Parallélisation et Implantation d'un Algorithme de Reconnaissance de Mots Chinois

Fan Yang[(1)], **Michel Paindavoine**[(1)] **et Hervé Abdi**[(2)]

[(1)]LE2I, Université de Bourgogne,
6, Bd Gabriel-21000 Dijon, France
email:fanyang(paindav)@u-bourgogne.fr

[(2)]LEAD, Université de Bourgogne,
6, Bd Gabriel-21000 Dijon, France
email:herve@u-bourgogne.fr

RÉSUMÉ

Nous présentons une parallélisation et une implantation d'un algorithme de reconnaissance de mots chinois ; cet algorithme est basé sur une approche neuronale utilisant une mémoire auto-associative. On peut distinguer 2 niveaux de parallélisme dans cet algorithme : parallélisme de la phase d'apprentissage et parallélisme de la phase de reconnaissance. Une application de reconnaissance de mots chinois a été implantée sur une architecture utilisant 3 DSPs TMS320C40. Les accélérations de 2.1 et 2.52 sont respectivement obtenues pour les phases d'apprentissage et de reconnaissance .

ABSTRACT

We present a parallelization and an implementation of a Chinese character recognition algorithm. This algorithm is based on a neuronal approach using auto-associative memories. We can make out two parallelism levels in this alogorithm: learning phase and recognition phase. A Chinese character recognition algorithm has been implemented using 3 DSPs TMS320C40 from Texas Instruments. Accelerations of 2.1 and 2.52 are respectively obtained for the learning phase and the recognition phase.

1 Introduction

Depuis quelques années, les réseaux de neurones artificiels font l'objet d'un renouveau d'intérêt, car ils correspondent à une approche performante pour la résolution d'un grand nombre de problèmes comme par exemple la reconnaissance de formes, la classification, le traitement du signal et des images. En outre, la plupart des calculs que demandent ces réseaux sont réguliers et dès lors, leur implantation sur des architectures parallèles permet de réduire considérablement le temps de calcul comparé aux machines séquentielles [1]. Nous illustrons cette propriété avec une application liée à la reconnaissance de formes, et plus particulièrement à la reconnaissance de caractères.
Dans une première de cet article nous présentons le modèle de réseau de neurones utilisé qui se décompose en deux phases : la phase d'apprentissage et la phase de reconnaissance. En deuxième partie nous abordons en termes d'adéquation algorithme-architecture la stratégie de parallélisation. Les troisième et quatrième parties sont consacrées à la description de la parallélisation des deux phases de l'algorithme.

2 Présentation de l'algorithme

Les modèles de réseaux de neurones utilisant des mémoires auto-associatives sont donc composés de neurones linéaires dont la réponse est une fonction linéaire de l'activation et fonctionnent comme des mémoires auto-adressables. Le but de ces modèles est de stocker des stimuli et de les récupérer avec une version partielle ou dégradée du stimulus. L'avantage de ces modèles par rapport à des modèles non linéaires est de pouvoir intégrer un très grand nombre de cellules dans le réseau. Leur implémentation est particulièrement aisée, puisque ces modèles s'analysent avec des notions de valeurs et vecteurs propres (matrices semi-définies positives). En outre, dans de nombreuses applications, utilisant des approches plus sophistiquées, ces modèles constituent alors la première étape de traitement.

L'auto-associateur est entraîné par un apprentissage supervisé. C'est à dire qu'en phase d'apprentissage, on applique en entrée du réseau un ensemble de données auquel on fait correspondre une sortie désirée. Nous avons utilisé la règle de Widrow-Hoff qui modifie les poids à chaque présentation d'un couple entrée-sortie et dont l'apprentissage procède par itérations. Des modèles plus récents utilisent les vecteurs propres et les valeurs propres comme outils d'analyse [2]. Rappelons que la matrice de prototypes $\mathbf{A}$ (ou stimuli d'apprentissage) peut se décomposer en valeurs singulières selon :

$$\mathbf{A} = \mathbf{P}\Delta\mathbf{Q}^{\mathbf{T}} \quad (1)$$

$\mathbf{P}$: matrice des vecteurs propres de $\mathbf{A}\mathbf{A}^{\mathbf{T}}$
$\mathbf{Q}$: matrice des vecteurs propres de $\mathbf{A}^{\mathbf{T}}\mathbf{A}$
Δ : matrice diagonale des valeurs singulières = $\Lambda^{1/2}$

Λ : matrice des valeurs propres de $\mathbf{AA^T}$ et $\mathbf{A^TA}$

Nous pouvons alors montrer [3] que la matrice de poids **W** converge vers :

$$\mathbf{W}^{(\infty)} = \mathbf{PP^T} \quad (2)$$

La phase de rappel revient à multiplier la matrice de nouveaux stimuli **X** par la matrice de poids **W** pour obtenir la matrice de réponse **O** :

$$\mathbf{O} = \mathbf{PP^TX} \quad (3)$$

La figure 1 montre les résultats d'une mémoire auto-associative qui est utilisée pour reconnaître des images bruitées de mots chinois. La mémoire reconstitue des stimuli stockés de manière satisfaisante.

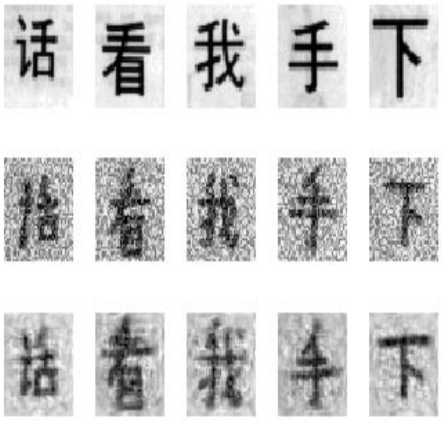

FIG. 1 — *Réponse d'un auto-associateur à des mots chinois manuscrits et bruités. La mémoire a été entraînée avec des mots chinois imprimés (en haut), au milieu :stimuli bruités du réseau et en bas : réponse du réseau*

3 Méthodologie et architecture parallèle

Les deux questions primordiales que l'on doit se poser pour implémenter un algorithme sur une architecture parallèle sont :

— `quelles sont les spécificités de l'algorithme ?`

— `quel type d'architecture pourrait le mieux répondre au besoin ?`

La première question consiste à exprimer l'algorithme initial de façon non plus séquentielle, mais de manière à mettre en évidence le parallélisme inhérent de l'algorithme. Cette étape peut être réalisée à l'aide d'un graphe de dépendance qui est une description formelle des activités espace-temps dans un algorithme. Pour la deuxième question, on propose une architecture à flot de données qui nous semble bien adaptée à notre algorithme de reconnaissance de mots chinois. L'architecture à flot de données ne comporte ni unité de commande, ni mémoire globale, mais est directement contrôlée par les données elles-mêmes [4]. Dans cette approche, l'arrivée d'une donnée d'un processeur voisin est interprétée comme un changement d'état et initialise une action. Ce type de machine permet à la fois de limiter le coût de gestion (sans horloge globale) et d'obtenir un haut degré de parallélisme.

4 Parallélisation de la phase d'apprentissage

L'apprentissage s'effectue en trois étapes (voir figure 2) :

- Calcul de la matrice $\mathbf{B=A^TA}$ avec **A** : l'ensemble des prototypes

- Extraction des vecteurs propres et des valeurs propres de la matrice **B** : [**Q**, λ]=eigen (**B**)

- Calcul de la matrice de vecteurs propres $\mathbf{P=AQ}$ $\text{inv}(\text{diag}(\lambda)) = \mathbf{AQ}\lambda^{-1}$

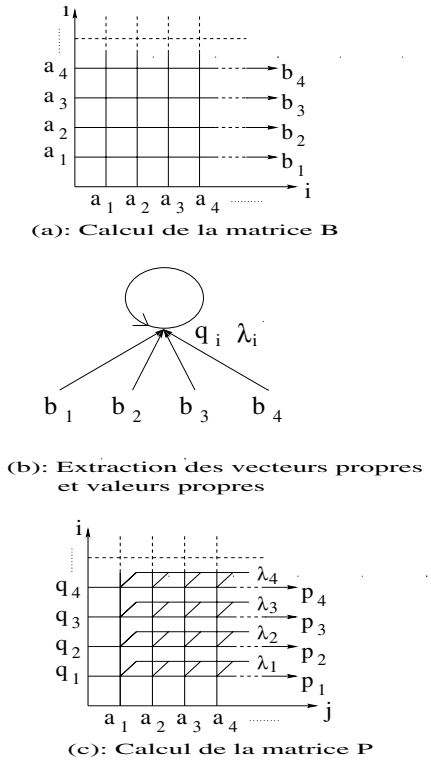

FIG. 2 — *Graphe de dépendance de la règle de Widrow-Hoff*

Une boucle est présente sur le graphe de la figure 2b. Ceci montre que le calcul en cours de $\mathbf{q_i}$, λ_i ne peut commencer seulement quand on a le résultat $\mathbf{q_{i-1}}$, λ_{i-1}, ce qui signifie que cette étape d'extraction des vecteurs propres et des valeurs propres n'est pas parallélisable. Par contre, les deux autres étapes de l'algorithme présentent une grande potentialité de parallélisme.

Nous avons implanté une application de reconnaissance de mots chinois sur 3 DSPs en suivant l'approche par flot de données. Chaque mot est composé de 40×40 pixels. La figure 3 montre le graphe flot de données de l'implémentation de la phase d'apprentissage qui a été obtenu avec l'outil SynDex [4]. Le chronométrage de l'exécution montre qu'une accélération

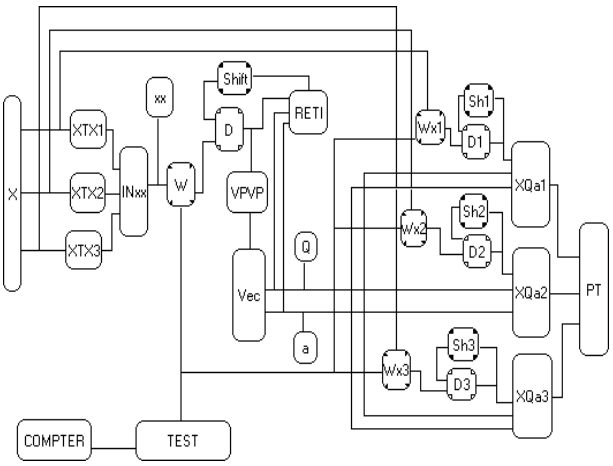

FIG. 3 — *Implantation de la phase d'apprentissage*

d'un facteur de 2.1 est obtenue.

5 Parallélisation de la phase de reconnaissance

Pour calculer la réponse du réseau, il suffit de multiplier la matrice des nouveaux stimuli **X** par la matrice de poids pour obtenir la matrice de réponse **O** selon l'équation (3). Dans le but de la parallélisation, l'équation (3) est factorisée et nous obtenons :

$$\mathbf{o_i} = \sum_{\mathbf{k=1}}^{\mathbf{N}} \mathbf{p_k}(\mathbf{p_k}^{\mathbf{T}}\mathbf{x_i}) = \sum_{\mathbf{m=1}}^{\mathbf{M}} (\sum_{\mathbf{k=1}}^{\mathbf{R}} \mathbf{p_k}(\mathbf{p_k}^{\mathbf{T}}\mathbf{x_i})) \qquad (4)$$

avec N le nombre de colonnes de la matrice P, M×R = N.
La figure 4 représente le graphe de dépendance de la phase

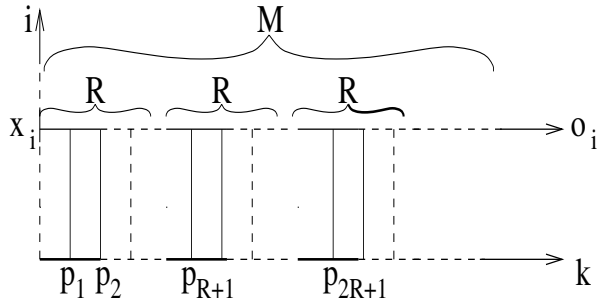

FIG. 4 — *Graphe de dependance*

de rappel. On peut obtenir la réponse $\mathbf{o_i}$ au stimulus $\mathbf{x_i}$ en effectuant des calculs parallèles sur M blocs de données, chaque bloc étant composé de R colonnes de P. Ainsi les blocs de **P** sont distribués sur tous les processeurs d'une manière équilibrée. A chaque flot de données, un stimulus $\mathbf{x_i}$ est transmis à l'architecture parallèle, chaque processeur réalise sa propre tâche en même temps que les autres d'une manière asynchrone. A la fin du calcul, tous les résultats sont regroupés vers la machine **root**. Cette approche peut être réalisée avec une topologie de type "grille". L'efficacité de cette approche parallèle est illustrée par la figure 5.

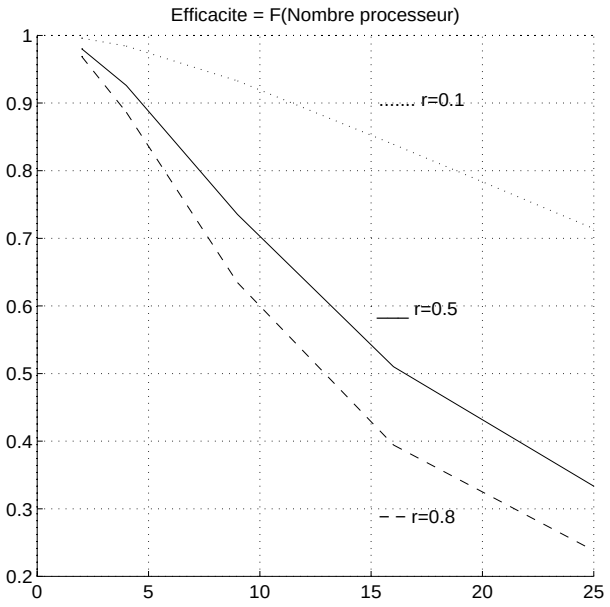

FIG. 5 — *Efficacité de parallélisation*

Nous avons r= T_{com}/T_{cal}, T_{com} qui est le temps de communication d'un vecteur entre deux processeurs voisins, et T_{cal} qui est le temps nécessaire pour traiter une ligne de donnée de la matrice de vecteurs propres **P**. Plus r est grand, plus l'efficacité baisse. Pour un nombre de processeurs inférieur à 8, on a toujours une approche parallèle assez performante et ceci indépendemment de la valeur de r. L'application de reconnaissance de mots chinois a été implantée sur 3 DSPs TMS320C40, la figure 6 montre le graphe flot de données (à gauche) et l'architecture du système (à droite). Nous avons

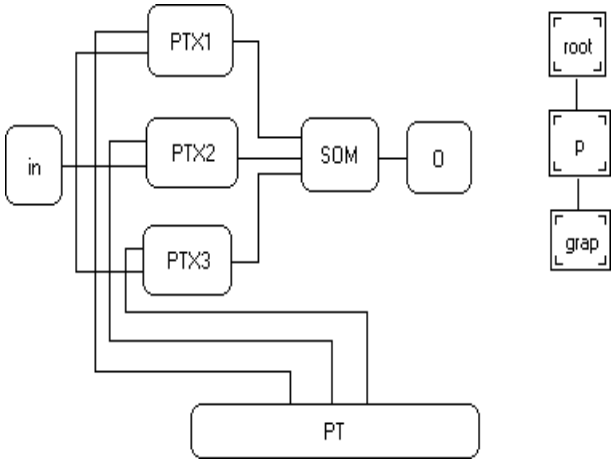

FIG. 6 — *Graphe Browser de l'implantation*

obtenu une accélération expérimentale de 2.52 (Eff_{exp} = 0.84). La vitesse de reconnaissance est de plus de 6 mots par seconde (6.25 mots/s, T_{para} = 160 ms).

6 Conclusion

Les analyses et résultats obtenus pour la parallélisation de notre algorithme de reconnaissance de mots chinois montrent

que l'algorithme de l'auto-associateur est un problème très parallélisable aussi bien pour la phase d'apprentissage que pour la phase de reconnaissance.
En perspective, un prétraitement utilisant un détecteur de contour multi-échelles [5] a montré que le taux de reconnaissance augmente de 17% [6]. Nous pensons implanter ce prétraitement sur un module multi-FPGA en maintenant l'implantation de l'auto-associateur sur une carte multi-DSP, ceci constituant un ensemble de machine hétérogène.

Références

[1] H.Yoon, JH.Nang and S.R.Maeng, *Parallel simulation of multiyered neural networks on distributed-memory multiprocessors*, Microprocessing and microprogramming, Vol.29, 1990.

[2] X.Jia and S.Nixon, *Extending the feature vector for automatic face recognition*, IEEE-Transactions on "Patterns Analysis and Machine intelligence "Vol.17, December 1995.

[3] D. Valentin and H. Abdi, *Connectionist models of face processing : a survey*, Pattern Recognition, Vol.27, 1994.

[4] C.Lavarenne, O.Seghrouchini, Y.Sorel, M.Sorine, *SynDex, un environnement de programmation pour applications de traitement du signal distribuées*, 13ème Colloque GETSI, Juan les Pins, Septembre 1991.

[5] Sariffudin, *Implantation sous forme de circuit spécialisé d'un algorithme de détection de contours mulitéchelles*, Thèse soutenue à l'Université de Bourgogne, Juillet 1995.

[6] F.Yang, M.Paindavoine and H.Abdi, *Multiscale edges detection by wavelet transform for model of face recognition*