Dix-septième colloque GRETSI, Vannes, 13-17 septembre 1999

Identification de visages sur une machine parallèle à DSPs : Implantation d'un réseau de neurones avec le logiciel SynDEx

Fan YANG[1], Michel PAINDAVOINE[1], Hervé ABDI[2]

[1]Laboratoire LE2i, Aile de l'Ingénieur
Faculté des Sciences Mirande
Université de Bourgogne
BP 400 - 21011 DIJON Cedex France

[2]University of Texas at Dallas
Program in Cognition, MS GR:4.1, Richardson TX
75083-0688, U.S.A.

fanyang@u-bourgogne.fr, paindav@u-bourgogne.fr,
http://www.utdallas.edu/ herve/

Résumé – Cette étude a été menée dans le cadre de l'élaboration d'un réseau de neurones multi-couches utilisé pour l'identification automatique de visages humains. Nous avons porté l'algorithme d'apprentissage par rétro-propagation sur une machine parallèle à DSPs. L'objectif principal était de réduire le temps de calcul pendant la phase d'apprentissage pour permettre de simuler rapidement différentes architectures.

Abstract – This research concerns the field of development of a multi-layer neural network applied to face automatic identification. The back-propagation learning algorithm has been realized onto multi-DSPs TMS320C40, using the SynDEx environment. The objective is to reduce the time of computation amounts of learning and to try rapidly different sets of parameters for the neural network.

1 Introduction

Un système automatique capable de localiser et identifier des visages humains est utile dans maintes applications : le contrôle d'accès, la criminologie, l'interface homme-machine, et la communication d'images *etc.* En fait, toutes les situations dans lesquelles l'identification d'une personne est nécessaire, sont potentiellement concernées. Ici, notre système d'identification de visages est réalisé avec un modèle connexionniste.

Les réseaux de neurones sont constitués d'un grand nombre d'unités de traitements élémentaires opérant en parallèle. Ils peuvent réaliser n'importe quelle approximation de fonction avec une précision arbitraire. De tel systèmes peuvent assurer des tâches de haut niveau comme la classification, l'optimisation ... Mais, les performances obtenues en approximation dépendent de plusieurs paramètres. Certains définissent l'architecture du réseau de neurones (nombre de couches cachées, nombre de cellules par couche), d'autres concernent la phase d'apprentissage (taux d'apprentissage, critère d'arrêt, ...). Trouver les paramètres optimaux pour un problème particulier n'a pas de réponse théorique. Aussi doit on procéder par une méthode d'essai-erreur, et choisir parmi un ensemble de paramètres ceux donnant les meilleurs résultats. Cette phase de recherche est longue et toutes les évaluations sont indépendantes les unes des autres, d'où l'idée d'une implantation parallèle qui accélère les calculs.

Dans cet article, nous présentons le réseau de neurones utilisé pour l'identification de visage humains, ainsi que la méthode de parallélisation de celui-ci. Les implantations parallèles à l'aide du logiciel SynDEx ont donné de bon résultats que nous montrons en fin article.

2 Le système d'identification de visages humains

2.1 Présentation du système

Notre système d'identification est composé de deux étages : le module de compression utilisant la technique de ACP (Analyse en Composantes Principales) pour réduire la taille de données à traiter et le module d'identification d'un réseau de multi-couches (voir Figure.1). Ce dernier est constitué de 3 couches : 1 couche d'entrée de 100 cellules de calcul, 1 couche cachée de 15 cellules de calcul, 1 couche de sortie de 11 cellules de calcul correspondant aux 10 personnes à identifier et à la classe "invité". Ces cellules de calcul élémentaire sont basées sur le principe du neurone formel[1].

2.2 Algorithme d'apprentissage : rétro-propagation des erreurs

L'apprentissage dans un tel réseau consiste à déterminer les poids des connexions entre les cellules de calcul qui

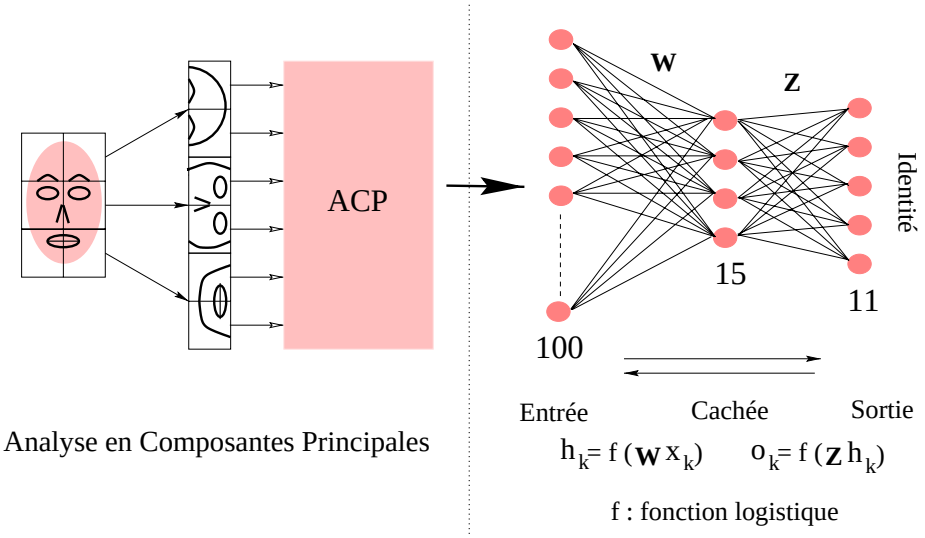

FIG. 1: Système d'identification de visages.

sont modifiés en fonction d'un ensemble d'exemples d'apprentissage. L'algorithme de rétro-propagation des erreurs utilise la technique du gradient pour minimiser une fonction coût égale à la moyenne des différences au carré entre la sortie désirée et la sortie observée.

En supposant que les valeurs des paramètres soient stockées dans une matrice $\mathbf{Z}$ et que la fonction à minimiser soit $y = g(\mathbf{Z})$, l'algorithme se déroule comme suit[2] :

– Choisir arbitrairement les valeurs $\mathbf{Z}_{[t]}$ pour $t = 0$ (de manière aléatoire, en règle générale).
– Calculer le gradient de g (noté $\bigtriangledown g$).

$$\frac{\partial g(\mathbf{Z}_{[t]})}{\partial \mathbf{Z}_{[t]}} = \bigtriangledown g. \qquad (1)$$

– Corriger $\mathbf{Z}_{[t]}$ en direction inverse du gradient de $\mathbf{Z}_{[t]}$ (avec η dénotant le coefficient d'apprentissage).

$$\mathbf{Z}_{[t+1]} = \mathbf{Z}_{[t]} + \Delta_t \mathbf{Z} = \mathbf{Z}_{[t]} - \eta \bigtriangledown g = \mathbf{Z}_{[t]} - \eta \frac{\partial g(\mathbf{Z}_{[t]})}{\partial \mathbf{Z}_{[t]}} \qquad (2)$$

– Continuer les deux dernières étapes tant que l'écart entre $\mathbf{Z}_{[t]}$ et $\mathbf{Z}_{[t+1]}$ est jugé important.

3 Parallélisation efficace: Adéquation Algorithme Architecture

3.1 Algorithme parallèle

Il existe trois variantes de l'algorithme de rétro-propagation des erreurs selon le nombre d'exemples b présentés au réseau entre chaque modification de la matrice synaptique[3]. Du point de vue de la parallélisation, l'algorithme du gradient total (b étant égal au nombre d'exemples dans la base d'apprentissage) semble idéal : l'estimation du gradient pour chaque exemple peut être effectuée indépendamment, ce qui permet des implantations très efficaces de l'algorithme sur des machines parallèles[3]. Cependant, il a été constaté expérimentalement que la vitesse de convergence de l'algorithme du gradient total est beaucoup plus lente que celle de l'algorithme du gradient stochastique ($b = 1$)[3]. Mais ce dernier est purement séquentiel.

Nous avons mené une étude sur la vitesse de convergence pour connaître les comportements de notre réseau avec différentes variantes de l'algorithme de rétro-propagation[4]. Le résultat des simulations nous montre que le nombre d'époques nécessaires pour atteindre la convergence avec l'algorithme du gradient total est beaucoup plus important que celui du gradient stochastique. Si on accepte une implantation parallèle de l'algorithme du gradient total, on gagne en parallélisme, mais on perd beaucoup en vitesse de convergence. Dans ce cas, il semble logique d'appliquer un apprentissage du gradient par bloc. Chaque sous-bloc calcule son gradient en même temps que les autres sous-blocs, le résultat de chaque sous-bloc est échangé mutuellement afin d'effectuer la mise à jour en parallèle. Ceci nous permet de réduire le temps de communication et surtout d'éviter le phénomène d'étranglement. Nous pouvons observer trois mises à jour des matrices de connexion. Chaque flot de données correspond à une époque d'apprentissage.

3.2 Performances prédites par SynDEx

L'environnement SynDEx[5][6] fournit la prédiction de performances, et permet de trouver une adéquation entre l'algorithme à implanter et l'architecture à utiliser dans une optique temps réel. Nous avons lancé l'heuristique de placement/ordonnancement de SynDEx pour des configurations matérielles allant de 2 à 9 processeurs. Les performances prédites sont représentées en figures 3 et 4. Nous supposons que le type de connexion entre les processeurs est de connexion totale, c'est-à-dire que l'on exploite le maximum de liens possédés par chaque processeur (6 pour le TMS320C40). Dans ce cas, nous obtenons une efficacité maximale de 0.91 pour un nombre de processeurs $N_{pr} = 3$. Quand on incrémente le nombre de processeurs, l'accélération de la machine parallèle augmente, mais c'est l'équilibre des charges qui joue le rôle le plus important. Prenons le nombre de processeurs $N_{pr} = 4$: puisque chaque bloc

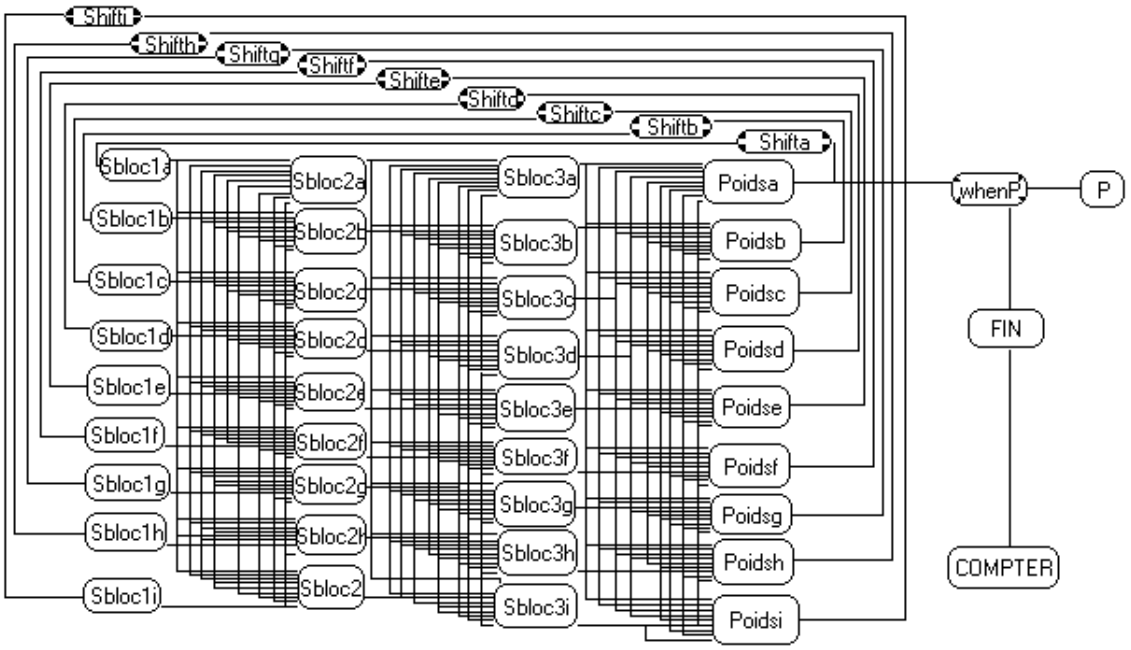

FIG. 2: Graphe flot de données avec 3×9 sous-blocs.

est divisé en 9 sous-blocs, avec $N_{pr} = 4$, il y a 4 sous-blocs qui peuvent s'exécuter en parallèle, pendant qu'un processeur traite le dernier sous-bloc, les trois autres processeurs restent inactifs, d'où une chute importante de l'efficacité du système. De même, quand le nombre de processeurs varie entre 5 et 9, l'efficacité du système diminue jusqu'à 8 processeurs et augmente à nouveau pour 9 processeurs, cela étant dû à la dissymétrie de répartition des tâches sur 8 processeurs d'un graphe logiciel optimum pour 9 processeurs.

Nous pouvons imaginer décomposer chaque bloc d'exemples d'une manière équilibrée en fonction du nombre de processeurs dont on dispose. Mais la réduction du grain de parallélisme va bientôt rencontrer une limite à partir de laquelle le rapport entre le temps de calcul et celui de communication rend la parallélisation peu rentable.

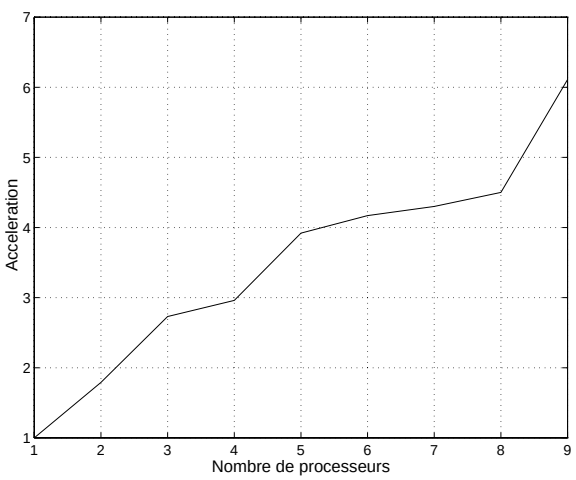

FIG. 3: Accélération de la machine parallèle.

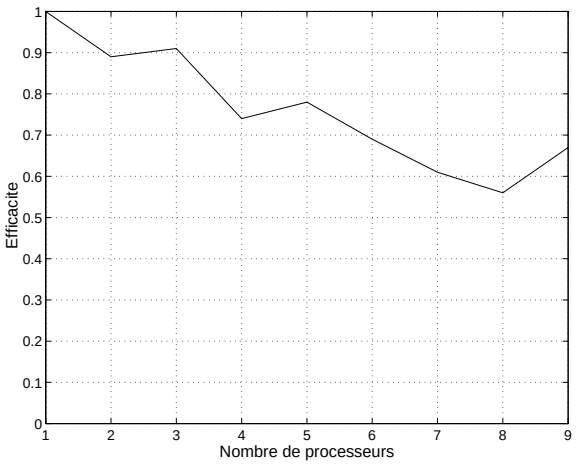

FIG. 4: Efficacité de la machine parallèle.

4 Implantation parallèle réalisée sur 3 DSPs

L'implantation a été réalisée avec une carte Transtech TDMB408, insérée dans un PC. Elle contient trois processeurs de traitement du signal TMS320C40 de Texas Instrument connectés en ligne par plusieurs liens.

En simplifiant le graphe flot de données représenté en figure 2, nous divisons les 324 exemples en 3 blocs de 108 exemples et chaque bloc est décomposé en 3 sous-blocs. Les 9 sous-blocs sont distribués sur les 3 DSPs dans la phase d'initialisation. L'implantation parallèle sur 3 DSPs est exposée dans la figure 5. SynDEx organise le placement du flot de données sur 3 DSPs et génère les exécutifs. Le résultat du chronométrage d'exécution sur 3 DSPs ainsi que la durée de chaque fonction sont donnés dans le tableau 1. D'après le diagramme temporel(voir la figure 6), le temps de communication est négligeable par rapport à

celui du calcul. Effectivement, l'exécution sur 3 DSPs permet d'obtenir un facteur d'accélération de 2.82. On peut dire que nous divisons pratiquement le temps de calcul par le nombre de processeurs avec la méthode de parallélisation utilisée. Ceci démontre une bonne parallélisation avec une répartition des charges de calcul qui s'avère optimum.

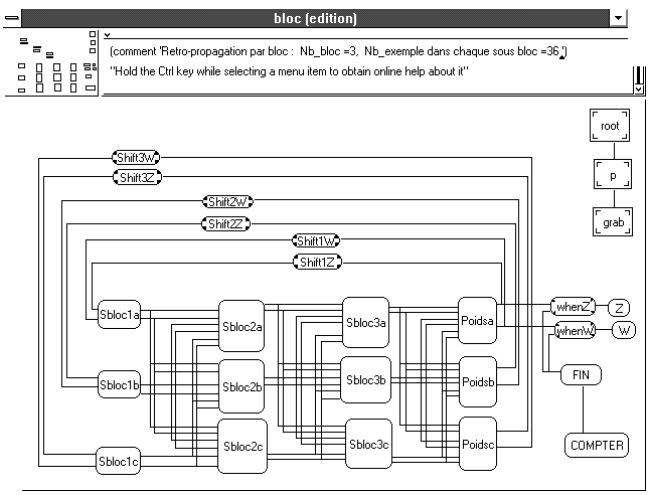

FIG. 5: Accélération et efficacité de la machine parallèle.

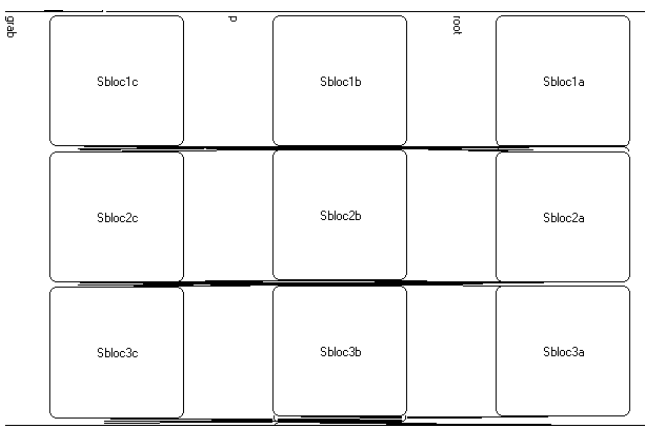

FIG. 6: Accélération et efficacité de la machine parallèle.

TAB. 1: Chronométrage de l'exécution pour une époque

Fonction	Sbloc1	Sbloc2	Sbloc3
Durée(μs)	260060	261818	261818
Fonction	Poids	T_{1DSP}	T_{3DSPs}
Durée(μs)	2203	2309182	818859

5 Conclusion

L'avantage principal de l'apprentissage du gradient par bloc est qu'il exhibe le parallélisme à grain fin. En répartissant la charge de calcul d'une manière équilibrée sur tous les processeurs, nous pouvons obtenir une implantation très efficace. Avec 3 DSPs TMS320C40 à notre disposition, nous avons réussi à diviser le temps d'apprentissage par 2.82 et l'efficacité de l'implantation parallèle a atteint 0.94. Cette machine parallèle nous a permis de réaliser l'apprentissage du réseau sur un nombre plus important d'individus et ainsi obtenir une meilleure estimation des zones de décision et une meilleure robustesse du réseau pour notre application d'identification de visages humains.

Références

[1] P. R. Lippman. *Pattern using neural networks.* IEEE. Communication magazine, Nov. 1989.

[2] H. Abdi. *Réseaux de neurones.* Presse universitaire de Grenoble, 1994.

[3] H. Paugam-Moisy. *On the convergence of a block-gradient algorithm for back-propagation learning.* Proc. IJCNN-92, no.3, 1992.

[4] F. Yang. *Traitement automatique d'images de visages : Algorithmes et Architecture.* Thèse soutenue à l'Université de Bourgogne, Juin, 1998.

[5] Y. Sorel et C. Lavarenne. *SYnDEx v4.2 INRIA: CAD for the optimized implementation of real-time algorithms on multicomponent architecture.* Technical Report, INRIA, 1996.

[6] C. Lavarenne et Y. Sorel. *Performance optimization of Multiprocessor Real-time Applications by Graph Transformations.* Parallel Computing, Grenoble, 1993.