Correspondance

Simulation de la machine de Boltzmann en temps réel.

Real-Time Simulation of Boltzmann Machine

par Jacques-Olivier KLEIN, Hubert PUJOL, Patrick GARDA

Institut d'Électronique Fondamentale - C.N.R.S. u.r.a. 22 - Bât. 220
Université de Paris Sud - 91405 Orsay - France
Tél : (1) 69 41 65 72 - Fax : 60 19 25 93

Résumé

Nous présentons une évaluation de la durée des simulations de la machine de Boltzmann synchrone en phase de relaxation sur des stations de travail usuelles et des supercalculateurs vectoriels. Nous étudions l'impact des paramètres les plus importants : la topologie du réseau, la nature des connexions entre couches, le codage des états, la précision des calculs et l'architecture des machines utilisées. Nous proposons une méthode pour prédire l'ordre de grandeur des performances des différentes machines. Enfin nous montrons que ces performances permettent d'envisager à court terme l'utilisation d'une machine de Boltzmann de taille moyenne dans des applications pratiques.

Mots clés : évaluation de performances, implémentation logicielle et matérielle, réseaux de neurones formels, machine de Boltzmann.

Abstract

We present a performance evaluation of usual workstations and supercomputers for the simulation of synchronous Boltzmann machines during relaxation. We consider the impact of several parameters, such as the network architecture, the type of relationship between layers, the coding of the neurons states, the arithmetic and the architecture of the hardware. We give a method to forecast the average performance of different computers. Finally, we conclude that the real time simulation of medium size Boltzmann machines will be achieved by the next generation of microprocessors.

***Key words** : performance evalution, hardware and software implementation, artificial neural networks, Boltzmann machines.*

1. Introduction

Les Perceptrons multicouches et la rétropropagation du gradient constituent le modèle le plus utilisé dans les applications de réseaux de neurones formels. La machine de Boltzmann est pourtant reconnue comme un modèle intéressant sur le plan théorique, parce qu'elle introduit une dynamique stochastique dans les réseaux de neurones qui donne d'excellents résultats dans les problèmes de classifications complexes. Cependant elle reste peu explorée par la communauté, tant au niveau expérimental qu'applicatif. Ceci est principalement dû à la lenteur de ses simulations, qui résulte de la quantité importante de calculs à effectuer. En effet un algorithme déterministe direct, comme celui des Perceptrons multicouches, est toujours plus rapide qu'un algorithme stochastique itératif, comme celui de la Machine de Boltzmann. Dans ce contexte, l'objectif de cette publication est de comparer ces deux algorithmes *du point de vue de leur vitesse de simulation*, et de montrer que l'apparition de microprocesseurs de plus en plus puissants rend dès aujourd'hui envisageables des applications temps réel pour des machines de Boltzmann en phase de relaxation.

Cette étude ne prend pas en compte les algorithmes d'apprentissage utilisés pour la mise au point des réseaux. En effet dans cette situation les durées des simulations sont moins critiques et elles ne conditionnent pas la faisabilité d'une application temps réel. Dans le cas contraire, les neuro-calculateurs apportent un compromis intéressant entre la réduction de ces durées et la souplesse d'utilisation [6, 9, 10, 16, 18] . Néanmoins, ils ne sont pas envisagés dans cette étude, qui porte sur des tâches de classification exécutées par des ordinateurs séquentiels standards.

Nous commencerons par introduire la machine de Boltzmann synchrone, puis les réseaux que nous avons simulés. Nous présenterons ensuite nos programmes de simulation et les méthodes que nous avons utilisées pour évaluer leurs performances. Nous décrirons en détail l'évaluation d'une plate-forme standard, puis nous comparerons différentes générations de machines et nous

montrerons comment extrapoler les résultats. Pour finir, nous déterminerons les tailles de réseaux quil est possible de traiter en temps réel.

2. La machine de Boltzmann

2.1. GÉNÉRALITÉS

Dans un réseau de neurones formels, les neurones sont reliés entre eux par des connexions pondérées, formant ainsi un graphe dont les neurones sont les sommets et dont les connexions synaptiques sont les arcs [15] . Les nouveaux états des neurones sont calculés selon les états de leurs voisins dans le graphe. L'influence des voisins du neurone i sur celui-ci, appelée potentiel post-synaptique V_i, est égale à la somme de leurs états X_i pondérés par les poids synaptiques W_{ij} (cf. équation (1)). Le nouvel état est calculé à partir de l'image du potentiel post-synaptique par une fonction sigmoïdale (cf. équation (2)).

$$V_i = \sum_{i \neq j} W_{ij} \cdot X_j \tag{1}$$

$$F(V_i) = \frac{1}{1 + e^{-(V_i/T)}} \tag{2}$$

Le Perceptron multicouche [15] est un modèle de réseau de neurones formels déterministes à états multivalués : les neurones prennent directement pour état la valeur $F(V_i)$. Les calculs des états de neurones se font couche par couche, l'information se propageant de l'entrée vers la sortie.
La machine de Boltzmann [11] est un modèle de réseau de neurones formels stochastiques à états binaires. La probabilité que l'état d'un neurone prenne la valeur 1 suit la loi donnée par l'équation (3) où le paramètre T est un coefficient de stabilisation appelé température. Elle est lentement décroissante en apprentissage et maintenue constante en reconnaissance. Par ailleurs, les connexions sont bidirectionnelles et le processus est itératif : l'état des neurones cachés et de sortie est calculé plusieurs fois, jusqu'à la stabilisation statistique du comportement du réseau.

$$P(X_i = 1) = F(V_i) = \frac{1}{1 + e^{-(V_i/T)}} \tag{3}$$

2.2. MACHINE DE BOLTZMANN ASYNCHRONE ET SYNCHRONE

Un premier intérêt de la machine de Boltzmann est de fournir d'excellents résultats en reconnaissance, comme l'a montré T. Kohonen dans une étude comparant trois modèles de réseaux de neurones [12] . Ces résultats sont obtenus au prix d'une grande durée des simulations, causée par la nécessité de faire évoluer le réseau jusqu'à l'équilibre selon un processus itératif. Or, dans la machine de Boltzmann asynchrone, les neurones modifient leur état un à un. Pour accélérer les simulations, un modèle synchrone a été développé par R. Azencott, dans lequel tous les neurones mettent à jour leur état simultanément [2, 4] . Les équations de relaxation sont les mêmes que pour le modèle asynchrone, la différence entre les deux modèles se situant au niveau de la loi asymptotique et de l'apprentissage. Par rapport à un modèle synchrone déterministe, la machine de Boltzmann synchrone présente l'avantage d'éviter les cycles asymptotiques.
Pour notre étude, nous avons simulé le modèle synchrone. En effet, il nous apporte une accélération par rapport au modèle asynchrone même sur les calculateurs séquentiels, lorsque leurs unités de traitement sont pipelinées, comme c'est le cas pour la majorité des microprocesseurs actuels et pour les calculateurs vectoriels.

3. Les réseaux

3.1. INTRODUCTION

Pour évaluer les performances de différentes plates-formes de simulation, nous avons choisi plusieurs réseaux représentatifs parmi ceux qui sont utilisés dans les tâches de classification. Leurs caractéristiques sont résumées dans le tableau 1. Nous décrivons successivement les traits communs à ces réseaux et les exemples que nous avons utilisés.

Tableau 1. – Récapitulatif des réseaux utilisés.

Réseaux	Silhouette	Crête	NETtalk
Modèle	Machine de Boltzmann synchrone	Machine de Boltzmann synchrone	Perceptron multicouche
# connex. ponctuelles	804	0	0
# connex. locales	444	0	0
# connex. complètes	1054	30200	18426
# total connexions	2302	30200	18426
# neurones d'entrée	150	100	203
# neurones cachés	250	100	80
# neurones de sortie	4	100	26
# total neurones	405	301	310

Nous avons vu qu'un réseau de neurones formels constitue un graphe de relations entre neurones. Dans les applications de classification ce graphe possède une structure dans laquelle les neurones sont regroupés dans des ensembles appelés couches. Dans les réseaux que nous avons simulés, les liaisons entre deux couches peuvent être classées parmi trois types : ponctuelles, locales ou complètes (cf. figure 1). Dans le cas des relations ponctuelles, le ième neurone d'une couche est relié au ième neurone de l'autre couche. Dans le cas des relations locales, le neurone de rang i d'une couche est relié aux neurones de rangs $i-1, i$ et $i+1$ de l'autre couche. Enfin dans le cas des connexions complètes, tous les neurones d'une couche sont

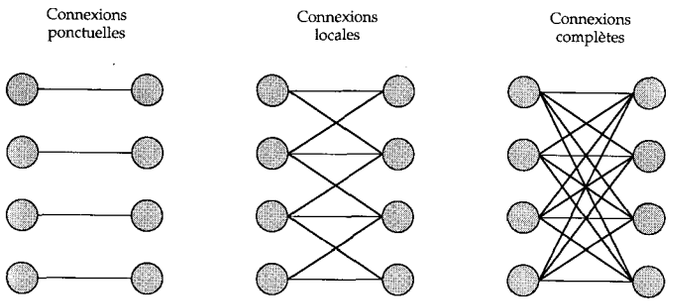


Figure 1. – Trois types de connexions entre couches.

reliés à tous les neurones de l'autre couche. Ces définitions des relations ponctuelles et locales, posées dans le cas de couches de même taille, peuvent être étendues au cas de couches de tailles différentes.

Ces trois types de relations sont représentatifs de l'architecture de la plupart des réseaux de neurones utilisés en reconnaissance de formes : des combinaisons de connexions locales pour le prétraitement et des connexions complètes pour la classification.

3.2. SILHOUETTE

Le premier réseau est destiné à la reconnaissance de silhouettes de bateaux extraites d'images infrarouges prétraitées [3, 4]. Il permet de reconnaître quatre classes de bateaux. C'est une machine de Boltzmann synchrone de 400 neurones et 2300 connexions. Il comprend trois couches d'entrée de 50 neurones, cinq couches cachées de 50 neurones et une couche de sortie de 4 neurones. Les relations entre couches de neurones appartiennent aux trois types définis plus haut : ponctuelles, locales et complètes, et le graphe des relations entres les couches est présenté sur la figure 2.

3.3. CRÊTE

Le deuxième réseau permet d'estimer les performances maximales d'une plate-forme matérielle pour la simulation de la machine de Boltzmann. En particulier, ce réseau est le seul qui donne la possibilité aux super-calculateurs dotés d'unités vectorielles pipelinées d'atteindre des performances proches de leurs performances crêtes. Ce réseau, qui n'est pas destiné à une application, est constitué de trois couches de 100 neurones : une couche d'entrée, une couche de sortie et une couche cachée complètement connectée aux deux autres.

3.4. NETtalk

NETtalk [17] est un réseau très connu qui a été introduit pour la synthèse de la parole. Il s'agit d'un réseau de 310 neurones répartis sur trois couches. La couche cachée (80 neurones) est

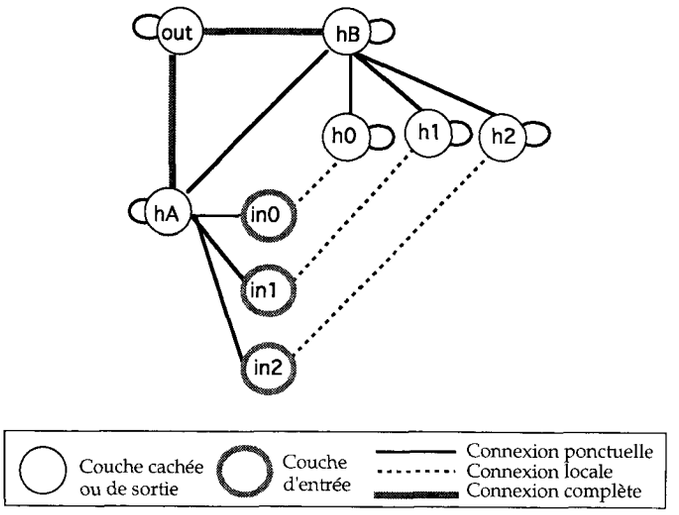


Figure 2. – Réseau Silhouette pour la classification de silhouettes de bateaux à partir d'images IR prétraitées.

entièrement connectée aux couches d'entrée (203 neurones) et de sortie (26 neurones). Les auteurs de [17] mentionnent l'utilisation d'une machine de Boltzmann pour la simulation de NETtalk, mais ces résultats n'ont pas été publiés. Nous utilisons ici le modèle du Perceptron multicouche pour comparer ses temps de simulation à ceux de la machine de Boltzmann.

Par ailleurs, la simulation de ce réseau nous permet de situer les performances de notre simulateur par rapport à celles d'un système de simulation de Perceptrons multicouches du domaine public : Aspirin/Migrains [14] . Ce logiciel est l'un des rares pour lesquels des mesures de vitesse de simulation de quelques réseaux de neurones soient données sur différentes plate-formes. Nous gagnons un facteur 1,5 en vitesse par rapport à Migrains pour la simulation de NETtalk. Comme les vitesses de simulation ont été soignées sur chacune des plates-formes sur lesquelles Aspirin/Migrains a été porté, ceci montre que les mesures que nous publions sont basées sur un logiciel optimisé.

4. Évaluation de performances

4.1. MESURE DE PERFORMANCES

4.1.1. Le paradoxe des MCPS

Pour les réseaux de neurones, il est habituel d'assimiler la vitesse de simulation à une vitesse moyenne de traitement des connexions V_s, exprimée en millions de connexions traitées par seconde (MCPS) [1], afin d'établir des comparaisons avec les résultats déjà publiés. Cependant cette mesure fournit une évaluation ambiguë des performances offertes par une plate-forme. Ceci est illustré par la variation des vitesses de simulation sur une SparcStation 2 des

réseaux NETtalk, Crête et Silhouette : 3,6 MCPS pour NETtalk, 3,3 MCPS pour Crête et 1,1 MCPS pour Silhouette. Nous avons recherché l'origine de ces variations en étudiant les différences entre ces réseaux. Trois facteurs peuvent avoir une influence sur les performances : le type de connexion, le modèle de neurone et le rapport entre les nombres des neurones et des connexions.

4.1.2. *Deux mesures plus fidèles*

Pour étudier l'influence de ces trois facteurs, la vitesse moyenne de traitement des connexions d'un réseau est insuffisante. Afin d'évaluer séparément l'influence des neurones et des connexions, nous avons défini deux nouvelles grandeurs. Nous mesurons d'une part le temps moyen de traitement d'une connexion (T_c) et d'autre part le temps moyen de mise à jour de neurone (T_n).

Il est alors possible de retrouver le temps de simulation (T_s) d'un réseau grâce à la relation (4), connaissant le nombre de connexions C, le nombre N de neurones, le nombre I d'itérations de relaxation et les temps de calcul moyen des connexions et des mises à jour d'état de neurone. Il est également possible de calculer la vitesse moyenne de traitement des connexions pour la simulation de ce réseau (relation 5).

$$T_s = I \cdot (N \cdot T_n + C \cdot T_c) \tag{4}$$

$$V_s = \frac{C}{N \cdot T_n + C \cdot T_c} \tag{5}$$

4.1.3. *Mesure du temps d'exécution*

Pour toutes nos mesures, le temps d'exécution est défini comme le temps *user* donné par la commande Unix *time*. Les initialisations présentes au début du programme (calcul d'une table pour la sigmoïde et initialisation des poids synaptiques) sont prises en compte mais elles ne faussent pas les mesures. En effet nous nous sommes assurés que la part de temps de calcul qui leur incombe est au plus de 1% alors que les temps d'exécution fluctuent d'environ 3%.

4.2. BENCHMARKS

Nous avons donc défini deux classes de benchmarks. Nous avons écrit d'une part des programmes destinés à mesurer séparément le temps moyen de calcul de chaque opération (T_c et T_n) et d'autre part des programmes de simulation de réseaux pour mesurer les vitesses moyennes de simulation (V_s).

Le temps moyen de traitement des connexions est estimé par la simulation de réseaux ne comprenant que des connexions et aucun neurone (relation (6)). Trois variantes sont envisagées pour les trois types de connexions : ponctuelles, locales, complètes.

De même le temps moyen de mise à jour des neurones est estimé par la simulation de réseaux composés uniquement de neurones, sans connexions, soit déterministes soit stochastiques (relation (7)).

Enfin la vitesse de simulation est estimée par la simulation des trois réseaux que nous avons introduits précédemment (relation (8)).

$$T_c = \frac{T_s}{I \cdot C} \quad \text{avec} \quad N = 0 \tag{6}$$

$$T_n = \frac{T_s}{I \cdot N} \quad \text{avec} \quad C = 0 \tag{7}$$

$$V_s = \frac{I \cdot N}{t_s} \tag{8}$$

Pour chacun de ces programmes nous avons écrit quatre versions correspondant à quatre modes de calculs : des calculs flottants simple précision, des calculs flottants vectorisés, des calculs entiers et des calculs entiers où le produit du poids synaptique par l'état du neurone (cf. équation (1)) est remplacé par un masquage. Dans ce cas l'état 0 est codé 0 et l'état 1 est codé -1 en complément à deux. Cet état sert de masque pour un ET logique appliqué sur le poids synaptique. Nous ne pouvons pas utiliser le masquage à la place de la multiplication pour NETtalk dont les états des neurones cachés ne sont pas binaires.

5. Impact des algorithmes

5.1. INTRODUCTION

Dans ce chapitre nous fournissons les résultats des mesures de performances sur une station de travail très répandue : la SparcStation 2. Nous donnons d'abord les performances des programmes n'effectuant que le traitement des connexions, puis celles des programmes n'effectuant que des mises à jour d'états de neurones, enfin nous examinons les performances des réseaux Silhouette, Crête et NETtalk.

5.2. INFLUENCE DU TYPE DE CONNEXION

La figure 3 donne les temps de traitement des connexions en fonction du type de relation entre couches et de l'arithmétique utilisée. Elle montre tout d'abord que les temps les plus courts sont obtenus en calcul entier en utilisant le masquage à la place de la multiplication. Elle montre ensuite que les calculs flottants sont plus rapides que les calculs entiers, prouvant ainsi que c'est la multiplication qui est pénalisante en calcul entier. Les meilleures performances des multiplications flottantes par rapport aux multiplications entières peuvent s'expliquer d'une part par la présence d'un coprocesseur flottant comportant un multiplieur câblé et d'autre part par l'exécution simultanée d'opérations entières et flottantes au sein du processeur. Par ailleurs, nous constatons de meilleures performances avec des connexions complètes, qui sont

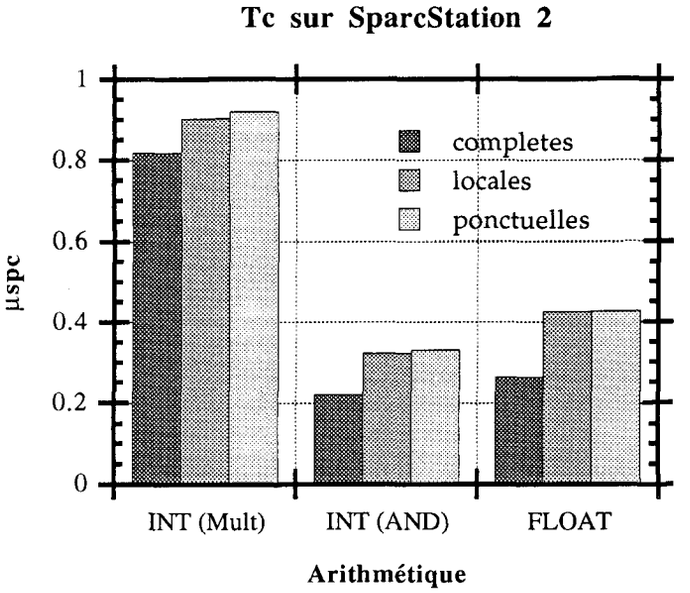


Figure 3. – Temps de calcul moyen des connexions (en μs par connexion) sur une SparcStation2.

Tableau 2. – Tableau récapitulatif des principales caractéristiques des programmes utilisés.

Grandeur mesurée	Réseau	Type de connexions utilisées	Type de mise à jour utilisée	Architecture réseau	Remarques
T_c	Connexion ponctuelle	Ponctuelles	–	100-100-100	Prétraitements locaux
T_c	Connexion locale	Locales	–	100-100-100	Prétraitements locaux
T_c	Connexion complète	Complètes	–	100-100-100	Classification
T_n	Mise à jour déterministe	–	Déterministe	100-100-100	Perceptron
T_n	Mise à jour stochastique	–	Stochastique	100-100-100	Machine de Boltzmann
V_s	NETtalk	Complètes	Déterministe	203-80-26	NETtalk
V_s	Silhouette	Tous	Stochastique	Cf. figure [2]	Silhouettes de bateaux IR
V_s	Crête	Complètes	Stochastique	100-100-100	Performances "crête"

plus régulières et minimisent le nombre d'accès à la mémoire : un registre est alors utilisé pour accumuler les potentiels post-synaptiques en cours de calcul. Pour les deux autres types de connexions (locales et ponctuelles) les accumulations sur le potentiel post-synaptique se font systématiquement avec deux accès supplémentaires à la mémoire : une lecture avant chaque somme, et une écriture après.

5.3. INFLUENCE DU MODÈLE DE NEURONE

Dans le cas de la rétropropagation du gradient, un simple accès à une sigmoïde tabulée effectue la mise à jour des états des neurones.

Dans le cas de la machine de Boltzmann il faut ajouter un tirage aléatoire et une comparaison. Le temps de calcul s'en trouve évidemment augmenté. Les mesures présentées sur la figure 4 montrent que l'on accroît ainsi d'un facteur 5 à 10 le temps de mise à jour des neurones, exprimé en μs par neurone (μspn).

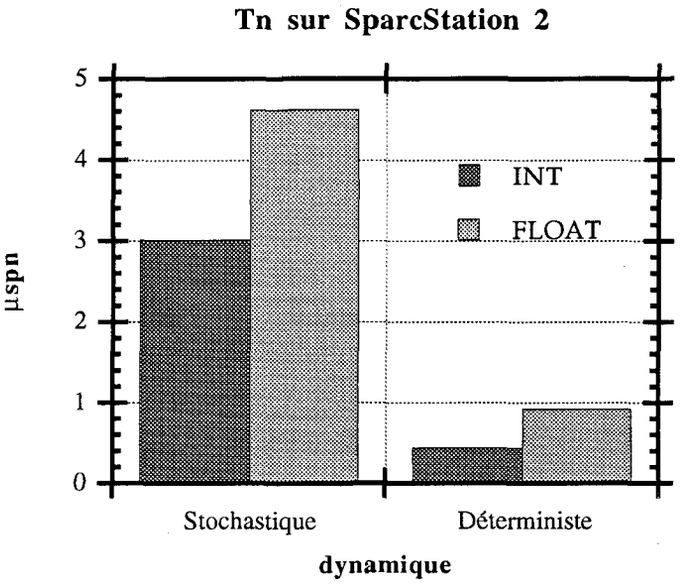


Figure 4. – Temps de calcul moyen des mises à jour d'états de neurones (en μs par neurone) sur une SparcStation 2.

5.4. INFLUENCE DE L'ARCHITECTURE DU RÉSEAU DE NEURONES

La simulation des réseaux de neurones implique des opérations linéaires, comme le calcul des potentiels (relation (1)), et non linéaires, comme la mise à jour des états des neurones (relation (2)). Utiliser la vitesse moyenne de traitement des connexions pour mesurer des performances suppose implicitement que l'opération prépondérante dans la simulation des réseaux de neurones est le traitement de la connexion, et que le temps pris par le calcul des nouveaux états de neurones est négligeable. Cette hypothèse est vérifiée pour un Perceptron multicouche à connexion complète, comme NETtalk, puisque la mise à jour de l'état des neurones représente moins de 2% du temps total (tableau 3). Cette hypothèse reste vraie dans une moindre mesure pour une machine de Boltzmann à connexions complètes, comme le réseau Crête, puisque la mise à jour de l'état des neurones représente moins de 10% du temps total. La différence provient de la

Tableau 3.

	Flottant			Entier		
Réseaux	V_s (MCPS)	% connexion	% neurone	V_s (MCPS)	% connexion	% neurone
Crête	3,3	90 %	10 %	4,0 (AND)	92 %	8 %
Silhouette	1,1	45 %	55 %	1,4 (AND)	49 %	51 %
NETtalk	3,6	99,7 %	0,3 %	1,2 (MULT)	98 %	2 %

mise à jour stochastique du neurone qui est plus lente que la mise à jour déterministe, comme nous l'avons vu au paragraphe précédent. Par contre, cette hypothèse devient franchement fausse dans une machine de Boltzmann utilisant des connexions locales et ponctuelles, comme le réseau Silhouette, puisque la mise à jour de l'état des neurones représente plus de 50% du temps total. La différence provient cette fois du rapport entre le nombre de connexions et de neurones qui est bien plus faible que dans un réseau n'utilisant que des connexions complètes.

5.5. CONCLUSION

Nous sommes maintenant en mesure d'expliquer les différences de vitesse de simulation de différents réseaux sur une même station de travail standard. D'abord l'arithmétique entière utilisant le masquage est plus rapide que l'arithmétique flottante, qui est elle même plus rapide que l'arithmétique entière qui utilise la multiplication. Ensuite les connexions complètes sont traitées plus rapidement que les connexions locales ou ponctuelles. De plus, le calcul des états des neurones est 5 à 10 fois plus long pour des neurones stochastiques que pour des neurones déterministes. Enfin, la part du temps de calcul consacrée au traitement des connexions n'est représentative de la durée de la simulation que si le rapport entre le nombre de connexions et de neurones est grand. Par ailleurs ceci confirme que la vitesse moyenne de la simulation n'est pas la mesure la plus pertinente.

6. Impact de l'architecture de la plate-forme

6.1. INTRODUCTION

Pour étudier la faisabilité d'une application temps réel utilisant une machine de Boltzmann il faut être capable de prévoir, même grossièrement, la durée des simulations. Pour cela il faut non seulement étudier l'impact de l'architecture des réseaux de neurones, comme nous l'avons fait précédemment, mais également étudier l'influence des plates-formes de simulations. Hélas, ce problème est très complexe car il fait intervenir un très grand nombre de caractéristiques architecturales ou technologiques telles que la fréquence d'horloge, le jeu d'instruction, la hiérarchie mémoire, le nombre d'unités fonctionnelles pouvant fonctionner en parallèle, les performances du compilateur, etc. Dans ce chapitre, nous allons donc, plus modestement illustrer l'impact de l'architecture en étudiant le cas d'une machine possédant des unités vectorielles : le Cray YMP-el. Ensuite, nous présenterons les vitesses de simulation de plusieurs générations de machines. Nous en déduirons enfin les tailles de réseau qui peuvent être simulés en temps réel.

6.2. LE CAS DU CRAY YMP-el

Dans cette section nous allons étudier le cas du Cray YMP-el qui est doté d'unités vectorielles. Le gain qui peut être obtenu par l'utilisation d'unités vectorielle est directement conditionné par la possibilité d'extraire de l'algorithme des opérations sur des vecteurs suffisamment longs. La longueur de ces vecteurs sera d'autant plus importante que la stratégie de parcours de la matrice des connexions pourra être librement choisie.

C'est ici qu'apparaît l'intérêt pour la simulation de la machine de Boltzmann synchrone. En effet, elle permet le choix de la stratégie adoptée pour parcourir, à chaque itération, tous les éléments non nuls de la matrice des connexions. En particulier, pour les connexions locales ou ponctuelles, nous pouvons parcourir les diagonales des sous-matrices représentant les connexions entre deux couches. Les vecteurs ont alors sensiblement la même taille que les couches de neurones, ce qui conduit à une utilisation beaucoup plus efficace des unités vectorielles. De plus, pour des connexions complètes, nous pouvons utiliser les produits matrice-vecteur de la librairie BLAS 2 qui sont particulièrement rapides puisque l'accélération qui en résulte est environ d'un facteur 10. Enfin la machine de Boltzmann synchrone permet de vectoriser les opérations de mises à jour des neurones bien qu'elles ne soient pas linéaires.

Dans le cas de la machine de Boltzmann asynchrone, le calcul des connexions locales conduit à des vecteurs de plus petites tailles, les produits matrice-vecteur de BLAS 2 pour les connexions complètes ne peuvent pas être utilisés et les mise à jour de neurones ne sont pas vectorisables. Ceci justifie donc le choix du modèle synchrone de la machine de Boltzmann.

Nous avons déjà présenté les résultats (cf. tableaux 5 et 6). Ils montrent que les unités de traitement vectorielles permettent un gain important pour les opérations linéaires (calcul du potentiel post-synaptique), par rapport aux architectures scalaires des autres plates-formes, comme le révèlent les bonnes performances sur le réseau *Crète*. Par contre, les opérations non linéaires (mise à jour des états des neurones), bien que vectorisées, sont exécutées d'une manière moins efficace. Ce point est mis en évidence par les performances obtenues sur le réseau *Silhouette*, qui implique 50% d'opérations non linéaires.

Bien qu'il utilise une technologie proche de celle des microprocesseurs classiques (CMOS à 33 MHz), le Cray YMP-el apporte une accélération significative des simulations. Nous pouvons nous attendre à retrouver cette accélération avec les nouveaux microprocesseurs dotés d'unités super-pipeline. Ceci montre que le choix d'un modèle synchrone de machine de Boltzmann se justifie même sur une plate-forme séquentielle.

Tableau 4. – Résultat des benchmarks Linpack et Dhrystone sur les machines utilisées.

Machines	Dhrystone (MIPS)	Linpack-100 (MFLOPS)
Apollo DN 3000	1,2	0,071
Apollo DN 4000	4	0,14
SUN SparcStation 2	28,5	6,1
IBM RS 6000-320	30	7
SparcStation 10-20	40	10
CRAY YMP-el	–	32

Tableau 5. – Performances de différentes générations de machines en entier.

Machines	Dhrystone	Connexion complète	Connexion locale	Connexion ponctuelle	Modèle stochastique	Modèle déterministe	Réseau silhouette	Réseau crête	Réseau nettalk
	Ve (en MIPS)	Tc (en μspc)	Tc (en μspc)	Tc (en μspc)	Tn (en μspn)	Tn (en μspn)	Vs (en MCPS)	Vs (en MCPS)	Vs (en MCPS)
Apollo DN 3000	1,20	3,94	4,06	4,11	34,10	4,90	0,12	0,21	0,05
Apollo DN 4000	4,00	1,70	1,85	1,92	14,97	1,76	0,29	0,54	0,20
SUN SparcStation 2	28,00	0,22	0,32	0,33	3,01	0,44	1,42	4,00	1,24
IBM RS 6000-320	30,00	0,30	0,37	0,37	2,06	0,41	1,56	3,20	2,48
SparcStation 10-20	40,00	0,15	0,15	0,16	2,39	0,18	2,19	5,77	1,23

Tableau 6. – Performances de différentes générations de machines en flottant.

Machines	Linpack	Connexion complète	Connexion locale	Connexion ponctuelle	Modèle stochastique	Modèle déterministe	Réseau silhouette	Réseau crête	Réseau nettalk
	Vf (en MFLOPS)	Tc (en μspc)	Tc (en μspc)	Tc (en μspc)	Tn (en μspn)	Tn (en μspn)	Vs (en MCPS)	Vs (en MCPS)	Vs (en MCPS)
Apollo DN 3000	0,07	47,20	44,90	45,70	144,90	30,90	0,01	0,02	0,02
Apollo DN 4000	0,14	21,30	20,60	20,50	69,70	12,90	0,03	0,04	0,04
SUN SparcStation 2	6,10	0,26	0,43	0,43	4,62	0,92	1,10	3,30	3,66
IBM RS 6000-320	7,00	0,32	0,26	0,26	5,47	1,88	1,05	3,59	3,57
SparcStation 10-20	10,00	0,16	0,26	0,26	3,52	0,44	1,64	6,00	5,91
CRAY YMP-el	32,00	0,02	0,12	0,22	0,93	0,46	1,96	27,10	24,00

6.3. PEUT-ON PRÉDIRE LES PERFORMANCES D'UNE MACHINE?

Dans ce chapitre, nous présentons les vitesses de simulation de plusieurs générations de machines. Le tableau 4 donne les performances attribuées à ces plates-formes par les benchmarks usuels en calcul flottant (Linpack - 100). Les tableaux 5 et 6 donnent les performances que nous avons mesurées en exécutant les programmes décrits dans le § 4.2 sur les mêmes plates-formes, pour différentes arithmétiques. Elles laissent entrevoir une progression analogue à celle des performances de calcul brut présentées sur le tableau 4. Nous pouvons vérifier que l'ordre de grandeur des vitesses moyennes de simulation varie sensiblement en proportion des vitesses moyennes de calcul flottant V_f (en MFLOPS-Linpack) (cf. figures 5 et 6). Nous pouvons donc prévoir un ordre de grandeur des performances de simulation de réseaux de neurones en extrapolant les performances des benchmarks numériques.

Pour aller au-delà, nous pouvons estimer sur plate-forme Y les produits $(V_f \cdot T_c)^Y$ et $(V_f \cdot T_n)^Y$ (cf. tableau 7), qui ont respectivement les dimensions d'un nombre d'opérations flottantes par connexion et d'opérations flottantes par neurone. Ceci nous permet théoriquement de prévoir a priori les temps de simulations T_n^X et T_c^X sur une nouvelle plate-forme X, connaissant la mesure de sa vitesse en calcul flottant V_f^X en MFLOPS, grâce aux relations (9) et (10).

$$T_n^X = \frac{(V_f \cdot T_n)^Y}{V_f^X} \tag{9}$$

Tableau 7. – Nombre moyen d'opérations flottantes par connexion et par neurone.

# Opérations flottantes	Connexions complètes	Autres connexions	Neurones
Moyenne	2,5	2,6	22
Ecart type	0,75	0,6	13

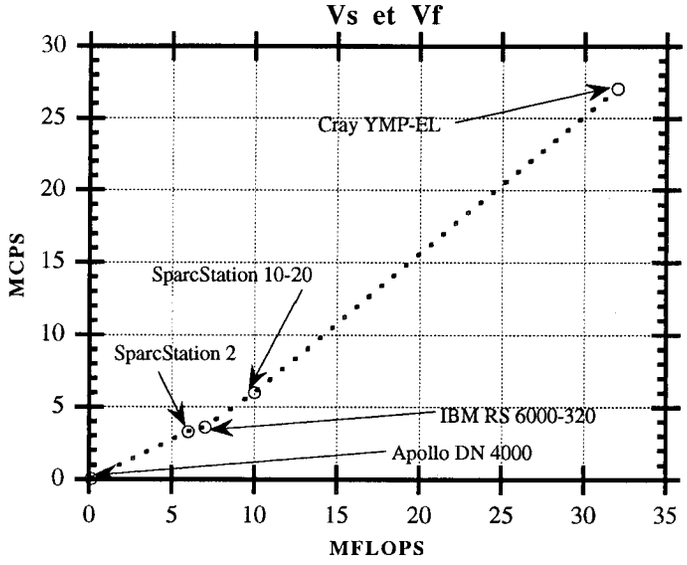


Figure 5. – Vitesse de simulation sur le réseau crête en fonction de la vitesse de calcul flottant des machines.

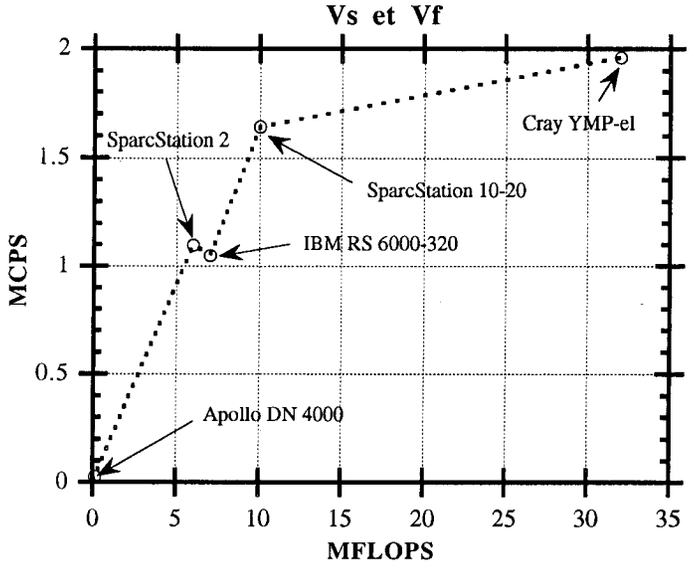


Figure 6. – Vitesse de simulation sur le réseau silhouette en fonction de la vitesse de calcul flottant des machines.

$$T_c^X = \frac{(V_f \cdot T_c)^Y}{V_f^X} \tag{10}$$

En fait, nos mesures montrent que cette approximation linéaire est raisonnable pour les connexions qui mettent en jeu des calculs linéaires, comme le benchmark utilisé pour estimer V_f, mais qu'elle est très grossière pour les mises à jour d'état de neurone, qui utilisent des calculs non linéaires. Dans la pratique, pour prédire les temps de simulation sur une plate-forme de cette manière, il nous faudra extrapoler à partir des mesures sur une plate-forme dont l'architecture soit proche. La prévision sera d'autant meilleure que la différence de vitesse en calcul flottant résultera de différences technologiques et non de différences d'architecture. Nous avons utilisé cette méthode pour prédire les vitesses de simulations sur des Cray YMP et le C90 utilisant une technologie plus agressive que le Cray YMP-el (cf. tableau 8). Comme ces processeurs sont parmi les plus rapides au monde, cette extrapolation permet d'évaluer les vitesses maximales de simulation qui peuvent être atteintes aujourd'hui par des processeurs séquentiels.

Tableau 8. – Estimation des performances des Cray.

Machine	Horloge (ns)	Linpack 100 (MFLOPS)	Silhouette (MCPS)	Crête (MCPS)
YMP 1-EL	30	32	1,9	27
YMP 1	6	161	9,5	135
C 90-1	4,2	387	23	325

La comparaison des performances en MCPS aux performances en MIPS, montre qu'une approche similaire est possible pour les calculs entiers (figure 7).

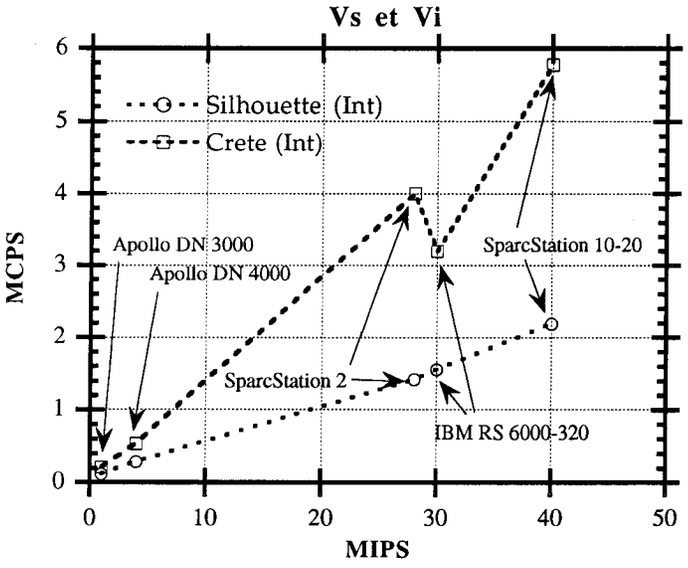


Figure 7. – Vitesse de simulation sur les réseaux crête et Silhouette en fonction de la vitesse de calcul flottant des machines.

6.4. TEMPS RÉEL ET TAILLE DE RÉSEAUX

Nous allons maintenant nous intéresser aux tailles des réseaux que l'on peut simuler en temps réel. Dans la suite, nous parlerons de traitement temps réel d'un réseau de neurones, si une classification peut être effectuée en une durée inférieur à T_{max}. Le temps de classification peut être calculé en utilisant la relation (4). Ceci nous conduit à préciser la taille des réseaux qui peuvent être traités en

temps réel par l'inégalité linéaire (11), où I représente le nombre d'itérations de relaxation synchrone, N le nombre de neurones calculés, C le nombre de connexions, T_n le temps de calcul d'un neurone et T_c le temps de calcul d'une connexion. Les tailles des plus grands réseaux pouvant être traités en temps réel sont alors données par la relation (2) qui est l'équation d'une droite dans le plan Neurone x Connexion.

$$I \cdot (N \cdot T_n + C \cdot T_c) < T_{max} \tag{11}$$

$$C = \frac{T_{max}}{I \cdot T_c} - N\frac{T_n}{T_c} \tag{12}$$

Nous pouvons ainsi représenter graphiquement les tailles de réseaux qui peuvent être traités en temps réel. Dans les exemples qui suivent nous avons pris $T_{max} = 40ms$ (par référence à la cadence vidéo) et $I = 100$ (nombre typique d'itérations pour la relaxation d'une machine de Boltzmann synchrones en phase de reconnaissance [13]). Sur les figures 8 et 9 chaque point correspond à une taille de réseau, avec le nombre de neurones en abscisse, et le nombre de connexions en ordonnée. Chaque segment représente la taille des plus gros réseaux qui peuvent être traités en temps réel sur une plate-forme avec des calculs entiers (cf. figure 8). Par exemple le réseau Silhouette peut être traité en temps réel par une Sparc 10-41 en entier. Par ailleurs, nous montrons sur la figure 9 les tailles de réseaux qui peuvent être traités en temps réel sur un processeur de l'un des supercalculateurs vectoriels les plus rapides au monde actuellement, le Cray C 90 : ils comportent de l'ordre de 2000 neurones et 100 000 connexions. Nous sommes donc loin de pouvoir traiter en temps réel les réseaux étudiés en traitement d'image, comportant quelques centaines de milliers de neurones et quelques millions de connexions [7, 13] . Pour ces réseaux des architectures parallèles spécialisées restent indispensables.

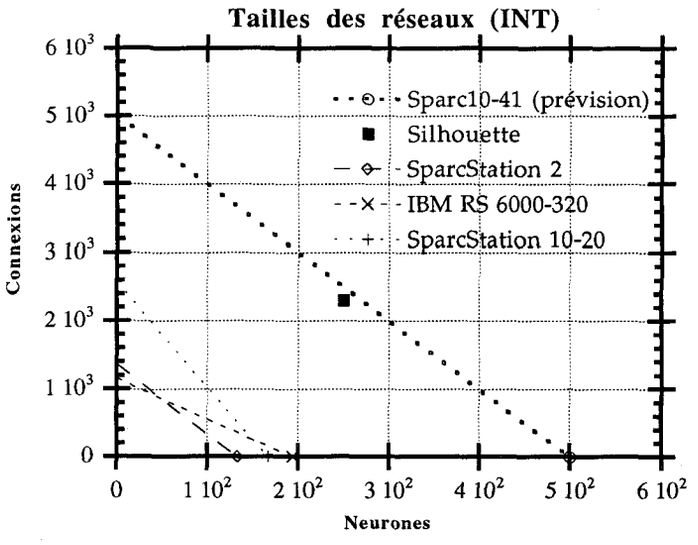


Figure 8. – Tailles des réseaux qui peuvent être traités en temps réel en calcul entier sur différentes machines.

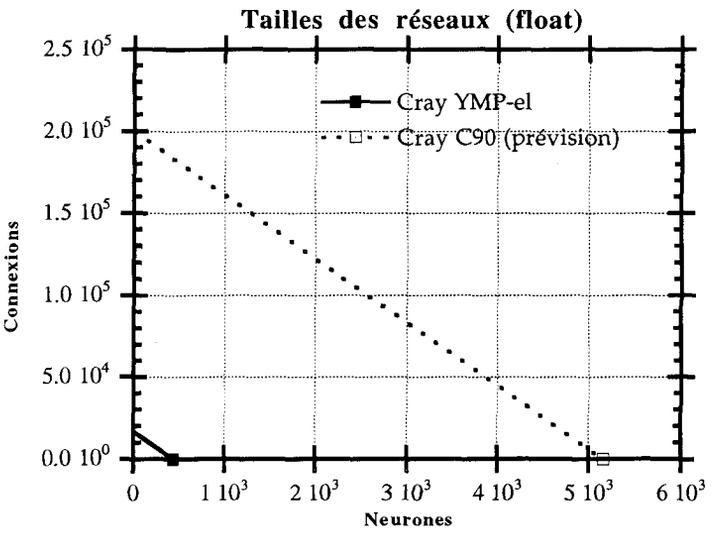


Figure 9. – Tailles des réseaux qui peuvent être traités en temps réel en calcul flottant sur un processeur du Cray C90 (380 Mflops).

7. Conclusion

Dans cette publication, nous avons évalué les performances de différentes plates-formes matérielles pour la simulation de machines de Boltzmann synchrones en phase de relaxation. Nous avons choisi des réseaux destinés à des tâches de classification, et nous avons introduit et utilisé de nouvelles unités de mesure. Nous avons montré ensuite que les vitesses des simulations progressent linéairement avec la vitesse de calcul flottant des plates-formes. Notre méthode permet d'obtenir un ordre de grandeur des performances suffisant pour faire des comparaisons et des prévisions.
Ainsi, avec une machine de Boltzmann de taille moyenne, telle que le réseau *Silhouette* de 400 neurones et 2300 connexions pour la reconnaissance de silhouettes de bateaux, une classification est effectuée en 160 ms seulement sur une station de travail de type Sparc 2 ou IBM RS 6000-320. Nous pouvons dès lors espérer une reconnaissance de formes « temps réel » (40 ms), avec la prochaine génération de plates-formes (IBM 560 ou SparcStation 10-41 par exemple), pour lesquelles un gain d'un facteur 4 sur la vitesse de calcul est attendu. Ceci prouve que, même quand les contraintes de vitesses sont importantes, la machine de Boltzmann est effectivement utilisable pour les applications où les performances de reconnaissance des réseaux de neurones déterministes sont insuffisantes.
D'un autre côté, les performances obtenues sur Cray YMP-el montrent que le traitement des très gros réseaux utilisés en imagerie, comportant quelques centaines de milliers de neurones et quelques millions de connexions [7, 13] ne peut pas être envisagé à court terme sur des plates-formes standards. En effet, des extrapolations à des réseaux de cette taille conduisent à des durées de simulation de l'ordre de quelques secondes qui sont incompatibles avec des contraintes de temps réel. Aujourd'hui, seules des architectures parallèles spécialisées permettent de fournir la

puissance de calcul nécessaire dans les applications au traitement d'image [5, 8].

Table des abréviations employées	
MCPS	Million de connexions par seconde
μspn	Micro seconde par neurone
μspc	Micro seconde par connexion
MFLOPS	Million d'opérations flottantes par seconde
MIPS	Million d'instructions par seconde
I	Million d'instructions de relaxation
N	Nombre de neurones
C	Nombre de connexions
T_c	Temps moyen de traitement par connexion (μspc)
T_n	Temps moyen de traitement par neurone (μspn)
Vi	Vitesse de calcul en entier (MIPS)
Vf	Vitesse de calcul en flottant (MFLOPS)
Ts	Temps de simulation d'un réseau (μs)
V_s	Vitesse de simulation d'un réseau (MCPS)

Remerciements

Cette étude a été mené dans le cadre du projet RA soutenu par le PRC/GDR ANM. Nous tenons à remercier tout particulièrement Robert Azencott, Jérôme Lacaille et Laurent Younès du DIAM, Eric Belhaire, Antoine Dupret, Antoine Dupuy, Franck Elie, Wang Wenqing et Yiming Zhu de l'opération *machine neuronales pour la vision* à l'IEF pour leur collaboration et leurs conseils.

BIBLIOGRAPHIE

[1] DARPA, Neural Network Study. 1988, *AFCEA International Press.*

[2] R. AZENCOTT, Synchronous Boltzmann Machines and Gibbs Field : Learning Algorithms. in *Neurocomputing : algorithms, architectures and applications*, 1990, Les arcs : Springer-Verlag.

[3] R. AZENCOTT, A. DOUTRIAUX, and L. YOUNES, Synchronous Boltzmann Machines and outline based classification. in *Proc. of International Neural Network Conf. Paris*, 1990, Boston : IEEE Kluwer Academic Publishers.

[4] R. AZENCOTT, A. DOUTRIAUX, and L. YOUNES, Synchronous Boltzmann Machines and Curve Identification tasks 1992, LMENS, ENS, 45 rue d'Ulm, 75005 Paris.

[5] E. BELHAIRE, Contribution à la réalisation électronique de Réseaux de Neurones Formels : Intégration Analogique d'une Machine de Boltzmann 1992, thèse de l'université Paris XI - Orsay.

[6] P. BESSIERE et al., From Hardware to software : Designing a « neurostation », in *VLSI Design of Neural Networks*, U.R.a.U. Rückert, Editors. 1991, Kluwer Academic Publishers pp. 311-335.

[7] B.E. BOSER, et al., An Analog Neural Network Processor with Programmable Topology, JSSC, 1991. **26** (12) pp. 2017-2025.

[8] P. GARDA and E. BELHAIRE, An analog chip set with digital I/O for Synchronous Boltzmann Machine, in *VLSI for Artificial Intelligence and Neural Network*, J.G.D.-F.a.W.R. Moore, Editors. 1990, Kluwer Academic : Boston. pp. 245-254.

[9] A. GUERIN et J. HERAULT, CRASY une architecture de calcul reconfigurable pour la simulation de réseaux de neuromimétiques. *Traitement du Signal*, 1988. **5** p. 177-186.

[10] D. HAMMERSTROM, A highly parallel digital architecture for neural network emulation, in *VLSI for Artificial Intelligence and Neural Networks*, J.G.D.-F.a.W.R. Moore, Editors. 1991, Plenum Press : New York. pp. 357-366.

[11] G.E. HINTON and T.J. SEJNOWSKI, Learning and Relearning in Boltzmann Machines, in *Parallel Distributed Processing*, D.E.R.a.J.L. McClelland, EditorÊditors. 1986, Bradford-MIT Press : Cambridge, MA. pp. 282-317.

[12] T. KOHONEN, R. CHRISLEY, and G. BARNA, Statistical Pattern Recognition with Neural Networks : Benchmarking Studies. in *Neural Networks from Models to Applications*, 1989, Paris : I.D.S.E.T.

[13] J. LACAILLE, Machine de Boltzmann : Théorie et Applications 1992, thèse de l'universite Paris XI.

[14] LEIGHTON, The Aspirin/MIGRAINES Software Tools User's Manual Release V5 1991, MITRE Corporation.

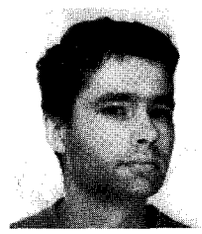
[15] M. MILGRAM, Reconnaissance des formes - Méthodes numériques et connexionnistes, ed. A. Colin. 1993.

[16] U. RAMACHER, SYNAPSE - A neurocomputer that synthesizes neural algorithms on parallel systolic engine, *Journal of Parallel Distributed Processing*, 1992. **14**, n° 3, pp. 306-318.

[17] T.J. SEJNOWSKI and C.R. ROSENBERG, Parallel Networks that Learn to Pronounce English Text, *Complex systems*, 1987, **1** pp. 145-168.

[18] J.B. THEETEN et al. The LNEURO-CHIP : A Digital VLSI With On-Chip Learning Mechanism. in International Neural Network Conference 90 Paris, 1990, Paris : IEEE, Kluwer Academic Publishers.

LES AUTEURS



Jacques-Olivier KLEIN est titulaire d'un DEA d'Électronique en architecture microélectronique des machines dédiées (Orsay 1990). Il prépare depuis octobre 1991 une thèse à l'Institut d'Électronique Fondamentale sur l'architecture d'un processeur mixte analogique numérique dédié à la simulation de réseaux de neurones en traitement d'images et en reconnaissance de formes. Il est responsable depuis 1992 du Réseau Doctoral en Architecture des Systèmes et Machines Informatiques pour le pôle Ile de France qu'il anime en collaboration avec Hubert Pujol.

Hubert PUJOL est titulaire d'une agrégation de génie électrique et d'un DEA d'Électronique en architecture microélectronique des machines dédiées (Orsay 1991). Il prépare depuis octobre 1991 une thèse à l'Institut d'Électronique Fondamentale sur les réseaux d'interconnexion pour machines parallèles. Il est responsable depuis 1992 du Réseau Doctoral en Architecture des Systèmes et Machines Informatiques pour le pôle Ile de France qu'il anime en collaboration avec Jacques-Olivier Klein.

Patrick GARDA est Chargé de Recherche au CNRS et Responsable du projet « Machines Neuronales pour la Vision » à l'Institut d'Électronique Fondamentale. Ses centres d'intérêt portent sur la conception et l'évaluation d'architectures parallèles pour les réseaux de neurones et la vision par ordinateur.

Manuscrit reçu le 7 juin 1993.