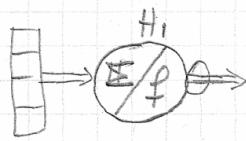


(2)

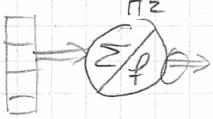
TraitementAvant :Neurone

$$\left(\frac{w}{2}\right)(t_* + t_+) + t_f = t_{h11}$$

Réseau

$$4g t_{h11} = t_{H11}$$

$$y = f(\sum_i w_i x_i)$$



$$\left(\frac{w}{4}\right)(t_* + t_+) + t_f = t_{h21}$$

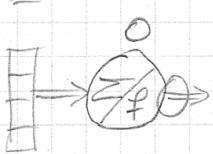
$$16g t_{h21} = t_{H21}$$

T:



$$\left(\frac{w}{2}\right)(t_* + t_+) + t_f = t_{h31}$$

$$6g t_{h31} = t_{H31}$$



$$26g(t_* + t_+) + t_f = t_{o1}$$

$$t_{o1} = t_o$$

$$t_{\text{seq avant}} = t_{H11} + t_{H21} + t_{H31} + t_{o1}$$

$$= 4g t_{h1} + 16g t_{h2} + 6g t_{h3} + t_o$$

$$= 4g \left( \frac{w}{2} (t_* + t_+) + t_f \right) + 16g \left( \frac{w}{4} (t_* + t_+) + t_f \right) + 6g \left( \frac{w}{2} (t_* + t_+) + t_f \right) \\ + (26g(t_* + t_+) + t_f)$$

$$= gw^2(t_* + t_+) + 4gt_f + gw^2(t_* + t_+) + 16gt_f + \frac{3}{2}gw^2(t_* + t_+) + 6gt_f \\ + 26g(t_* + t_+) + t_f$$

$$= (gw^2 + gw^2 + \frac{3}{2}gw^2 + 26g)(t_* + t_+) + (4g + 16g + 6g + 1)t_f$$

$$= \frac{1}{2}(7gw^2 + 52g)(t_* + t_+) + (26g + 1)t_f$$

$$\boxed{t_{\text{seq avant}} = \frac{g}{2}(7w^2 + 52)(t_* + t_+) + (26g + 1)t_f} \quad \begin{matrix} \text{(Pour une} \\ \text{fenêtre)} \end{matrix}$$

sait  $g = 3$     $t_+ = t_* = 3$  cycles  
 $t_f = (\text{min}, \text{max}) = (180, 3300)$  cycles

Pour une fenêtre

$$w = 20$$

$$\underline{\text{min}} \quad t_{\text{seq avant}} = 39888 \text{ cycles}$$

$$w = 40$$

$$\underline{\text{max}} \quad t_{\text{seq avant}} = 115488 \text{ cycles.}$$

$$\underline{\text{min}} \quad t_{\text{seq avant}} = 286368 \text{ cycles}$$

$$\underline{\text{max}} \quad t_{\text{seq avant}} = 361968 \text{ cycles}$$

(3)

ErreurNeurone

$$\begin{aligned} \Delta e &= t_- + 2t_* \\ e_i &= 26g t_* \end{aligned} \quad \left. \begin{aligned} &+ \\ &= \end{aligned} \right\} = t_{O_2}$$

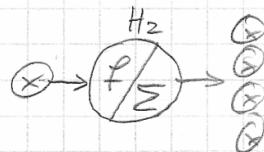
Réseau

$$t_{O_2} = t_{O_2}$$



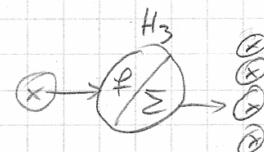
$$\Delta e_i = t_- + 2t_* = t_{H_{12}}$$

$$4g t_{H_{12}} = t_{H_{12}}$$



$$\Delta e_i = t_- + 2t_* = t_{H_{22}}$$

$$16g t_{H_{22}} = t_{H_{22}}$$



$$\Delta e_i = t_- + 2t_* = t_{H_{32}}$$

$$6g t_{H_{32}} = t_{H_{32}}$$

$$\begin{aligned} t_{\text{seq}, \text{erreur}} &= t_{O_2} + t_{H_{12}} + t_{H_{22}} + t_{H_{32}} \\ &= t_o + 4g t_{H_1} + 16g t_{H_2} + 6g t_{H_3} \\ &= ((t_- + 2t_*) + 26g t_*) + 4g(t_- + 2t_*) + 16g(t_- + 2t_*) + 6g(t_- + 2t_*) \\ &= (1 + 4g + 16g + 6g)(t_- + 2t_*) + 26g t_* \\ &= (26g + 1)(t_- + 2t_*) + 26g t_* \\ &= (26g + 1)t_- + (52g + 26g + 2)t_* \end{aligned}$$

$$\boxed{t_{\text{seq}, \text{erreur}} = (26g + 1)t_- + (78g + 2)t_*} \quad \left( \begin{array}{l} \text{Pour une} \\ \text{fonction} \end{array} \right)$$

$$\text{soit } g = 3 \quad t_+ = t_* = 3 \text{ cycles}$$

$$t_f = (\text{moy}, \text{max}) = (180, 3300) \text{ cycles.}$$

Pour une fonction

$$\underline{w = 20}$$

$$t_{\text{seq}, \text{erreur}} = 945 \text{ cycles}$$

$$\underline{w = 40}$$

$$t_{\text{seq}, \text{erreur}} = 945 \text{ cycles.}$$

(4)

Arrière

$$0 \Rightarrow \begin{bmatrix} x \\ x \\ x \\ x \end{bmatrix} = H_1 = \left( \frac{w}{2} \right)^2 (t_+ + t_*) + t_* = t_{h_{13}}$$

Réseau

$$4g t_{h_{13}} = t_{H_{13}}$$

$$0 \Rightarrow \begin{bmatrix} x \\ x \\ x \\ x \end{bmatrix} = H_2 = \left( \frac{w}{4} \right)^2 (t_+ + t_*) + t_* = t_{h_{23}}$$

$$16g t_{h_{23}} = t_{H_{23}}$$

$$0 \Rightarrow \begin{bmatrix} x \\ x \\ x \\ x \end{bmatrix} = H_3 = \left( \frac{w}{2} \right)^2 (t_+ + t_*) + t_* = t_{h_{33}}$$

$$6g t_{h_{33}} = t_{H_{33}}$$

$$0 \Rightarrow \begin{bmatrix} x \\ x \\ x \\ x \end{bmatrix} = O = 26g (t_+ + t_*) + t_* = t_{O_3}$$

$$t_{O_3} = t_{O_3}$$

$$\begin{aligned} t_{\text{éq arrière}} &= t_{H_{13}} + t_{H_{23}} + t_{H_{33}} + t_{O_3} \\ &= 4g t_{h_1} + 16g t_{h_2} + 6g t_{h_3} + t_O \\ &= 4g \left( \frac{w}{2} \right)^2 (t_+ + t_*) + 4g t_* + 16g \left( \frac{w}{4} \right)^2 (t_+ + t_*) + 16g t_* + 6g \left( \frac{w}{2} \right)^2 (t_+ + t_*) + 6g t_* \\ &\quad + 26g (t_+ + t_*) + t_* \\ &= (gw^2 + gw^2 + \frac{3}{2}gw^2 + 26g)(t_+ + t_*) + (4g + 16g + 6g + 1)t_* \\ &= \frac{8}{2}(7w^2 + 52)(t_+ + t_*) + (26g + 1)t_* \end{aligned}$$

$$\boxed{t_{\text{éq arrière}} = \frac{8}{2}(7w^2 + 52)t_+ + \left( \frac{78}{2}w^2 + 52g + 1 \right)t_*} \quad \text{(Pour une fenêtre)}$$

sait  $g = 3$      $t_+ = t_* = 3$  cycles

$$t_f = (\text{moy}, \text{max}) = (180, 3300) \text{ cycles.}$$

Pour une fenêtre

$$\underline{w = 20}$$

$$t_{\text{éq arrière}} = 25.905 \text{ cycles.}$$

$$\underline{w = 40}$$

$$t_{\text{éq arrière}} = 101.505 \text{ cycles}$$

(5)

Pour une fenêtre

$$\begin{aligned}
 t_{\text{seq}} &= t_{\text{seq avant}} + t_{\text{seq en cours}} + t_{\text{seq arriver}} \\
 &= \frac{g}{2} (7\omega^2 + 52)(t_+ + t_*) + (26g + 1)t_f \\
 &\quad + (26g + 1)t_- + (78g + 2)t_* \\
 &\quad + \frac{g}{2} (7\omega^2 + 52)t_+ + \frac{g}{2} (7g\omega^2 + 52g + 1)t_* \\
 &= (\frac{7}{2}g\omega^2 + 26g + 26g + 1 + \frac{7}{2}g\omega^2 + 26g)t_+ \\
 &\quad + (\frac{7}{2}g\omega^2 + 26g + 78g + 2 + \frac{7}{2}g\omega^2 + 52g + 1)t_* \\
 &\quad + (26g + 1)t_f
 \end{aligned}$$

$$t_{\text{seq}} = (7g\omega^2 + 78g + 1)t_+ + (7g\omega^2 + 156g + 3)t_* + (26g + 1)t_f$$

Pour toutes les fenêtres

$$T_{\text{seq}} = N t_{\text{seq}} = N(7g\omega^2 + 78g + 1)t_+ + N(7g\omega^2 + 156g + 3)t_* + N(26g + 1)t_f$$

$$\text{soit } \gamma = 3 \quad t_+ = t_* = 3 \text{ cycles}$$

$$t_f = 180 \text{ cycles} \quad t_{f,\max} = 3300 \text{ cycles.}$$

Pour une fenêtre

alors

$$\omega = 20$$

$$N = 40$$

$$t_{\text{seq,moy}} = 66738 \text{ cycles}$$

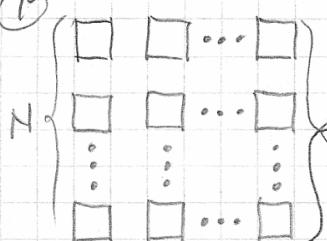
$$t_{\text{seq,moy}} = 217938 \text{ cycles.}$$

$$t_{\text{seq,max}} = 313218 \text{ cycles}$$

$$t_{\text{seq,max}} = 464418 \text{ cycles.}$$

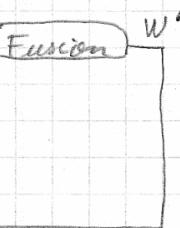
## ① Domaine $\Rightarrow$ Fenêtres

10



Préseaux

Chacun des P réseaux traite N/P fenêtres



Q : Comment fusionner les poids générés par les P réseaux ?

- sol 1 :
- Chaque réseau fait la mise à jour de ses poids au fur et à mesure qu'il traite les fenêtres dont il est responsable, selon un principe d'apprentissage "on-line".
  - La fusion consiste alors à calculer la moyenne des poids générés par chacun des P réseaux et de diffuser les poids "fusionnés".

Avantages : - apprentissage "on-line" partiel donc apprentissage plus rapide.

Inconvénients : - chaque réseau doit avoir sa propre copie des poids.

- Effet du calcul de la moyenne des poids sur l'apprentissage.

- sol 2 :
- Chaque réseau accumule ( $\sum \Delta W_i$ ) les variations des poids ( $\Delta W$ ), mais ne fait pas la mise à jour des poids.
  - La fusion fait la somme des  $\Delta W$  produits par les P réseaux et applique les  $\Delta W_f$  aux poids.

Avantages : - Les réseaux partagent la même copie des poids (pas de diffusion).

Inconvénients : - Apprentissage "Batch" donc plus lent.

(7)

sol 3: - chaque réseau accumule ( $\sum e_i$ ) seulement l'erreur locale (erreur de chaque neurone) mais ne fait pas le calcul de  $\Delta W$  ni la mise à jour des poids.

- La fusion fait la somme des  $E$  produites par les  $P$  réseaux et calcule et applique les  $\Delta W$  aux poids.

Avantages : - Les réseaux partagent la même copie des poids (pas de diffusion)

- La somme  $E_T = \sum E_i$  faite par la fusion est moins grosse que la somme  $\Delta W_T = \sum \Delta W_i$  de la sol 2.

Inconvénients : - Apprentissage "Batch" donc plus lent.

- La fusion fait plus de travail car elle doit calculer  $\Delta W$  à partir de  $E_T$  et ce travail est fait séquentiellement.

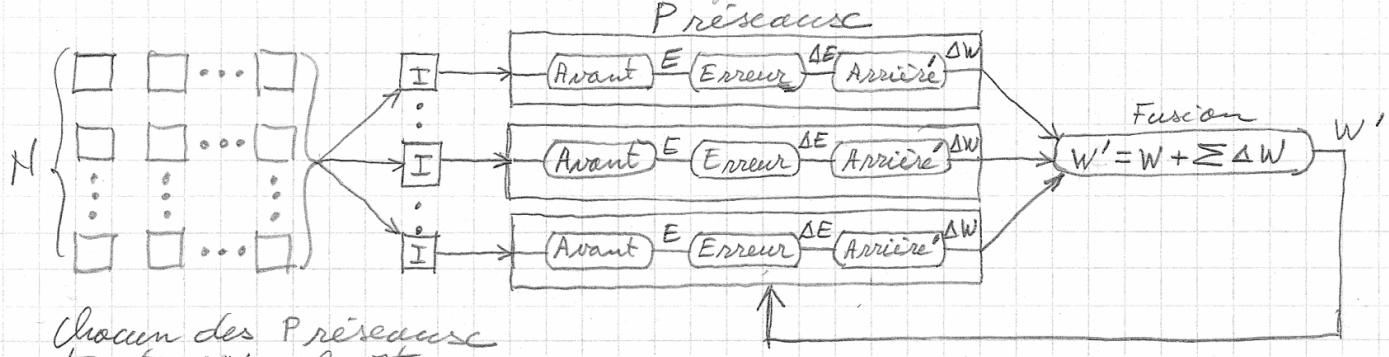
Le choix entre ces solutions n'est pas facile car elles ont toutes des avantages évidents. Pour faire un choix "optimal", il faudrait les essayer toutes.

Choix : - La sol 1 a certainement l'avantage d'un apprentissage "on-line" partiel mais l'effet du calcul de la moyenne sur l'évolution de l'apprentissage n'est pas clair. Une analyse détaillée et une expérimentation seraient nécessaires pour mettre cet effet en lumière.

- La sol 2 a l'avantage de réduire l'importance du traitement séquentiel de la Fusion, par rapport à la sol 3. Et bien que l'apprentissage "Batch" est plus lent que dans le cas de sol 1, il ne contient pas de traitement avec effets non-maitrisés, tel que le calcul de la moyenne de la sol 1.

Conclusion : Moyennant une étude plus en profondeur de la sol 1 et selon l'état des connaissances actuelles, la sol 2 serait préférable.

Donc, selon la sol 2, le système sera



Chacun des Précause  
traité  $N/P$  fenêtres.

Pour un réseau :

$$t_{\text{esce}} = \frac{N}{P} (t_{\text{avant}} + t_{\text{erreur}} + t_{\text{arrière}})$$

$$\text{où } t_{\text{avant}} = t_{\text{seq avant}}$$

$$\text{erreur} = t_{\text{seq erreur}}$$

$$\text{arrière} = t_{\text{seq arrière}}$$

Note : Le travail d'accumulations des  $\Delta W_i$ , c'est-à-dire  $\Delta W = \sum_{N/P} \Delta W_i$  est équivalent au travail de mise à jour des poids, c'est-à-dire  $N/P \times (W = W + \Delta W)$

$$t_{\text{com}} = \underbrace{\frac{N}{P} (t_s + w^2 t_w)}_{\text{fenêtres}} + \underbrace{(t_s + \frac{g}{2} (7w^2 + 52) t_w)}_{\Delta W} + \underbrace{(t_s + \frac{g}{2} (7w^2 + 52) t_w)}_{W'}$$

$$t_{\text{com}} = \frac{N}{P} (t_s + w^2 t_w) + (2t_s + g(7w^2 + 52) t_w)$$

Pour Fusion :

$$t_{\text{esce}} = P \left( \frac{g}{2} \right) (7w^2 + 52) t +$$

$$t_{\text{com}} = P \underbrace{(t_s + \frac{g}{2} (7w^2 + 52) t_w)}_{\Delta W} + \underbrace{(t_s + \frac{g}{2} (7w^2 + 52) t_w)}_{W'}$$

$$t_{\text{com}} = (P+1) (t_s + \frac{g}{2} (7w^2 + 52) t_w)$$

Donc pour une itération

(9)

$$T_{\text{exec}} = T_{\text{exec}}(\text{réseau}) + t_{\text{exec}}(\text{Fusion})$$

$$T_{\text{exec}} = \frac{N}{P} (t_{\text{avant}} + t_{\text{erreur}} + t_{\text{arrière}}) + P \left( \frac{g}{2} (7w^2 + 5z) t_w + \right. \quad (4) \quad (5)$$

$$\begin{aligned} T_{\text{com}} &= t_{\text{com}}(\text{réseau}) + t_{\text{com}}(\text{Fusion}) \\ &= \frac{N}{P} (t_s + w^2 t_w) + (2t_s + g(7w^2 + 5z)t_w) \\ &\quad + (P+1) \left( t_s + \frac{g}{2} (7w^2 + 5z) t_w \right) \end{aligned}$$

$$T_{\text{com}} = \frac{N}{P} (t_s + w^2 t_w) + (P+3) \left( t_s + \frac{g}{2} (7w^2 + 5z) t_w \right) \quad (6)$$

et  $T_p = T_{\text{exec}} + T_{\text{com}}$

d'où

$$G = \frac{T_{\text{seq}}}{T_p} \quad \text{mais}$$

$E \neq \frac{T_{\text{seq}}}{PT_p}$  puisque  
le traitement  
de Fusion est  
séquentiel

donc  $E = \frac{T_{\text{seq}}}{P(T_{\text{exec}}(\text{réseau}) + t_{\text{com}}(\text{réseau})) + (T_{\text{exec}}(\text{Fusion}) + T_{\text{com}}(\text{Fusion}))}$

sach  $g = 3$   $t_+ = t_* = 3 \text{ cycles}$   $t_s = 0$

$N = 30000$   $t_F = (\text{moy}) = (180 \text{ cycles})$   $t_w = 1 \text{ cycle.}$

P  $\underline{w = 20}$

2  $G = 1.988$

$E = 0.994$

4  $G = 3.976$

$E = 0.994$

8  $G = 7.948$

$E = 0.994$

$\underline{w = 40}$

$G = 1.985$

$E = 0.993$

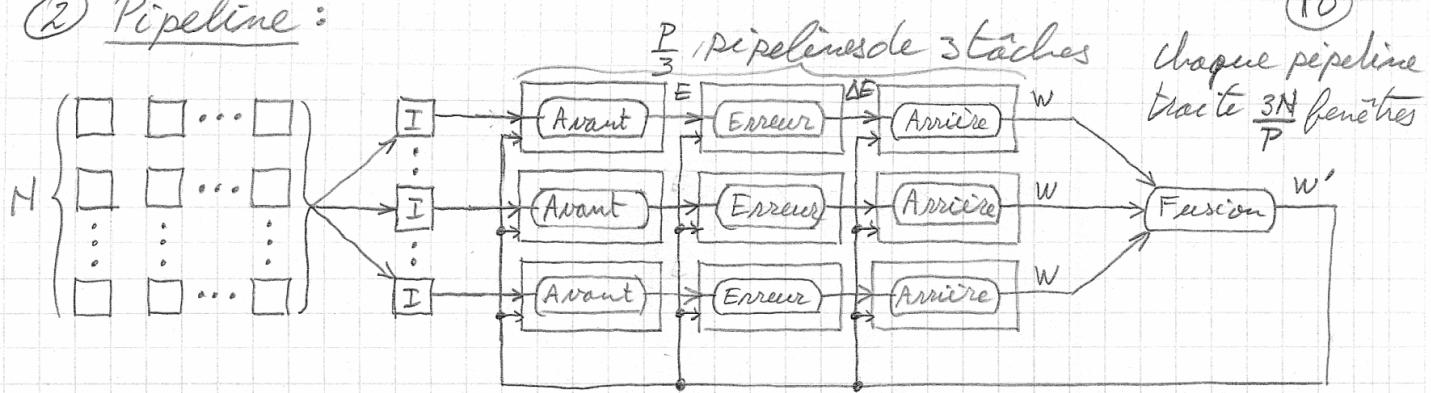
$G = 3.970$

$E = 0.993$

$G = 7.936$

$E = 0.993$

## ② Pipeline :



Q: Comment fusionner les poids générés par les  $\frac{P}{3}$  pipelines ?

Q: Comment gérer le partage des poids entre les 3 tâches qui composent chaque pipeline, en sachant que ces 3 tâches s'exécutent simultanément ?

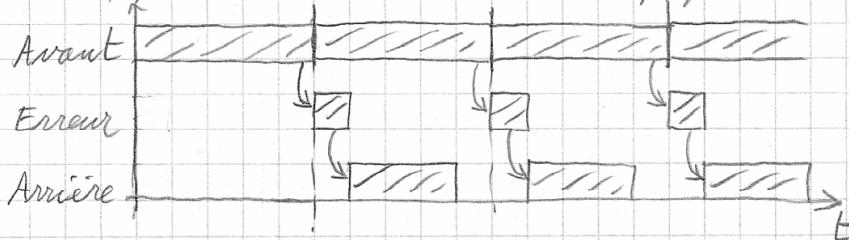
→ Cette question n'est pas simple puisque la tâche "Arrière" modifie les poids qui sont utilisés simultanément par les 2 autres tâches.  
D'où un problème de collision.

Pour ces deux questions, la sol 2 vue précédemment semble plus appropriée.

De plus, les temps d'exécution des 3 phases (tâches) du traitement en pipeline ne sont pas de "poids" égales.  
En fait,

$$terreur \ll \text{arrière} < \text{avant}.$$

Donc, selon le traitement en pipeline :



Donc, le travail est cadencé par la tâche la plus longue et les autres subissent des temps mort pouvant être important.

En fait, selon les estimations :

Pour  $W=20$

moyenne:  $\text{arrière} \approx \text{avant}/1.5$   
 $\text{terreur} \approx \text{avant}/42$

maximum:  $\text{arrière} \approx \text{avant}/11$   
 $\text{terreur} \approx \text{avant}/303$

Pour  $W=40$ :

$\text{arrière} \approx \text{avant}/1.01$   
 $\text{terreur} \approx \text{avant}/122$

$\text{arrière} \approx \text{avant}/3.6$   
 $\text{terreur} \approx \text{avant}/383$

(11)

Dans le cas de la tâche "Avant", son temps d'exécution est largement dominé par le calcul de TANH, dont le temps d'exécution varie largement.

Selon les estimations :

moyenne

$$t_{\text{tanh}} \approx 180 \text{ cycles}$$

maximum

$$t_{\text{tanh}} \approx 3300 \text{ cycles.}$$

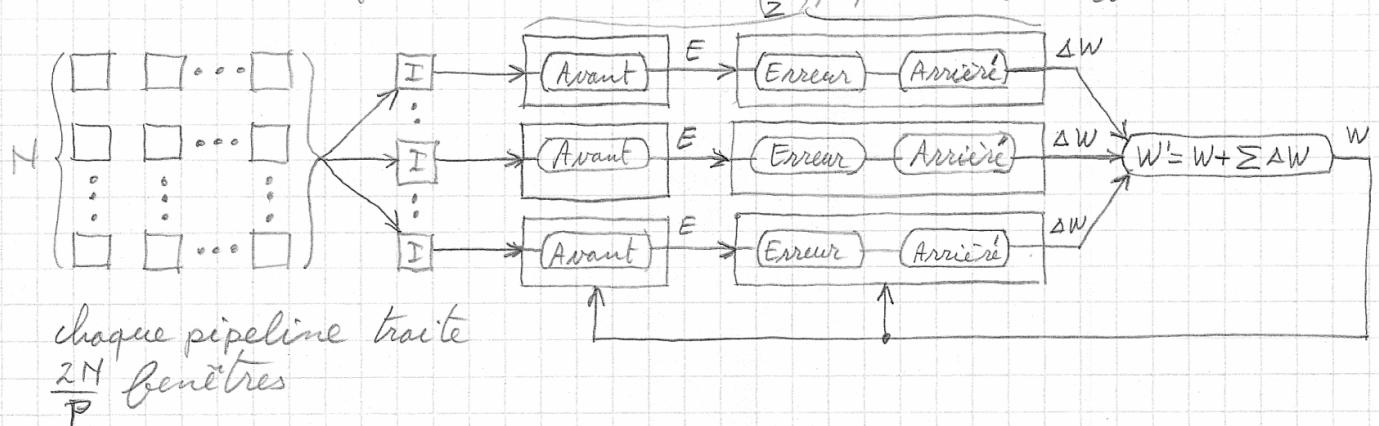
Par contre, pour des fenêtres plus grande (par exemple  $w \geq 40$ ), le temps d'exécution de TANH représente une portion plus petite du traitement total car il ne varie pas avec la taille de la fenêtre.

Aussi, avec des fenêtres suffisamment grande, le temps de traitement de la tâche "Arrière" dépasse celui de la tâche "Avant".

Finalement, selon l'analyse, le temps de traitement de la tâche Erreur, est très petit en proportion des deux autres et ne varie pas avec la taille des fenêtres, seulement avec la taille du réseau (nombre de groupes).

En conclusion, il serait plus judicieux d'intégrer la tâche Erreur dans l'une ou l'autre des deux autres tâches. Normalement, on l'intégrerait dans celle des deux autres qui a le temps d'exécution le plus court. Dans le cas présent et puisque on ne pense pas utiliser des fenêtres de plus que  $w=40$ , la tâche Erreur sera intégrée à la tâche Arrière.

D'où le système devient :  $\frac{P}{2}$  pipelines de 2 tâches.



Pour un pipeline :

$$t_{\text{exce}} = \frac{2N}{P} \text{ mase(tavant, terreut tarrière)}$$

$t_{\text{exce}} = \frac{2N}{P} \text{ tavant}$	(pour des fenêtres $w \leq 40$ )
---	----------------------------------

① pour Avant :

$$t_{com} = \underbrace{\frac{2N}{P} (ts + w^2 tw)}_{\text{fenêtres}} + \underbrace{\frac{2N}{P} (ts + tw)}_{\text{Erreur.}} + \underbrace{(ts + \frac{9}{2}(7w^2 + 52)tw)}_{\substack{\text{Poids} \\ \text{fusionnés}}}$$

② pour Erreur+Arrière :

$$t_{com} = \underbrace{\frac{2N}{P} (ts + tw)}_{\text{Erreur.}} + \underbrace{(ts + \frac{9}{2}(7w^2 + 52)tw)}_{\Delta W} + \underbrace{(ts + \frac{9}{2}(7w^2 + 52)tw)}_{\substack{\text{Poids} \\ \text{fusionnés.}}}$$

Par contre, on peut considérer que les communications de ces deux tâches sont simultanées.

Ainsi, le temps de communication sera dominé par le  $t_{com}$  de la tâche pour laquelle ( $t_{exe} + t_{com}$ ) est le plus long.

Dans le cas présent, il a déjà été déterminé que la tâche Avant a le  $t_{exe}$  le plus long. Et l'analyse des  $t_{com}$  ci-dessus montre que la tâche Avant a aussi le  $t_{com}$  le plus long (à cause des fenêtres).

Donc

$$t_{com} = \frac{2N}{P} (2ts + (w^2 + 1)tw) + (ts + \frac{9}{2}(7w^2 + 52)tw)$$

Pour Fusion :

$$t_{exe} = \frac{P}{2} \left( \frac{9}{2} \right) (7w^2 + 52) t +$$

$$t_{com} = \underbrace{\frac{P}{2} (ts + \frac{9}{2}(7w^2 + 52)tw)}_{\Delta W} + \underbrace{(ts + \frac{9}{2}(7w^2 + 52)tw)}_{W'}$$

$$t_{com} = \frac{1}{2}(P+2)(ts + \frac{9}{2}(7w^2 + 52)tw)$$

Donc pour une itération

$$T_{exe} = t_{exe}(\text{pipeline}) + t_{exe}(\text{Fusion}).$$

$$T_{exe} = \frac{2N}{P} t_{avant} + P \frac{9}{4} (7w^2 + 52) t +$$

$$T_{com} = t_{com}(\text{pipeline}) + t_{com}(\text{Fusion})$$

(13)

$$T_{com} = \frac{2N}{P} (2t_s + (\omega^2 + 1)t_w) + \left( t_s + \frac{\Delta}{2} (7\omega^2 + 52) t_w \right) + \frac{(P+2)}{2} \left( t_s + \frac{\Delta}{2} (7\omega^2 + 52) t_w \right)$$

$$\boxed{T_{com} = \frac{2N}{P} (2t_s + (\omega^2 + 1)t_w) + \frac{(P+4)}{2} \left( t_s + \frac{\Delta}{2} (7\omega^2 + 52) t_w \right)}$$

et pour finir

$$\boxed{T_p = T_{seq} + T_{com}}$$

d'où

$$\boxed{G = \frac{T_{seq}}{T_p}} \quad \text{mais} \quad E \neq \frac{T_{seq}}{PT_p} \quad \text{puisque la Fusion est séquentielle}$$

donc

$$\boxed{E = \frac{T_{seq}}{P(t_{seq}(\text{pipeline}) + t_{com}(\text{pipeline})) + (t_{seq}(\text{Fusion}) + t_{com}(\text{Fusion}))}}$$

soit  $\omega = 3$        $t_+ = t_* = 3$  cycles       $t_s = 0$

$N = 30000$        $t_f = (\text{moy}) = 180$  cycles       $t_w = 1$  cycle.

P

$\omega = 20$

$\omega = 40$

2

$$G = 1.656$$

$$G = 1.861$$

$$E = 0.828$$

$$E = 0.931$$

4

$$G = 3.313$$

$$G = 3.722$$

$$E = 0.828$$

$$E = 0.931$$

8

$$G = 6.624$$

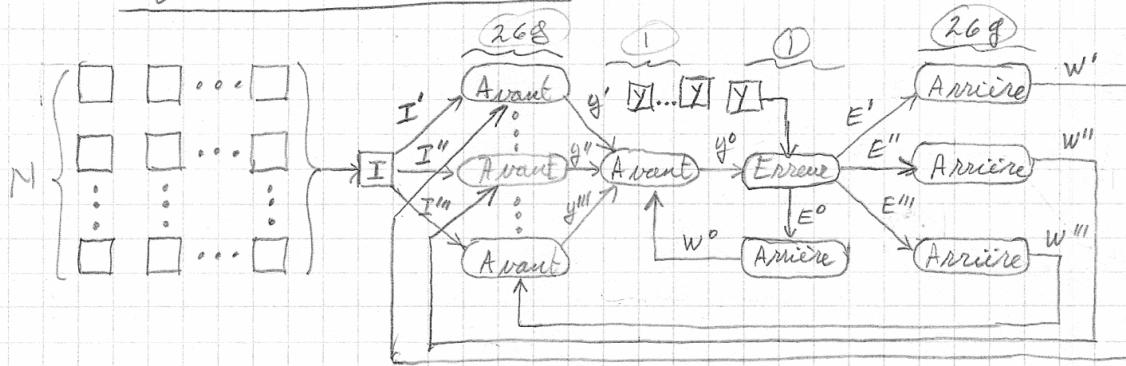
$$G = 7.443$$

$$E = 0.828$$

$$E = 0.931$$

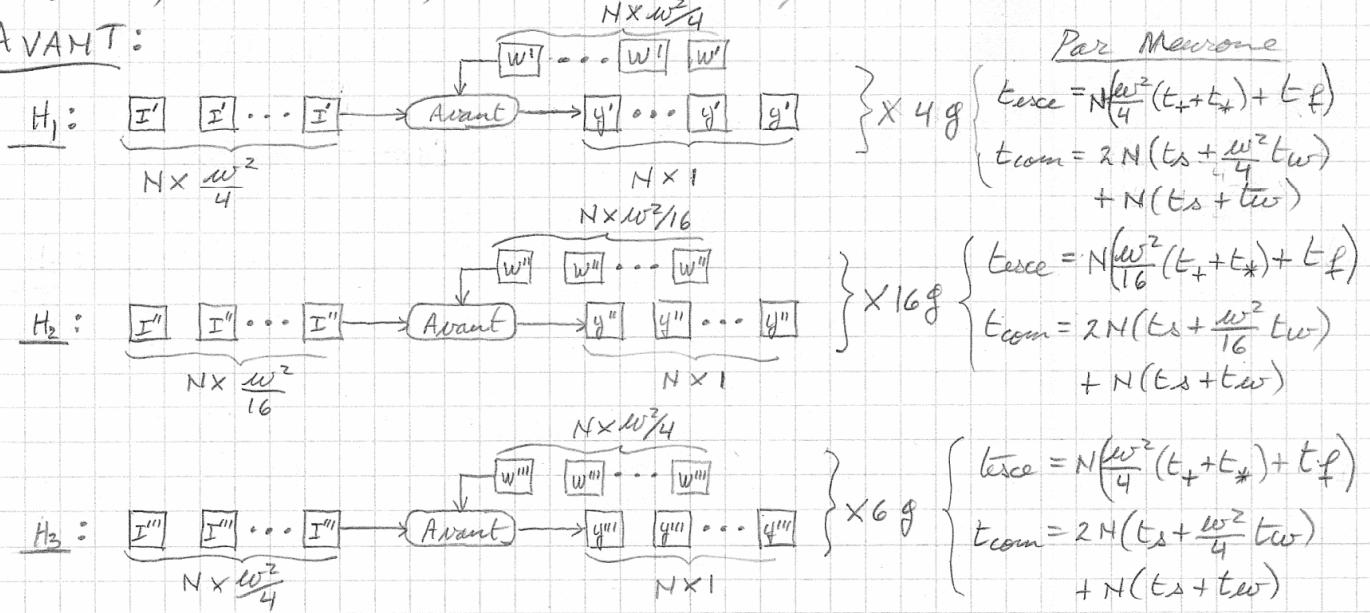
(14)

### ③ Domaine $\Rightarrow$ Neurone

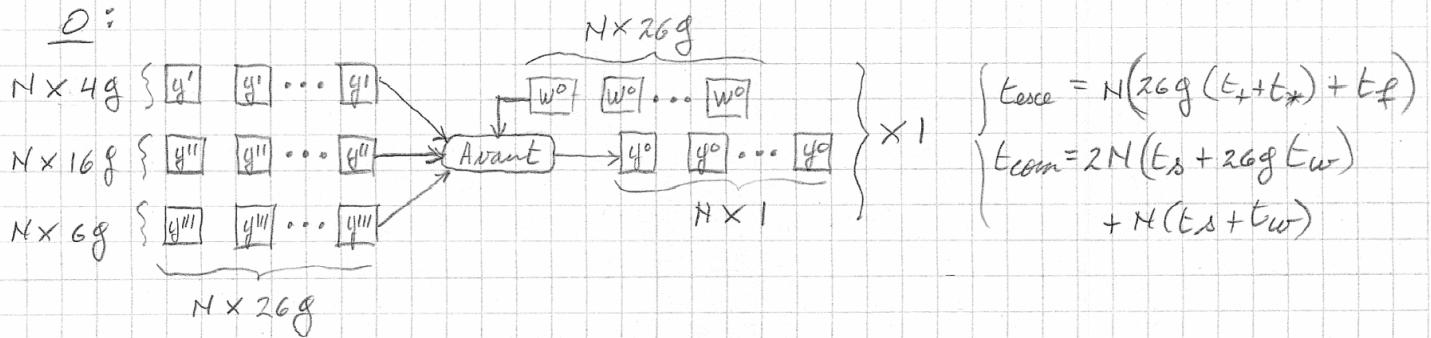


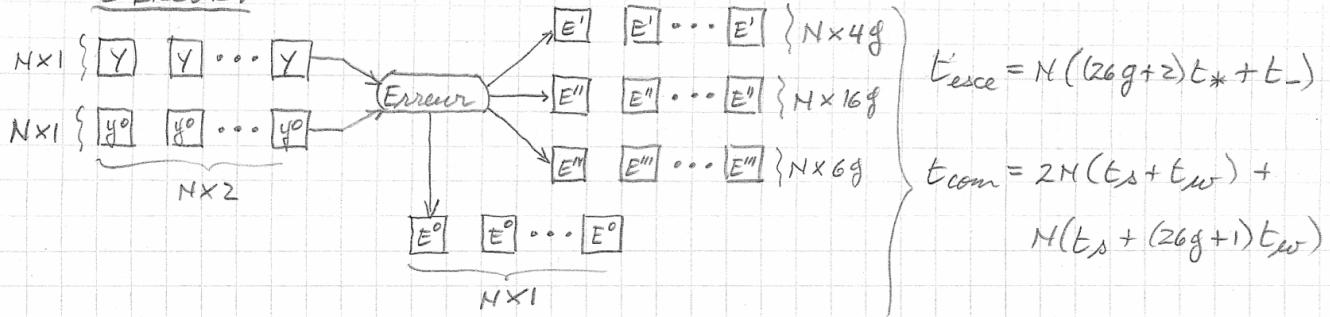
Dans la stratégie ci-dessus, chaque étape du traitement de chaque neurone est marquée par des tâches différentes. Donc, en détail, les tâches du système seraient :

#### AVANT:



#### O:



ERREUR:ARRIÈRE:

$$H_1 : \underbrace{[E' \ E' \cdots E']}_{N \times 1} \rightarrow \text{Arrrière} \rightarrow \underbrace{[W' \ W' \cdots W']}_{N \times \frac{w^2}{4}} \left\{ \begin{array}{l} t_{exec} = N(\frac{w^2}{4} + 1)(2t_* + t_+) \\ t_{com} = N(t_s + \frac{w^2}{4}t_{sw}) + N(t_s + t_{sw}) \end{array} \right.$$

$$H_2 : \underbrace{[E'' \ E'' \cdots E'']}_{N \times 1} \rightarrow \text{Arrrière} \rightarrow \underbrace{[W'' \ W'' \cdots W'']}_{N \times \frac{w^2}{16}} \left\{ \begin{array}{l} t_{exec} = N(\frac{w^2}{16} + 1)(2t_* + t_+) \\ t_{com} = N(t_s + \frac{w^2}{16}t_{sw}) + N(t_s + t_{sw}) \end{array} \right.$$

$$H_3 : \underbrace{[E''' \ E''' \cdots E''']}_{N \times 1} \rightarrow \text{Arrrière} \rightarrow \underbrace{[W'''' \ W'''' \cdots W''']}_{N \times \frac{w^2}{4}} \left\{ \begin{array}{l} t_{exec} = N(\frac{w^2}{4} + 1)(2t_* + t_+) \\ t_{com} = N(t_s + \frac{w^2}{4}t_{sw}) + N(t_s + t_{sw}) \end{array} \right.$$

$$\textcircled{1} : \underbrace{[E^0 \ E^0 \cdots E^0]}_{N \times 1} \rightarrow \text{Arrrière} \rightarrow \underbrace{[W^0 \ W^0 \cdots W^0]}_{N \times 26g} \left\{ \begin{array}{l} t_{exec} = N(26g + 1)(2t_* + t_+) \\ t_{com} = N(t_s + 26gt_{sw}) + N(t_s + t_{sw}) \end{array} \right.$$

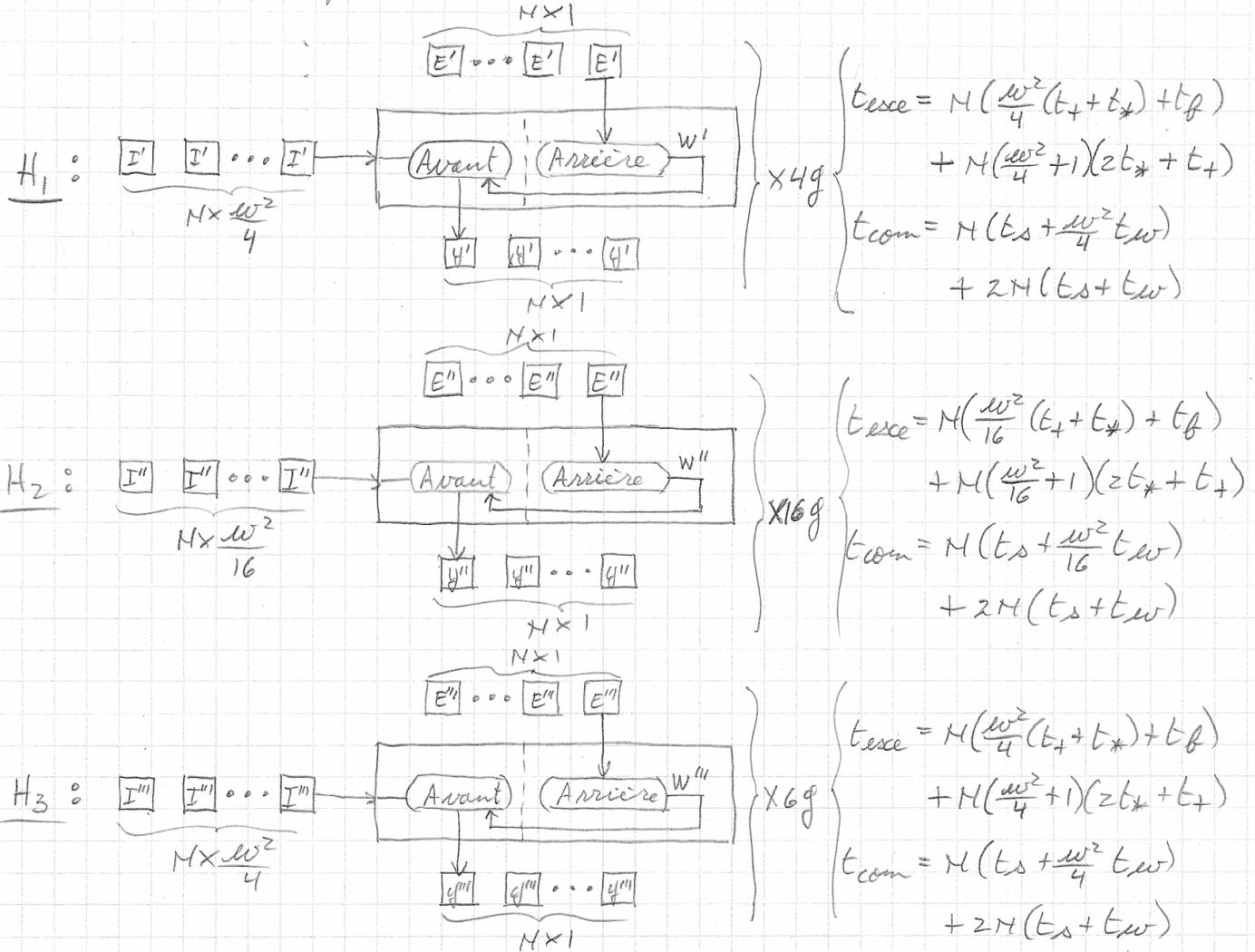
Après analyse, on constate que la stratégie proposée, qui consiste à séparer en tâches les 3 phases du traitement (Avant, Erreur et Arrrière) impose beaucoup de communications entre les tâches. Plus particulièrement, elle impose la communication des poids entre les phases Avant et Arrrière.

Pour éviter ces communications de poids, les phases Avant et Arrrière des neurones pourraient être regroupées dans la même tâche, qui serait associée au neurone correspondant. Ainsi, les poids du neurone seraient conservés "localement" et n'auraient pas besoin d'être communiqués.

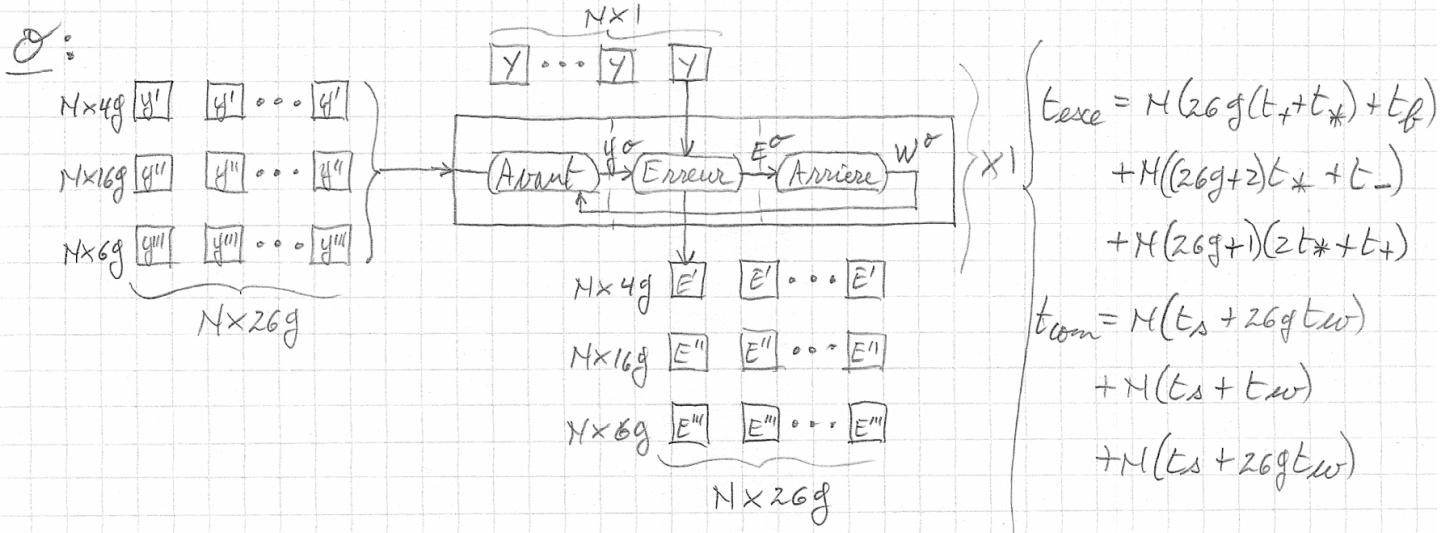
Evidemment, dans ce cas, les phases de traitement des tâches ainsi créées auraient besoin d'être "synchronisées" entre elles avec des mécanismes appropriés.

(16)

Selon ce regroupement, le système devient :



Finalement, on choisit de regrouper la phase Erreur avec les deux autres phases, dans la tâche du neurone de sortie. Ceci n'apporte pas vraiment d'avantages autre que de "rationnaliser" le système.



En définitive, le traitement est marqué par 3 phases distinctes, dont une est purement séquentielle.

(17)

<u>Phase Avant</u>	<u>Phase Erreur</u>	<u>Phase Arrière</u>
$T_{\text{avant}} = \text{Max}(T_{H_1 \text{ avant}},$ $T_{H_2 \text{ avant}},$ $T_{H_3 \text{ avant}})$ $+ T_{\text{O avant}}$	$T_{\text{erreur}} = T_{\text{O erreur}}$	$T_{\text{arrière}} = \text{Max}(T_{H_1 \text{ arrière}},$ $T_{H_2 \text{ arrière}},$ $T_{H_3 \text{ arrière}},$ $T_{\text{O arrière}})$

Pour Tavant : Note :  $T_{\text{O avant}}$  doit être traité après AVANT ( $H_1, H_2, H_3$ ) car le neurone de sortie a besoin des sorties des neurones  $H_1, H_2$  et  $H_3$

Il est clair que  $T_{H_1 \text{ avant}} = T_{H_3 \text{ avant}}$

$$T_{H_2 \text{ avant}} \leq (T_{H_1 \text{ avant}}, T_{H_3 \text{ avant}})$$

Donc

$$\boxed{T_{\text{AVANT}} = N(26g(t_+ + t_*) + tf) + N(t_s + 26g t_w) + N(t_s + t_w)} \\ \underbrace{N(\frac{\omega^2}{4}(t_+ + t_*) + tf)}_{t_{\text{esce}}} \quad \underbrace{N(t_s + \frac{\omega^2}{4} t_w) + N(t_s + t_w)}_{t_{\text{com}}}$$

Pour Terreur :

$$\boxed{T_{\text{erreur}} = N((26g+2)t_* + t_-) + N(t_s + 26g t_w) + N(t_s + t_w)} \\ \underbrace{N((26g+2)t_* + t_-)}_{t_{\text{esce}}} \quad \underbrace{N(t_s + t_w)}_{t_{\text{com}}}$$

Pour Tarrière :

Ici aussi, il est clair que  $T_{H_1 \text{ arrière}} = T_{H_3 \text{ arrière}}$

$$T_{H_2 \text{ arrière}} < (T_{H_1 \text{ arrière}}, T_{H_3 \text{ arrière}})$$

$$\text{Tarrière} < (T_{H_1 \text{ arrière}}, T_{H_3 \text{ arrière}}) \text{ pour } \frac{\omega^2}{4} > 26g$$

Donc

$$\boxed{T_{\text{arrière}} = N(\frac{\omega^2}{4} + 1)(2t_* + t_+) + N(t_s + t_w)} \\ \underbrace{N(\frac{\omega^2}{4} + 1)(2t_* + t_+)}_{t_{\text{esce}}} \quad \underbrace{N(t_s + t_w)}_{t_{\text{com}}}$$

Et le temps total du traitement parallèle sera :

(18)

$$T_p = T_{\text{Avant}} + T_{\text{Traitement}} + T_{\text{Après}}$$

Par contre, ceci suppose que nous avons 1 CPU/Neurone, ce qui n'est pas nécessairement le cas.

Donc, pour des nombres de CPU inférieurs au nombre de neurones, il convient de faire des regroupements en rapport avec le nombre de CPU disponibles.

Pour faire ces regroupements, il faut tenter d'équilibrer la charge des CPU de manière possible. Et à ce titre, une première constatation est que la charge des neurones n'est pas équilibrée. En effet, les neurones de  $H_1$  et  $H_3$  ont plus de travail à faire que celles de  $H_2$  et  $O$ . Et  $H_2$  est celle qui a le moins de travail à faire.

Afin de bien équilibrer les charges, nous pourrions faire une évaluation en profondeur des charges respectives des neurones et d'en déduire une répartition optimale. Mais c'est un travail ardu.

On bien, nous pourrions simplement laisser l'OS gérer l'ensemble des tâches, afin de répartir dynamiquement les tâches selon la disponibilité des CPU. Par contre, on aura à subir de nombreux changements de contexte pour l'ordonnancement en cours d'exécution.

Une autre alternative serait de regrouper le traitement des neurones dans les tâches, ce qui évite les changements de contexte. Et ce regroupement pourrait être fait de façon à répartir les neurones le plus équitablement possible dans les tâches.

Par exemple, en établissant la liste des neurones et en attribuant les neurones de la liste, une à la fois, à chaque CPU à tour de rôle. C'est un ordonnancement statique cyclique.

Du moment que  $P < (26g + 1)$ , il y aura des CPU qui auront plus qu'un neurone à traiter. Avec l'attribution cyclique, un CPU donné aura :

$$\text{Pour } 1 \text{ CPU} \Rightarrow \left\{ \begin{array}{l} \text{max ou min} \\ M_{H_1} \text{ ou } (M_{H_1} - 1) \text{ de } H_1 \\ M_{H_2} \text{ ou } (M_{H_2} - 1) \text{ de } H_2 \\ M_{H_3} \text{ ou } (M_{H_3} - 1) \text{ de } H_3 \\ 1 \text{ ou } 0 \text{ de } O \end{array} \right.$$

(19)

Alors, toujours pour un CPU séquentiel, nous avons

$$T_{\text{AVANT}} = M_{H_1} T_{H_1 \text{ avant}} + M_{H_2} T_{H_2 \text{ avant}} + M_{H_3} T_{H_3 \text{ avant}} + T_{O \text{ avant}}$$

+

$$T_{\text{erreur}} = T_{O \text{ erreur}}$$

$O$  doit être traité  
après  $H_1, H_2$  et  $H_3$ .

+

$$T_{\text{arrière}} = M_{H_1} T_{H_1 \text{ arrière}} + M_{H_2} T_{H_2 \text{ arrière}} + M_{H_3} T_{H_3 \text{ arrière}} + M_O T_{O \text{ arrière}}$$

$$\boxed{T_P = M_{H_1} (T_{H_1 \text{ avant}} + T_{H_1 \text{ arrière}}) + M_{H_2} (T_{H_2 \text{ avant}} + T_{H_2 \text{ arrière}}) \\ + M_{H_3} (T_{H_3 \text{ avant}} + T_{H_3 \text{ arrière}}) + (T_{O \text{ avant}} + T_{O \text{ erreur}})}$$

Donc, tout dépend de la répartition des tâches dans les tâches attachées à un CPU séquentiel.

En fait, il faut considérer le pire cas, car les CPU qui ont moins de travail à faire doivent attendre que celui qui en a plus termine. Cense-là auront un "temps mort" égal à la différence entre leur temps d'exécution et celui du pire cas.

$$\text{donc } T_P = (M_{H_1} + M_{H_3}) N \left[ \frac{w^2}{4} (t_+ + t_*) + t_f + \left( \frac{w^2}{4} + 1 \right) (2t_* + t_+) + \left( t_s + \frac{w^2}{4} t_w \right) \right] \\ + 2(t_s + t_w) \\ + M_{H_2} N \left[ \frac{w^2}{16} (t_+ + t_*) + t_f + \left( \frac{w^2}{16} + 1 \right) (2t_* + t_+) + \left( t_s + \frac{w^2}{16} t_w \right) + 2(t_s + t_w) \right] \\ + N \left[ 26g (t_+ + t_*) + t_f + 2(t_s + 26gt_w) + (t_s + t_w) + (26g + 2)t_* + t_- \right] \\ + 2(t_s + t_w) + (t_s + (26g + 1)t_w)$$

$$\text{et } G = \frac{T_{\text{seq}}}{T_P}$$

mais  $E \neq \frac{T_{\text{seq}}}{P T_P}$  car  $O_{\text{avant}}$  et  $O_{\text{erreur}}$  sont strictement séquentiels.

alors

$$\boxed{E = \frac{T_{\text{seq}}}{P \left[ M_{H_1} (T_{H_1 \text{ avant}} + T_{H_1 \text{ arrière}}) + M_{H_2} (T_{H_2 \text{ avant}} + T_{H_2 \text{ arrière}}) + M_{H_3} (T_{H_3 \text{ avant}} + T_{H_3 \text{ arrière}}) \right] \\ + (T_{O \text{ avant}} + T_{O \text{ erreur}})}}$$

Pour un nombre  $P$  de CPU, il faut déterminer le pire cas afin de calculer  $G$  et  $E$ . Le pire cas contiendra le plus possible de neurones de type  $H_1$  et  $H_3$ , car ceux-ci ont des temps de traitement les plus longs.

Voici quelques règles de paix pour choisir ce pire cas.

$(26g + 1)/P \rightarrow$  Arrondit à l'entier supérieur donne le nombre maximum de neurones par CPU

$H_1: 4g/P$       }  
 $H_2: 16g/P$       }  
 $H_3: 6g/P$       }  
 → { Arrondit à l'entier supérieur donne le nombre maximum de ces types de neurones par CPU. Et le nombre minimum est  
 min = max - 1  
 Sauf s'il n'y a pas de partie décimale. Dans ce cas, min = max.

Avec ça, on détermine le pire cas pour les  $P$  suivants:

$P$	$M_{H_1}$	$M_{H_2}$	$M_{H_3}$	$M_0$	<u>pour <math>g = 3</math></u>
2	6	24	9	1	
4	3	12	5	0	
8	2	6	2	0	
16	1	3	1	0	
24	1	2	1	0	
32	1	1	1	0	
40	1	0	1	0	

Avec ça et avec :  $g = 3$        $t_f = t_s = 3$  cycles       $t_s = 0$   
 $N = 30000$        $t_f = (max) = 180$  cycles       $t_w = 1$  cycle.

$P$	<u><math>M = 20</math></u>	<u><math>M = 40</math></u>
2	$G = 1.582$ $E = 0.802$	$G = 1.524$ $E = 0.765$
4	$G = 2.960$ $E = 0.769$	$G = 2.892$ $E = 0.731$
8	$G = 5.637$ $E = 0.769$	$G = 5.699$ $E = 0.731$

Pour finir, chacune de ces 3 stratégies ont été comparées au temps de traitement séquentiel. Par contre, pour chacune de ces stratégies, le temps de communication des données a été comptabilisé même si le système est à "mémoire partagée".

En fait, dans ce cas, les données sont communiquées par la mémoire. C'est-à-dire que pour accéder à la donnée, il suffit de la lire en mémoire. Donc, la version séquentielle doit faire ces accès aux données, elle aussi, mais n'en a pas été "tassé".

Ainsi, les évaluations de G et de E qui ont été faites pour les 3 stratégies sont biaisées. Afin d'en faire une évaluation plus juste, il faudrait ajouter ces temps d'accès au calcul de la version séquentielle ou éliminer ces temps d'accès des 3 stratégies.

Par contre, dans ce dernier cas, certains temps d'accès ne peuvent pas être éliminés lorsque ils correspondent à des copies des données d'un emplacement à un autre pour les seuls besoins de l'algorithme de la stratégie.

Néanmoins, l'évaluation théorique, qui a été faite, donne des indications sur les avantages de chacune de ces stratégies, même si pessimistes.

En définitive, la stratégie ① donne les meilleurs résultats au niveau du gain de performance, suivie de la stratégie ② et ensuite la stratégie ③.

Par contre, les stratégies ① et ② utilisent un apprentissage de type "Batch", qui est classiquement plus lent que l'apprentissage "on-line" utilisé par la stratégie ③ ainsi que la version séquentielle.

Donc, finalement, la stratégie ③ sera utilisée, car il est presque certain que son apprentissage "on-line" plus rapide compensera sa plus faible performance par rapport aux deux autres stratégies.