Introduction au calcul GP-GPU

Partie 3 : Intérêt en deep-learning

Laurent Risser

Institut de mathématiques de Toulouse
lrisser@math.univ-toulouse.fr

Prédiction

I =
Image RGB 200*200

Image de chien ou chat

Classifieur (boite noire)

$h^1(\boldsymbol{I})$

$h^2(\boldsymbol{I})$

« Probabilité » de prédiction

- Si chien : $h^1(\boldsymbol{I}) == \boldsymbol{0}$
- Si chat : $h^1(\boldsymbol{I}) == \boldsymbol{1}$
- Si gentil : $h^2(\boldsymbol{I}) == \boldsymbol{0}$
- Si agressif : $h^2(\boldsymbol{I}) == \boldsymbol{1}$

Prédiction

Image de chien ou chat

Classifieur (boite noire)

0.98

0.02

- Si chien : $h^1(\boldsymbol{I}) == \boldsymbol{0}$
- Si chat : $h^1(\boldsymbol{I}) == \boldsymbol{1}$
- Si gentil : $h^2(\boldsymbol{I}) == \boldsymbol{0}$
- Si agressif : $h^2(\boldsymbol{I}) == \boldsymbol{1}$

Prédiction

Image de chien ou chat

Classifieur (boite noire)

0.93

0.99

- Si chien : $h^1(\boldsymbol{I}) == \boldsymbol{0}$
- Si chat : $h^1(\boldsymbol{I}) == \boldsymbol{1}$
- Si gentil : $h^2(\boldsymbol{I}) == \boldsymbol{0}$
- Si agressif : $h^2(\boldsymbol{I}) == \boldsymbol{1}$

Prédiction

Image de chien ou chat

Classifieur (boite noire)

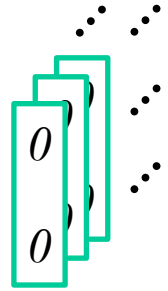
0.01

0.02

- Si chien : $h^1(I) == 0$
- Si chat : $h^1(I) == 1$
- Si gentil : $h^2(I) == 0$
- Si agressif : $h^2(I) == 1$

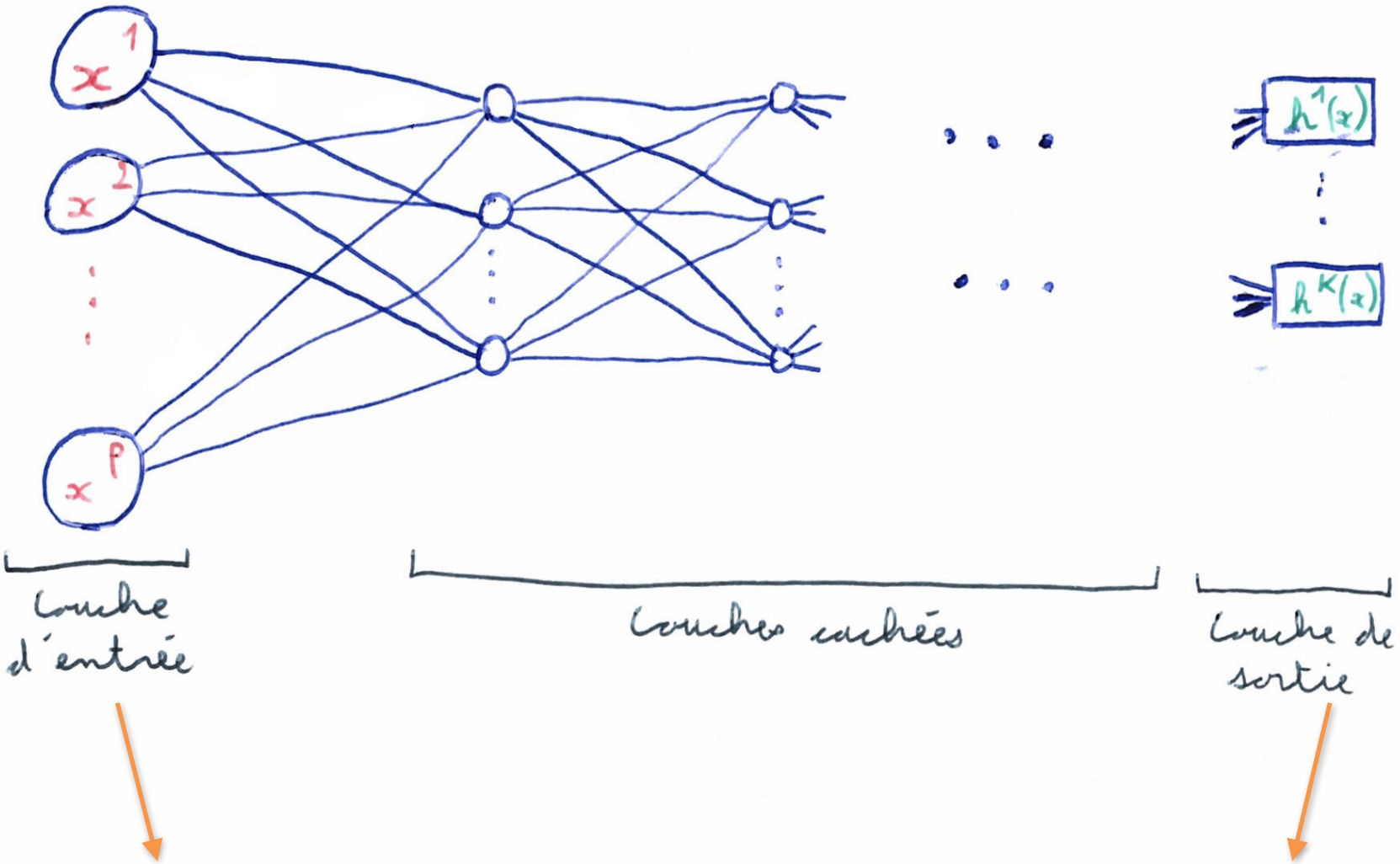
Apprentissage : Optimisation des paramètres pour obtenir les meilleurs prédictions en moyenne

Classifieur (*boite noire*)



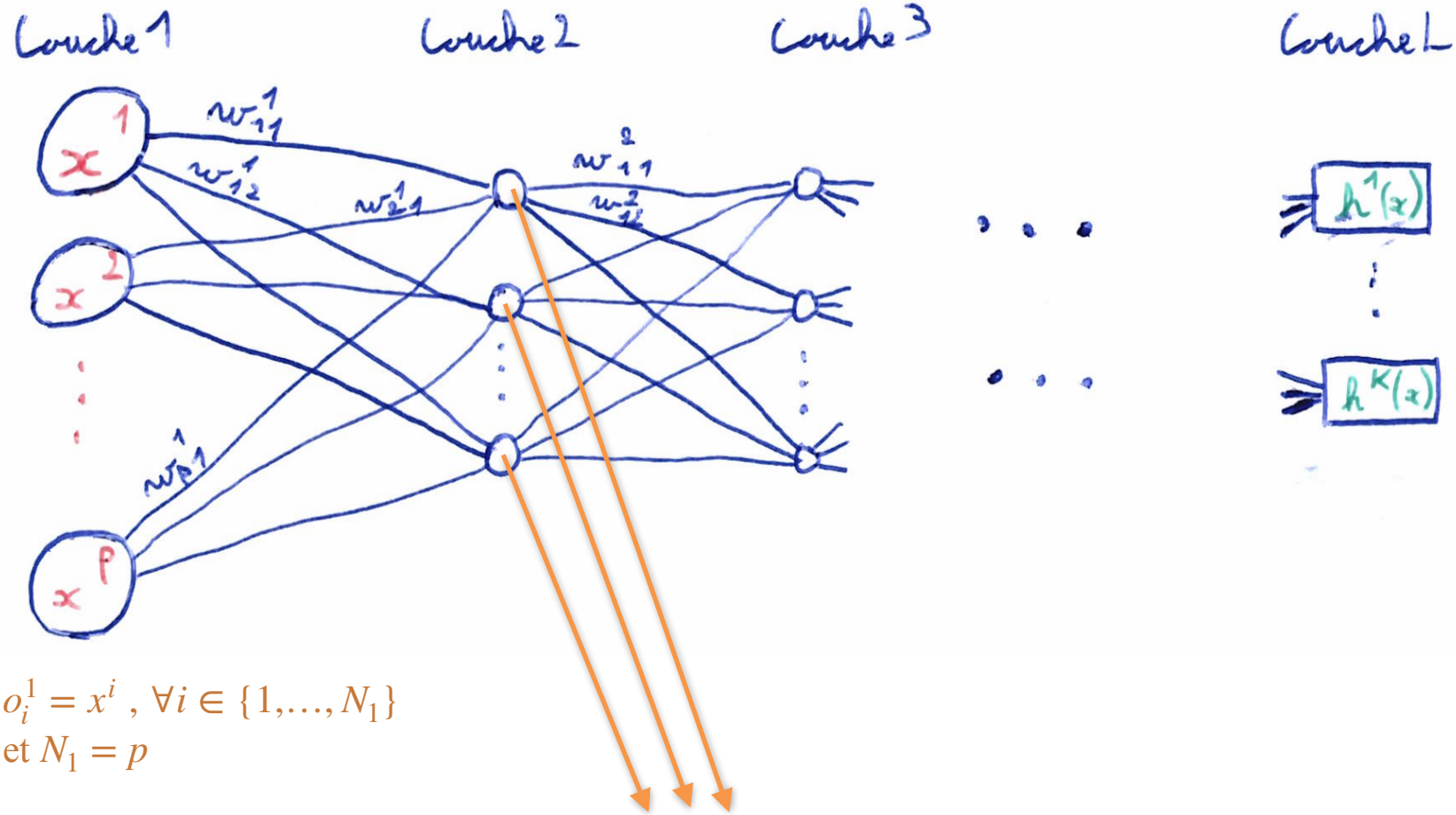
Données d'apprentissage (1/2)
Beaucoup d'images de chien ou chat

Données d'apprentissage (2/2)
Labels de chaque image



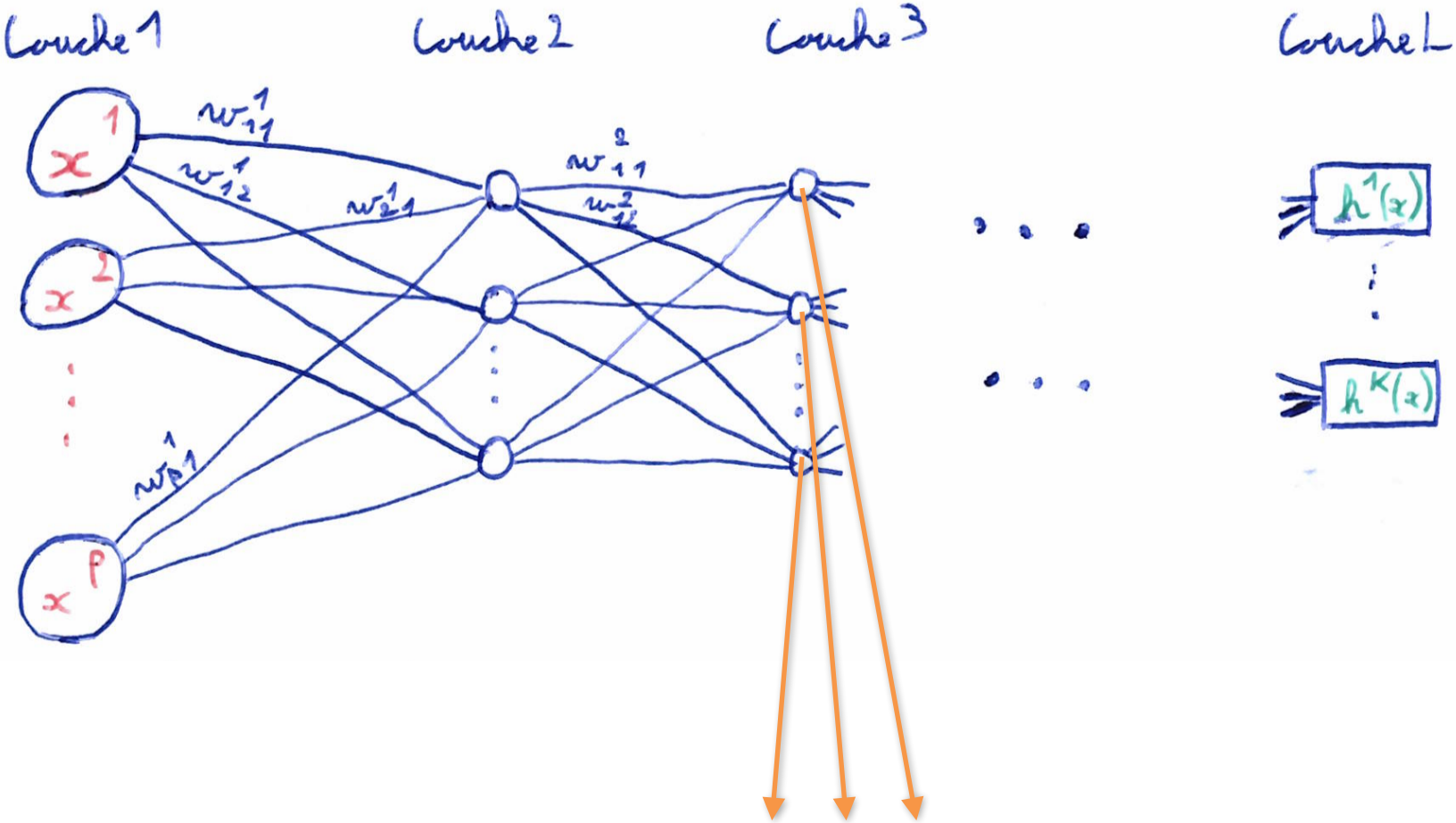
Les x^i peuvent être les intensités de l'image I sur chaque canal RGB

Labels prédits

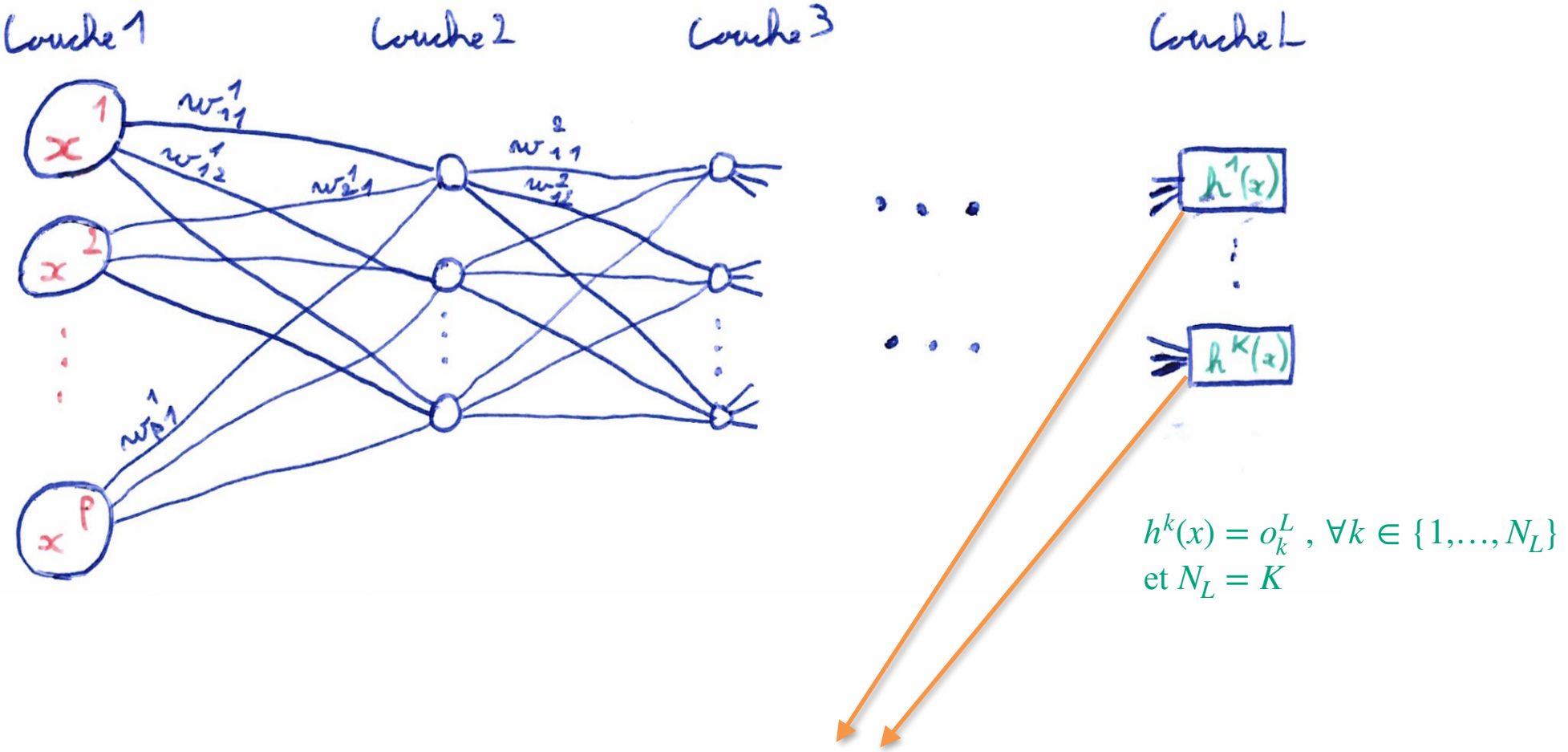


$o_i^1 = x^i$, $\forall i \in \{1,\dots,N_1\}$
et $N_1 = p$

$$\begin{cases} s_k^{l+1} = \sum_{p\in\{1,\dots,N_l\}} w_{pk}^l o_p^l \\ o_k^{l+1} = \varphi(s_k^{l+1}) \end{cases}$$

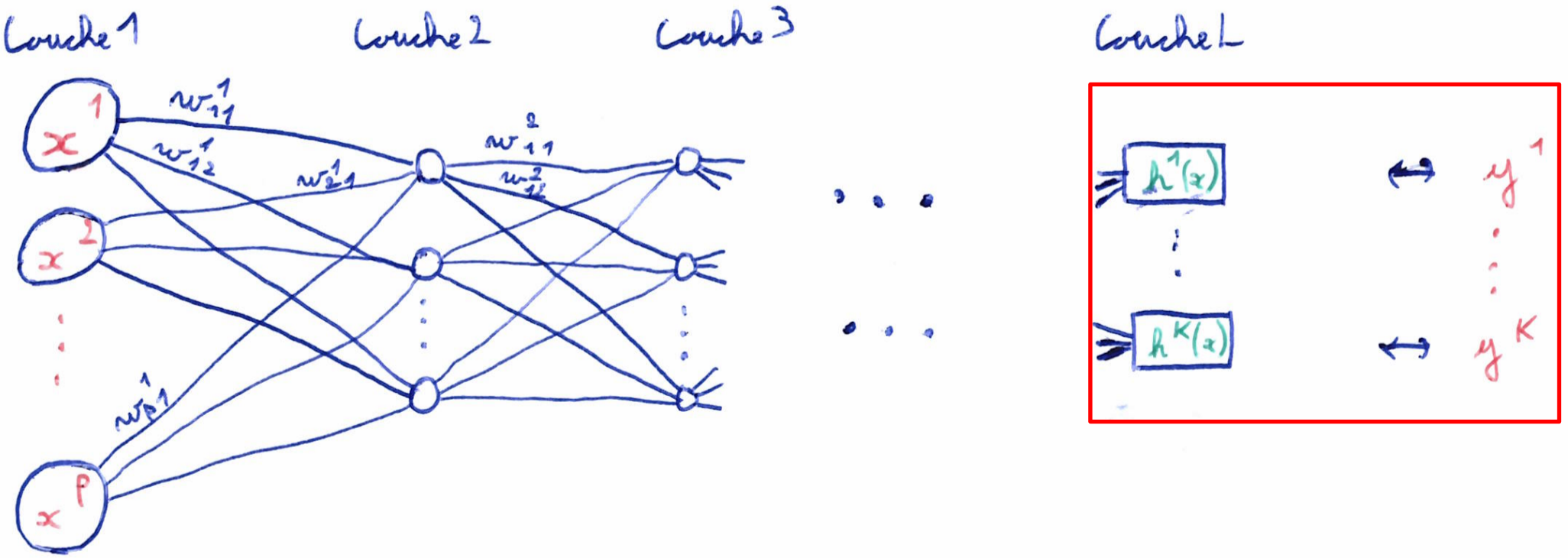


$$\begin{cases} s_k^{l+1} = \sum_{p \in \{1,\dots,N_l\}} w_{pk}^l o_p^l \\ o_k^{l+1} = \varphi(s_k^{l+1}) \end{cases}$$



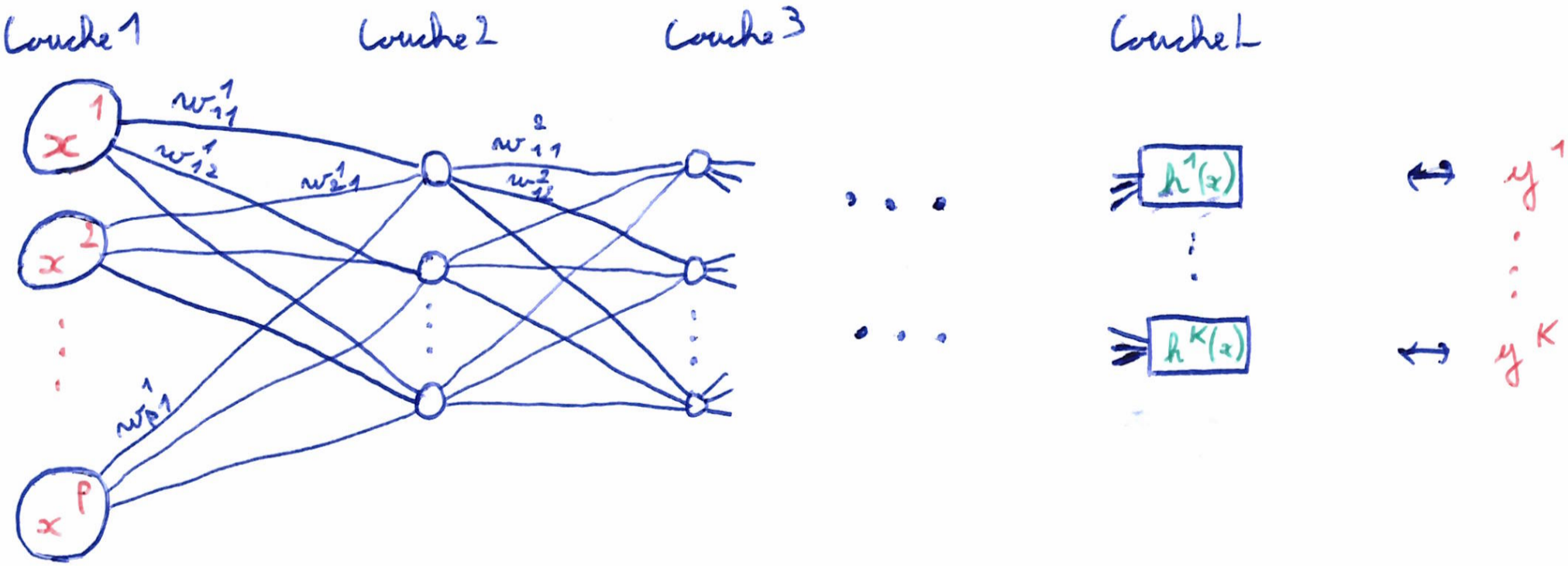
$h^k(x) = o_k^L$, $\forall k \in \{1,\dots,N_L\}$
et $N_L = K$

$$\begin{cases} s_k^{l+1} = \sum_{p\in\{1,\dots,N_l\}} w_{pk}^l o_p^l \\ o_k^{l+1} = \varphi(s_k^{l+1}) \end{cases}$$



Erreur d'estimation dans un mini-batch :

$$E(\underbrace{x^1, \ldots, x^p}_{x}) = \sum_{k=1}^{K} (h^k(x) - y^k)^2$$

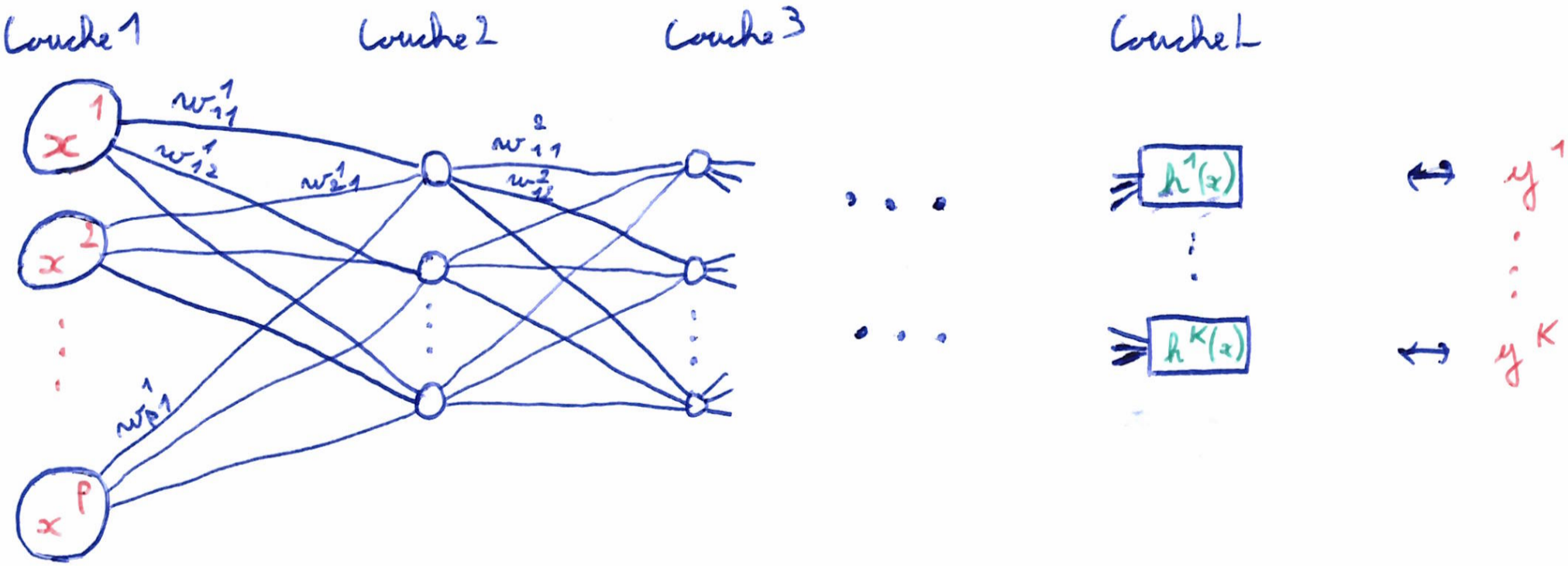


Erreur d'estimation dans un mini-batch :

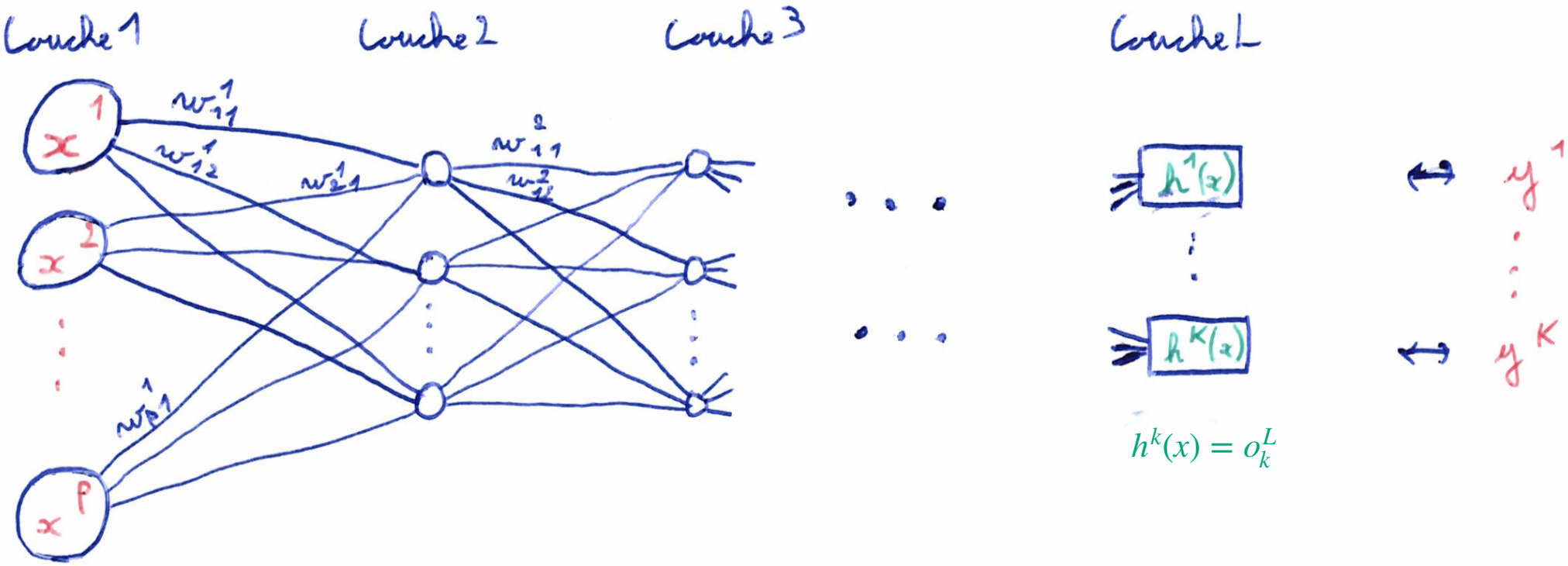
$$E(\underbrace{x^1, \ldots, x^p}_{x}) = \sum_{k=1}^{K} (h^k(x) - y^k)^2$$

Apprentissage par descente de gradient stochastique ≈

- Descente de gradient sur l'erreur moyenne (*risque empirique*) par rapport aux paramètres w^l_{pk} du réseau
- A chaque itération, l'erreur moyenne est approchée avec un sous-ensemble des observations (*mini-batch*)



Itération de la descente de gradient stochastique : $w^l_{pk} = w^l_{pk} - \eta \frac{\partial E}{\partial w^l_{pk}}$, $\forall \{p, k, l\}$

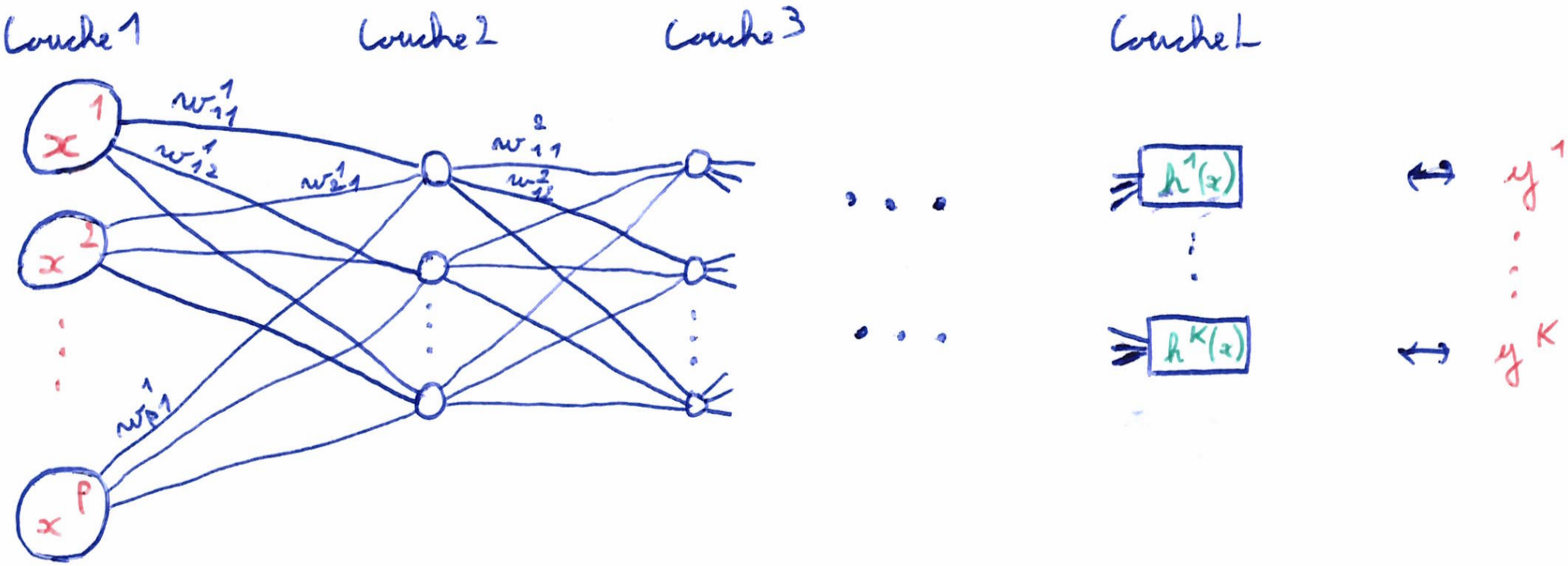


$h^k(x) = o_k^L$

Calcul des dérivées $\frac{\partial E}{\partial w_{pk}^l}$ du risque empirique :

(1) Calcul analytique de $\frac{\partial E}{\partial o_k^L}$ à la sortie du réseau ; par exemple avec la MSE :

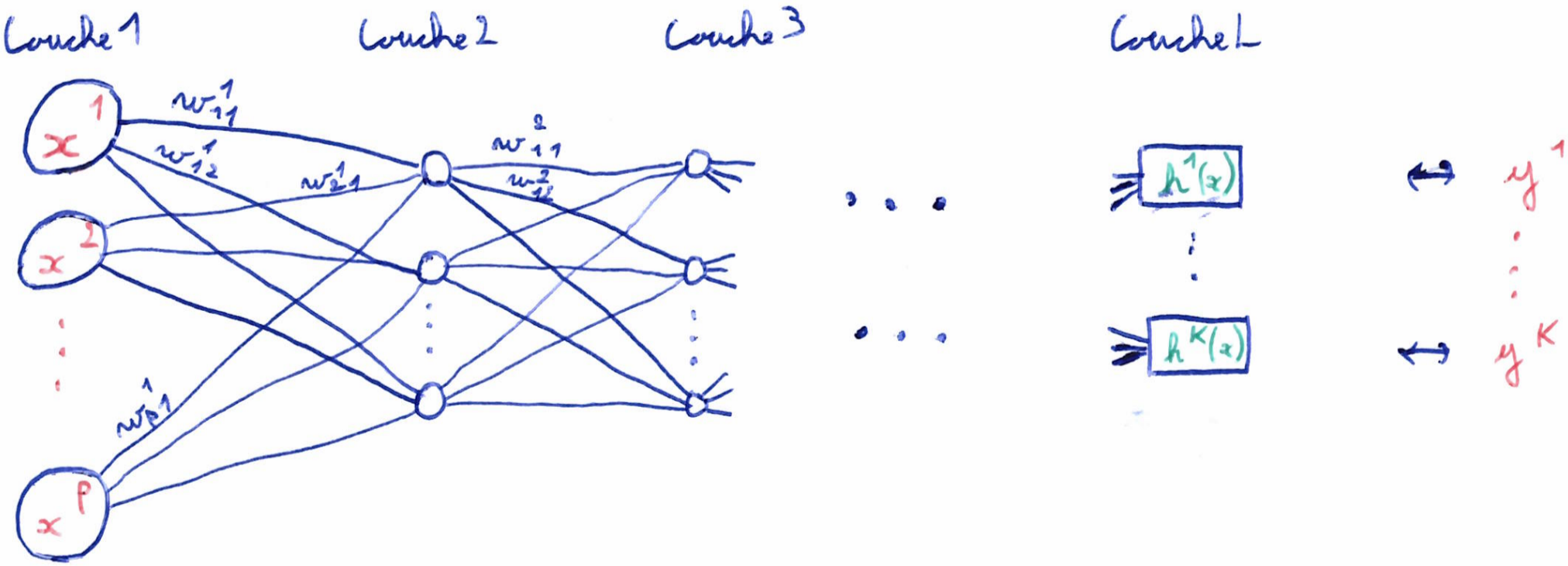
$$\frac{\partial E}{\partial o_k^L} = \frac{\partial}{\partial o_k^L} \frac{1}{2n} \sum_{i=1}^n (\underbrace{h^k(x_i)}_{=o_k^L \text{ for obs. } i} - y_i^k)^2 = \frac{1}{n} \sum_{i=1}^n \left(h^k(x_i) - y_i^k \right)$$



Calcul des dérivées $\dfrac{\partial E}{\partial w^l_{pk}}$ du risque empirique :

(2) Backpropagation des $\dfrac{\partial E}{\partial o^l_k}$:

$$\frac{\partial E}{\partial o^l_k} = \sum_{p=1}^{N_{l+1}} \underbrace{\frac{\partial E}{\partial o^{l+1}_k}}_{\text{Connu à la couche } l+1} \underbrace{\frac{\partial o^{l+1}_k}{\partial s^{l+1}_k}}_{\text{Derivee de } \varphi} \underbrace{\frac{\partial s^{l+1}_k}{\partial o^l_k}}_{=w^l_{pk}}$$

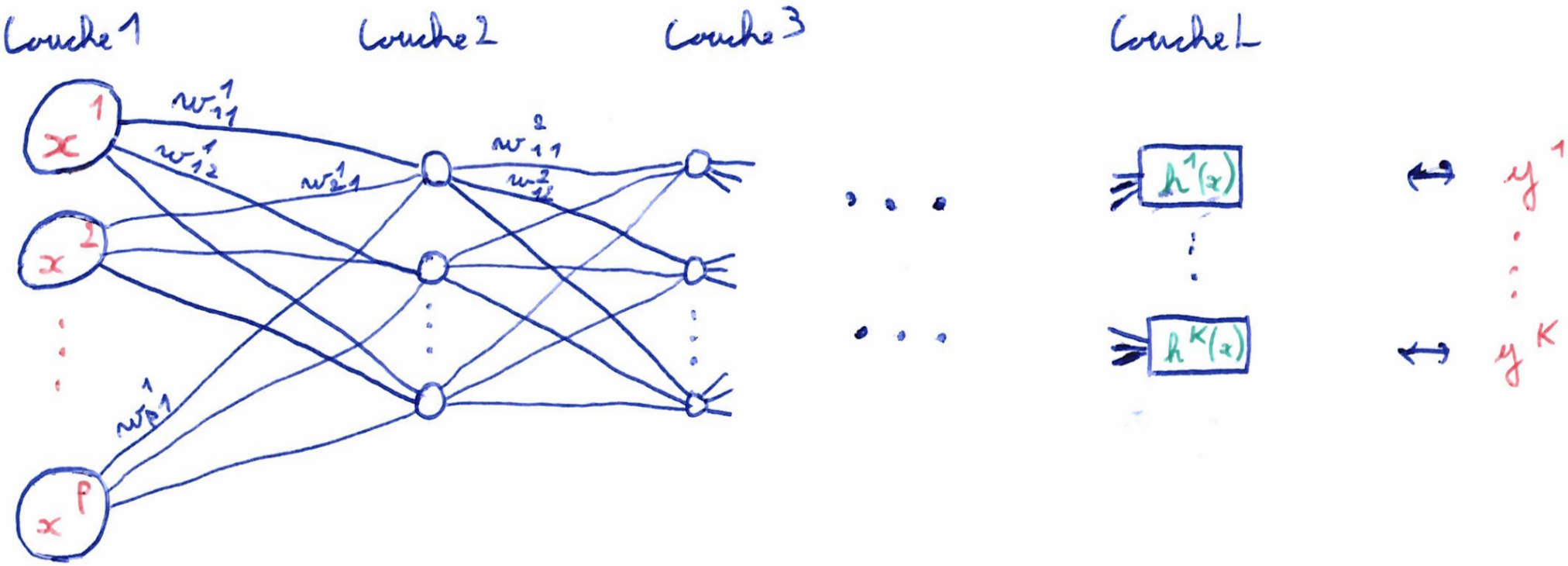


Calcul des dérivées $\dfrac{\partial E}{\partial w^l_{pk}}$ du risque empirique :

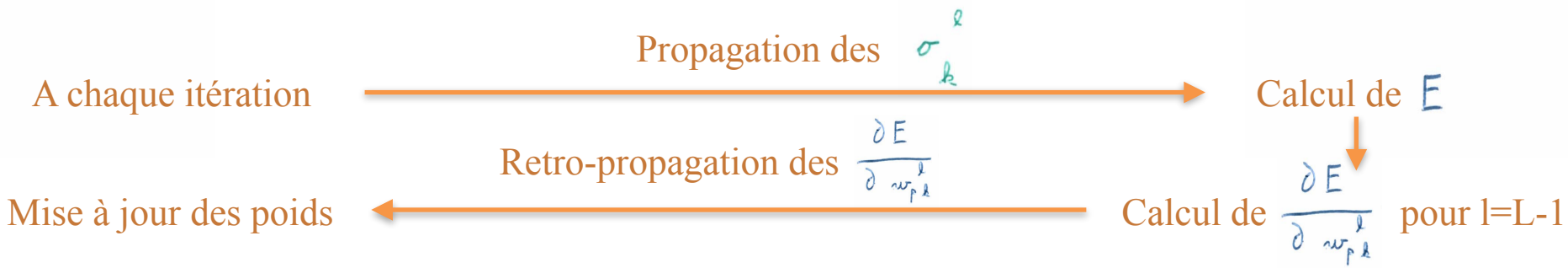
(3) Calcul en parallèle des $\dfrac{\partial E}{\partial w^l_{pk}}$:

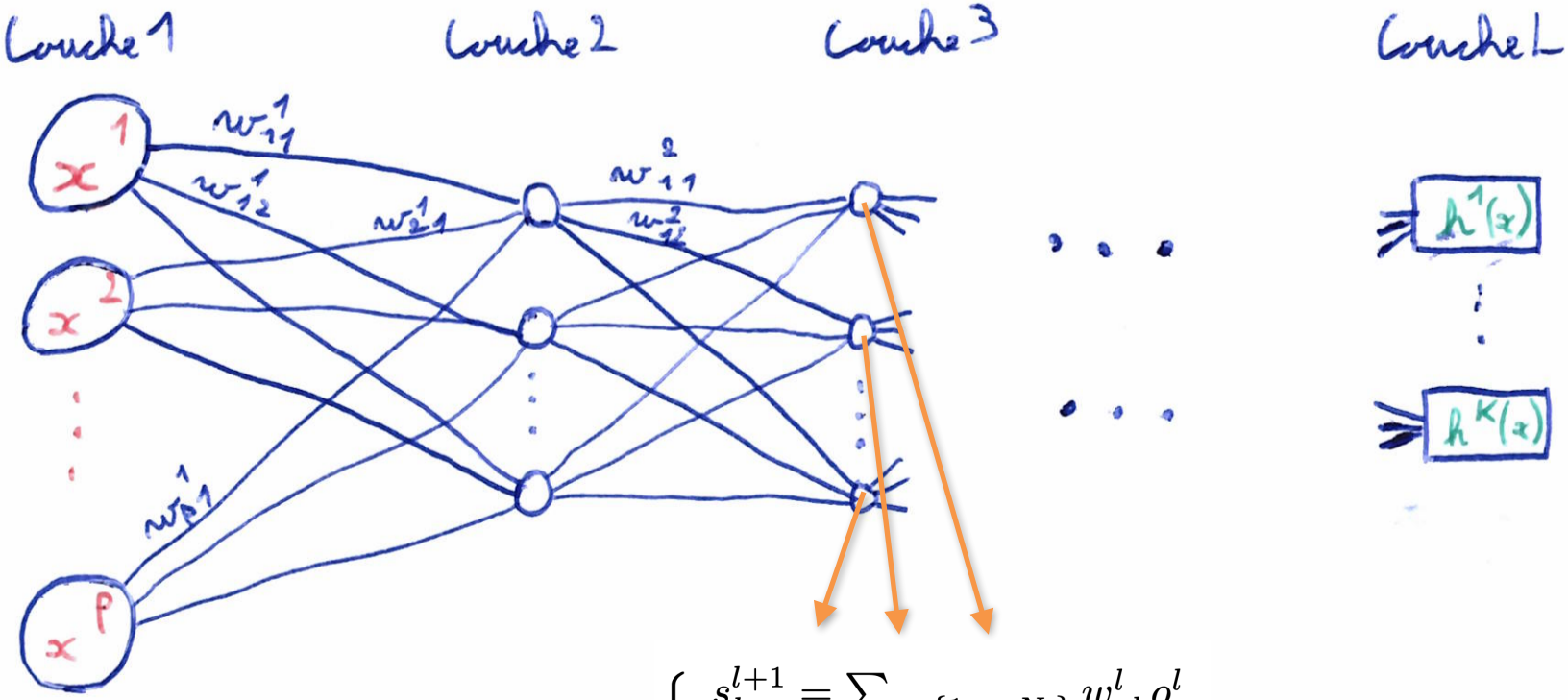
$$\frac{\partial E}{\partial o^l_k} = \sum_{p=1}^{N_{l+1}} \underbrace{\frac{\partial E}{\partial o^{l+1}_k}}_{\text{Connu à la couche } l+1} \underbrace{\frac{\partial o^{l+1}_k}{\partial s^{l+1}_k}}_{\text{Derivee de } \varphi} \underbrace{\frac{\partial s^{l+1}_k}{\partial o^l_k}}_{=w^l_{pk}}$$

$$\boxed{\frac{\partial E}{\partial w^l_{pk}} = \frac{\partial E}{\partial o^{l+1}_k} \frac{\partial o^{l+1}_k}{\partial s^{l+1}_k} \frac{\partial s^{l+1}_k}{\partial w^l_{pk}}}$$



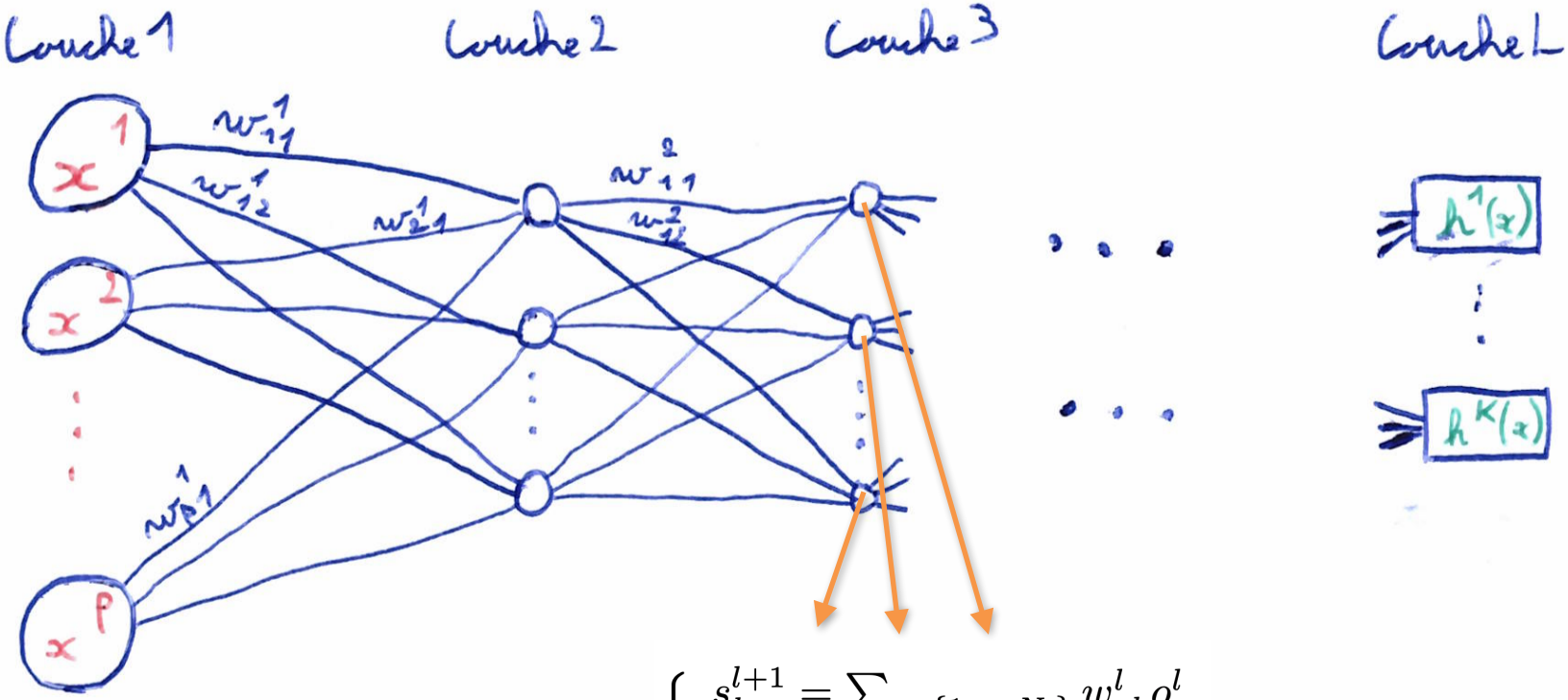
Descente de gradient stochastique (principe) :





$$\begin{cases} s_k^{l+1} = \sum_{p\in\{1,\dots,N_l\}} w_{pk}^l o_p^l \\ o_k^{l+1} = \varphi(s_k^{l+1}) \end{cases}$$

Forme matricielle pour le calcul des o_k^l de la couche l ?



$$\begin{cases} s_k^{l+1} = \sum_{p\in\{1,\dots,N_l\}} w_{pk}^l o_p^l \\ o_k^{l+1} = \varphi(s_k^{l+1}) \end{cases}$$

Forme matricielle pour le calcul des o_k^l de la couche l ?

$$S^l = W^{l-1} O^{l-1} \text{ puis } O^l = \varphi(S^l)$$

$\in \mathbb{R}^{(\#N_l)\times(\#N_{l+1})}$ $\in \mathbb{R}^{\#N_{l+1}}$

```
__kernel void mv_mul(const int N,__global float *W_lm1, __global float *O_lm1, __global float *S_l)
{
  int i;
  int k = get_global_id(0);
  float tmp;   //in private memory

   tmp = 0.0f;
    for (i = 0; i < N; i++)
      tmp += W_lm1[k*N+i]*O_lm1[i];
    S_l[k] = tmp;
  }
```

$$\begin{cases} s_k^{l+1} = \sum_{p\in\{1,\dots,N_l\}} w_{pk}^l o_p^l \\ o_k^{l+1} = \varphi(s_k^{l+1}) \end{cases}$$

Forme matricielle pour le calcul des o_k^l de la couche l ?

$$S^l = W^{l-1}O^{l-1} \text{ puis } O^l = \varphi(S^l)$$

```
__kernel void mv_mul(const int N,__global float *W_lm1, __global float *O_lm1, __global float *S_l)
{
  int i;
  int k = get_global_id(0);
  float tmp;   //in private memory

   tmp = 0.0f;
    for (i = 0; i < N; i++)
      tmp += W_lm1[k*N+i]*O_lm1[i];
    S_l[k] = tmp;
  }
```

OpenCL

```
import tensorflow as tf
N=784
O_lm1 = tf.placeholder(tf.float32, [None, N])
W_lm1 = tf.Variable(tf.zeros([N,128]))
o_l = tf.matmul(O_lm1, W_lm1)
```

Tensorflow/Python

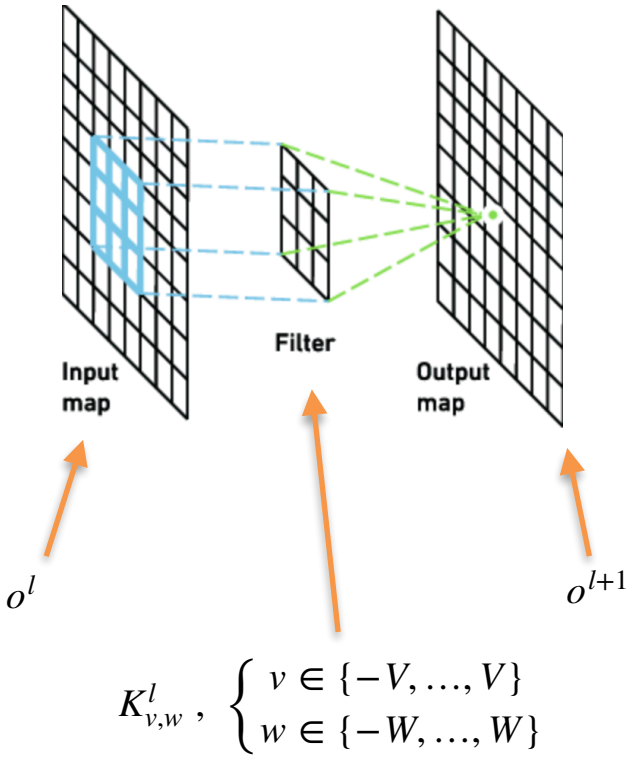
Forme matricielle pour le calcul des o_k^l de la couche l ?

$$S^l = W^{l-1}O^{l-1} \text{ puis } O^l = \varphi(S^l)$$

Parallélisation encore plus intéressantes dans des réseaux convolutionels :

- Modèle *forward* :

$$\begin{cases} s_{f,g}^{l+1} = \sum_{v=-V}^{V} \sum_{w=-W}^{W} K_{v,w}^{l} o_{f+v,g+w}^{l} \\ o_{f,g}^{l+1} = \varphi(s_{f,g}^{l+1}) \end{cases}$$



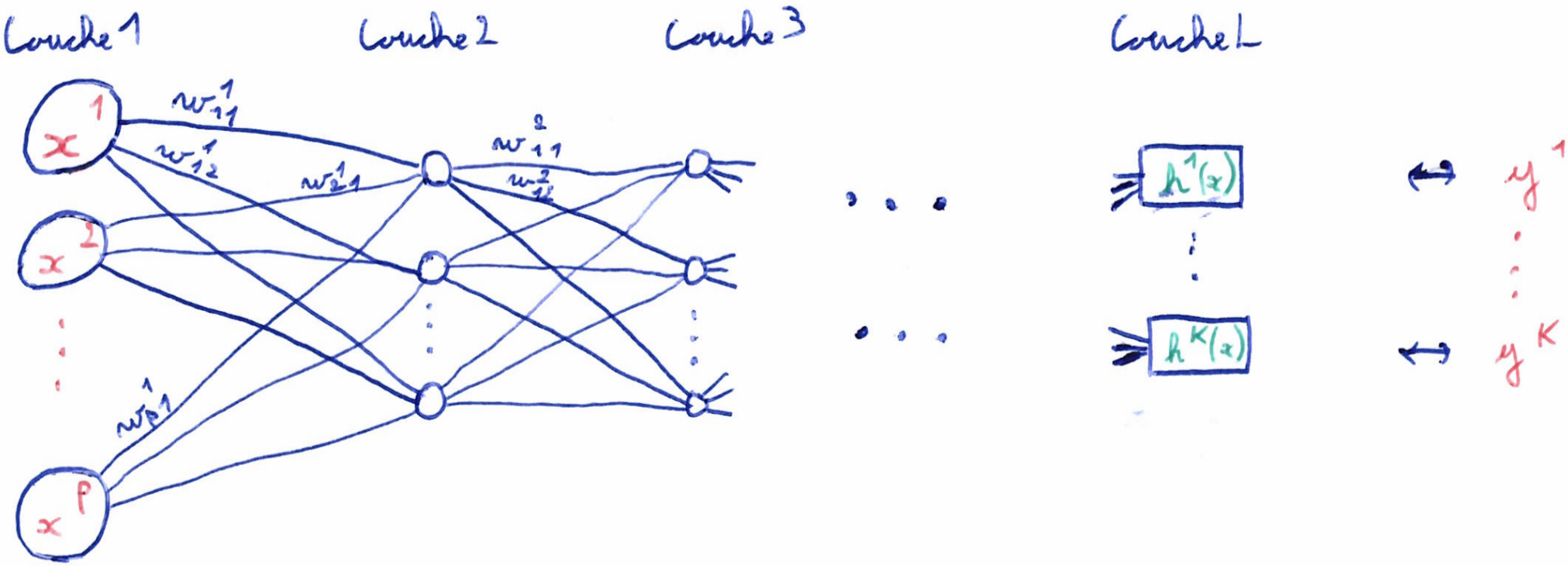
- Modèle *backward* :

$$\frac{\partial E}{\partial o_{m,n}^{l}} = \sum_{v=-V}^{V} \sum_{w=-W}^{W} \frac{\partial E}{\partial o_{m+v,n+w}^{l+1}} \frac{\partial o_{m+v,n+w}^{l+1}}{\partial s_{m+v,n+w}^{l+1}} \frac{\partial s_{m+v,n+w}^{l+1}}{\partial o_{m,n}^{l}}$$

$$= \frac{\partial E}{\partial o_{m+v,n+w}^{l+1}} \varphi'(s_{m+v,n+w}^{l+1}) K_{-v,-w}^{l}$$

$$\frac{\partial E}{\partial K_{v,w}^{l}} = \sum_{m=1}^{M} \sum_{n=1}^{N} \frac{\partial E}{\partial o_{m,n}^{l+1}} \frac{\partial o_{m,n}^{l+1}}{\partial s_{m,n}^{l+1}} \frac{\partial s_{m,n}^{l+1}}{\partial K_{v,w}^{l}}$$

$$= \frac{\partial E}{\partial o_{m,n}^{l+1}} \varphi'(s_{m,n}^{l+1}) o_{m+v,n+w}^{l}$$



Dans le cas général :

- Prédiction et apprentissage particulièrement simples à paralléliser sur GPU
- Librairie *Nvidia cuDNN* massivement utilisée et sous jacente à Caffe, Keras, TensorFlow, Theano…