CHAPITRE 3

Parallélisation du classifieur par partage de l'espace d'entrée

Introduction

Ce chapitre traite d'une parallélisation du classifieur par partage de l'espace d'entrée. Les exemples sont partagés[1] entre les processeurs, chaque processeur possède donc un morceau de chaque exemple.

Choix d'une configuration matérielle

Nous avons mis au point cette première parallélisation sur une machine parallèle, ce choix est justifié par une analyse de la granularité d'une parallélisation par partage de l'espace d'entrée (section 3.1). Les caractéristiques principales de la machine parallèle utilisée et la topologie d'interconnexion des processeurs choisie sont détaillées.

Prétraitement

Nous avons choisi de paralléliser l'application de reconnaissance de dessins au trait présentée dans la section 1.2. Il a été vu au chapitre 1 qu'il est indispensable de paralléliser également le prétraitement du problème d'OCR car il utilise intensivement le processeur, notamment au regard du temps de classification (voir la figure 1.5 page 21). La parallélisation du prétraitement est étudiée section 3.2, l'image est découpée en bandes horizontales. Chaque région ainsi formée, est confiée à un processeur. Les cellules à champ récepteur sont alors activées par chaque processeur (détection des petits segments orientés) sur la région dont ils ont la charge. Le prétraitement terminé, chaque processeur possède un bloc de la matrice tridimensionnelle (nous dirons « 3D ») des cellules à champ récepteur.

Classification

La parallélisation de l'algorithme d'apprentissage du classifieur, c'est-à-dire l'activation et la mise à jour des prototypes, laisse plus de liberté. Deux stratégies sont

1. La notion de « partage » est ambigü e en français. Dans cette thèse, « partager un exemple entre des processeurs » signifie que chaque processeur possède un morceau de chaque exemple. Dans les prochains chapitres, nous parlerons de la « distribution des exemples entre les processeurs » pour signifier que chaque processeur possède une partie de l'ensemble des exemples.

étudiées (section 3.3), le partage (ou découpage) des prototypes et la distribution des prototypes. La deuxième solution est retenue et mise en œuvre (section 3.4). L'évolution des temps de prétraitement et de classification est suivie durant une série d'apprentissages pour déterminer l'accélération atteinte.

3.1 Granularité du partage de l'espace d'entrée

Nous qualifions de « moyenne » la granularité de la parallélisation par partage de l'espace d'entrée. La notion de granularité étant relative, ce choix impose une explication. Le partage de l'espace d'entrée revient à confier aux processeurs des morceaux de réseau. Ce cas de figure est un compromis entre la granularité la plus faible (c'est-à-dire une cellule par processeur) et la granularité la plus grossière (le réseau complet sur chaque processeur).

Choix d'une machine architecture

Le réseau étant découpé, la propagation de l'influx va susciter de nombreuses communications entre les processeurs. Nous avons vu que les performances en communication des systèmes distribués, notamment sous l'environnement PVM, restent inférieures à ce que peuvent faire les machines parallèles (autant pour la bande passante que pour la latence). Aussi, nous choisissons d'utiliser exclusivement une machine parallèle plutôt qu'un réseau de stations pour cette parallélisation à grain moyen.

La Volvox IS-860

La machine parallèle Volvox IS-860, de la société Archipel [Arc92], a été décrite dans le chapitre introduisant le parallélisme (section 2.5). La figure 3.1 présente la machine configurée en « anneau ». Comme annoncé, bien que chaque nœud de la Volvox soit composé de deux processeurs, nous n'utilisons explicitement que les processeurs de calcul *i*860.

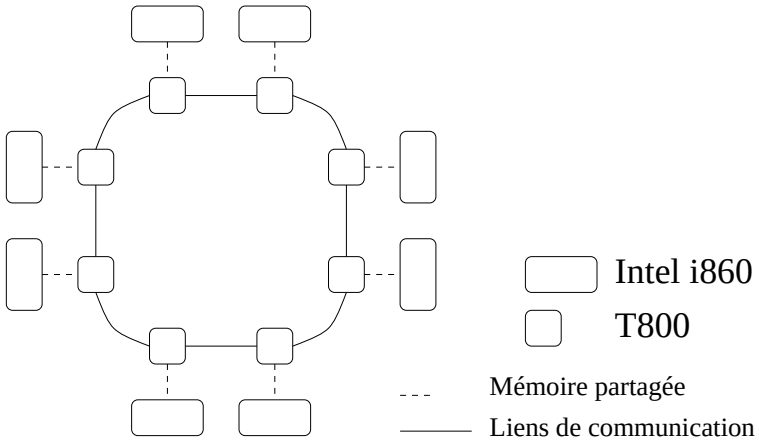

FIG. 3.1 – *Configuration de 8 processeurs en anneau sur une Volvox*

3.2 Le prétraitement du problème d'OCR

Le classifieur étant indépendant du type de formes à reconnaître, un prétraitement adapté au domaine d'application est nécessaire. Le schéma du couplage du classifieur incrémental et d'un réseau de prétraitement dédié à la reconnaissance de dessins au trait [Azc93, AA92] a déjà été présenté (figure 1.4, page 1.4). Ce dernier est de type *feedforward* avec connexions latérales.

3.2.1 Détail du prétraitement

Le système doit reconnaître des images monochromes discrétisées de 16×16 pixels. La figure 3.3 donne un exemple d'entrées possibles. Cette matrice de pixels est présentée à la couche d'entrée du réseau de prétraitement. La deuxième et dernière couche comprend des connexions latérales, l'activation se fait donc en deux étapes (voir l'annexe A.3).

Extraction de petits segments orientés

Le module de prétraitement est composé de cellules à champ récepteur [Har40] chargées de reconnaître de petits segments orientés dans la matrice de pixels. Un champ récepteur est une cellule qui s'active si une forme donnée apparaît dans une zone précise. La détection est effectuée sur une zone 5×5 de la matrice de pixels. Les cellules sont réparties en 12 groupes de 16×16. Chaque groupe est chargé de reconnaître une orientation de segment précise grâce à un des champs récepteurs présentés figure 3.2. Toutes les cellules d'un même groupe sont chargées de détecter une même orientation et possèdent donc le même champ récepteur. La figure 3.4 montre la superposition des segments détectés par les 12 groupes à partir de la figure 3.3.

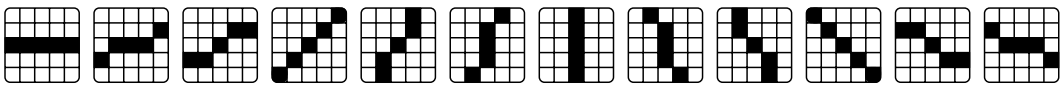

FIG. 3.2 – *12 groupes de cellules à champ récepteur détectent 12 orientations*

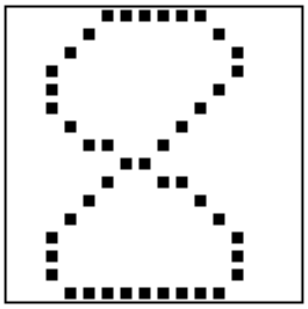

FIG. 3.3 – *Exemple d'image discrétisée (pixels) en entrée du système*

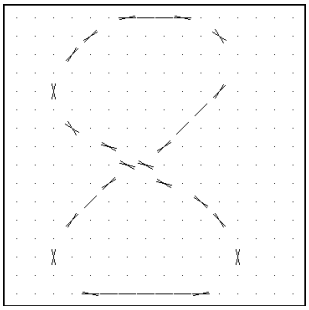

FIG. 3.4 – *Activation des cellules à champ récepteur des 12 orientations*

Diffusion des segments orientés

Les cellules à champ récepteur sont liées entre voisines immédiates grâce à des connexions latérales. De plus, chacun des 12 groupes forme une tranche d'une matrice 3D, chaque cellule à champ récepteur a donc 26 voisines ($3^3 \Leftrightarrow 1$). Les $12 \times 16 \times 16$ cellules de la deuxième couche du réseau de prétraitement composent donc une matrice 3D de 3072 cellules à champ récepteur (voir la figure 3.5).

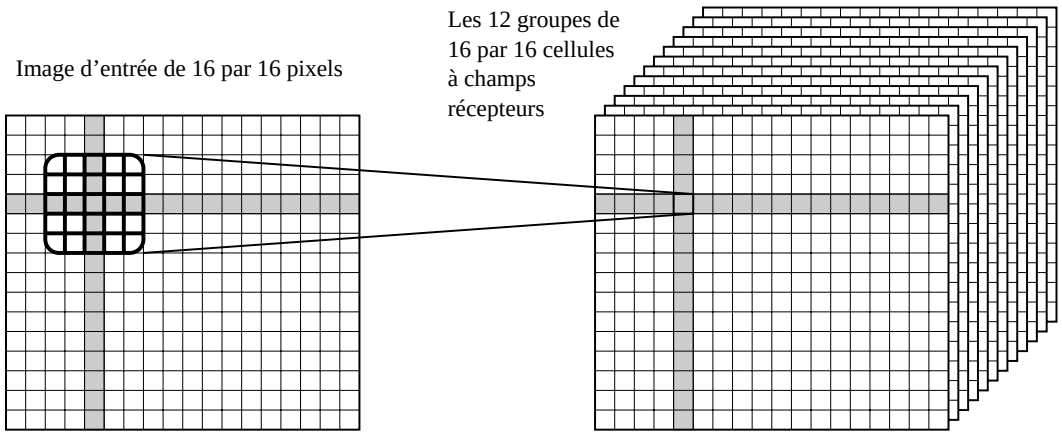

FIG. 3.5 – *Chacune des* $12 \times 16 \times 16$ *cellules à champ récepteur scrute une zone de* 5×5 *pixels dans l'image d'entrée à la recherche d'un seul segment (c'est-à-dire d'une direction donnée, et d'une position donnée)*

L'activation d'une cellule à champ récepteur de la matrice 3D est favorisée par l'activation de ses 26 voisines :

- les voisines de la même tranche favorisent la détection d'un segment si un segment de la même orientation est détecté à proximité (au sens de l'image) ;
- les voisines des tranches adjacentes favorisent la détection d'une orientation si un segment d'une orientation proche apparaît dans la même zone de l'image.

Sortie du module de prétraitement

La figure 3.6 fait suite à la figure 3.4, elle présente la superposition des segments orientés détectés après la prise en compte des connexions latérales. L'activation des cellules à champ récepteur est ensuite seuillée (figure 3.7). Le résultat de seuillage est la sortie du module de prétraitement, et donc l'entrée du classifieur neuronal incrémental.

Résistance au bruit

Le vecteur à apprendre ou à reconnaître, formé par l'activation des cellules à champ récepteur, offre une bonne tolérance à un bruit aléatoirement distribué dans l'image. Seule une altération de l'image affectant plusieurs points alignés risque d'engendrer la détection erronée d'un segment orienté. Par ailleurs, les connexions latérales diminuent l'importance de la position et de l'orientation du segment dans l'image. L'entrée du classifieur offre ainsi une grande tolérance aux petites variations en translation et aux petites déformations.

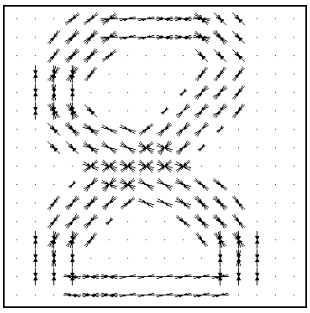

FIG. 3.6 – *Influence des connexions latérales sur les segments détectés*

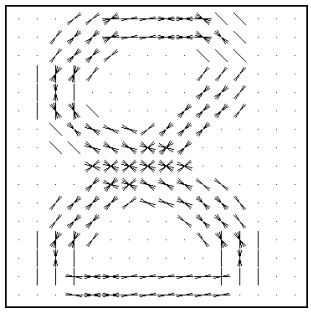

FIG. 3.7 – *Sortie du module de prétraitement (entrée du classifieur)*

3.2.2 Parallélisation du prétraitement

Nous pouvons maintenant détailler la parallélisation. Cette solution fait intervenir un processeur maître, ainsi que des processeurs esclaves :

- le processeur maître découpe l'image (l'espace d'entrée) en bandes de tailles égales. Le découpage est arbitrairement effectué horizontalement ;
- le processeur maître envoie les sous-images aux autres processeurs, sauf une qu'il garde à sa charge ;
- chaque processeur active les cellules à champ récepteur correspondant à la sous-image dont il a la charge.

Activation des cellules à champ récepteur en parallèle

L'activation d'une cellule à champ récepteur, en limite d'une sous-image, nécessite une information contenue dans une autre sous-image, et donc placée sur un autre processeur. De plus, la cellule à champ récepteur est influencée par ses voisines, dont certaines se trouvent sur un autre processeur. Une communication entre cellules à champ récepteur serait donc nécessaire pour achever le prétraitement. Une telle opération serait très coûteuse et doit être évitée. La solution passe par un recouvrement des sous-images.

Recouvrement des sous-images

La taille des sous-images non-adjacentes à un bord de l'image à reconnaître est augmentée de 4 lignes pour permettre l'extraction des petits segments orientés (2 au-dessus de la bande et 2 au-dessous), et de 2 lignes supplémentaires pour permettre la diffusion des segments (une au-dessus et une au-dessous). Le processeur maître envoie donc les sous-images en ajoutant 3 lignes de la sous-image supérieure, et 3 lignes de la sous-image inférieure.

Exemple

La figure 3.8 présente l'envoi des sous-images avec un exemple de configuration à 4 processeurs (3 esclaves et 1 maître). Une figure récapitulative est donnée en fin de chapitre (figure 3.16, page 60) avec la suite du traitement parallélisé.

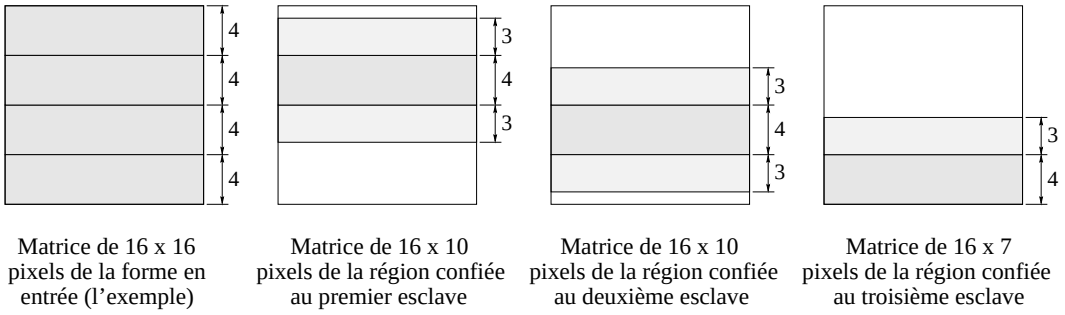

FIG. 3.8 – *Sous-images confiées aux processeurs (avec le recouvrement)*

Utilisation de masques avec 4 processeurs

Récapitulons en considérant l'utilisation de masques. Avec 4 processeurs, la diffusion des segments orientés nécessite le balayage de 768 cellules à champ récepteur par chaque processeur : 4 lignes de 16 cellules sur chacun des 12 groupes. La figure 3.9 montre que la mise en œuvre du masque $3 \times 3 \times 3$ impose la présence d'une ligne de cellules à champ récepteur supplémentaire au-dessus et au-dessous des 4 lignes à balayer. Nous avons donc besoin de 6 lignes de cellules à champ récepteur.

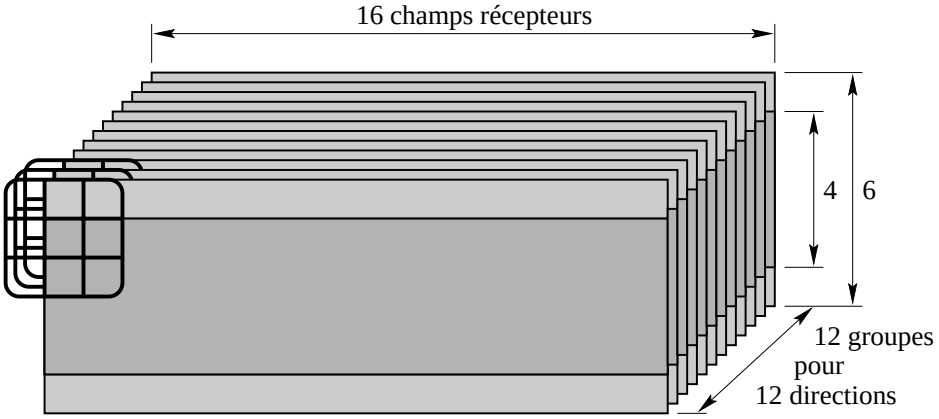

FIG. 3.9 – *Diffusion des segments orientés par un balayage des 12 tranches de la matrice 3D des cellules à champ récepteur avec un masque* $3 \times 3 \times 3$

Les 6 lignes de cellules à champ récepteur nécessaires sont initialement activées par le balayage de l'image à reconnaître avec un masque 5×5 correspondant à une des 12 orientations que doit détecter une tranche donnée de la matrice. Comme l'illustre la figure 3.10, il est nécessaire d'ajouter deux lignes de l'image au-dessus et au-dessous de la bande confiée au processeur.

Utilisation d'un pipeline

La méthode est implémentée en utilisant un pipeline sur les exemples. Les sous-images sont transmises alors que les processeurs calculent l'activation des cellules à champ récepteur de l'exemple précédent. Cette technique, difficile à programmer,

présente l'avantage de masquer les temps de communication. Cependant, le recouvrement des communications par les calculs est délicat au-delà de 4 processeurs sur le problème de la reconnaissance de chiffres manuscrits car les images sont petites. Des formes en entrée d'une plus haute résolution, ou non carrées (par exemple l'image envoyée par un satellite balayant la Terre), permettraient l'utilisation d'un nombre supérieur de processeurs.

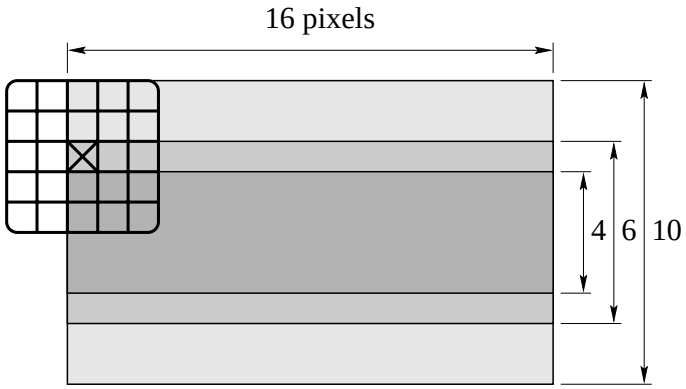

FIG. 3.10 – *Activation des segments orientés par un balayage de l'image de pixels en entrée avec un masque* 5×5

3.3 Activation des prototypes en parallèle

Le prétraitement est parallélisé, chaque processeur possède un bloc de la matrice 3D des cellules à champ récepteur. La discussion qui suit concerne la parallélisation de l'activation des prototypes, c'est-à-dire la classification en elle-même.

3.3.1 Partage des prototypes

Chaque processeur possédant une partie de la matrice 3D, une solution simple est de distribuer l'activation des prototypes en les découpant. Chaque processeur possède et active, crée ou modifie, tous les prototypes, mais uniquement la partie $(w_k, \ldots, w_l)_{1 \leq k < l \leq n}$ dont il a besoin.

Point de vue connexionniste

La terminologie connexionniste facilite la compréhension de cette solution. Chaque processeur possède toutes les cellules prototypes, mais ne s'occupe que des connexions provenant des cellules à champ récepteur dont il a déjà la charge. De même, en cas de modification d'un prototype durant l'apprentissage, tous les processeurs travaillent mais ne modifient que les poids des connexions dont ils ont la charge. Notons que s'il est nécessaire de créer une nouvelle cellule prototype, le processeur possède toute l'information localement (cellules à champ récepteur).

Exemple avec 4 processeurs

La figure 3.11 présente la situation. Chaque processeur possède un bloc de la matrice des cellules à champ récepteur et la totalité des prototypes. Pour ne pas alourdir la figure, on ne représente les liens que d'un seul prototype (le même) par processeur.

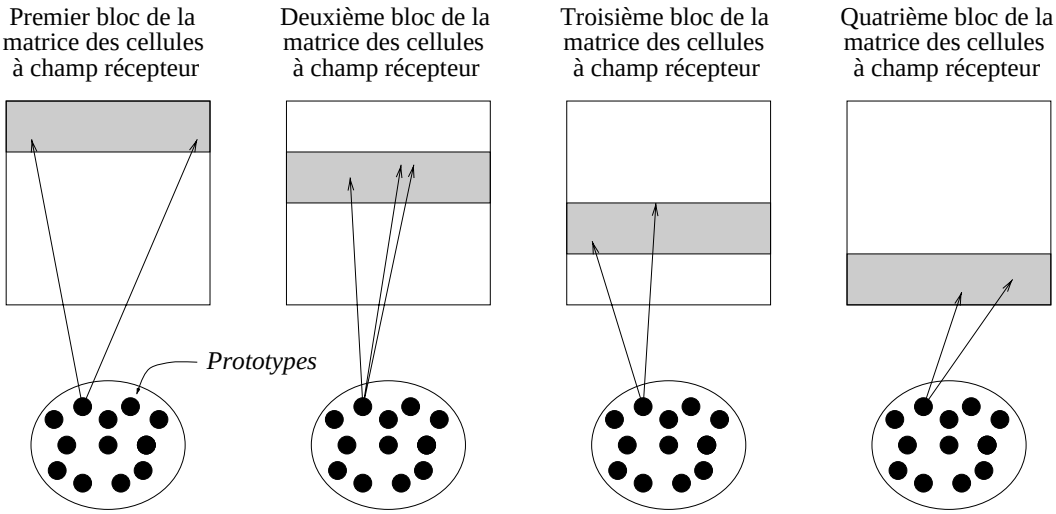

FIG. 3.11 – *Chaque processeur possède tous les prototypes et quelques cellules à champ récepteur (c'est-à-dire quelques connexions entrantes)*

Multidiffusion des calculs intermédiaires

Une multidiffusion de faible volume est nécessaire pour calculer l'activation des prototypes. En effet, considérons la mesure utilisée pour le problème d'OCR (équation 3.1), le processeur P_i, chargé des lignes k à l (avec $1 \leq k < l \leq n$) possède les entrées x_j (c'est-à-dire un des blocs de la matrice 3D) et les w_{ij} (c'est-à-dire un des morceaux de chaque prototype) avec $j \in [k, l]$ et peut donc calculer les trois sommes partielles données par l'équation 3.2.

$$s(X, P_i) = \frac{2 \sum_{j=1}^{n} w_{ij} x_j}{\left(\sum_{j=1}^{n} w_{ij}^2\right) + \left(\sum_{j=1}^{n} x_j^2\right)} \quad (3.1)$$

$$\begin{cases} 2 \sum_{j=k}^{l} w_{ij} x_j \\ \sum_{j=k}^{l} w_{ij}^2 \\ \sum_{j=k}^{l} x_j^2 \end{cases} \quad (3.2)$$

Recouvrement des calculs et communications

Les numérateur et dénominateur complets peuvent être facilement calculés au fur et à mesure de leur multidiffusion. Si on prend l'exemple de processeurs connectés en anneau, chaque processeur peut envoyer les sommes partielles qu'il possède (soit trois scalaires par prototype) à un de ses voisins et recevoir simultanément les sommes partielles de son autre voisin. Chaque processeur peut alors additionner les sommes locales et les sommes reçues tout en faisant suivre les sommes reçues et en recevant de nouvelles sommes. Les calculs et les communications se font simultanément, on parle de recouvrement. Dans notre exemple d'une configuration en anneau à p processeurs, $p \Leftrightarrow 1$ phases de calculs et communications suffisent pour que chaque processeur connaisse la similitude de chaque prototype avec l'entrée (c'est-à-dire la matrice 3D entière) et puisse donc déterminer quel est le prototype gagnant $P_{gagnant}$. Enfin,

sans aucune communication, chaque processeur peut :

- modifier le morceau de $P_{gagnant}$ dont il a la charge :

$$Pour\ i\ \ tel\ que\ \ P_i = P_{meilleur} \qquad pour\ j \in [k,l]\ \ w_{ji} \leftarrow w_{ji} + \alpha(x_j \Leftrightarrow w_{ji})$$

- ou (le cas échéant) créer un morceau du nouveau prototype :

$$Pour\ i\ \ tel\ que\ \ P_i = P_{nouveau} \qquad pour\ j \in [k,l]\ \ w_{ji} \leftarrow x_j$$

3.3.2 Distribution des prototypes

Une alternative au partage (c'est-à-dire au découpage) des prototypes est leur distribution. Chaque processeur a la charge d'un sous-ensemble des prototypes mais prend en compte toutes les connexions dans l'activation. L'algorithme de classification est identique à la version séquentielle, mais les p processeurs travaillent avec p fois moins de prototypes. Chaque processeur cherche à élire son propre prototype gagnant, une communication permet de connaître le gagnant global.

Exemple

La figure 3.12 présente un exemple de distribution des prototypes entiers, avec 4 processeurs. Chaque processeur possède la totalité des cellules à champ récepteur et quelques prototypes. Pour ne pas alourdir la figure, on ne représente complètement qu'un seul prototype par processeur.

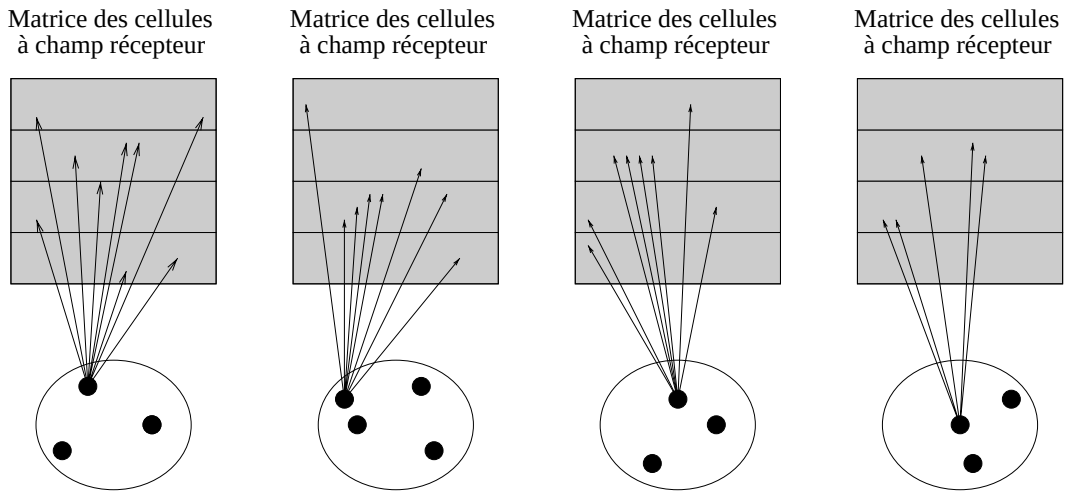

FIG. 3.12 – *Chaque processeur possède des prototypes différents et toutes les cellules à champ récepteur (c'est-à-dire toutes les connexions de ses prototypes)*

3.3.3 Choix d'une stratégie

Communications

La distribution des prototypes nécessite la présence du résultat complet du prétraitement sur chaque processeur alors que le prétraitement choisi (section 3.2.2) conduit à n'avoir qu'un bloc de la matrice 3D sur chaque processeur. Afin de fusionner les blocs de la matrice 3D, tous les processeurs doivent diffuser le bloc de la matrice 3D qu'ils ont localement calculé aux autres processeurs.
Le partage des prototypes évite les communications de fusion de la matrice 3D, seules des sommes partielles ont à circuler, cette charge de communications est partiellement recouvrable par les calculs.

Equilibrage des calculs

Cependant, partager les prototypes présente un défaut rédhibitoire intimement lié au problème de la reconnaissance de dessins au trait. En effet, certaines zones de l'image d'entrée peuvent être presque vides et d'autres très chargées en information. Pour une forme donnée, certains processeurs auront donc beaucoup de travail alors que d'autres seront inactifs. Enfin, la charge de chaque processeur ne peut être nivelée sur plusieurs exemples car l'algorithme doit être synchrone pour recourir à un pipeline.
Notons que ce problème ne peut pas être résolu par un équilibrage dynamique de la charge, c'est-à-dire par une modification du nombre de lignes confiées à chaque processeur en fonction de l'exemple, car la répartition des lignes est dicté par le prétraitement. De plus, les formes étant présentées dans un ordre aléatoire, l'équilibrage devrait être fait entre chaque exemple au prix de nombreuses communications.

Choix de la distribution des prototypes

Nous choisissons donc d'implémenter la distribution des prototypes. En suivant cette stratégie, la charge des processeurs est équilibrée aisément au fur et à mesure des créations de prototypes. En effet, chaque processeur peut mesurer le temps dont il a besoin pour activer son sous-ensemble de prototypes. Si la création d'un nouveau prototype est nécessaire, il est créé « sur » (et « par ») le processeur le moins chargé, c'est-à-dire celui activant ses prototypes le plus rapidement.

Implémentation sur la Volvox

Cependant, la Volvox ne dispose pas de fonctions de mesure du temps suffisamment précises. Cependant la charge peut être équilibrée efficacement en créant le nouveau prototype sur le processeur gérant le moins de prototypes. Notons que cette solution est légèrement moins performante que la mesure effective du temps car toutes les formes n'induisent pas la même charge.
Par exemple, les chiffres « 1 » et « 8 » ont des prototypes de tailles très différentes. Enfin, la mesure réelle du temps permet d'équilibrer la charge même si les processeurs sont de puissances différentes.

3.4 Mise en œuvre du classifieur à granularité moyenne

Nous présentons maintenant l'algorithme parallèle correspondant à nos choix :

- un partage des exemples entre les processeurs pour le prétraitement ;
- une distribution des prototypes pour la classification.

L'algorithme suivant présente la boucle principale du classifieur parallèle, son fonctionnement est illustré par le schéma donné en fin de chapitre (figure 3.16, page 60). Cette boucle est exécutée séquentiellement par tous les processeurs sur chaque exemple à apprendre.

Algorithme d'apprentissage à moyen grain

Pour le processeur maître :

1 Présenter un exemple.

2 Découper l'exemple en sous-images se recouvrant et les diffuser.

Pour les autres processeurs :

2 Recevoir une sous-image.

Pour tous les processeurs :

3 Calculer l'activation du bloc de la matrice 3D des cellules à champ récepteur correspondant à la sous-image à traiter.

4 Diffuser le bloc de la matrice 3D calculé,
recevoir les autres blocs et les fusionner.

5 Calculer l'activation des prototypes locaux sur la matrice 3D.

6 Diffuser les résultats de l'élection locale,
recevoir les résultats des autres processeurs.

7 Dépouiller les résultats et modifier ou créer un prototype.

Envoi des morceaux d'exemples

Une bonne accélération impose de minimiser les temps d'attente des processeurs. Il est donc nécessaire de communiquer autant que possible durant les calculs pour que les données soient toujours disponibles. Afin de masquer les communications induites par l'envoi des morceaux d'exemples, le prétraitement (étapes *1*, *2*, et *3*) met en œuvre un pipeline d'exemples. Cette technique, délicate à programmer et à mettre au point, autorise le commencement du prétraitement d'un nouvel exemple alors que le travail à faire sur l'exemple précédent n'est pas terminé.

Envoi des blocs de la matrice 3D

Nous avons vu dans la section précédente qu'une parallélisation par distribution des prototypes impose la fusion des blocs de la matrice 3D des cellules à champ récepteur (étape *4*). Cette multidiffusion est effectuée de manière « synchrone » afin d'utiliser au mieux la configuration en anneau de la machine parallèle.
En d'autres termes, le programme orchestre précisément les envois et réceptions, dans ces conditions seules $p \Leftrightarrow 1$ phases de communications sont nécessaires avec p processeurs. La figure 3.13 donne l'exemple de la multidiffusion avec 4 processeurs, à chaque instant, chaque processeur envoie et reçoit simultanément une bande de l'image.

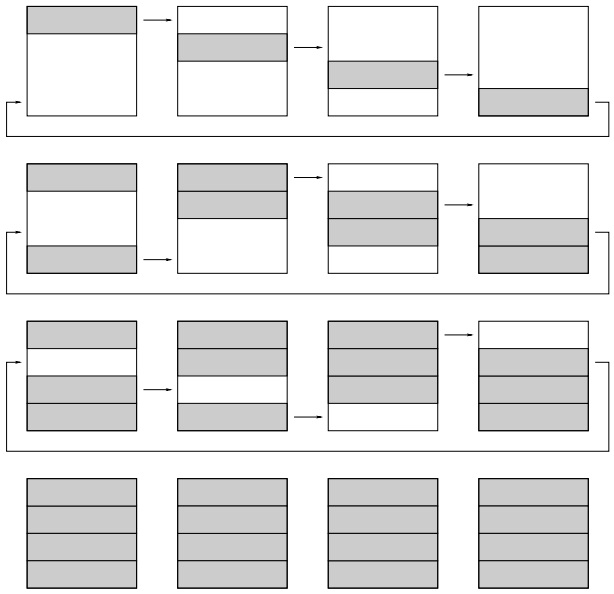

FIG. 3.13 – *Multidiffusion des blocs de la matrice 3D sur un anneau*

Envoi du résultat des élections partielles

Par ailleurs il est nécessaire de multidiffuser les résultats des élections locales (étape *6*) pour que chaque processeur puisse savoir s'il heberge le gagnant ou s'il n'y a pas de gagnant. Cette multidiffusion est implémentée de la même manière que la multidiffusion des blocs de la matrice 3D. Bien que le volume d'information à envoyer soit très faible, cette communication permet à chaque processeur de terminer l'apprentissage de l'exemple sans qu'il soit de nouveau nécessaire d'échanger des messages. En effet, chaque processeur dépouille les résultats des élections partielles et détermine s'il est **le** processeur devant travailler à l'étape *7*:

- le processeur possédant le prototype gagnant le modifie ;
- s'il n'y a pas de gagnant, chaque processeur vérifie s'il est le moins chargé en prototypes et, dans l'affirmative, crée un nouveau prototype à partir de l'exemple à apprendre (c'est-à-dire la matrice 3D).

3.5 Étude du temps de prétraitement en apprentissage

Implémentation

Un long travail d'implémentation a permis de tester cet algorithme sur la Volvox configurée en anneau. Les temps d'activation des cellules à champ récepteur (prétraitement) et des prototypes (classification) sont présentés par la figure 3.14 pour 2 processeurs, et par la figure 3.15 pour 4 processeurs :

- les courbes supérieures suivent le prétraitement (étapes *1*, *2* et *3*) ;
- les courbes inférieures montrent le temps de classification (étapes *4* à *7*).

Au-delà de 4 processeurs

L'utilisation de 4 processeurs impose l'envoi de 62.5 % de l'exemple en entrée à chaque processeur (10 lignes), pour ne calculer que 25 % de la matrice 3D des cellules à champ récepteur (4 « lignes »). Il est donc inutile d'utiliser un nombre supérieur de processeurs sur des images de seulement 16×16 pixels.

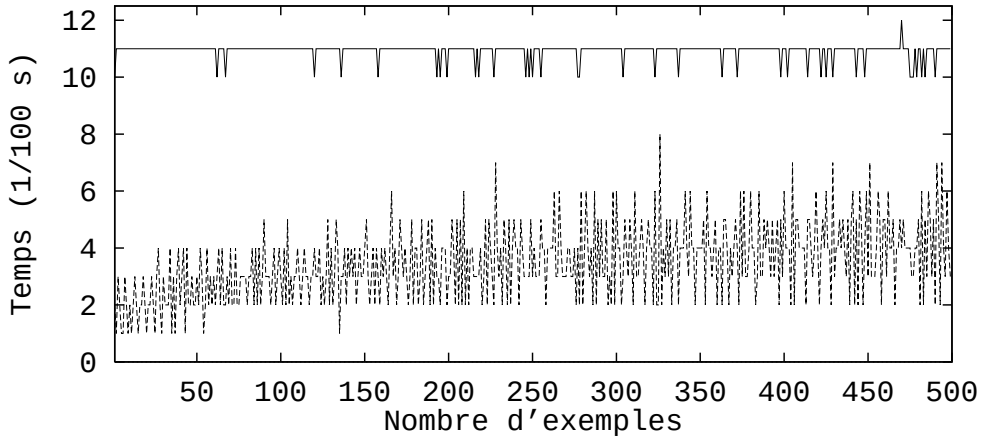

FIG. 3.14 – *Temps de prétraitement et de classification avec 2 processeurs*

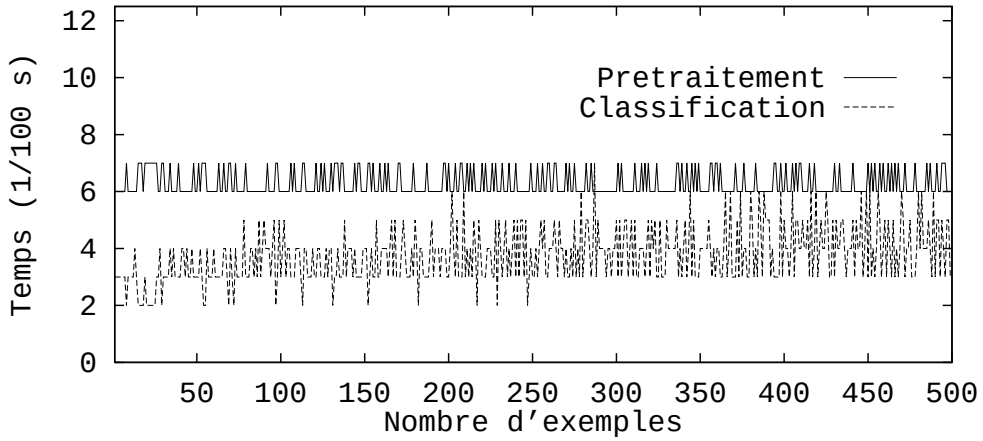

FIG. 3.15 – *Temps de prétraitement et de classification avec 4 processeurs*

Les temps d'exécution du processus d'apprentissage complet des 500 caractères, ainsi que les accélérations correspondantes sont donnés table 3.1, pour 2 et 4 processeurs.

Nombre de processeurs	2 processeurs	4 processeurs
Temps d'apprentissage (s)	72.65	50.87
Accélération	1.50	2.15

TAB. 3.1 – *Temps d'apprentissage et accélérations*

Conclusion

Accélération

Les accélérations obtenues sont tout a fait correctes, cependant nous obtiendrons des résultats bien meilleurs dans les prochains chapitres. Notons que la parallélisation souffre de la forte latence du réseau de communication de la Volvox et que de meilleures accélérations sont possibles avec une machine ayant un rapport *puissance de calcul / capacité à communiquer* plus équilibré. Cette particularité de la machine d'Archipel devait être corrigée par une version pourvue de meilleurs processeurs de communication (voir section 2.5).

Prétraitement

Nombre de processeurs

La limitation de l'intérêt de la parallélisation du prétraitement à 4 processeurs est plus gênante car elle conditionne l'accélération maximale. Cependant, un nombre bien supérieur de processeurs peut être utilisé avec des images plus grandes. En effet, une taille de 16×16 pixels est un cas extrême. De nombreuses applications traitent de plus grandes images et peuvent trouver intérêt à l'utilisation de cette méthode de parallélisation.

Expertise

La parallélisation effective du prétraitement a été longue et délicate, il a été nécessaire de comprendre parfaitement et de modifier en de très nombreux points un programme existant et mettant en œuvre des structures de données complexes, notamment des listes chaînées imbriquées[2]. De façon plus générale, la mise au point d'un programme complexe et dépourvu d'erreurs peut prendre des mois, il est donc

2. A titre d'exemple, chaque prototype est un maillon d'une première liste. Chaque prototype est une liste chaînée de pointeurs. Chacun de ces pointeurs donne l'adresse d'une des composantes de la structure de données utilisée pour la matrice 3D des cellules à champ récepteur (elle-même assez complexe), enfin chaque champ récepteur est une liste de pointeurs donnant l'adresse des pixels de l'image à tester pour détecter la présence d'un petit segment orienté précis. On imagine facilement la difficulté d'un découpage et d'une distribution de ces structures, avec les parcours de listes et les allocations ou libérations de mémoire que cela implique. En fait, cette architecture imaginée par Azcarraga est très efficace en séquentiel (et éventuellement pour une machine à très faible granularité) mais se révèle lourde pour une parallélisation à moyen grain.

important de préserver cet investissement. Notons que pour des applications industrielles, le programme source des prétraitements peut ne pas être disponible. Il est donc préférable d'avoir recours à une stratégie de parallélisation permettant l'utilisation directe des programmes déjà écrits (par exemple pour le prétraitement). Nous espérons que cette « expertise » pourra guider le lecteur désirant lui-même paralléliser une application. Pour notre part, cette conclusion est mise à profit dans les parallélisations présentées dans les prochains chapitres.

Classification

Parallélisation de la classification

Outre le prétraitement, la parallélisation présentée dans ce chapitre met en œuvre une solution efficace pour la classification proprement dite. Distribuer les prototypes est une solution performante. L'unique communication nécessaire est indépendante de la taille des prototypes et peut donc être totalement recouverte par les calculs si les prototypes sont d'une dimension suffisante. De plus, l'équilibrage dynamique de la charge assure une bonne accélération [3].

Apprentissage non supervisé et généralisation

Nous avons étudié la parallélisation de l'apprentissage supervisé. L'algorithme non supervisé du classifieur est plus simple car l'absence d'étiquette supprime le cas d'un exemple à apprendre mal classifié par les prototypes existants. La parallélisation présentée dans ce chapitre peut donc être directement utilisée pour un apprentissage non supervisé. Pour l'algorithme de généralisation, la parallélisation est également facile à implémenter à partir du travail déjà réalisé. Cependant, nous verrons dès le prochain chapitre qu'une parallélisation à forte granularité est préférable.

Qualité de l'apprentissage

Ce chapitre ne présente aucun pourcentage d'erreur en généralisation, et donc aucune mesure de la qualité de classification. Cette étude est inutile car l'algorithme parallèle respecte exactement le fonctionnement de l'algorithme séquentiel, en conséquence, les taux de reconnaissance en généralisation sont identiques à ceux de la version séquentielle. Les prototypes émergeant de l'apprentissage sont les mêmes qu'en séquentiel, quel que soit le nombre de processeurs utilisés. Les prototypes créés ne dépendent que de la base à apprendre et de l'ordre de présentation des exemples.

3. Notons que contrairement aux techniques d'équilibrage classiques, il n'est pas nécessaire de transférer la charge *via* de coûteuses communications.

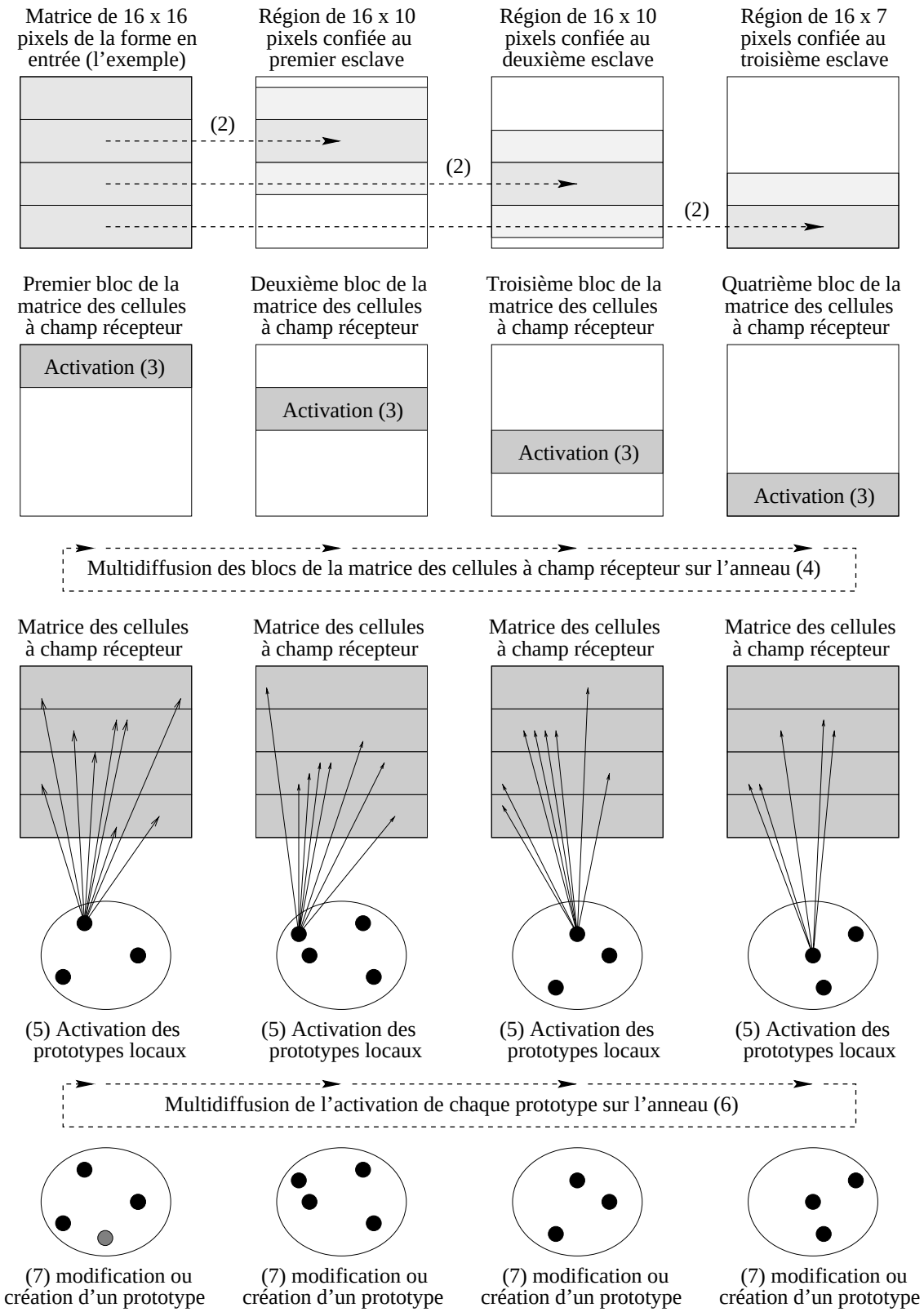

FIG. 3.16 – *Schéma complet (les numéros correspondent à l'algorithme section 3.4)*