.

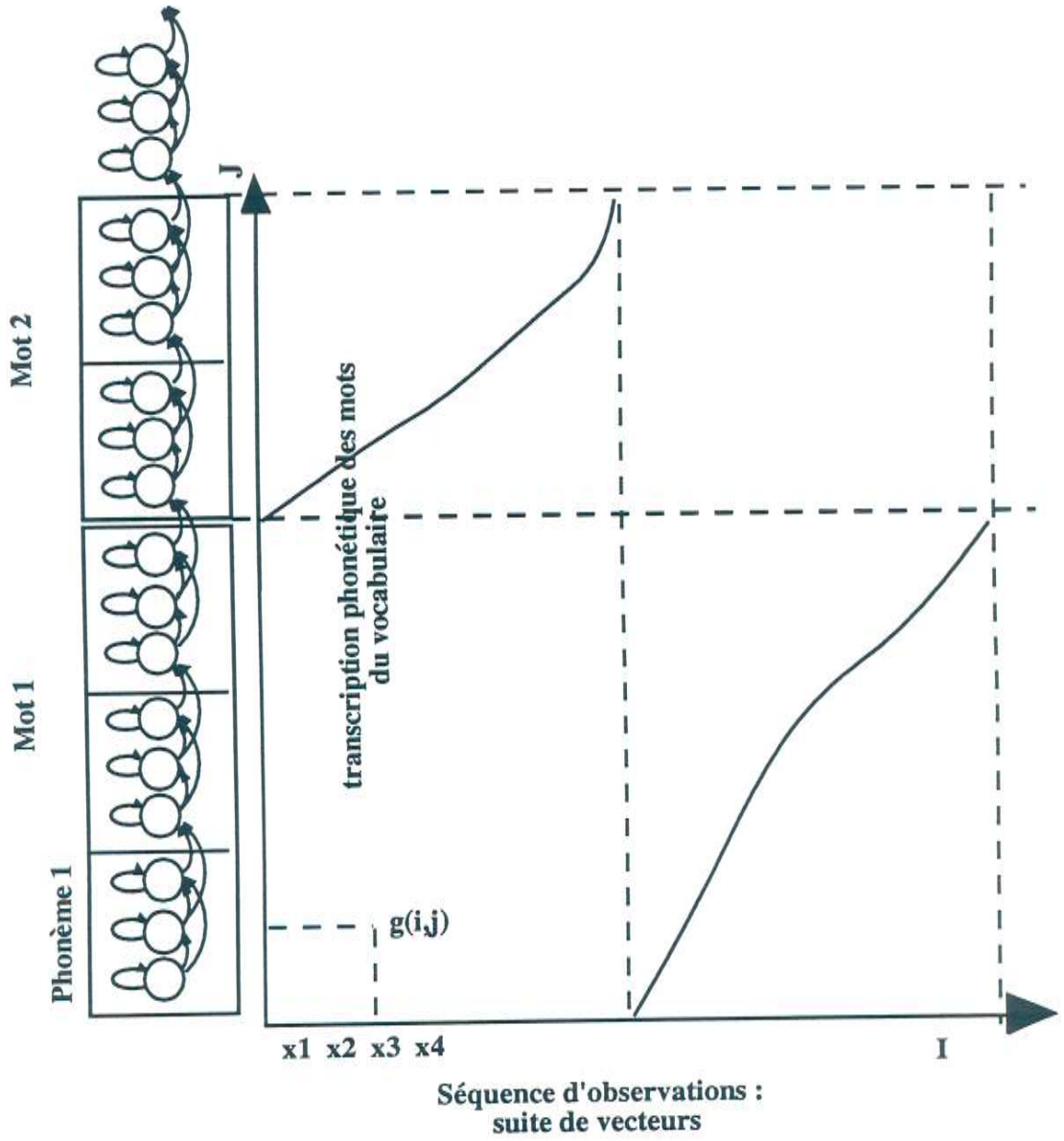

Fig. 8.2 : ***Reconnaissance avec l'algorithme de Viterbi***

8.1.3 Apprentissage avec l'algorithme de Viterbi

L'apprentissage de Viterbi (cf chapitre 2) comprend tout d'abord une phase d'initialisation qui fixe l'ensemble des paramètres $a_{i,j}$ et b_i des modèles. Puis de façon itérative, tant que le critère de convergence n'est pas atteint, un alignement temporel est effectué entre les modèles phonétiques concaténés correspondant aux phrases et les séquences acoustiques qui permet d'associer chaque vecteur acoustique à un état d'un modèle, et une réestimation des paramètres des modèles en fonction des observations collectées par chaque état.

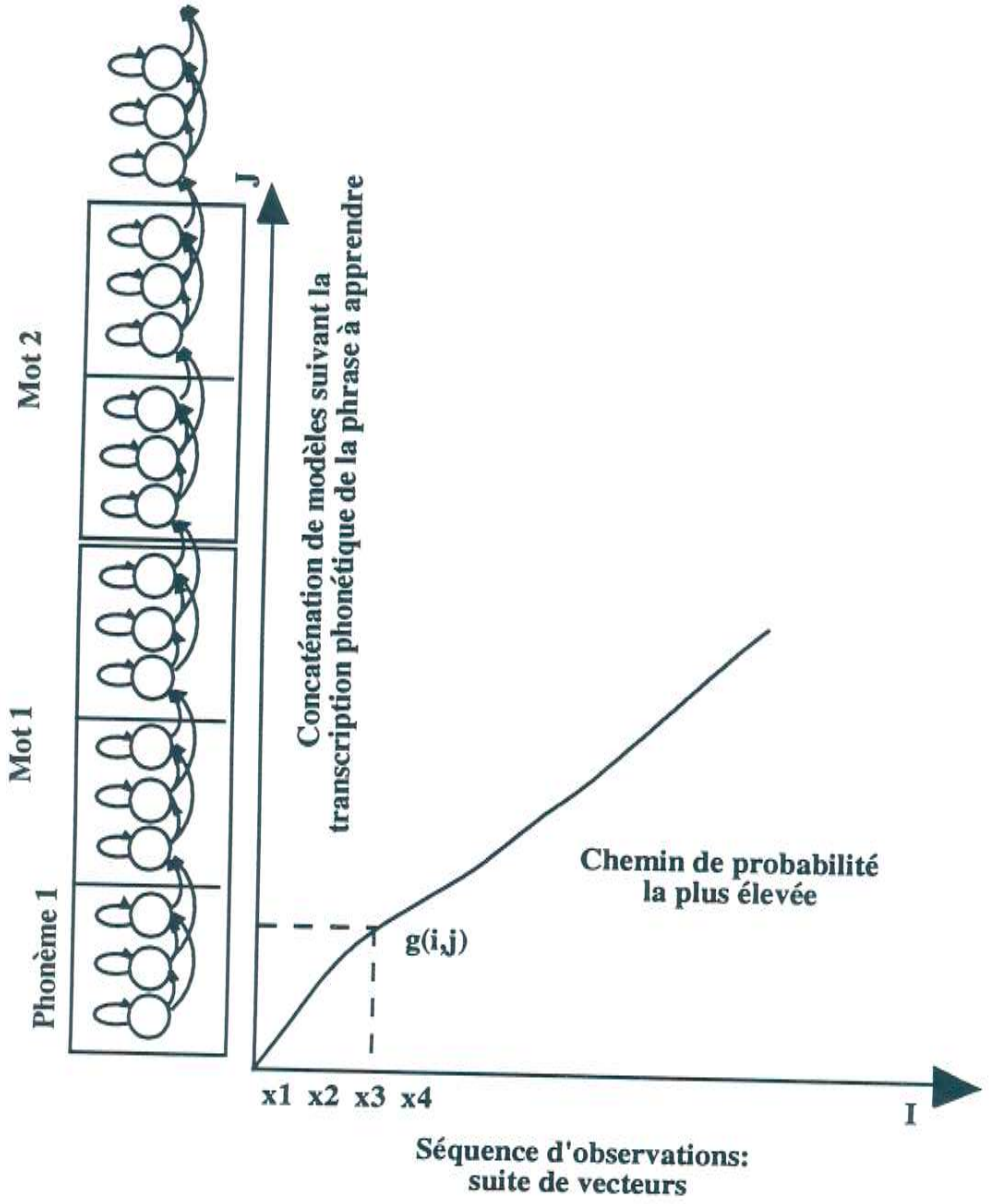

Fig. 8.3 : ***Apprentissage avec l'algorithme de Viterbi***

Soit les vecteurs observés x d'indice i=1,...,I (notés sur le schéma 8.3 x_1 x_2 x_3 x_4... pour symboliser une séquence de vecteurs), soit j=1...J l'axe sur les états, g(i,j) le coût du chemin pour arriver au point (i,j):

$$g(i,j) = \max_{\text{états } j'} \left(g(i-1,j')\, a_{j',j}\, b_j(x_i) \right)$$

$a_{j',j}$ correspond à la probabilité de transition entre l'état j' et l'état j,

Si $b_j(x_i)$ la probabilité d'émission de x_i dans l'état j est modélisée par $p(x_i/j)$.

on a alors l'équation : $g(i,j) = \max_{\text{états } j'} \left(g(i-1,j')\, a_{j',j}\, p(x_i/j) \right)$

On code pour des facilités de calcul les probabilités comme des entiers en utilisant la transformation logarithmique des probabilités: $-\log_b(p)$

où b est la base choisie afin que toutes les probabilités soient représentées comme des entiers de taille donnée. Par exemple, si on code les probabilités sur 2 octets, on a : $b = e^{1/65535}$ où 65535 est le nombre le plus grand pouvant être stocké.

L'équation devient alors : $$g(i,j) = \min_{\substack{\text{états } j' \\ \text{précédents } j}} \left(g(i-1,j') - \log(a_{j',j}) - \log(p(x_i/j)) \right)$$

Pour chaque méthode hybride ou non, nous explicitons le terme $-\log(p(x_i/j))$ de cette équation. Dans le cas de modèles discrets, ce terme est en fait : $-\log(P(x_i/j))$. L'apprentissage des modèles discrets et continus est décrit dans les paragraphes suivants. On trouvera une introduction à l'algorithme de Viterbi dans le chapitre 2.

8.2 Modèles Markoviens cachés

8.2.1 Modèles discrets: DHMM

Soit un modèle donné, de paramètres $a_{j',j}$ et $b_j(x_i)$:

Si le modèle est discret, la loi d'émission b_j peut être donnée exhaustivement, c'est une probabilité discrète. Les vecteurs d'observations (notés par x_1 x_2 x_3 x_4 sur le schéma, cf fig 8.3) sont alors des vecteurs quantifiés décrits par un code $QV(x_i)$. Supposons un ensemble de vecteurs quantifiés de 256 prototypes (ou codes), les $b_j(x_i)$ sont alors des vecteurs de 256 coefficients.

L'apprentissage avec l'algorithme de Viterbi permet de calculer les b_j, c'est-à-dire les probabilités d'associer les vecteurs quantifiés aux états des modèles par de simples comptages cf §2.5.3.3.

En reconnaissance, la séquence inconnue est quantifiée et on calcule très facilement pour chaque état j la probabilité :

$$-\log(P(x_i)/j) = -\log(P(QV(x_i)/j))$$

Le coût total g(I,J) minimal indique le meilleur chemin et permet de reconstituer la suite de mots recherchés (ou de phonèmes). Soit l'axe I des séquences observées et l'axe J des modèles, soit le point (I,J) le dernier point du chemin, $a_{j',J}$ correspond à une probabilité de transition entre l'état j' et le dernier état J, l'équation globale du coût total est :

$$g(I,J) = \min_{\substack{\text{états } j' \\ \text{précédents } J}} \left(g(I-1,j') - \log(a_{j',J}) - \log(P(QV(x_I)/J)) \right)$$

8.2.2 Modèles continus: CHMMs

Soit un modèle donné, de paramètres $a_{j',j}$ et de lois d'émission $b_j(x_i)$:

Dans le cas des modèles continus ou semi-continus, une densité de probabilité au lieu d'une probabilité représente la loi d'émission $b_j(x_i)$. Cette densité de probabilité est souvent modélisée par une ou plusieurs gaussiennes ou laplaciennes. Dans le cas du modèle de Bakis décrit au §8.1.1, le phonème est modélisé par 3 segments, une densité de probabilité est alors associée à chacun d'eux.

$$b_j(x_i) = p(x_i/j)$$

Les modèles HMMs de "*mixtures densities*" utilisent des "mélanges" de densités de probabilités [Ney 90]. Ces densités sont obtenues lors de l'apprentissage. Chaque loi d'émission $b_j(x_i)$ est alors constituée d'une densité de probabilité d'émission qui est la somme de fonctions gaussiennes ou laplaciennes de la forme:

$$b_j(x_i) = p(x_i/j) = \sum_k C_k \, p(x_i/k)$$

où $p(x_i/k)$ est une distribution normale, k le nombre de gaussiennes ou laplaciennes et C_k est un vecteur de pondération de ces fonctions. Chaque densité représentée par une somme de distributions normales unimodales est une approximation d'une densité multimodale. On parlera souvent dans la suite de ce

chapitre des densités associées à un segment, c'est un abus de langage, en fait chaque segment est défini par une densité représentée par des mélanges de gaussiennes ou de Laplaciennes. Dans nos expériences, les distributions utilisées sont des laplaciennes et l'opérateur $\sum$ est remplacé par l'opérateur Max.

Soit $d(x_i/k)$ correspondant à la distance entre le vecteur observé x_i et le vecteur moyen de la laplacienne k, A(v) un facteur de normalisation, v le vecteur de variance des paramètres, Σ_v la matrice positive des covariances de la loi normale et Cte une constante de pondération.

L'équation suivante décrit une fonction Laplacienne :

$$p(x_i/k) = A(v)\, e^{-d(x_i/k)}$$

avec le facteur de normalisation A(v):

$$A(v) = \frac{1}{Cte\sqrt{Det(\Sigma_v)}}$$

Pour une loi gaussienne avec une matrice de covariance diagonale, la distance $d(x_i/k)$ est la suivante:

$$d(x_i/k) = \frac{1}{2}\sum_c \left(\frac{x_{i,c}-m_c(k)}{v_c}\right)^2$$

avec c la dimension des vecteurs, $m_c(k)$ le cième coefficient du vecteur moyen de la gaussienne k et v_c de cième coefficient du vecteur de variance des paramètres.

Pour une laplacienne, la distance $d(x_i/k)$ est la suivante:

$$d(x_i/k) = \sum_c \left|\frac{x_{i,c}-m_c(k)}{v_c}\right|$$

Dans le cas des modèles **semi-continus**, toutes les densités de probabilité sont communes aux modèles, seuls les Ck sont associés aux états. Pour les modèles **continus**, les densités de probabilité dépendent du modèle.

Apprentissage des modèles continus

Les p(x/i) sont obtenues par apprentissage en utilisant les algorithmes : de Viterbi, de "split" et des k-moyennes. La figure 8.4 schématise l'apprentissage.

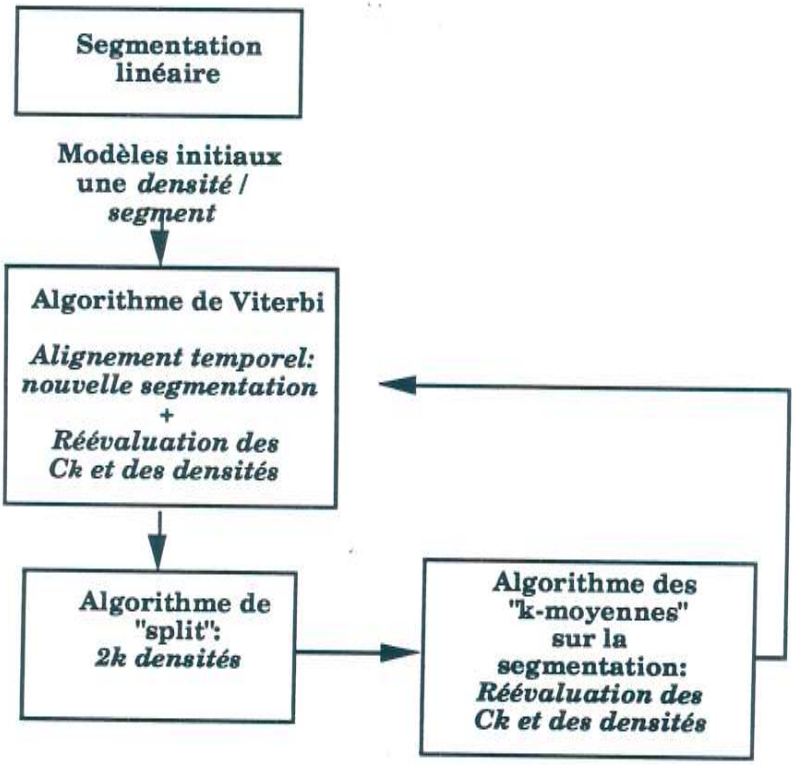

Fig. 8.4 : ***Apprentissage des p(xi/j) pour des modèles continus de type "mixtures densities"****, un "mélange" de laplaciennes dans notre cas.*

Initialisation des modèles avant apprentissage :

Une segmentation linéaire est tout d'abord évaluée sur une partie des données de parole afin d'initialiser les modèles. Les $p(x_i)$ sont initialisées avec une densité par segment. La détection de début et fin de phrase est obtenue à partir de la courbe d'énergie.

Pour mieux ajuster les frontières des phonèmes, la procédure d'alignement temporel est appliquée deux fois sur tout le corpus d'apprentissage. Après ces deux itérations, l'apprentissage est fait sur toutes les phrases sans détection de début et fin de phrase.

Apprentissage :

Prennons comme exemple la base de données Darpa, le nombre de phonèmes est de 47. Chaque modèle contient 3 segments. Pour un phonème P, on a 3 étiquettes P1 P2 P3, on modélise le début du phonème P1, le centre P2 et la fin du phonème P3. Le nombre de segments est donc de 3 * 46 +1 = 139 segments (d'où 139 étiquettes dont P1, P2, P3, B1, B2, B3...), le phonème qui n'est modélisé que par 1 segment est le silence. Le nombre initial de densités est de 139. Une densité est représentée par un vecteur moyen et un vecteur de variance identique pour tous les phonèmes. Ce vecteur de variance est calculé sur toutes les données.

L'algorithme de Viterbi permet d'obtenir une nouvelle segmentation et de calculer très simplement par comptage les C_k qui sont les pondérations associées aux densités et de réévaluer les densités. Après quelques itérations de l'algorithme de Viterbi (de l'ordre de 6), la distorsion moyenne ne varie plus, on utilise alors l'algorithme de "*split*".

L'algorithme de "*split*" permet d'augmenter le nombre de densités. Les densités sont dupliquées sous certaines conditions. L'algorithme de "*split*" remplace le vecteur moyen existant par deux nouveaux vecteurs infimement modifiés par rapport à ce vecteur initial. Les conditions pour créer deux nouveaux vecteurs sont d'avoir un nombre suffisant de données associées à chaque nouvelle distribution et bien sûr une distribution initiale (avant le "*split*") dont la variance soit supérieure à un certain seuil.

L'algorithme utilisé est une variante des k-moyennes qui permet de réévaluer les distributions des densités et les C_k sachant la segmentation.

Les algorithmes de Viterbi, de "*split*" et des k-moyennes sont réitérés jusqu'à obtenir le nombre de densités désirées. Les expériences ont montré qu'environ 50 densités par segment étaient nécessaires à partir de la segmentation linéaire pour obtenir une bonne segmentation.

En reconnaissance, on calcule pour chaque état j de chaque modèle phonétique :

$$-\log(p(x_i/j)) = \min_{k \text{ densités}} \left(-\log C_k - \log A(v) + \sum_c \frac{(x_{i,c} - m_c(k))}{v_c}\right)$$

Le coût total g(I,J) minimal indique le meilleur chemin et permet de reconstituer la suite phonétique recherchée. Soit I l'axe correspondant aux sorties du réseau, J l'axe des modèles, (I,J) le dernier point du chemin, $a_{j',J}$ la probabilité de transition de l'état j' au dernier état du chemin J, g(I,J) sécrit :

$$g(I,J) = \min_{\substack{\text{états } j' \\ \text{précédents } J}} \left(g(I-1,j') - \log(a_{j',J}) - \log(p(x_I/J)) \right)$$

8.3 TDNN comme pré-traitement d'un HMM

Les réseaux de neurones sont souvent comparés aux modèles Markoviens (HMMs). En réalité, cette comparaison n'a de sens que pour la classification de formes statiques puisque les réseaux ne possèdent pas de mécanisme d'alignement temporel efficace permettant d'intégrer des séquences temporelles. Tandis que les modèles markoviens cachés sont typiquement des systèmes capables de gérer des séquences d'évènements. L'algorithme de reconnaissance de Viterbi, classiquement utilisé dans les HMMs, permet de faire une approximation de la meilleure séquence d'états des modèles à partir des distributions de probabilités d'émission associées à chaque état. Une réelle comparaison se situe au niveau de la modélisation de ces distributions. Elles peuvent être des mélanges de densités continues "*continuous mixture densities*", mais aussi des sorties de classifieurs discrimants tels que les MLPs... On parle alors de système hybride neuronal et markovien où les réseaux sont utilisés comme prétraitements des modèles Markoviens.

8.3.1 Utilisation des sorties des TDNNs

Les sorties des réseaux sont des transformations non linéaires des vecteurs d'entrée x. Un vecteur de sortie étant de dimension égale au nombre de formes à classifier n, les coefficients des vecteurs de sorties peuvent aussi être considérés comme des probabilités a posteriori des classes $C(C_1,...,C_n)$ pour un x_i donné. La figure 8.5 schématise les coopérations mises en oeuvre entre les modèles TDNN et HMM où le modèle neuronal sert de prétraitement au modèle markovien.

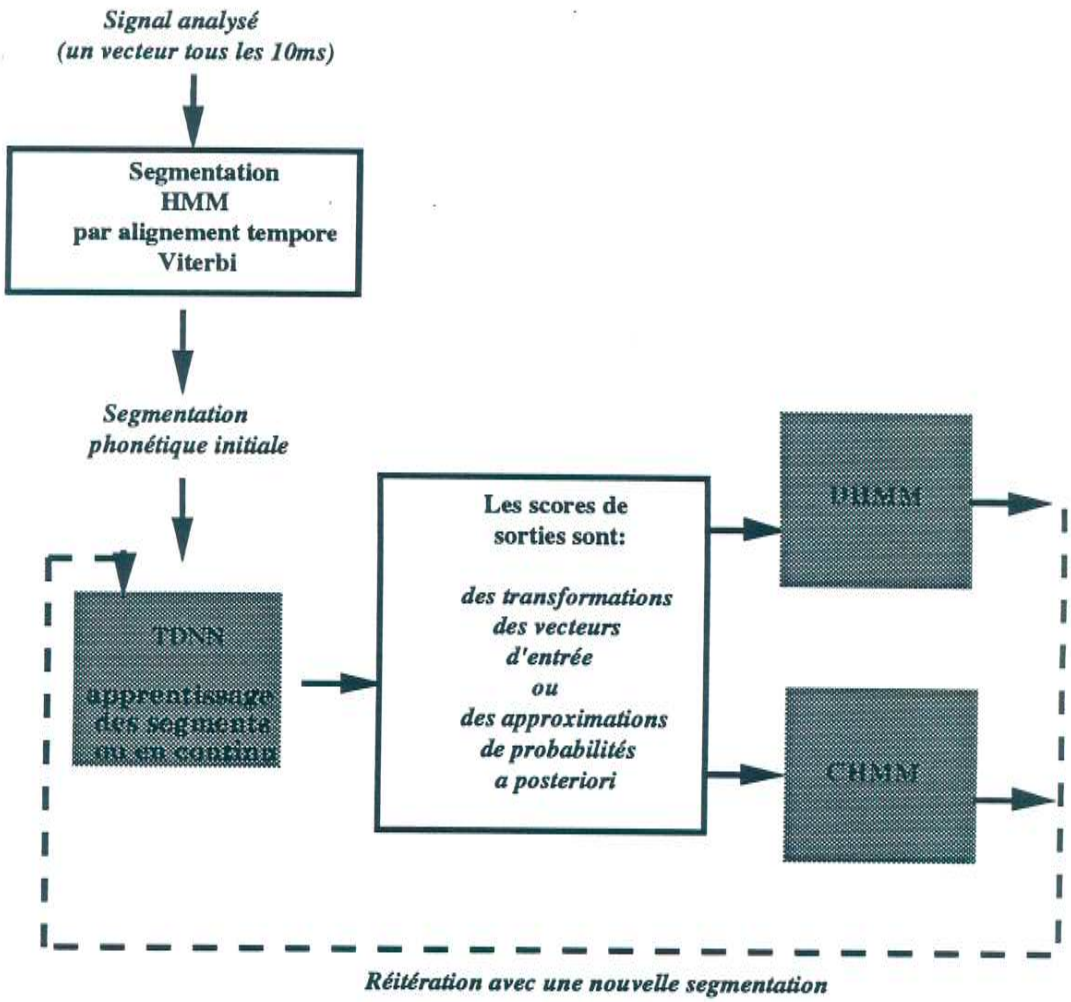

*Fig. 8.5 : **TDNN comme prétraitement d'un HMM.***

Transformation des vecteurs d'entrée

Un réseau multicouche est une suite de filtres linéaires ou non linéaires qui transforme un vecteur d'entrée de dimension m en un vecteur de sortie de dimension n permettant de classifier l'entrée. On peut le comparer d'une certaine façon à une analyse discriminante. Gallinari & al [Gallinari 88] [Gallinari 91] ont montré une équivalence entre l'analyse discriminante linéaire et les perceptrons multicouches linéaires à une couche cachée.

L'intérêt de l'analyse discriminante linéaire (LDA) est de réduire la dimension de l'espace tout en regroupant les formes de même classe. L'analyse discriminante permet de projeter les formes d'un espace X dans un espace plus petit N(X), suivant une application de la matrice N, où la distance entre les classes est la plus grande possible et les vecteurs appartenant à la même classe sont bien regroupés. C'est une transformation linéaire de l'espace. La matrice N est évaluée à partir des exemples d'apprentissage [De Bollivier 92]. Dans le cas de fonctions non linéaires, un réseau

multicouche fait une projection et une séparation de classes sur ses couches cachées. De couche en couche, on observe une meilleure séparation des données.

Approximation de probabilités a posteriori

Il a été montré [Bourlard, Wellekens 89], [White 89] que les sorties d'un réseau multicouche entraîné avec un critère discriminant tel que critère de MSE ou de MAP peuvent être considérées comme des approximations des probabilités a posteriori cf §2.4.5.1. L'optimum global de la fonction d'erreur est obtenu si les sorties des réseaux sont exactement les probabilités des classes. Cependant, dans la réalité, cette approximation est d'autant plus fiable que le réseau répond à certaines conditions :

- qu'il ait une structure ayant suffisamment de cellules dans la couche cachée,
- que le nombre d'échantillons utilisés soit assez grand pour que l'algorithme d'apprentissage soit capable de trouver un optimum global.

Dans notre approche, les scores de sortie des réseaux sont forcés à être des probabilités et le critère utilisé à l'apprentissage est celui de MSE et non de MAP. Chaque valeur du vecteur de sortie représente approximativement une probabilité d'être de classe C_i c'est à dire $p(C_i/x)$ à l'instant t.

Intégration continue/discrète

Nous proposons plusieurs intégrations dans un HMM continu et discret pour les réseaux élaborés dans nos premières expériences. Nous avons utilisé les sorties des réseaux comme des vecteurs d'observation pour un HMM afin de modéliser les transitions entre phonèmes qui n'avaient pas été apprises par les réseaux. Les vecteurs de sortie générés par les réseaux pour ces parties transitions sont quasiment aléatoires, ce sont ces vecteurs qui seront plus particulièrement codés par plusieurs densités dans un CHMM, cf 8.3.4.
Les derniers réseaux construits sur les données en continu ne permettent plus de coder leurs sorties avec plusieurs densités par segment dans un HMM. Les sorties sont alors réellement des approximations de probabilités a posteriori.

Réitération avec une nouvelle segmentation

La boucle de réitération est indiquée en pointillé sur le schéma. Réapprendre avec la segmentation issue de l'intégration des scores des réseaux dans un algorithme de Viterbi permettrait de moins dépendre de la segmentation issue des HMMs et d'obtenir de meilleurs résultats.

Toutefois deux problèmes majeurs existent qui font que cette boucle de réitération n'a jamais en fait été utilisée :

- l'apprentissage est beaucoup trop long sur une machine monoprocesseur pour le répéter même un petit nombre de fois,

- les bornes calculées par le système hybride TDNN-DHMM à partir des scores de sortie des réseaux ne donnent pas une segmentation réutilisable. Avec les TDNN-CHMM, l'expérience n'a pas été menée mais devrait être plus intéressante.

8.3.2 Systèmes hybrides discrets: TDNN-DHMM

Dans cette intégration, les sorties des réseaux sont considérées comme des approximations des probabilités a posteriori. Les probabilités a posteriori divisées par la probabilité a priori de la classe considérée sont des approximations des probabilités conditionnelles p(x/j). On peut alors les associer aux b_j des modèles.

Dans notre approche, la somme des scores est forcée à être égale à 1. On intègre les sorties des réseaux dans le HMM en utilisant la règle de Bayes sachant que les probabilités a posteriori peuvent être exprimées comme des probabilités conditionnelles directement utilisables dans un HMM :

$$p(x/j) = \frac{P(j/x)}{P(j)} p(x) \quad \text{avec } p(x) = \sum_j p(x/j)P(j)$$

les P(j) sont implicitement décrits dans les HMMs par les différentes source de connaissances: modèle de langage...

Soit $S(j/x_i)$ un score de sortie du tdnn, x_i l'observation, on obtient une approximation de la probabilité a posteriori de j, j étant considéré comme une classe de sortie pour le TDNN et comme un état d'un modèle pour le HMM, nous garderons la notation j qui désigne en fait l'état q_j dans le HMM.

$$P_{tdnn}(j/x_i) \approx S(j/x_i)$$

$$\text{d'où } P_{tdnn}(x_i/j) \approx \frac{P_{tdnn}(j/x_i)}{P(j)}$$

Les vecteurs de sortie du TDNN sont obtenus à partir de différentes architectures, elles sont décrites en détail dans le chapitre 7. Pour l'architecture G-TDNN, les vecteurs de sortie contiennent directement la connaissance de tous les phonèmes.

Pour les architectures hiérarchiques comme HT-TDNN et H-TDNN, l'intégration de la structure est faite au niveau des sorties des sous-réseaux.

Pour l'architecture H-TDNN, "*Hierarchical TDNN*", la règle d'intégration des scores de sorties des réseaux utilisées dans les expériences relatées dans ce chapitre est la suivante :

Soit S(j/x) le score du phonème j pour l'observation x qui résulte de l'intégration des différents sous-réseaux, K_k la macro-classe contenant j, $S(K_k/x)$ le score de K_k dans le réseau de macro-classes et $S(j/K_k,x)$ le score de j dans le réseau classifiant les phonèmes de la macro-classe K_k ,

$$S(j/x) = S(j/K_k,x)S(K_k/x)$$

Soit $P_{tdnn}(j/x_i)$ l'approximation de la probabilité réalisée par rapport à cette règle d'intégration :

$$P_{tdnn}(j/x_i) = \frac{S(j/x_i)}{\sum_j S(j/x_i)}$$

Pour l'architecture HT-TDNN, "*Hierachical TDNN with Traps*", la règle d'intégration des scores de sorties des réseaux utilisées dans les expériences relatées dans ce chapitre est la suivante :

Soit S(j/x) le score d'intégration des différents réseaux, K_k la macro-classe contenant j, $S(K_k/x)$ le score de K_k dans le réseau de macro-classes et $S(j/K_{ki},x)$ le score de j dans le ième réseau classifiant les phonèmes de K_k et une trappe pour la macro-classe K_i,

$$S(j/x) = S(K_k/x) \sum_{i \neq k} \lambda_i \, S(j/K_{ki},x)$$

$$\text{avec} \sum_{i \neq k} \lambda_i = 1$$

Soit $P_{tdnn}(j/x_i)$ l'approximation de la probabilité réalisée par rapport à cette règle d'intégration :

$$P_{tdnn}(j/x_i) = \frac{S(j/x_i)}{\sum_j S(j/x_i)}$$

Les probabilités de transition entre états sont fixées empiriquement, on privilégie la boucle sur un état par rapport au saut d'un état pour éviter les élisions. Le nombre d'états d'un modèle est fonction de la longueur moyenne du phonème en nombre de trames. Les modèles sont découpés en trois segments auxquels sont associés une probabilité d'émission des phonèmes suivant la topologie présentée au §8.1.1.

Deux stratégies ont été adoptées pour construire un système hybride TDNN-DHMM, une sans apprentissage et l'autre avec un apprentissage Viterbi :

- la première appelée TDNN-DHMM est de considérer les probabilités issues du TDNN comme les b_j des modèles. La probabilité associée à l'émission d'un phonème est la même sur tous les états de ce modèle.

- la deuxième appelée (P)TDNN-DHMM, le P signifiant pondéré, est de faire un apprentissage Viterbi (fig. 8.6) avec le critère MLE pour calculer des vecteurs de probabilité de confusion. Elles sont initialisées à des valeurs de 1-eps (eps étant proche de 0) pour le phonème correspondant au modèle et 1/(n-1) eps pour les n-1 èmes autres coefficients, eps étant une valeur très faible qui permet d'éviter les probabilités à zéro. Après un alignement temporel de type Viterbi entre les modèles phonétiques des phrases d'apprentissage et les phonèmes de plus grande probabilité obtenus en sortie du réseau, on obtient la probabilité de confusion C entre le meilleur phonème et le phonème associé à l'état j en comptant les phonèmes reconnus par chaque état.

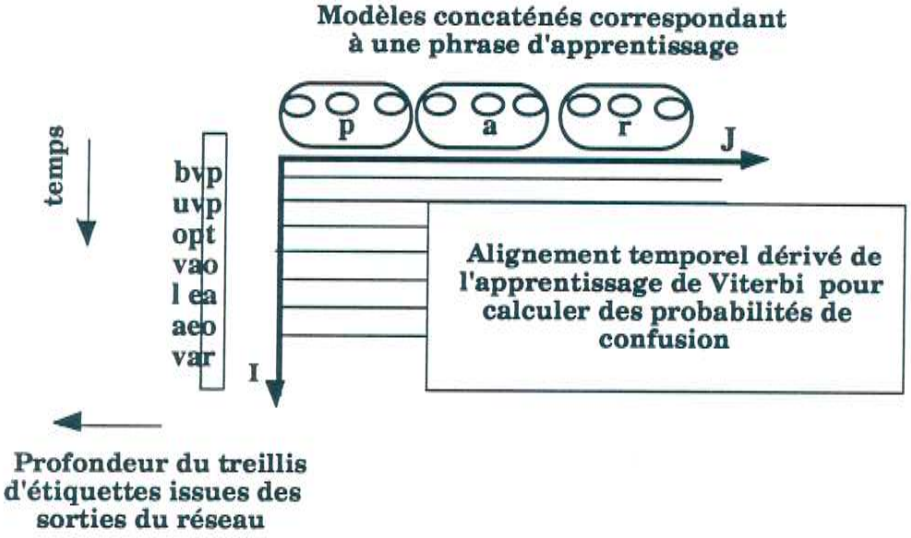

*Fig. 8.6 : **Apprentissage de la matrice de confusion***

Si on modélise chaque phonème par 3 segments cf §8.1.1 correspondant aux étiquettes (P1, P2, P3,...) on obtient 139 segments pour la base Darpa. Les vecteurs de confusion sont donc de dimension 139 et 3 vecteurs de probabilités de confusion sont associés à un modèle phonétique pour le segment de début, de milieu et de fin du phonème.

Soit $a_{j',j}$ la probabilité de transition de l'état j' à j, soit I l'axe des sorties du TDNN et J l'axe des modèles, en reconnaissance, le coût local est :

$$g(i,j) = \min_{\substack{\text{états } j' \\ \text{précédents } j}} \left(g(i-1,j') - \log(a_{j',j}) - \log(P(x_i/j)) \right)$$

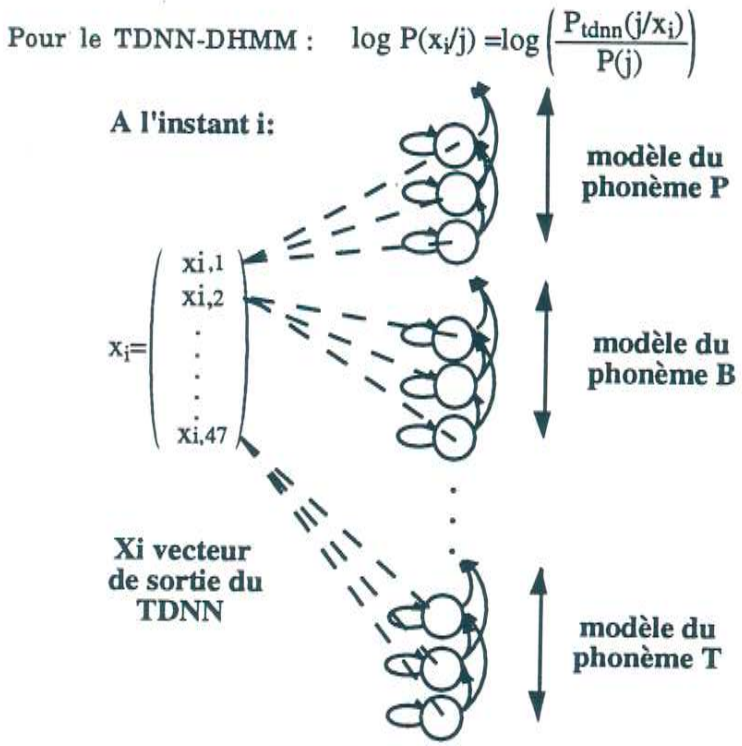

*Fig.8.7 : **TDNN-DHMM.***

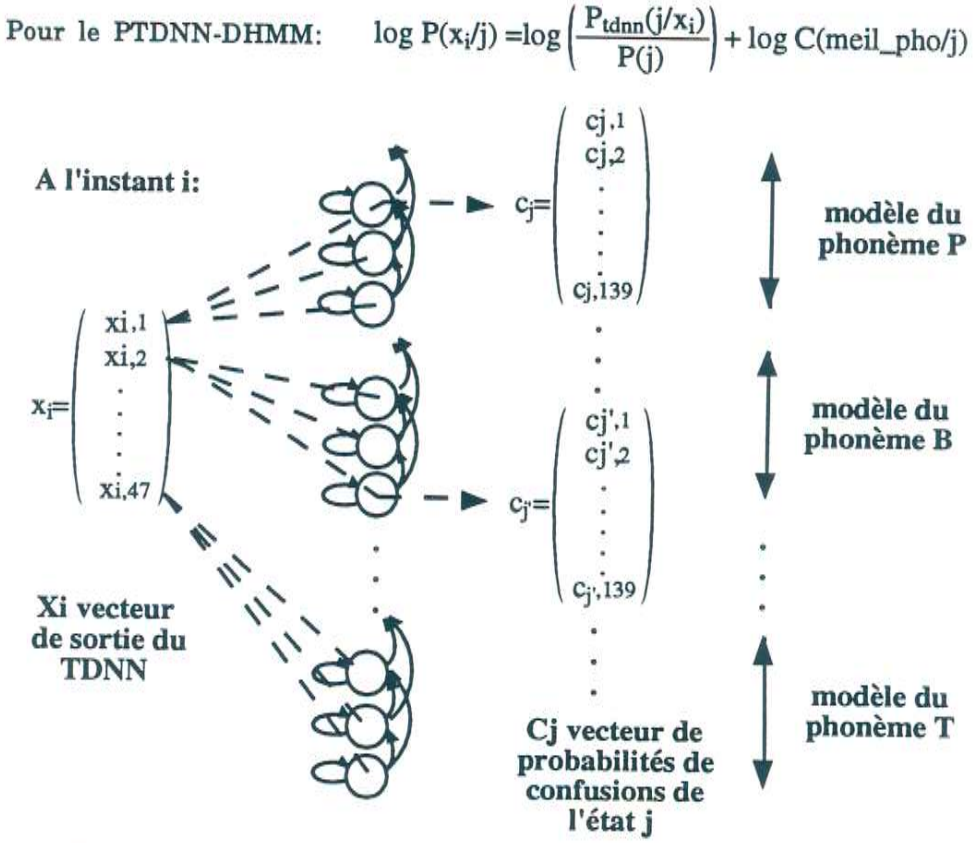

*Fig. 8.8 : **(P)TDNN-DHMM***

8.3.3 Comparaison entre TDNN-DHMM, (P)TDNN-DHMM et CHMM.

Nous comparons deux méthodes HMMs, l'une discrète (DHMM) et l'autre continue (CHMM) avec des systèmes hybrides avec apprentissage d'une matrice de confusion ((P)TDNN-DHMM) et sans apprentissage (TDNN-DHMM). Chacune de ces méthodes a été décrite précédemment. Nous récapitulons dans un tableau le coût local de chaque méthode calculé lors de la reconnaissance.

Soit les vecteurs observés x d'indice i=1,...,I, soit j=1...J l'axe sur les états, g(i,j) le coût du chemin pour arriver au point (i,j) est :

$$g(i,j) = \min_{\substack{\text{états } j' \\ \text{précédents } j}} \left(g(i-1,j') - \log(a_{j',j}) - \log(p(x_i/j)) \right)$$

$a_{j',j}$ correspond à la probabilité de transition entre l'état j' et l'état j,

Pour chacune des approches, $\log(p(x_i/j))$. est calculé comme suit :

pour le DHMM : $-\log(P(x_i)/j) = -\log(P(QV(x_i)/j))$

Pour le TDNN-DHMM : $-\log P(x_i/j) = -\log\left(\frac{P_{tdnn}(j/x_i)}{P(j)}\right)$

Pour le (P)TDNN-DHMM :

$$-\log P(x_i/j) = -\log\left(\frac{P_{tdnn}(j/x_i)}{P(j)}\right) - \log C(meil\_pho/j)$$

Pour le CHMM : $-\log(p(x_i/j)) = \min_{k \text{ densités}} \left(-\log C_k - \log A(v) + \sum_c \frac{(x_{i,c} - m_c(k))}{v_c}\right)$

De manière générale, pour tous les tests effectués, le taux d'erreur calculé sur les mots ou sur les phonèmes correspond à la somme des ajouts, insertions et substitutions de mots (ou de phonèmes) divisés par le nombre de mots (ou de phonèmes) de la phrase testée. Les tableaux décrivant les résultats donnent des indications sur la méthode (DHMM, CHMM ou systèmes hybrides), le mode d'apprentissage pour les modèles discrets, le nombre de densités pour les modèles continus et sur les données utilisées (apprentissage ou test). Le champ "mode" permet de préciser pour les systèmes hybrides discrets si on utilise une matrice de confusion (MC) ou non (-) et le champ densité précise pour les systèmes continus le nombre de laplaciennes par segment.

8.3.3.1 Tests en français

La base de donnée utilisée sur un locuteur est décrite en annexe 1. Dans cette expérience, le nombre de phonèmes à reconnaître est de 31 et le nombre de mots du vocabulaire de 247. Une des particularités de cette base de données qui a servi à nos premières expériences est d'avoir un vocabulaire très hétérogène entre la base de test et d'apprentissage, 134 mots pour l'apprentissage et 137 pour le test avec une vingtaine de mots fonctionnels en commun. Cette base de données a été créée pour contenir tous les contextes CV (consonne-voyelle) et VC au moins une fois, pour se faire elle contient à la fois des phrases de parole continue et des logatomes qui permettent d'enrichir les contextes (APATAKA-ABADAGA...). Les méthodes hybrides TDNN-DHMM et PTDNN-DHMM décrites ci-dessus ont été comparées à un CHMM. L'analyse du signal est la même dans tous les cas. L'architecture TDNN utilisée est celle décrite sous le nom de G-TDNN dans le §7.2.2. L'apprentissage de cette architecture a été fait à partir de données segmentées et non en continu, la conséquence de ce choix est que le réseau n'a pas appris les transitions entre phonèmes. Ces tests ont été faits sans modèle de langage.

Tableau de performances des architectures G-TDNN-DHMM et CHMM *(TDNN : non apprentissage des transitions, vecteurs d'entrée de 17 coefficients : 16 coef. FFT + Energie. HMM : vecteurs d'entrée de 17 coefficients : (16 coef. FFT + Energie).*

%Erreur/mots	*Mode*	*Apprentissage*	*Test*
G-TDNN-DHMM	-	17.7	30.0
	P	9.1	27.6
Erreur/mots	***Densités***	***Apprentissage***	***Test***
CHMM	8	22.3	67.5

Tab. 8.1 : ***Comparaison entre l'architecture hybride TDNN-DHMM et les CHMMs.*** *Pourcentage d'erreurs sur les mots pour le corpus d'apprentissage et de test.*

Les résultats sur le corpus en français (tab. 8.1) montrent que les systèmes hybrides donnent de meilleurs résultats que les modèles markoviens continus. La mauvaise performance des CHMMs dans cette expérience vient du manque de données, en effet certains contextes phonétiques sont présents dans le test et ne sont jamais appris. On ne peut d'ailleurs pas beaucoup accroître le nombre de densités de probabilité par segment à cause du peu de données. Les TDNNs sont dans ce cas plus robustes pour traiter des données inconnues. Le système hybride (P) G-TDNN-

DHMM donne de meilleurs résultats que l'intégration simple du TDNN dans le HMM. Cet apprentissage permet surtout de modéliser les probabilités de transition entre phonèmes qui n'ont pas été codées par le TDNN. Il permet aussi d'estimer une probabilité de confusion différente pour le début, le centre et la fin des modèles afin d'avoir une information plus précise.

8.3.3.2 Tests sur DARPA

La base de donnée DARPA RM "Speaker Dependent" est décrite dans l'annexe 2. Ces tests ont été menés sur un locuteur JWS0_4 de difficulté moyenne. Le nombre de phonèmes à classifier est de 47. Tous les scores d'erreur sont donnés avec ou sans utilisation du modèle de langage. La "perplexité" est de 991 sans modèle de langage et devient 63 avec le modèle de langage.

L'apprentissage des architectures TDNNs : G-TDNN, H-TDNN, HT-TDNN a été menée dans cette première expérience à partir de données segmentées, c'est-à-dire sans apprendre les transitions entre les segments phonétiques cf §6.4. Les vecteurs d'entrée des réseaux sont constitués de 16 paramètres FFT moyennés par l'energie de la trame qui est ajoutée en 17ème paramètre. La moyenne des scores des sous-réseaux utilisés est de 78.3% de reconnaissance sur le corpus de test et de 65.4 pour les macro-classes.

Tableau de performances des architectures G-TDNN, H-TDNN, HT_TDNN intégrées dans un HMM *(TDNN: non apprentissage des transitions, vecteurs d'entrée de 17 coefficients : 16 coef. FFT + Energie).*

Test	*Mode*	*Erreurs/phon.*	*Erreurs/mots*
G-TDNN-DHMM	-	60.5	74.6
	MC	50.4	50.8
H-TDNN-DHMM	-	55.9	65.5
	MC	52.2	56.9
HT-TDNN-DHMM	-	52.4	52.8
	MC	51.0	41.1

*Tab. 8.2 : **Différentes architectures TDNN-DHMM.** Pourcentage d'erreurs sur les phonèmes et les mots pour le corpus de test.*

Les résultats du tableau 8.2 ne sont pas très bons. La structure HT-TDNN-DHMM permet cependant d'obtenir les meilleurs scores. Il faut rappeler que le réseau n'a pas appris les transitions entre les phonèmes aussi la phase d'apprentissage avec

Viterbi est très importante. Cet apprentissage sur les sorties des réseaux permet d'améliorer nettement les scores sur les mots et moins nettement celui des phonèmes. L'explication de ce phénomène est assez simple: les probabilités de confusion calculées par apprentissage HMM pondèrent les scores issus du réseau. Les scores des phonèmes vont se rapprocher mais ne modifient pas le classement des phonèmes. En revanche, à cause du contexte phonétique si les scores des phonèmes sont plus proches, on peut plus facilement passer d'un mot à un autre et trouver finalement le bon candidat.

L'analyse du signal pour les CHMMs est aussi une analyse FFT. Toutes les 10ms de parole, 30 coefficients sont calculés sur une fenêtre de 25 ms de parole. L'energie moyenne d'une trame est soustraite aux coefficients et ajoutée au vecteur comme 31ième valeur. 15 coefficients correspondant aux dérivées premières et 15 coefficients correspondant aux dérivées secondes obtenus à partir des dérivées premières sont calculés sur 70 ms de parole (30ms de contexte gauche et 30 ms à droite) et ajouté au vecteur. Les vecteurs d'entrée sont donc de dimension 63. Des tests avec les TDNNs utilisant la même analyse sont décrits au §6.2.3.2 .

Tableau de performances des CHMMs *(63 coefficients : 30 coefficients FFT + Energie + dérivées premières et secondes)*

Test	*Densités*	*Erreurs/phon.*	*Erreurs/mots*
CHMM	1	55.1	52.8
	2	49.1	46.2
	-	-	-
	58	24.4	14.3
	116	21.2	12.2
	210	20.5	11.5

Tab. 8.3 : ***Variations du nombre de densités par segment pour les CHMMs.*** *Pourcentage d'erreurs sur les phonèmes et les mots pour le corpus de test.*

Les modèles markoviens continus donnent de meilleurs résultats (Tab. 8.3), ceci est principalement dû au codage de plusieurs laplaciennes par segment. Le TDNN-DHMM code un vecteur de probabilités par segment. Comparé à un CHMM avec une densité par segment, la méthode hybride discrete TDNN-DHMM obtient de meilleures performances avec un codage en entrée plus simple (17 coefficients). On peut cependant difficilement comparer les deux méthodes puisque le codage en entrée est différent.

Les tableaux de résultats suivant regroupent des tests obtenus avec des architectures H-TDNN-DHMM et HT-TDNN-DHMM dont les TDNNs ont été entraînés sur les données en continu. Le codage des transitions élimine en particulier les nombreuses trames de sorties "aléatoires" produitent par les parties du signal jamais apprises. Les vecteurs d'entrée des TDNNs sont composés de 63 coefficients issus la même analyse que celle utilisée par les CHMMs.

Les performances obtenues sont meilleures avec cette architecture hiérarchique simple sur les données en continu qu'avec l'architecture hiérarchique plus complexe avec trappes n'ayant pas appris les transitions (Tab 8.4). L'amélioration vient principalement du codage des transitions, mais aussi d'un meilleur apprentissage. L'utilisation de vecteurs plus précis en entrée (presque 4 fois plus de coefficients), l'ajout des dérivées premières et secondes, permettent aussi d'améliorer nettement les résultats et de comparer réellement les performances avec celles des CHMMs.

Tableau de performances des architectures H-TDNN-DHMMs et HT-TDNN-DHMMs *(avec les nouvelles stratégies d'apprentissage des TDNNs : gestion des transitions, vecteurs d'entrée de 63 coefficients...)*

Test	*Mode*	*Erreurs/phon.*	*Erreurs/mots perplexité 991*	*Erreurs/mots perplexité 63*
H-TDNN-DHMM	-	33.6	27.3	-
	MC	32.9	25.0	4.7
HT-TDNN-DHMM	-	32.6	25.0	-
	MC	32.2	23.6	4.1

Tab. 8.4 : ***Architectures hybrides TDNN-DHMMs dont les TDNNs résultent d'un apprentissage en continu sur des vecteurs d'entrée de 63 coefficients.*** *On donne le pourcentage d'erreurs sur les phonèmes et les mots pour le corpus de test.*

Le taux d'erreur sur les mots de la meilleure configuration hybride est comparable à celui obtenu avec des CHMMs utilisant environ 8 laplaciennes par segment. Le minimum de différence entre les taux d'erreur avec et sans matrice de confusion est obtenu avec l'architecture la plus optimale HT-TDNN. Il est remarquable de voir que l'algorithme de Viterbi apporte beaucoup moins d'informations supplémentaires de confusion avec un meilleur entraînement des réseaux. La matrice de confusion sert de toute évidence à récupérer la "non-optimalité" des des

réseaux et des architectures modulaires. La même observation peut être faite avec l'utilisation du modèle de langage. La configuration HT-TDNN apporte aussi beaucoup moins d'amélioration des résultats avec la nouvelle stratégie d'apprentissage des réseaux. Sachant que l'entraînement des réseaux avec trappes prend au moins 8 fois plus de temps calcul que pour l'architecture plus simple H-TDNN, les améliorations observées pour ces tests ne justifient pas la dépense de temps calcul. Pour effectuer des tests sur deux autres locuteurs, nous avons utilisé la structure hybride H-TDNN-DHMM. Le tableau suivant (tab 8.5) regroupe ces tests (les données de chaque locuteur ont été entraînées indépendamment).

Locuteur	*Sous réseaux*	*Macro classe*	*Erreurs/mots perplexité 991*	*Erreurs/mots perplexité 63*
JWS0	9.9	28.9	25.0	4.7
CMR0	14.8	29.0	20.7	4.5
BEF0	13.7	34.2	36.4	8.9

Tab. 8.5 : ***Taux d'erreur sur l'ensemble de test (100 phrases) pour trois locuteurs avec les sous-réseaux (en moyenne), le réseau de macro-classe et le système hybride H-TDNN-DHMM (avec matrice de confusion).*** *Les TDNNs ont été entraînés avec les nouvelles stratégies d'apprentissage. On donne le pourcentage d'erreurs sur les mots pour le corpus de test avec (perplexité 63) et sans grammaire (perplexité 991).*

Les résutats obtenus sont très dépendants des performances du réseau de macro-classe pour cette architecture (H-TDNN), on le remarque clairement pour le locuteur BEF0.

8.3.4 Systèmes hybrides continus: TDNN-CHMM

Un défaut important des réseaux de type MLP est de ne générer qu'une sortie par classe. Les réseaux RBFs ou LVQ n'ont pas ce défaut lié à un apprentissage discriminant global. Un des moyens de pallier ce problème est d'utiliser une LVQ sur des représentations issues des TDNNs, on obtient alors plusieurs références neuronales par classe. Cette méthode fait partie d'un système hybride TDNN-LVQ-DTW alliant en fait les deux modèles neuronaux à apprentissage global et local à un algorithme de programmation dynamique. Elle est développée au LRI par X. Driancourt et P. Gallinari [Driancourt 92].

Une autre approche est de faire une quantification vectorielle des sorties des réseaux avec un algorithme des k-moyennes et d'utiliser un algorithme de split pour modéliser plusieurs distributions des formes. Une intégration dans un HMM continu permet de modéliser les sorties des TDNNs de façon précise en associant une ou plusieurs densités de probabilités à chaque segment des modèles.

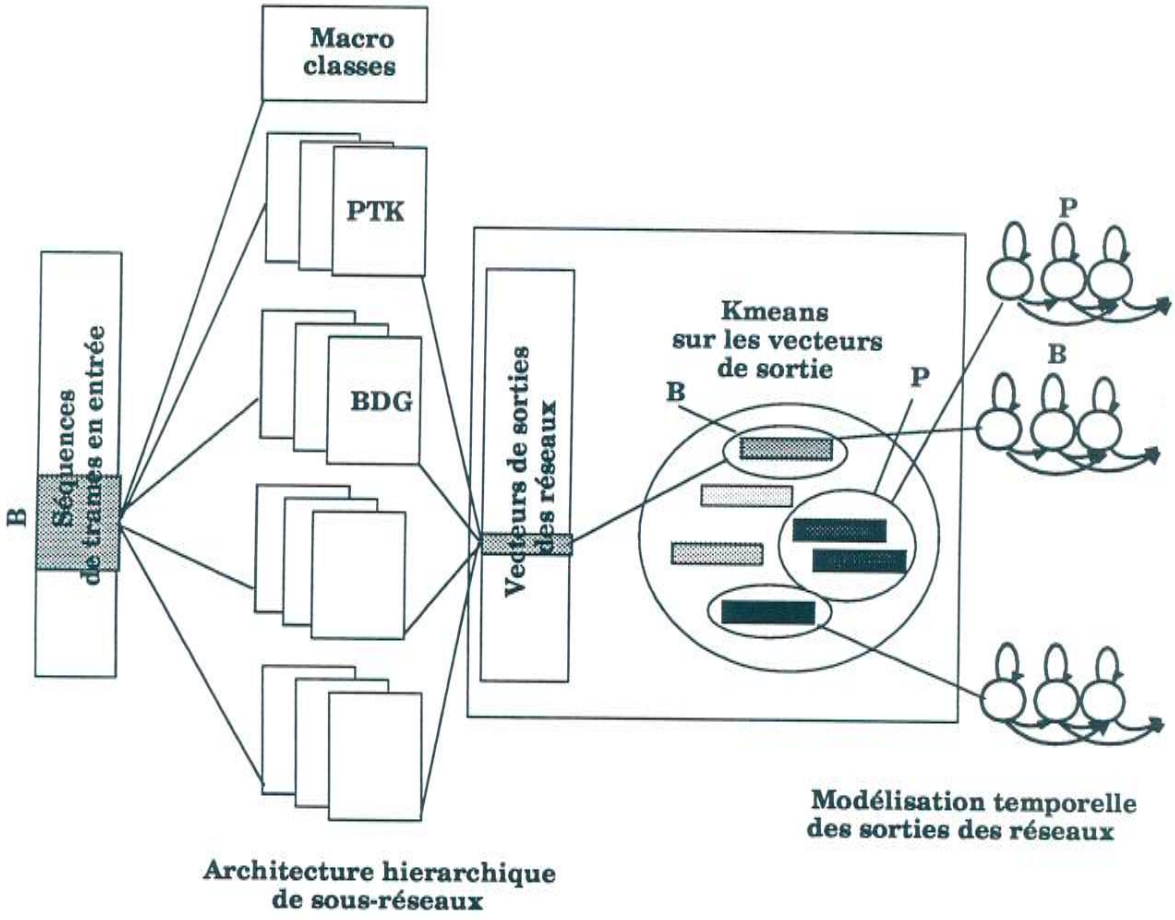

Fig. 8.9 : ***TDNN-CHMM.***

Cette deuxième intégration utilise donc les sorties des réseaux comme des vecteurs d'observation pour un HMM. Il était intéressant de construire cette intégration avec les premiers réseaux développés qui n'avaient pas appris les transitions entre phonèmes. L'apprentissage HMM permet alors de coder les sorties "aléatoires" du réseau correspondant aux transitions entre les phonèmes.

L'apprentissage des modèles est pratiquement le même que dans le cas des CHMMs avec une initialisation différente. La segmentation initiale utilisée est une segmentation issue de l'apprentissage de CHMMs seuls. Une quantification vectorielle, de type k-moyennes, est appliquée sur les sorties des TDNNs en fonction de cette segmentation.

Pour le système hybride TDNN-CHMM, les algorithmes de Viterbi, de "split" et des k-moyennes sont réitérés jusqu'à obtenir le nombre de densités désirées.

Pour le TDNN-CHMM, on calcule pour chaque état j de chaque modèle phonétique la probabilité $p(x_i/j)$ d'être dans cet état. Soit k le nombre de densités par segment :

$$-\log(p(x_i/j)) = \min_{k \text{ densités}} \left(-\log C_k - \log A(v) + \sum_c \frac{(x_{i,c} - m_c(k))}{v_c}\right)$$

8.3.5 Comparaison entre les systèmes hybrides discrets et continus et CHMMs sur la base DARPA

Parmi les architectures TDNNs, la structure HT-TDNN a été testée avec les différentes intégrations, utilisant des scalaires ou des vecteurs. Les réseaux de la structure hiérarchique ont été entraînés sur les données segmentées sans apprendre les transitions entre phonèmes. L'analyse du signal est la même pour tous les systèmes hybrides (17 coefficients par vecteur). Les CHMMs sont les modèles cités au §8.3.4. Les systèmes continus sont comparés avec le même nombre de densité.

Tableau de performances de l'architecture HT-TDNN intégrée de différentes façons dans un HMM et des CHMMs *(TDNN : vecteurs d'entrée de 17 coefficients FFT, non apprentissage des transitions. CHMM: vecteurs d'entrée de 63 coefficients)*

Test HMM continus	***Densités***	***Erreurs/mots Perplexité 991***	***Erreurs/mots Perplexité 63***
HT-TDNN-CHMM	116	16.3	2.7
CHMM	116	12.2	2.9
CHMM	210	11.5	2.2

Tab. 8.6 : ***Comparaison entre les méthodes hybrides discretes et continues et les CHMMs avec le même nombre de densités par segment pour les modèles continus.*** *Pourcentage d'erreurs sur les mots pour le corpus de test sans grammaire (perplexité 991) et avec grammaire (perplexité 63).*

On obtient des résultats beaucoup plus intéressants dans cette expérience pour l'architecture HT-TDNN avec une intégration dans un HMM continu (2.7% d'erreur au lieu de 17.8% avec l'intégration discrète). On peut ainsi modéliser les transitions entre phonèmes qui n'avaient pas été apprises dans les réseaux. La variabilité présente dans les sorties des réseaux au niveau des transitions permet de construire des "*mixtures densities*". A nombre de densités égales, les CHMMs obtiennent une meilleure performance de classification sur les mots sans modèle

de langage tandis que le modèle hybride est un peu meilleur avec le modèle de langage. Le codage des réseaux n'apporte pas de meilleures performances par rapport au codage réalisé par les CHMMs. Les CHMMs sont clairement plus performants lorsqu'on utilise plus de densités. Dans le cas du système hybride, on ne peut pas modéliser les entrées (en l'occurence les sorties du HT-TDNN et principalement les vecteurs correspondant aux transitions) avec plus de laplaciennes.
Les réseaux obtenus avec les nouvelles stratégies d'apprentissage dont notamment l'apprentissage des transitions et un prétraitement utilisant 63 coefficients dont les dérivées premières et secondes (cf chapitre 6) sont plus performants que les réseaux utilisés dans ces tests. On ne peut plus modéliser les sorties par de nombreuses densités, les sortie des réseaux sont plus proches de véritables probabilités.

Tableau de performances des architectures H-TDNN et HT-TDNN intégrées de différentes façons dans un HMM et des CHMMs *(TDNN : vecteurs d'entrée de 63 coefficients, apprentissage des transitions. CHMM: vecteurs d'entrée de 63 coefficients)*

Test HMM discrets	***Mode***	***Erreurs/mots Perplexité 991***	***Erreurs/mots Perplexité 63***
H-TDNN-DHMM	MC	25	4.7
Test HMM continus	***Densités***	***Erreurs/mots Perplexité 991***	***Erreurs/mots Perplexité 63***
H-TDNN-CHMM	2	44	12.5
H-TDNN-CHMM	4	41	16
CHMM	116	12.2	2.9

Tab. 8.7 : ***Comparaison entre les méthodes hybrides discretes et continues et les CHMMs.*** *Pourcentage d'erreurs sur les mots pour le corpus de test sans grammaire (perplexité 991) et avec grammaire (perplexité 63).*

Dans cette expérience, l'intégration discrète montre à l'inverse des tests menés jusqu'à présent une meilleure performance qu'une intégration continue. Les transitions entre phonèmes ont cette fois-ci été apprises dans les réseaux ce qui explique en grande partie l'impossibilité de modéliser les sorties par de nombreuses densités. L'apprentissage des sous-réseaux est aussi plus optimal et les sorties générées sont plus proches de probabilités effectives.

Brièvement en conclusion sur cette partie correspondant à l'utilisation des réseaux comme prétraitement des modèles markoviens, on peut noter que les derniers

résultats obtenus avec des méthodes hybrides TDNN-DHMM sont très encourageants, ils résultent surtout d'architectures modulaires hybrides relativement simples.

8.4 Coopération des scores TDNN et HMM

8.4.1 Principe et but recherché

Un exemple de coopération entre TDNN et HMM est décrit dans la figure 8.10. Il s'agit pour cet exemple de l'architecture hybride TDNN-CHMM/CHMM.

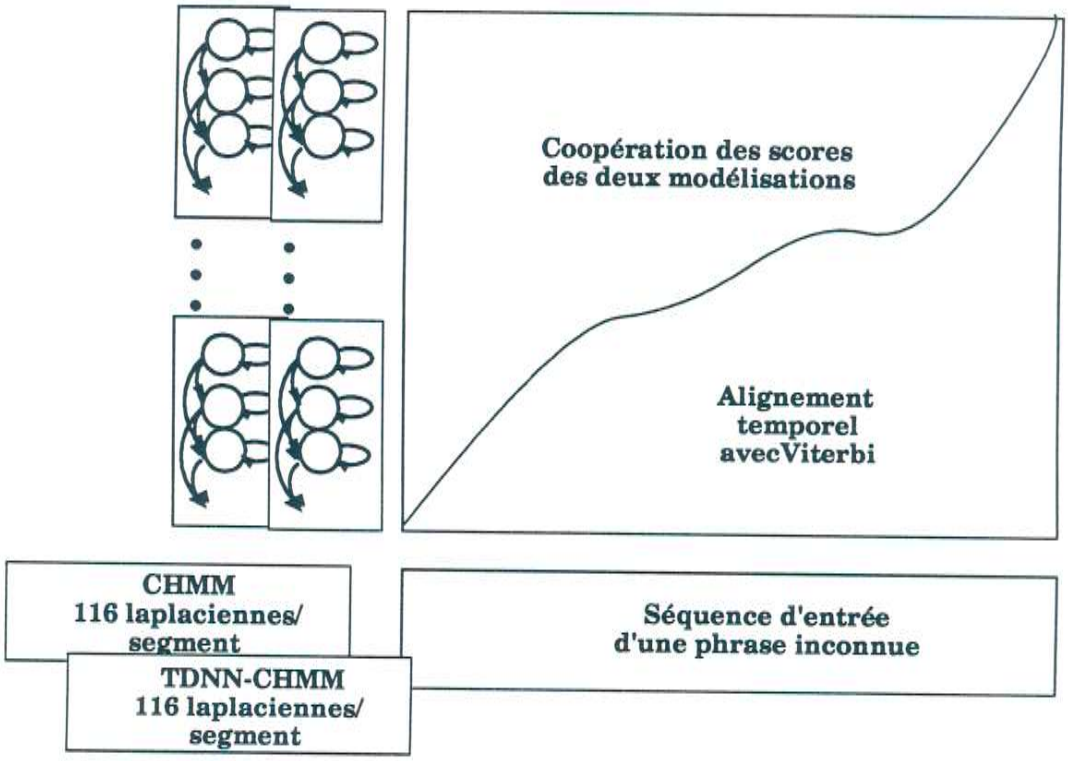

Fig. 8.10 : ***Schéma de la coopération au niveau de la reconnaissance des CHMMs, et des TDNN-CHMMs.***

La coopération des probabilités issues des modélisation TDNN et CHMM se fait au niveau de la reconnaissance. L'idée était de mesurer l'apport des connaissance extraites par les réseaux. Renals, Morgan et al [Renals 92] présente une approche semblable mais à partir d'un système hybride MLP/HMM, cf chapitre 4.

Lors du calcul du meilleur chemin avec l'algorithme Viterbi, on associe les deux modélisations très simplement.

La règle d'intégration utilisée pour un état q_j avec une coopération TDNN-CHMM/CHMM est :

$$\log(P(x/q_j)) = \lambda_1 \log(P_{tdnn\text{-}chmm}(x/q_j)) + \lambda_2 \log(P_{hmm}(x/q_j))$$

La règle d'intégration utilisée pour une coopération TDNN/CHMM est la suivante :

$$\log(P(x/q_j)) = \lambda_1 \log\frac{(P_{tdnn}(q_j/x))}{P(q_j)} + \lambda_2 \log(P_{hmm}(x/q_j))$$

Les λ_i sont empiriquement optimisés avec comme seule contrainte que leur somme soit égale à 1. Les λ_i peuvent être automatiquement évalués par une méthode de *"Deleted interpolation"*.

8.4.2 Résultats sur DARPA

Pour cette expérience, les valeurs des λ_i sont égales à 0.5. Le tableau 8.8 regroupe les résultats obtenus avec une coopération TDNN-CHMM/CHMM.

TEST	***Densités***	***Erreurs/mots Perplexité 991***	***Erreurs/mots Perplexité 63***
HT-TDNN-CHMM	116	16.3	2.7
CHMM	116	12.2	2.9
HT-TDNN-CHMM + CHMM	116 + 116	11.0	1.3
CHMM	210	11.5	2.2

Tab. 8.8 : ***Coopération des scores TDNN-CHMM et CHMMs.*** *Pourcentage d'erreurs sur les mots pour le corpus de test avec et sans modèle de langage.*

Au niveau des mots sans modèle de langage, la coopération n'apporte pas beaucoup d'informations, par contre au niveau des résultats avec modèle de langage, on obtient une baisse de 40% du taux d'erreur par rapport au CHMM le plus performant et de 50% par rapport au CHMM utilisant 116 densités/ segment.

Le tableau suivant (Tab. 8.9) détaille les erreurs des deux systèmes mis en compétition, le CHMM avec 116 densités/segment et le système faisant coopérer les scores des modèles HT-TDNN-CHMM et CHMM. Les TDNNs utilisés n'ont pas appris les transitions et ont codé des vecteurs de 17 coefficients en entrée.

TEST	*Erreurs/mot Perplexité 63*	*Insertions*	*Suppressions*	*Substitutions*
HT-TDNN-CHMM / CHMM	1.3	0	4	7
CHMM	2.9	5	10	15

Tab. 8.9 : ***Comparaison entre les résutats obtenus par le système HT-TDNN-CHMM/CHMM et le CHMM utilisé seul au niveau des insertions, suppressions et susbstitutions.***

Voici des exemples de substitutions qui n'apparaissent plus avec la combinaison des scores : Lats -> last, conquest's -> conquest a, from -> for, last -> lat, of -> two, since -> and,16 -> 60, the -> all, their -> the.

Le mot "*the*" qui était 4 fois élidé ne l'est plus.

Ces exemples montrent que le système hybride a appris des connaissances différentes du système CHMM. Il peut notamment rattraper de nombreuses erreurs dues à des substitutions grâce au codage discriminant des réseaux.

Ce résultat a été vérifié avec une architecture plus simple: la coopération entre le réseau hybride H-TDNN et les CHMMs. Pour cette expérience, les valeurs des λ_i sont de 0.1 pour les scores des réseaux et de 0.9 pour les CHMMs. D'autres expériences faisant varier le degré de coopération des deux systèmes sont en cours.

Les résultats obtenus pour trois locuteurs avec ce réseau hybride sont donnés dans le tableau 8.10.

Locuteur	*H-TDNN-DHMM*	*CHMM 116densités/seg.*	*H-TDNN + CHMM*
JWS0	4.2	2.9	1.8
CMR0	4.5	2.5	2.2
BEF0	8.9	4.2	3.5
Moyenne	6.0	3.2	2.5

Tab. 8.10 : ***Coopération des scores TDNN et CHMMs.*** *Pourcentage d'erreurs sur les mots pour le corpus de test avec modèle de langage.*

La coopération des deux systèmes permet d'obtenir un taux d'erreur en moyenne de 2.5% au lieu de 3.2% avec les CHMMs utilisés seuls avec un système beaucoup moins onéreux que la coopération décrite précédemment (les taux d'erreur donnés sont sur les mots en utilisant la grammaire DARPA). Ces scores sont très encourageants pour la coopération des deux systèmes. L'architecture neuronale utilisée est très simple, elle comprend seulement 9 sous-réseaux et un réseau de macro-classe, soit environ 40000 poids. Le problème le plus critique reste cependant le temps d'apprentissage des réseaux sur un ordinateur mono-processeur. Avec une puissance de calcul de 20MIPS, le temps total d'apprentissage de la structure H-TDNN est d'environ un mois (soit 15 jours pour le réseau de macro-classes et 2 jours pour chaque sous-réseau) contre un jour pour les CHMMs. Au contraire pour la reconnaissance, sur une machine de 20MIPS, les probabilités des réseaux sont calculées en temps réel tandis que les CHMMs nécessitent quatre fois le temps réel.

En conclusion, cette expérience montre l'intérêt du codage discriminant des réseaux utilisé en coopération avec des HMMs. Malgré la "non-optimalité" des architectures modulaires, des informations complémentaires des HMMs ont été codées par les réseaux. Ces informations sont en particulier des discriminations fines entre des phonèmes acoustiquement proches qui ont été codés par les sous-réseaux.

Les systèmes basés sur des modèles markoviens sont maintenant largement utilisés avec des apprentissages discriminants suivant une phase préalable d'apprentissage avec un algorithme EM. Il serait intéressant de montrer ce qu'apporte une architecture neuronale par rapport à un algorithme de "*corrective training*" ou de MMI. La mise en oeuvre de ces procédés n'est cependant pas toujours facile et l'utilisation de réseaux de neurones peut être un moyen plus aisé d'ajouter localement (ici grâce au codage des sous-réseaux) des informations discriminantes aux HMMs.

8.5 Conclusion

Dans l'état actuel du domaine de recherche et de nos travaux, nous pouvons uniquement avancer des conclusions partielles sur ce type de techniques. Nous résumons dans cette conclusion les principaux résultats obtenus.

Nous avons obtenu des résultats très intéressants avec une architecture hybride faisant coopérer les probabilités issues des réseaux et de modèles markoviens

CHMMs lors de la reconnaissance. Pour les expériences menées sur trois locuteurs de DARPA, cette coopération des probabilités a diminué de 20% le taux d'erreur des CHMMs utilisés seuls qui était déjà de 2.5%. Ce résultat est particulièrement encourageant car il a été obtenu avec une architecture modulaire très simple (H-TDNN). Ces expériences montrent que le pouvoir de discrimination apporté par les réseaux permet de rattraper des erreurs des CHMMs. La majorité des erreurs corrigées correspondent à des substitutions de mots courts telles que lats (last), their (the)... phonétiquement très proches.

Les différents systèmes hybrides TDNN-HMMs testés, où les probabilités des réseaux sont directement utilisées dans un formalisme markovien n'ont pas donné de meilleures performances que les CHMMs. Parmi les architectures modulaires proposées, l'architecture modulaire "hiérarchique" liant les scores des réseaux lors de la reconnaissance est la plus intéressante. L'architecture hiérarchique avec trappes permet d'être moins dépendant des performances de la macro-classe. Cependant, le bénéfice de cette architetcure n'est pas justifiée si l'on tient compte du temps de calcul nécessaire pour l'apprentissage de tous les sous-réseaux.

Les premiers réseaux développés, n'ayant pas appris les transitions entre phonèmes, ont été utilisés dans des intégrations HMM discrètes et continues. Pour la même architecture HT-TDNN, l'utilisation d'un HMM continu a permis de modéliser les transitions entre les phonèmes avec de nombreuses densités et d'obtenir de meilleures performances qu'avec une intégration discrète. Cette tendance s'est inversée avec les réseaux construits plus récemment qui ont appris les transitions entre phonème lors de l'apprentissage neuronal et ont bénéficié d'un apprentissage plus optimal. De réelles probabilités sont maintenant générées en sortie des réseaux.

Dans nos systèmes, les différents modules ont été entraînés séparément et leurs sorties combinées pour produire des systèmes hybrides. Ces systèmes pourraient être améliorés par une optimisation globale des paramètres. La poursuite de nos travaux devrait aller dans ce sens.

En conclusion de cette partie "systèmes hybrides", on peut répondre à la question posée dans l'introduction. Les réseaux de neurones ne peuvent à l'heure actuelle remplacer des méthodes comme les HMMs qui bénéficient de nombreux mécanismes très performants comme la gestion intégrée des connaissances acoustiques, syntaxiques... et de vingt années d'expériences. En revanche, tout comme d'autres méthodes basées sur la minimisation du critère d'erreur, les méthodes neuronales peuvent améliorer les performances des HMMs entraînés

avec des algorithmes EM. Les résultats actuels montrent que la coopération de ces deux méthodes dans des systèmes hybrides est une voie de recherche très prometteuse.

Conclusion

9. Conclusion de l'étude

Ce travail regroupe un ensemble d'études sur l'utilisation des réseaux de neurones en parole. Les premières expériences qui ont été menées alors que le domaine en était à ses débuts, ont permis de mettre au point des algorithmes et des architectures de réseaux de neurones et de sélectionner les plus efficaces pour la parole.

Le traitement de la parole nécessite un traitement des séquences et une prise en compte des distorsions temporelles. Afin de traiter des applications en reconnaissance de parole continue, nous avons développé des systèmes hybrides alliant les capacités de discrimination des réseaux et la gestion temporelle ainsi que les mécanismes d'intégration des HMMs. Les problèmes inhérents à l'apprentissage des réseaux ont été résolus par des architectures modulaires de sous-réseaux. Nous avons conçu différents types d'intégration de ces sous-réseaux dans des architectures globales qui ont permis de tester de grandes bases de données.

Les systèmes hybrides et modulaires sont certainement une voie de recherche prometteuse. Notre travail a été une exploration de cette voie qui s'est ouverte récemment et qui suscite l'intérêt de nombreux chercheurs à l'heure actuelle. De nombreux travaux sont cependant encore nécessaires pour développer d'autres algorithmes et améliorer les structures actuelles, et pour avoir plus de certitudes quant à l'apport des réseaux et des méthodes hybrides dans le domaine de la reconnaissance de la parole.

Les méthodes connexionnistes sont souvent utilisées dans les modèles hybrides soit comme pré-processeurs, extracteurs d'indices pour un HMM, soit comme correcteurs

d'erreur après un HMM. Peu de résultats montrent cependant à l'heure actuelle de meilleures performances avec ces méthodes neuronales hybrides que celles obtenues avec des systèmes basés sur les modèles markoviens seuls. L'un des l'objectifs de notre étude était justement de construire un système hybride neuronal et markovien, tirant partie des qualités des deux systèmes et offrant de meilleures performances que l'un ou l'autre pris séparément.

Nous avons obtenu des résultats particulièrement intéressants en faisant coopérer lors de la reconnaissance les probabilités issues de modèles neuronaux et markoviens. La coopération des probabilités issues d'un système H-TDNN et de modèles markoviens CHMMs a diminué le taux d'erreur de 20% des mêmes modèles markoviens utilisés seuls pour les trois locuteurs de DARPA testés. Ces résultats sont très encourageants car l'architecture neuronale utilisée est très simple et devrait pouvoir être optimisée. En résumé, les deux systèmes, neuronal et markovien, stockent des informations de type différent qui sont complémentaires. Ces expériences montrent tout l'intérêt du codage discriminant des réseaux utilisés en coopération avec des HMMs.

Ce travail est une étape, il peut se prolonger sur de nombreux axes de recherche dont les suivants :

- L'étude d'algorithme comme la rétropropagation dans le temps pour entraîner des réseaux multi-couches récurrents afin de mieux gérer les distorsions temporelles,
- le découpage automatique de la tâche de classification en sous-tâches et l'apprentissage coopératif dans une architecture modulaire neuronale,
- l'étude de coopérations optimales des algorithmes d'apprentissage utilisés dans les systèmes hybrides.

Le travail mené au cours de cette thèse a été à la fois conceptuel, théorique et pratique. Nous avons conçu de nouvelles structures modulaires et intégrations dans les HMMs et mis en oeuvre de nombreuses architectures modulaires et hybrides. Des comparaisons entre les systèmes hybrides et des approches purement markoviennes très performantes développées par Philips ont été effectuées. Ces tests ont été menés sur la base de données DARPA RM nécessitant des temps de calcul assez importants mais permettant de situer les performances réalisées au niveau international. Cette étude a permis d'obtenir de très bons résultats de reconnaissance avec des systèmes hybrides et de montrer la complémentarité des probabilités calculées par les HMMs et les réseaux.

Annexes
et
Références
bibliographiques

Annexe 1

Base de données en français

Cette base de données a été enregistrée au LIMSI par M.Adda pour sa thèse [Adda 88]. L'intérêt de ce corpus malgré sa petite taille est d'être un corpus de parole continue comportant toutes les suites de consonnes-voyelles et voyelles-consonnes du français. Deux débits de parole sont utilisés, un débit lent et normal.

Les données proviennent d'un signal échantillonné à 10KHz, analysé par une FFT toutes les 12.8ms, puis filtré sur 16 canaux répartis suivant une échelle de Bark (100Hz - 5000Hz). Chacune des seize valeurs du vecteur aussi appelé trame est codée suivant une échelle logarithmique sur 8 bits. Puis, l'énergie moyenne est retranchée à chaque coefficient et rajoutée comme 17ième coefficient du vecteur. Les valeurs de ces vecteurs sont normalisées entre -1 et 1 afin d'être homogènes avec les bornes choisies pour les valeurs d' activité des cellules du réseau.

Le corpus a été étiqueté et segmenté manuellement, l'unité phonétique choisie est le phone. L'ensemble de ces phonèmes comprend 13 voyelles, 17 consonnes et une unité correspondant au silence. La longueur moyenne des phonèmes est de 6 trames.

Cette base est intéressante pour vérifier la robustesse d'un système aux distorsions temporelles. Elle a été scindée en deux parties qui servent à l'apprentissage et au test. La partie test comprend des contextes phonétiques inconnus de la partie apprentissage ce qui nous permettra de vérifier le pouvoir de généralisation du modèle d'apprentissage.

Pour un locuteur, nous disposons d'un corpus de logatomes alternant consonnes et voyelles dans différents contextes de type /apataka/, d'un corpus de 50 phrases de parole continue à débit naturel (environ 11 phonèmes par secondes) et de ces mêmes 50 phrases à débit lent (environ 8 phonèmes par secondes). Ces phrases contiennent l'inventaire exhaustif des diphones CV (consonnes - voyelles) et VC du français.

Ce corpus recouvre une grande diversité phonémique en ce sens que les phonèmes sont issus de corpus très différents; naturel (parole continue) et artificiel (logatomes et parole à débit lent) mais il n'y a souvent qu'une occurrence pour chaque contexte

phonétique. Le corpus est de taille réduite; la base d'apprentissage comporte 2173 segments phonétiques, celle de test 1964 segments.

Le corpus d'apprentissage correspond à environ 60% du corpus mixte (2/3 des logatomes, la première moitié du corpus des phrases à débit normal, la deuxième moitié du corpus de phrases à débit lent) et les 40% restant servent de corpus de test.

La construction de ce corpus mixte a été motivée par plusieurs raisons; le corpus des logatomes ne recouvre pas tous les contextes existant dans le corpus de phrases et la différence de débit de parole entraîne des distorsions temporelles et fréquentielles. Un corpus mixte permet d'avoir une représentation plus robuste des phonèmes.

Les tableaux suivants correspondent aux symboles phonétiques des 31 classes utilisées en norme LIMSI et API:

Code API	Code LIMSI	Exemples
p	P	papa
t	T	tango
k	K	cou
b	B	bravo
d	D	delta
g	G	golf
f	F	photo
s	S	nation
ʃ	X	chat
v	V	rêve
z	Z	rose
ʒ	J	je
m	M	matin
n	n	animal
ɲ	%	agneau
l	L	lapin
r	R	carotte
silence	#	-

Code API	Code LIMSI	Exemples
ɔ	+	donner
a	A	patte
ɑ̃	*	sans
i	I	vie
y	U	vêtu
e	)	jouer
ɛ	(	jouet
œ	E	peur
ø	=	peu
o	O	mot
ɔ̃	/	bon
u	W	roue
ɛ̃	<	calin

Annexe 2

Base de données en américain *DARPA RM1 speaker-dependent*

La base "*DARPA Ressource Management 1, Speaker Dependent*" contient 1000 mots de vocabulaire et a été enregistrée pour la recherche militaire. Elle comporte 12 locuteurs ayant chacun prononcé 700 phrases de parole continue issue de textes lus dans un environnement non bruité. Elle est échantillonnée à 16kHz, disponible sur disques compacts, avec une syntaxe de type "*word-pair grammar*" (grammaire bi-grams de mots) et comporte un jeu de test permettant d'évaluer les résultats obtenus avec de nombreux systèmes.

Les expériences ont été menées trois locuteurs JWS0, CMR0 et BEF0. Le corpus d'apprentissage est de 600 phrases par locuteur, soit 23000 segments phonétiques et 160000 trames. la durée des segments est en moyenne est de 7 trames. Le corpus de test comporte 100 phrases, soit 3500 segments. les différents contextes phonétiques (triphones) de la base d'apprentissage sont de l'ordre de 2500.

Les phrases sont traduites phonétiquement à l'aide de 47 symboles phonétiques mais ne sont pas étiquetées sur le signal. Les tailles des segments phonétiques sont très variables.

Les phonèmes les plus courts (1 à 2 trames) sont pour les voyelles le "schwa" /ax/ et /ix/ et pour les consonnes le "flap" /dx/ et les phonèmes qui peuvent être élidés. Les exemples marqués d'un astérix dans le tableau suivant sont des cas de mots où certains phonèmes peuvent être élidés en position terminale. Des classes spéciales ont été créées pour ces phonèmes, il existe les classes /pd/, /td/, /kd/ et /dd/.

Au contraire, certains phonèmes comme les diphtongues /ay/, /oy/, /aw/... sont souvent très longs (15 trames).

Le débit d'élocution est normal, cependant très rapide pour un non anglophone. Les phonèmes sont particulièrement instables comparés à ceux du français. la base de données a été segmentée avec des modèles markoviens continus développés par Philips (Aix-la-Chapelle) qui donne environ 2% d'erreur sur les mots de la base de données (les HMMs sont décrits dans le chapitre 8).

Liste des symboles phonétiques de DARPA

API	DARPA	exemple
w	w	wet
r	r	red
l	l	let
y	y	yet
s	s	see
v	v	view
ʃ	sh	she
f	f	free
θ	th	thief
z	z	zeal
ð	dh	thee
p	p	pea
t	t	tea
k	k	key
b	b	bee
d	d	deal
g	g	geese
m	m	meet
n	n	neat
ŋ	ng	sing
ʧ	ch	church
ʤ	jh	judge
h	hh	heat
2 (flap)	dx	edit
silence	si	-
d^	dd	edited(*)
p^	pd	group(*)
t^	td	edit(*)
k^	kd	frederick(*)

API	DARPA	exemple
i	iy	beat
I	ih	bit
e	ey	bait
ɛ	eh	bet
æ	ae	bat
ɑ	aa	bob
ɔ	ao	bought
ʌ	ah	but
o	ow	boat
U	uh	book
u	uw	boot
3^	er	burst
ə	ax	about
ɨ	ix	roses
a^{y}	ay	bite
$ɔ^{y}$	oy	boy
a^{w}	aw	about
n̩	en	didn't

Annexe 3

Validité statistique

Pour tous les tests de reconnaissance, il est préférable de préciser un intervalle de confiance qui donne une mesure de la validité statistique des résultats obtenus.. Determiner cet intervalle est important pour comparer les performances avec différents classifieurs.

Il indique qu'il y a x% de chance que le taux de reconnaissance obtenu soit valide statistiquement. Il dépend du taux de reconnaissance (P), du taux d'erreur (Q) et du nombre de test effectué (N) et du niveau de confiance α.

Si on suppose que le taux d'erreur suit une distribution normale, on calcule la précision r qui permet d'indiquer l'intervalle de confiance à α%: [P-r, P+r], P étant le taux de reconnaissance. Nous avons utilisé la formule suivante [Montacie,Chollet 87] :

$$r = Z_\alpha * \sqrt{\frac{P * Q}{N}}$$

pour $\alpha = 95\%$ $Z_\alpha = 1.96$

Rappel

Etant donné une variable aléatoire X dont on connait la distribution dans une population E. Déterminer l'intervalle de confiance revient à construire un intervalle $[B_{inf}, B_{sup}]$ qui contienne toute valeur X tirée au hasard dans E avec une probabilité $1-\alpha$ (α étant le risque).

Ceci revient à déterminer la quantité positive Z_α tel que :

$$P[-Z_\alpha < X < +Z_\alpha] = 1 - \alpha$$

Z_α se déduit simplement de la table, par exemple, de la loi Normale.

Bibliographie

[**Abramowitz 64**]

Abramowitz, Milton, Stegun, Irene A. : "*Handbook of Mathematical Functions*", Applied Mathematics Series, vol. 55, réédité 1968 par "Dover Publications", New York, 1964.

[**Ackley 85**]

Ackley D.H., Hinton G.E., Sejnowski T.J.: *A learning algorithm for Boltzmann machines* - Cognitive Science, 9, pp 147-169, 1985.

[**Adda-Decker 88**]

Adda-Decker Martine : "*Evaluation d'unités de décision pour la reconnaissance de la parole continue*", Thèse de 3ème cycle, Orsay, 1988.

[**Andreewsky 87**]

Andreewsky, A., Devillers L., Ringot P. : "*Segmentation et reconnaissance en parole continue à l'aide de références issues du système VARAP : Validation Automatique de Références en APprentissage*", XIth International Congress of Phonetic Sciences, Tallin (URSS), 1987.

[**Austin 92**]

Austin S., Zavaliagkos, G., Makhoul, J., Schwartz, R. : "*Speech Recognition using segmental neural nets*", ICASSP, vol. I-625, San Francisco, 1992.

[**Baker 75**]

J.K. Baker: *"The Dragon System - An overview"*, IEEE Trans. on Acoustics, Speech and Signal Processing, Vol. ASSP-23, pp 24-29, Feb. 1975.

[**Bedworth 89**]

M.D. Bedworth, L. Bottou, J.S Bridle, F. Fallside, L. Flynn, F. Fogelman, K.M. Ponting, R.W. Prager :"Comparison of neural and conventional classifiers on a speech recognition problem", IEEE, 1989.

[**Bengio 91**]

Bengio. J, De Mori R., Flammia G., Kampe R. :"*Artificial Neural Networks and their Application to Sequence Recognition*", thèse de doctorat, Université McGill, Montréal, 1991.

[**Bengio 91**]

Bengio. J, De Mori R., Flammia G., Kampe R. :"*A comparative study on hybrid acoustic phonetoc decoders based on artificial neural networks*", Eurospeech 91, Gênes, Italie, 1991.

[Bengio 91]
Bengio. J :"*Phonetically motivated acoustic parameters for continuous speech recognition using artificial neural networks*", Eurospeech 91, Gênes, Italie, 1991.

[Béroule 85]
D. Béroule : "Un Modèle de Mémoire Adaptative, Dynamique et Associative pour le Traitement Automatique de la Parole", Thèse 3ème cycle, Orsay, 1985.

[Biem 91]
A. Biem : "*Study on combining HMM and Neural Networks for Speech Recognition*", rapport ATR, 1991.

[Bimbot 91]
F. Bimbot, G. Chollet, J. P. Tubach : "*TDNNs for phonetic features extraction a visual exploration*", Eurospeech 91, Gênes, Italie, 1991.

[Blanchet 89]
P. Blanchet : "*Multilayer Perceptron Architectures for Data Compression Tasks*", Proc. Eurospeech, Paris, Septembre 1989.

[Bodenhausen, Waibel 91]
U.Bodenhausen, A.Waibel :"*The tempo 2 algorithm : adjusting time-delays by supervised learning*". Advances in Neural Information Processing Systems vol. 3, pp155-161, 1991

[Boiteau 92]
D. Boiteau :"*Classification segmentale de voyelles avec des unités gaussiennes ou sigmoïdales*", JEP92 Bruxelles, 1992.

[Bonneau 87]
H. Bonneau : "Quantification vectorielle et adaptation au locuteur", Thèse de doctorat en sciences, Université de Paris 11, 1987.

[Booker 89]
L. B. Booker, D. E. Goldberg, J. H. Holland : "Classifier Systems and Genetic Algorithms", Artificial Intelligence, vol. 40, numbers 1-3, 235-282, Sept. 1989.

[Bottou 91]
Bottou L.: "*Une Approche théorique de l' Apprentissage Connexioniste; Applications à la reconnaissance de la Parole*". Thèse de doctorat Université de Paris Sud, Orsay, 1991.

[Bottou 90]
Bottou L., Fogelman Soulié F., Blanchet P., Liénard J.S.: *Speaker independent isolated digit recognition: Multilayer perceptron vs Dynamic Time Warping*, Neural Networks, vol3, pp 453-465, 1990.

[**Bourlard 92**]

Bourlard, H., Morgan, N., Wooters, C. Renals, S. : "CDNN: *A context dependent neural network for continuous speech recognition*", ICASSP, vol. II-349, San Francisco, 1992.

[**Bourlard 92**]

H. Bourlard :"*Reconnaissance Automatique de la Parole : Modèles Stochastiques et/ou Modèles Connexionnistes*", 19ème JEP, Bruxelles, 1992.

[**Bourlard 91**]

Bourlard, H., N. Morgan, Ch. Wooters :"*Connectionist Approaches to the Use of Markov Models for Speech Recognition*", Advances in Neural Information Processing Systems 3", Morgan Kaufman, 1991.

[**Bourlard 90**]

Bourlard, H., N. Morgan, Ch. Wooters :"*Continuous speech recognition on the Ressource Management database using connectionist probability estimation*", ICSLP90, Kobe, 1990.

[**Bourlard,Wellekens 89**]

H. Bourlard, C.J. Wellekens : *"Links between Markov Models and Multilayer Perceptrons"*; Advances in Neural Information Processing Systems, Vol. 1, pp. 502-510, Morgan Kaufman, Denver 1989

[**Bourlard 87**]

Bourlard, H. : *"Auto-Association by Multilayer Perceptron and Singular Value Decomposition"*, manuscript M217, Philips Research Laboratory, Brussels, Belgium, Novembre 1987.

[**Bridle 90**]

J.S. Bridle :"*Neural networks or Hidden Markov Models for automatic speech recognition : is there a choice?*", proc NATO ASI on Speech Recognition and Understanding, Cetraro, Italy, 1990.

[**Bridle 90**]

J.S. Bridle :"*Training stochastic model recognition algorithms as networks can lead to Maximum Mutual Information Estimation of parameters*", Advances in Neural Information, 1990.

[**Bridle 89**]

J.S. Bridle :"*Alpha-Nets : A recurrent "neural" network architecture with a Hidden Markov Model Interpretation*", DRA-SRU research report, 1989.

[**Brown 87**]

P. F. Brown: "*The Acoustic Modeling problem in Automatic Speech Recognition*", Ph.D. thesis, Carnegie-Mellon University, May 1987.

[**Burnod 89**]

Yves Burnod : "*An adaptative neural network : The cerebral cortex*", Masson, Paris, 1989.

[Calliope 89]
Calliope : *"La parole et son traitement automatique"*, collection technique et scientifique des télécommunications, Masson, Paris, 1989.

[Chow 90]
YL Chow : *"Maximum Mutual Information Estimation of HMM parameters for continuous speech recognition using the N-Best algorithm"*, ICASSP90, p701, 1990.

[DeBollivier 92]
M. De Bollivier : *"Incorporation de modularité dans les réseaux connexionnistes par décomposition de tâche"*, Thèse de doctorat, LRI, Orsay, 1992.

[DeBollivier 90]
M. DeBollivier, P. Gallinari, S. Thiria :"Cooperation of neural nets for robust classification", IJCNN90, pp113-120, 1990.

[Devijver 82]
Devijver P.A, Kittler J.: *"Pattern Recognition: A Statistical Approach"* - Prentice Hall International, 1982.

[Devillers,Dugast 91]
Devillers, L., Dugast, Ch. : *"Comparison of Continuous Mixture Densities and TDNN in a Viterbi-Framework: Experiment on Speaker Dependent Darpa RM1"*, Eurospeech, Genova, Italy, 24-26 Sept. 1991.

[Devillers 90]
Devillers L. : *"Reconnaissance monolocuteur des phonèmes du français au moyen d'une architecture de réseaux à masques temporels"*, JEP, Montréal, Canada, May 1990.

[Devillers 87]
Devillers L. : *"Segmentation automatique de parole continue"* - Rapport de stage, DEA d'Electronique option traitement de l'information, Université de Paris Sud, Orsay, 1987.

[Dugast,Devillers 92]
Dugast, Ch.,Devillers, L. : *"Incorporating acoustic-phonetic knowledge in hybrid TDNN/HMM frameworks"*, ICASSP, vol. I-421, San Francisco, 1992.

[Driancourt 92]
Driancourt, X.D., Gallinari P. : *"A speech reconizer optimaly combining learnng vector quantization, dynamic programming and multi-layer perceptron":* ICASSP, vol. I-609, San Francisco, 1992.

[Driancourt 91]
Driancourt, X.D., Bottou L., Gallinari P. : *"Learning Vector Quantization, Multi-layer Perceptron and Dynamic Programming: Comparison and Cooperation"*, IJCNN 91, vol. 2, pp 815-819, Seattle, 1991.

[**Doutriaux 89**]

A. Doutriaux, D. Zipser :"*Unsupervised discovery of speech segments using recurrent networks*", Proc. Connectionist Models Summer School, Morgan Kaufmann, 1990.

[**Duda 73**]

Duda R.O., Hart P.E. : "*Pattern classification and Scene analysis*", Wiley 1973.

[**Elman 88**]

J.L. Elman :"*Finding Structure in time*", Cognitive Science, vol 14, pp179-211, 1988.

[**Escande 91**]

P. Escande, D. Béroule, P. Blanchet : "*Speech recognition experiments with guided propagation*", IJCNN 91, Singapore, 18-21 Nov. 1991.

[**Fallside 92**]

F. Fallside : "*Issues in speech recognition using neural networks*", Artificial Nreural Networks, 2, Elsevier Science Publishers, 1992.

[**Fallside 90**]

F. Fallside, H. lucke & al :"*Continuous speech recognition for the TIMIT database using neural networks*", ICASSP90, p445, 1990.

[**Flamenbaum**]

G. Flamenbaum, J. Thiery, J.P. Benzecri :"*Agrégation eb boules de rayon fixe et centres optimisés*", les cahiers de l'analyse de données, vol IV.

[**Franzini,Waibel 91**]

M. Franzini, A. Waibel, K.F. Lee :"*Recent work in continuous speech recognition using the connectionist Viterbi training procedure*", ICASSP91, p1213, 1991.

[**Franzini,Waibel 90**]

M. Franzini, K.F. Lee, A. Waibel :"*Connectionist Viterbi Trainig: A new hybrid method for continuous speech reognition*", ICASSP90, P425, 1990.

[**Franzini 89**]

M. Franzini, M. Witbrock, K.F. Lee :"*Speaker-Independent Recognition of Connected Utterances Using Recurrent and Non-recurrent Neural Networks*", IJCNN89, Washington, 1989.

[**Franzini 88**]

M. Franzini :"*Learning to recognizie spoken words: A study in connectionist speech recognition*", Connectionist school at CMU, Morgan Kaufman, 1988.

[**Gallinari 91**]

Gallinari P., Thiria S., Badran F., Fogelman Soulié F.: "*On the Relations between Discriminant Analysis and Multi-layer Perceptrons*"- Neural Networks, vol.4, n° 3, pp 349-360 1991

[Gallinari 88]
Gallinari P., Thiria S., Fogelman Soulié F.: "*Multilayer Perceptrons and Data Analysis*" - Procs of IEEE 2nd ICNN, San Diego, vol I, pp 391-401, 1988

[Garofolo 88]
J.S Garofolo : "*Getting Started with the DARPA TIMIT CD-ROM: An acoustic phonetic continuous speech database*", NIST, 1988.

[Gauvain 92]
J.L. Gauvain, C.H. Lee :"*MAP Estimation of continuous density HMM : Theory and Applications*", Proc. DARPA Speech and Natural Language Workshop, 1992.

[Gauvain 82]
J.L. Gauvain : "*Reconnaissance de mots enchaînés et détection de mots dans la parole continue*", Thèse de 3ème cycle, LIMSI, Orsay, 1982.

[Gear 71]
Gear C., William : "*Numerical Initial Value Problems in Ordinary Diffrential Equations*", Englewood Cliffs, NY, Prentice-Hall, chap 2, 1971.

[Goldberg 88]
D. Goldberg :"*Genetic Algorithms in MAchine Learning, Optimization and Search*". Addison-Wesley, 1988.

[Haeb-Umbach 92]
R. Haeb-Umbach, H. Ney :"*Linear discriminant analysis for improved large vocabulary continuous speech recognition*", ICASSP92, San Francisco, 1992.

[Haffner 92]
P. Haffner :"*Connectionist word-level classification in speech recognition*", ICASSP92, San Francisco, 1992.

[Haffner 91]
P. Haffner, A. Waibel :"*Time-delay neural networks embedding time alignment: a performance analysis*", Eurospeech 91, Gênes, Italie, 1991.

[Haffner 88]
P. Haffner, A. Waibel, H. Sawai, S. Shikano :"*Fast Back-Propagation Learning Methods for Neural Networks in Speech*", ATR report, 1988.

[Hataoka,Waibel 89]
N. Hataoka, A. Waibel : "*Speaker-Independent Phoneme Recognition on TIMIT Database using Intergrated Time-Delay Neural Netaworks (TDNNs)*", CMU-CS-89-190, November 1989.

[Hebb 49]
D. Hebb :"*The Organisation of Behavior*", New York : Wiley, 1949.

[Hebb 61]
D. Hebb ;"*Organization of behavior*", Science Edition, 1961.

[Hinton 86]
G.E. Hinton, D.E. Rumelhart, R.J. Williams : *"Learning internal representation by error propagation"*, in D.E. Rumelhart and J.L. McClelland (eds) Parallel Distributed Processing, MIT Press, vol. 1, chap 8, 1986.

[Hopfield 82]
J.J. Hopfield : "Neural networks and physical systems with emergent collective computational abilities", Proc. Natl. Acad. Sct. USA, Vol. 79, pp. 2554-2558, April 1982.

[Huang, Lippmann 89]
W.Y.Huang, R.P. Lippmann :*"HMM Speech Recognition with Neural Net Discrimination"*, NIPS89, Denver, 1989.

[Huang, Lippmann 87]
W.Y.Huang, R.P. Lippmann :*"Comparison between Neural Net and conventional classifiers"*, IEEE, 1987.

[Iso 90]
K. Iso, T. Watanabe :*"Speaker-Independent Word Recognition Using a Neural Prediction Model"*, IEEE90, 1990.

[Jelinek 76]
F.Jelinek :*"Continuous Speech Recognition by Statistical Methods"*, Proc. of the IEEE, Vol. 64, pp. 532-556, April 1976.

[Jacobs 91]
Jacobs R.A, Jordan M.I., Nowlan S.J., Hinton G.E.: *"Adaptive mixtures of local experts"*. Neural Computation, vol 3, n° 1, pp 79-87,1991.

[Jacobs 90]
Jacobs R.A.: *"Task Decomposition Through Competition in a Modular Connectionist Architecture"*. PhD Dissertation, University of Massachussets at Amherst, 1990.

[Jacobs 90]
Jacobs R.A., Jordan M.I., Barto A.G.: *"Task Decomposition Through Competition in a Modular Connectionist Architecture: the What and Where Tasks"*, Cognitive Science, in Press, 1990.

[Jordan 86]
M. Jordan :*"Attractor Dynamics and Parallelism in a Connectionist Sequential Machine"*, Proc. 8ème conférence -"Cognitive Science Society", pp531-546, 1986.

[Jouvet 88]
D. Jouvet :*"Reconnaissance de Mots Connectés Indépendamment du Locuteur par des Méthodes Statistiques"*. PhD thesis, Ecole Normale Supérieure des Télécommunications, Juin 1988.

[Kohonen 88]
Kohonen T., Barna G., Chrisley R. : "*Statistical Pattern Recognition with neural networks: Benchmarking Studies*" - Second International Conference on Neural Networks, San Diego, IEEE proc. of ICNN, vol I, pp 61-68, 1988.

[Kohonen 84]
T. Kohonen : "*Self organization and associative memory*", series in Information Sciences, vol. 8, Springer-Verlag, Berlin-Heidelberg-New York, Tokyo, 1984.

[Kubala 88]
E. Kubala, Y. Chow et al :"*Continuous speech recognition results of the Byblos system on the Darpa 1000 word Ressource Management database*", ICASSP88, p291, 1988.

[Kuhn 90]
G. Kuhn, R. L. Watrous, & B. Ladendorf :"*Connected Recognition with a Recurrent Network*", Speech Communication, vol. 9, pp41-48, 1990.

[Lang 88]
Lang K., Hinton G. : "*the development of the Time Delay Neural Network Architecture for Speech Recognition*", Carnegie Mellon University TR CMU-CS-88-152, 1988.

[Le Cun 87]
Y. LeCun : "*Modèles Connectionistes de l'Apprentissage*"- Thèse de doctorat de l'Université Pris VI, 1987.

[Lee 90]
K.F. Lee, H.W. Hon et al :"*Recent progress and future outlook of the SPHINX speech recognition system*", Computer Speech and Language, 57-69, 1990.

[Lee 89]
K.F. Lee, H.W. Hon :"*Speaker-independent phone recognition using HMMs*", IEEE Transactions on Acoustics, Speech and Signal Processing, 37, 1641-48,1989.

[Lee 88]
K.F. Lee : "*Large Vocabulary Speaker-Independent Continuous Speech Recognition: The SPHINX System*" PhD thesis, Carnegie Mellon Computer Science Department, CMU-CS-88-148, April 1988.

[Levin 90]
E. Levin : "*Word recognition using Hidden Control Neural Architecture*", ICASSP 90, pp433-436, 1990.

[Levinson 89]
S.E. Levinson, L.R. Rabiner, M.M. Sondhi :"*An Introduction to the application of the theory of probabilistic functions of a Markov process to automatic speech recognition*", BSTJ, 62, 1035-1074, 1983.

[Liénard 92]

Liénard, J.S. : *"Evaluation perceptive d'un corpus de viyelles françaises émises isolément par plusieurs locuteurs selon diverses forces de voix"* 19èmeJEP, Bruxelles, 1992.

[Liénard 77]

Liénard, J.S. : *"les processus de la communication parlée"*, Edition Masson, 1977.

[Linch]

Y. Linch, A. Buzo, R.M Gray :*"An algorithm for vector quantization design"*, IEEE Trans on communication, vol. com-28.

[Lippman 89]

Lippmann R.P.: *"Review of Neural Networks for Speech Recognition"* - Neural Computation, vol 1, pp 1-38, 1989.

[MCCulloch 43]

MCCulloch W., Pitts W. : *"A logical calculus for the ideas immanent in nervous activity."* Bull. Math. Biophysics, 5, pp 115-133, 1943.

[MCDermott 92]

MCDermott, E., Katagiri, Sh. : *"Prototype-based discriminative training for various speech units"*, ICASSP, vol. I-417, San Francisco, 1992.

[MCDermott 89]

MCDermott, E., Katagiri, Sh. : *"Shift-Invariant, Multi-Category pPhoneme Recognition using Kohonen's LVQ2"*, ICASSP 89, p81, 1989.

[Merialdo 88]

B. Merialdo :*"Phonetic Recognition using Hidden Markov Models and Maximum Mutual Information Training"*, ICASSP88, S3.4, 1988.

[Minsky 68]

Marvin Minsky, Seymour Papert : "Perceptrons", MIT Press, Cambridge MA, 1968.

[Morgan 90]

N. Morgan, H. Bourlard : *"Continuous speech recognition using Multilayer Perceptrons with Hidden Markov Models"*, ICASSP 90, pp 413-416, 1990.

[Montacie,Chollet 87]

C. Montacié, G. Chollet :*"Systèmes de références pour l'évaluation d'applications et la caractérisation de bases de données en reconnaissance automatique de la parole"*, 16ème JEP, Hammamet, 1987.

[Nadas 83]

A. Nadas :*"A decision-theoric formulation of a training problem in speech recognition and a comparison of training by unconditional versus conditional maximum likelihood"*, IEEE Transactions on Acoustics, Speech and Signal Processing, vol. ASSP-31, pp814-817, 1983.

[Nakagawa 90]
Nakagawa, S., Yoshimitsu, H. :"*Comparison among time-delay neural network, LVQ2, discrets parameter HMM and continuous parameter HMM*", ICASSP 90, p509, 1990.

[Ney 90]
H. Ney :"*Acoustic modelling of phoneme units for continuous speech recognition*", Signal Processing V :Théories and Applications, Elsevier Science Publishers B.V. 1990.

[Ney 84]
H. Ney :"*The Use of a One-Stage Dynamic Programming Algorithm for Connected Word Recognition*", IEEE Trans. on Acoustics, Speech and Signal Processing, Vol. ASSP-32, No. 2, pp. 263-271, April 1984.

[Niles 92]
L.T. Niles :"*Error-correcting training for phoneme spotting*", ICASSP92, San Francisco, Vol I-425, 1992.

[Niles 91]
L.T. Niles :"*Timit phoneme recognition using an HMM-derived recurrent neural network*", Eurospeech91, p559, Gênes, Italie, 1991.

[Niles 90]
L.T Niles, H.F. Silverman : "*Combining Hidden Markov Model and Neural Network Classifiers*", ICASSP90, pp417-420, 1990.

[Normandin 91]
Y. Normandin, D. Morgera :"*An Improved MMIE Training Algorithm for Speaker-Independent Small Vocabulary, Continuous Speech Recognition*", ICASSP91, pp537-540, 1991.

[Parker 85]
D.B. Parker :"*Learning Logic*"- Technical Report MIT, 1985.

[Peterson 52]
G.E. Peterson, H.L. Barney :"*Control methods used in a study of vowels*", The journal of the Acoustic Society of America", vol. 24, pp175-84, 1952.

[Poggio 89]
Poggio T., Girosi :"*A theory of networks for approximation and learning*". MIT AT Memo No. 1140. 1989.

[Pollard 85]
D.V. Pollard :"*Quantization and the méthods of K-means*", IEEE Trans on ASSP, 1985

[Price 88]
P. Price, W. Fisher, M. Bernstein, D. Pallet : *"The DARPA 1000-word Ressource Management database for continuous speech recognition"*, ICASSP88, pp651-654, 1988.

[Rabiner,Juang 86]
L.R. Rabiner, B.H. Juang :*"An introduction to hidden markov models"*. IEEE ASSP magazine, 3(1):4-16, 1986.

[Rander,Unnikrishnan 92]
P.W. Rander, K.P. Unnikrishnan : *"Learning the time-delay characyeristics in a neural network"*, ICASSP, vol. II-285, San Francisco, 1992.

[Renals 92]
Renals, S., Morgan, Cohen M., Franco, H. : *"Connectionist probability estimation in the decipher speech recognition system"*, ICASSP, vol. II-349, San Francisco, 1992.

[Robinson 92]
A. Robinson :*"A real-time recurrent error propagation network word recognition system"*, ICASSP, vol. I-617, San Francisco, 1992.

[Robinson,Fallside 91]
A. Robinson, F. Fallside :*"A recurrent error propagation network speech recognition system"*, Computer Speech and Language, pp259-274, 1991.

[Rosenblatt 58]
Rosenblatt F. : *"The Perceptron: A probabilistic model for information storage and organization in the brain"*, Psychological Review, Vol. 65, pp 386-407, 1958.

[Rosenblatt 57]
Rosenblatt F.: *The Perceptron: a perceiving and recognizing automaton* - Project PARA, Cornell Aeronautical Lab., Report 85-460-1, 1957.

[Rumelhart 86]
D.E. Rumelhart, G.E. Hinton, R.J. Williams : *"Learning internal representation by error propagation"*, in D.E. Rumelhart and J.L. McClelland (eds) Parallel Distributed Processing, MIT Press, vol. 1, 318-362, 1986.

[Russel 89]
Russell M. : *"Segment-Level Hidden Markov Model"*, IS2 Research Note, DRA-SRU (GB), 1991.

[Sakoe,Shiba 71]
H. Sakoe, S. Shiba : "A dynamic progamming approach to continuous speech rcognition", 7th ICA, Budapest, 1971

[Sawai, Waibel 89]
H. Sawai, A. Waibel, M. Miyatake, K. Shikano :*"Spotting Japanese CV-Syllabes and Phonemes Using Time-Delay Neural Networks"*, ICASSP91, p25, 1989.

[**Tebelskis, Waibel 91**]
J.Tebelskis, A.Waibel :"*Continuous speech recognition using linked predictive neural networks*", ICASSP91, p61, 1991.

[**Tebelskis, Waibel 90**]
J.Tebelskis, A.Waibel :"*Large Vocabulary Recognition using linked predictive Neural Networks*", IEEE magazine (1990).

[**Vintsjuk 86**]
T.K. Vintsjuk : "*Reconnaissance de mots par programmation dynamique*", Kybernetika 1, page 81-88, 1968.

[**Waibel 90**]
A. Waibel, H. Sawai :"*Integrated Training for Spotting Japanese Phonemes Using Large Phonemic Time-Delay Neural Networks*", ICASSP90, pp449-452, 1990.

[**Waibel 89**]
A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, K. Lang :"*Phoneme Recognition using Time Delay Neural Networks*".*IEEE Trans ASSP*, 1989.

[**Waibel 88**]
A.Waibel: "*Connectionist Glue: Modular Design of Neural Speech Systems*". Proceedings of the 1988 Connectionist Models Summer School, Carnegie Mellon University, pp 417-421, 1988.

[**Waibel 88**]
Waibel A., Hanazawa T., Hinton G., Shikano K., Lang K.: "*Phoneme recognition: neural networks vs. Hidden Markov Models*". Proceedings ICASSP 88, S-Vol.1, 107-110, 1988.

[**Waibel 87**]
A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, K. Lang:"*Phoneme recognition using Time Delay Neural Networks*", ATR Interpreting Telephony Research Laboratories, Octobre 1987.

[**Watrous 87**]
Raymond L. Watrous, Lokendra Shastri : "Learning Phonetic Features Using Connectionist Networks : An Experiment in Speech Recognition", pp. 381-388, vol. 4, IEEE, San Diego, California, June 21-24, 1987.

[**White 89**]
H. White : "Learning in artificial neural networks: A statistical perspective", Neural Computation, vol.1, no. 4, pp. 425-464, 1989.

[**Widrow 60**]
Widrow B., Hoff M.E.: "*Adaptive switching circuits*" - IRE WESCON Conv. record, part 4 1960, pp 96-104, 1960.

[Young 90]
S.J. Young :"*Competitive Training: A connectionist Approach to the Discriminative Training of Hidden Markov Models*", CUED/F-INFENG/TR.41, 1990