SETIT 2007
4th International Conference: **S**ciences of **E**lectronic, **T**echnologies of **I**nformation and **T**elecommunications
March 25-29, 2007 – TUNISIA

Compression d'Images par une Technique Hybride de Parallélisation de Réseau de Neurones[1]

E.M. Daoudi [*], A.hajji [**] et E.M. Jaara[**]

[*]*Université Mohamed 1er, Ecole Nationale des Sciences Appliquées d'Oujda*

mdaoudi@ensa.univ-oujda.ac.ma

[**] *Université Mohamed 1er, Faculté des Sciences d'Oujda*

a.hajji@sciences.univ-oujda.ac.ma

jaara@sciences.univ-oujda.ac.ma

Résumé: Pour minimiser le coût de stockage et de transmission d'images on utilise les techniques de compression qui permettent de réduire la taille des données représentant l'image. L'objet de ce travail est l'étude de la parallélisation des méthodes de compression basées sur les réseaux de neurones multicouches. La technique de parallélisation proposée est une technique hybride qui est un compromis entre la technique statique, qui a l'avantage d'avoir un surcoût de communication réduit mais a l'inconvénient de déséquilibrage de charge; et la technique dynamique, qui présente un meilleur équilibrage de charge au détriment d'un surcoût de communication élevé. Les trois techniques, à savoir techniques statique, dynamique et hybride ont été implémentées et comparées sur un réseau de processeurs. Les tests numériques montrent que les meilleurs résultats peuvent être obtenus avec la méthode hybride.

Mots clés : Compression d'images, Réseau de neurones, Réseau de processeurs, Equilibrage de charge.

INTRODUCTION

Parmi les difficultés rencontrées dans le domaine du traitement d'images, il y a la quantité énorme de données qui représentent une image numérique [ABD 1]. Par conséquent la transmission et le stockage des images sont très coûteux, d'ou l'intérêt des techniques de compression qui deviennent nécessaires. Parmi les méthodes utilisées, on cite la compression par réseau de neurones Multi-Couches [ABD 1] dont le principe consiste à chercher les poids de connexions adéquats pour construire un réseau compresseur, composé de la couche d'entrée et la couche cachée, et un réseau décompresseur, composé de la couche cachée et la couche de sortie (voir figure 1), de telle façon que les données qui représentent l'image au niveau de la couche d'entrée, seront réduit selon un taux de compression calculé par le rapport entre le nombre de neurones de la couche cachée et celui de la couche d'entrée. Au niveau de la couche de sortie, on doit avoir presque les mêmes données d'entrées.

Vu la quantité de données que représentent les images numériques et la longue durée de la phase d'apprentissage qui demande des milliers d'itérations sur les machines monoprocesseur, le recours au traitement parallèle est naturel.

[1] Soutenu par l'UMP dans le cadre du PGR.

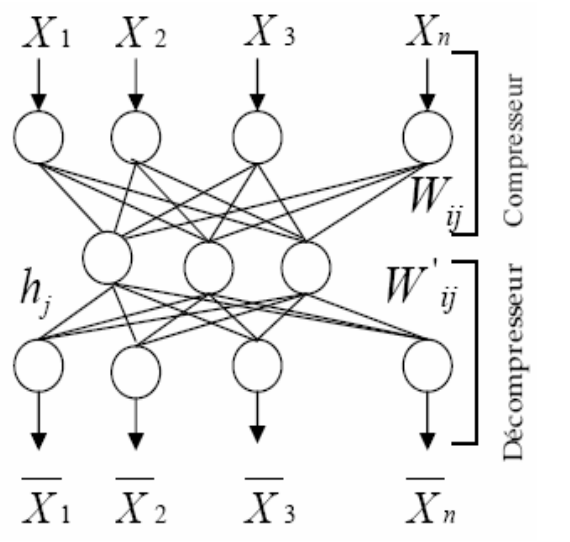

Figure 1 : *Réseau Compresseur -Décompresseur*

X_i : l'entrée de la cellule i,

$\overline{X_i}$: la sortie de la cellule i,

h_i : l'état de la cellule i dans la couche cachée,

$W_{i,j}$: la matrice des poids entre la couche d'entrée et la couche cachée,

$W'_{i,j}$: la matrice des poids entre la couche cachée et la couche de sortie.

1. L'algorithme d'apprentissage utilisé

Dans notre implémentation le réseau de neurones utilisé a les caractéristiques suivantes :

Une couche d'entrée de 64 cellules,

Une couche cachée de 4 cellules,

Une couche de sortie de 64 cellules.

La base d'exemples d'apprentissage est formée des images subdivisées en blocs de taille (8x8) pixels.

Description de l'algorithme de rétro-propagation :

initialisation des matrices des poids $(w_{ij})_{1\ 1}^{i\ 64}{}_{j\ 4}$ et $(w'_{ij})_{1\ 1}^{i\ 4}{}_{j\ 64}$ par des valeurs aléatoires dans l'intervalle [-0.5, 0.5],

répéter pour chaque bloc

- affecter les X_i (les données du bloc) aux cellules de la couche d'entrée.
- **tant que** l'erreur quadratique est supérieure à un seuil,
  - calculer les h_i (les états des neurones de la couche cachée) puis les $\overline{X_i}$ (les états des neurones de la couche sortie).
  - calculer les gradients des cellules de la couche de sortie puis ceux de la couche cachée.
  - mise à jour des poids par l'algorithme du gradient.
- fin **tant que**.

fin **répéter**

2. Technique statique

La méthode statique [DAO 3][JAA 4] exploite le parallélisme de données. Toute image de la base d'exemples d'apprentissage est découpée en n blocs de mêmes tailles. On affecte à chaque processeur $\left\lceil \frac{n}{p} \right\rceil$ blocs ou p est le nombre de processeurs. Un PMC local est implémenté sur chaque processeur. Les blocs reçus dans chaque processeur forment la base d'exemples d'apprentissage du PMC local [DAO 2][PAU 6], qui utilise l'algorithme de rétro-propagation du gradient [NAI 5] de façon indépendante. Bien que cette approche permette d'éviter les échanges de données, l'expérimentation de cet algorithme montre que la charge des processeurs n'est pas équilibrée. En effet, quand un processeur termine son travail (apprentissage des blocs locaux), il devient inactif même s'il existe des blocs dans d'autres processeurs en attente.

L'algorithme parallèle de la méthode statique :

affecter à chaque processeur $\left\lceil \frac{n}{p} \right\rceil$ blocs.

pour chaque processeur : Traitement des blocs locaux d'une façon séquentielle en utilisant l'algorithme de rétro-propagation du gradient.

Fin **pour**

3. La technique dynamique

Dans le but d'équilibrer la charge des processeurs, une technique de parallélisation dynamique [DAO 3][JAA 4], basée sur l'architecture maître-esclave, est introduite dont la description est la suivante :

L'image est subdivisée en n blocs comme dans le cas de la méthode statique.

Il y a (p-1) processeurs esclaves et un processeur maître.

Au début du traitement, chaque processeur esclave commence à traiter un bloc, le processeur maître affecte les blocs aux processeurs esclaves au fur et à mesure qu'ils deviennent inactifs.

Le processus est répété tant qu'il reste encore des blocs non traités.

L'algorithme parallèle de la méthode dynamique :

Processeur maître :

envoie d'un bloc à chaque processeur esclave,

Tant qu'ils existent des blocs en attente

- réception de la demande d'un processeur inactif Y,
- envoie d'un bloc au processeur Y,

fin **Tant que.**

informer tous les processeurs esclaves de l'épuisement du travail.

Processeur esclave :

initialisation des poids du PMC local,

réception d'un bloc du processeur maître,

traitement du bloc reçu,

tant qu'il reste du travail (il existe des blocs non encore traités),

- envoie la demande au processeur maître,
- réception d'un bloc du maître,
- traitement du bloc reçu,

fin **Tant que.**

4. Technique hybride

La troisième technique qu'on propose dans ce travail représente un compromis entre la technique statique et la technique dynamique, de telle façon qu'on utilise les avantages des deux méthodes. La façon de distribution des blocs est la suivante :

il y a (p-1) processeurs esclaves et un processeur maître.

Au début du traitement le processeur maître envoie k blocs pour chaque processeur esclave, avec $1 \quad k \quad \left\lceil \frac{n}{p} \right\rceil$, (pour k=1, c'est le cas dynamique et pour $k \quad \left\lceil \frac{n}{p} \right\rceil$ c'est le cas statique)

Les $(n \quad pk)$ blocs restants seront distribués aux processeurs esclaves un par un de la même façon que la méthode dynamique au fur et à mesure qu'ils terminent leur travail (apprentissage des blocs locaux).

5. Résultats expérimentaux

Nous avons implémenté cette technique sur un réseau de 5 processeurs sous l'environnement de programmation parallèle MPI [PET 7]. Les tests sont effectués sur une image de taille (312 312) pixels (voir figure 2), subdivisée en (39 39) blocs chacun de taille (8 8) pixels.

Le maître est un Pentium 4 cadencé à 3.2 Ghz et équipé de 512 Mo de RAM,

les 4 esclaves sont des Pentium 2 cadencés à 300 Mhz. La taille de RAM pour chacun est de 64 Mo.

Les différentes PCs sont connectés à un HUB.

Figure 2 : *Image avant compression*

Figure 3 : *Image après décompression*

k	Processeur numéro	Nombre de blocs traités	Temps d'exécution (s)
1	0	397	652
	1	372	650
	2	385	656
	3	385	688
16	0	387	550
	1	377	522
	2	386	558
	3	371	560
24	0	377	500
	1	391	502
	2	373	498
	3	380	503
32	0	365	545
	1	380	558
	2	394	530
	3	382	558
64	0	382	766
	1	376	767
	2	371	791
	3	392	783
128	0	377	812
	1	404	801
	2	348	806
	3	392	798
232	0	377	834
	1	365	846
	2	406	870
	3	373	829
256	0	388	954
	1	383	968
	2	365	953
	3	385	980
264	0	390	863
	1	375	852
	2	371	840
	3	385	856
300	0	373	807
	1	381	813
	2	375	832
	3	392	821
380	0	380	700
	1	380	782
	2	381	847
	3	380	794

Table1: *Temps d'exécution en fonction de la variation de k*

Dans la table 1, on reporte le temps d'exécution (par 1000 itérations) obtenu en secondes pour l'apprentissage suivant le nombre de blocs k initialement affectés à chaque processeur esclave. Dans la colonne 3 on reporte le nombre de blocs réellement traités par chaque processeur et dans la colonne 4 on reporte le temps d'exécutions (calcul + communication) pour chaque processeur. Les résultats expérimentaux montrent :

un déséquilibrage de charge pour la technique statique (pour k=380). Le temps d'exécution des différents processeurs varie entre 700s et 847s.

une amélioration de l'équilibrage de charge pour la technique dynamique (pour k=1). Le temps d'exécution des différents processeurs varie entre 650s et 688s.

un meilleur équilibrage de charge est obtenu pour la technique hybride avec k=24. Le temps d'exécution pour les différents processeurs varie entre 498s et 503s.

Remarque : la valeur k=24 est obtenue expérimentalement. Une étude théorique est nécessaire pour estimer la valeur de k qui donne le temps d'exécution minimum.

6. Conclusion

Dans ce travail, nous avons exposé les avantages et les inconvénients de deux techniques d'apprentissage des réseaux de neurones conçus pour la compression d'images. La première technique statique est basée sur une distribution uniforme des blocs, le grand avantage de cette méthode se résume dans l'élimination des communications entre les différents processeurs, mais l'inconvénient majeur qui se pose pour cette méthode est le problème d'équilibrage de charge. La seconde méthode dynamique résoud le problème d'équilibrage de charge posé par la technique statique, mais pose le problème de communications entre les processeurs esclaves et le processeur maître.

La technique hybride, proposée dans cet article, représente un compromis entre les deux techniques citées précédemment, pour minimiser le temps d'exécution. Au début, le processeur maître distribue d'une façon uniforme un nombre de blocs k à tous les processeurs esclaves, et si un processeur termine son travail sur les blocs locaux, il demande au processeur maître un bloc non encore traité. Ce qui explique la réduction du coût de communications, et par conséquent elle rassemble les avantages des deux techniques. Les études expérimentales que nous avons effectuées pour cette technique nous permettent de trouver le nombre de blocs k affectés au départ à chaque processeur pour avoir un temps d'exécution minimal. En perspectives :

Effectuer le test sur plusieurs images.

étude préalable des données des images qui forment la base d'apprentissage avant la distribution des blocs aux processeurs.

Faire une étude théorique incluant les paramètres de la machine et les caractéristiques de l'image afin d'estimer le nombre de blocs k réalisant le temps d'exécution minimum.

REFERENCES

[1] O. ABDEL-WAHHAB et M.M. FAHMY, *"Image Compression using Multilayer Neural Networ*", IEEE Proc - Vis. Image Signal Process, Vol. 144. N° 5 October 1997.

[2] E.M. DAOUDI, E.M. JAARA, *"Parallel Methods of Training for Multilayer Neural Network*" 5th International Euro-Par Conference, Toulouse, France, August/September 1999, Lecture Notes in Computer Science 1685, p.686 – 690.

[3] E.M. DAOUDI, E.M. JAARA, H. NAIT CHERIF, *"Étude de la Parallélisation de la Compression d'Images par Réseaux de Neurones Multicouches*", CARI'2000 ANTANANARIVO (Madagascar) Session 8A Architecture dédiées.

[4] E.M. JAARA, "Étude et Implémentation Parallèles des Réseaux de Neurones ", Thèse de Doctorat, Faculté des Sciences d'Oujda, 19 Septembre 2000.

[5] H. NAIT CHARIF, *"A Fault Tolerant Learning Algorithm for Feedforward Neural Networks*" Conference FTPD 1996, Hawaï.

[6] H. PAUGAM-MOISY, "Réseaux de Neurones Artificiels : Parallélisme, Apprentissage et Modélisation" Habilitation à Diriger des Recherches, Ecole Normale Supérieure de Lyon, 6 Janvier 1997.

[7] PETER S. PATCHECO, *"Parallel Programming with MPI"*, Morgan Kaufman Publishers, 1997.