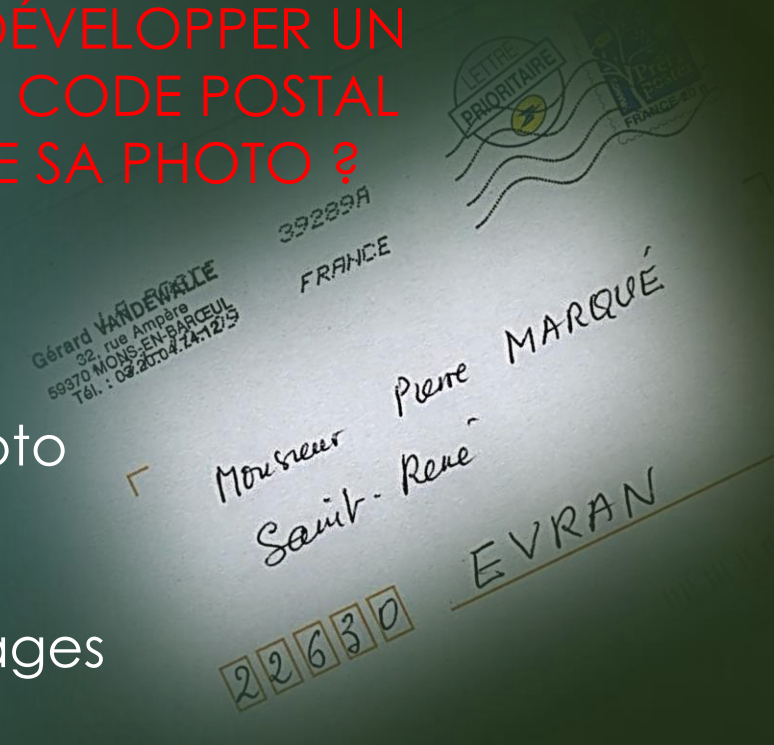
Reconnaissance du code postal sur une enveloppe

PROBLÉMATIQUE : COMMENT PEUT-ON DÉVELOPPER UN PROGRAMME PERMETTANT D'OBTENIR LE CODE POSTAL INSCRIT SUR UNE ENVELOPPE À PARTIR DE SA PHOTO ?



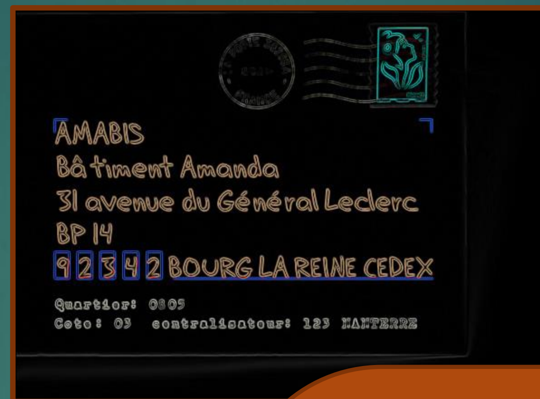
Plan :

I – Récupération des chiffres sur une photo d'enveloppe

II – Méthodes de reconnaissances d'images
- Classifiation Bayesienne
- Perceptron multicouche
- Deep Belief Network

Présentation de l'algorithme

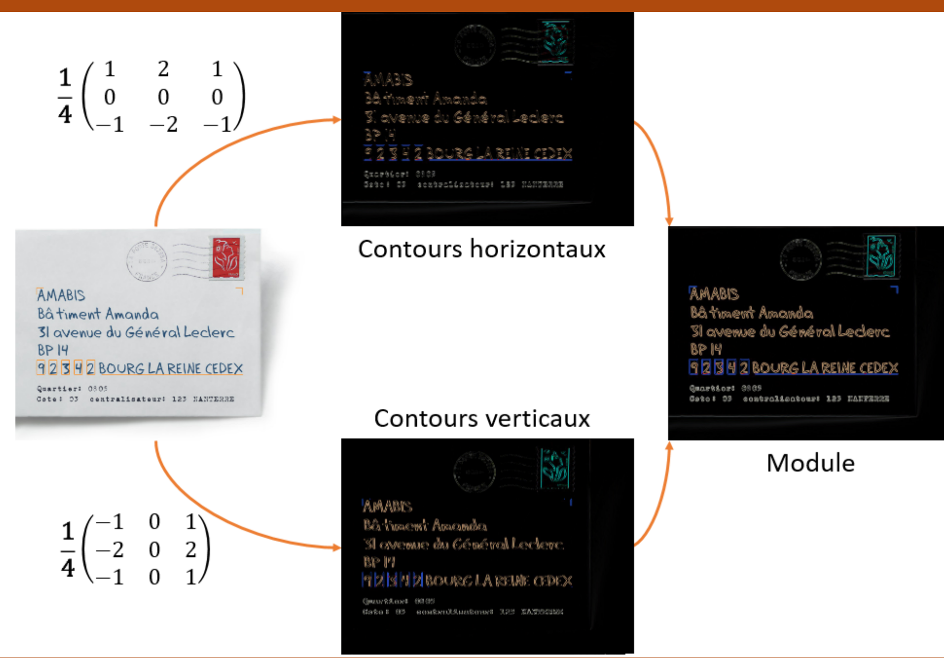
Détection de contour



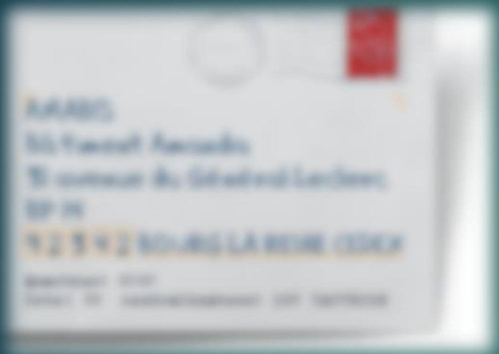
Détection de contour par opérateur de Sobel

$$\frac{1}{4}\begin{pmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{pmatrix}$$

$$\frac{1}{4}\begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix}$$



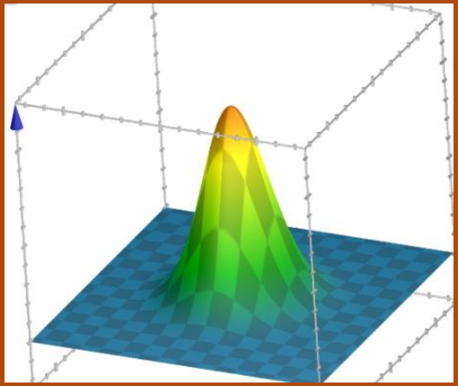
Filtrage sur les couleurs



Filtrage sur les couleurs

Lissage

Filtre gaussien

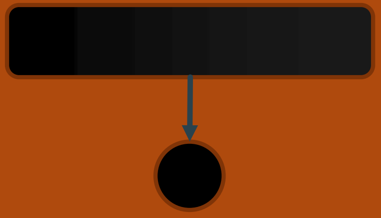


$$exp\left(-\left(\frac{x^2+y^2}{2\,\sigma^2}\right)\right)$$

Exemple pour $\sigma = 1{,}6$

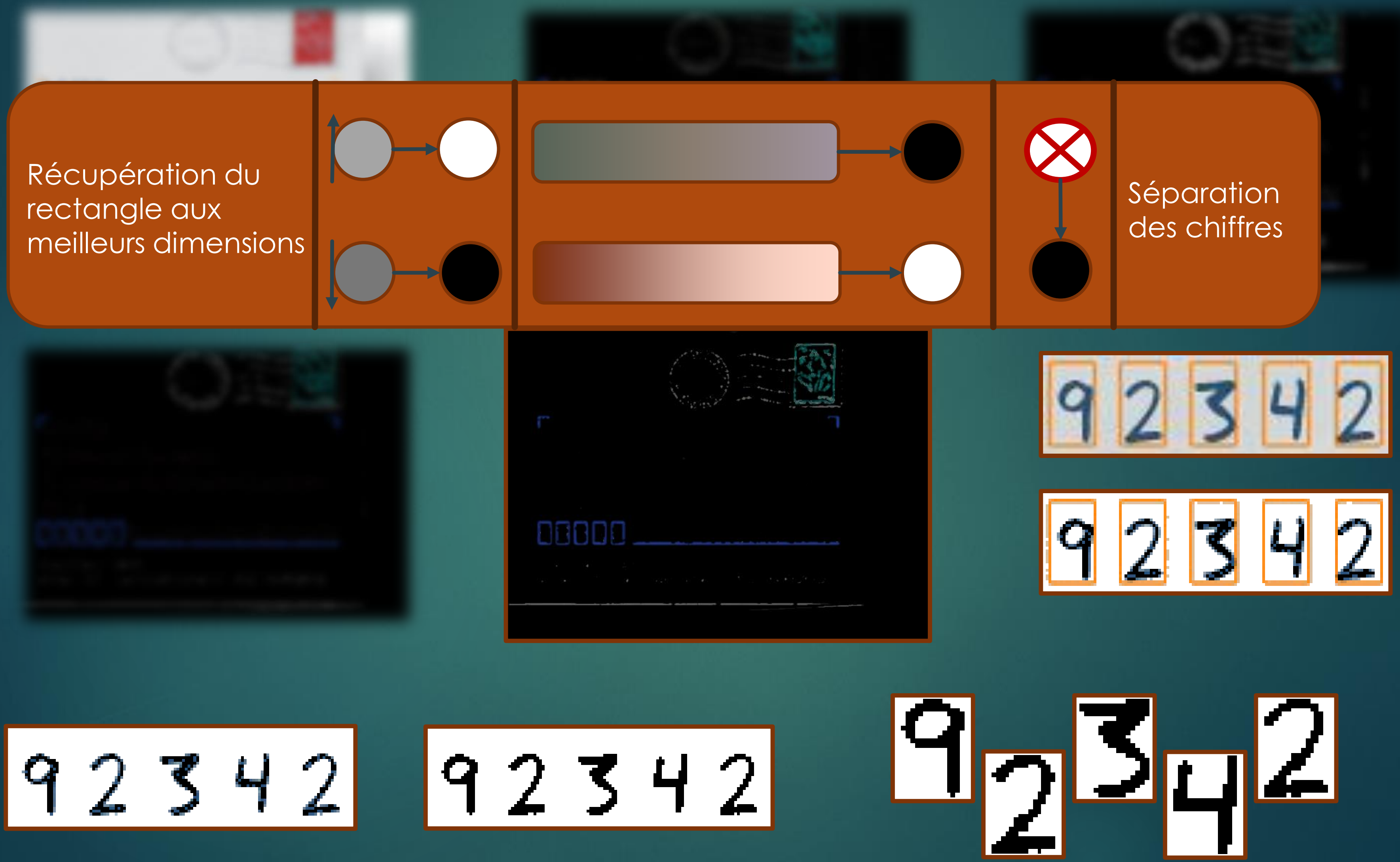
$$M = \frac{1}{103}\begin{pmatrix} 10 & 12 & 10 \\ 12 & 15 & 12 \\ 10 & 12 & 10 \end{pmatrix}$$

Filtrage sur les couleurs

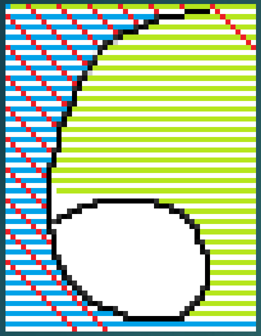


(+ filtre médian)

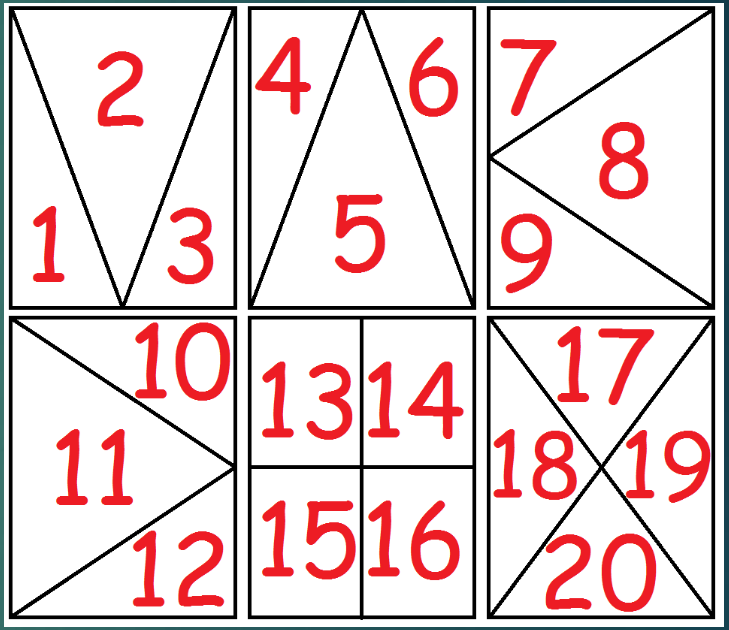
Fin de l’algorithme



Variables caractéristiques



- Profil orienté (PFO)
- Profil gauche (PFG)
- Profil droit (PFD)

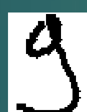


Total de 100 variables :
$X_1, X_2, \ldots X_{100}$ dans $[0, 1]^{100}$

Classification naïve Bayésienne

C variable de classe dans $[\![0, 9]\!]$. Les X_i sont indépendants.

Formule de Bayes : $\mathbb{P}(C = k \mid X_1 \ \dots \ X_{100}) = \frac{\mathbb{P}(C=k)}{\mathbb{P}(X_1 \dots X_{100})} \prod_{i=1}^{100} \mathbb{P}(X_i \mid C = k)$



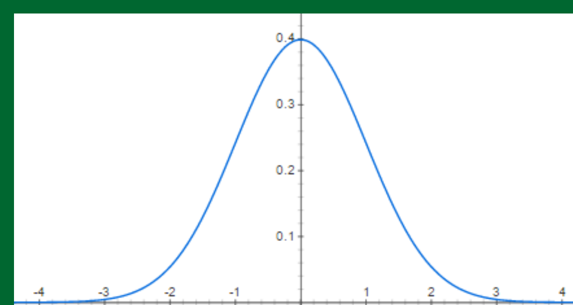
Calcul des X_i

Calcul des $\mathbb{P}(C = k \mid X_1 \ \dots \ X_{100})$

Renvoie de la valeur k pour laquelle $\mathbb{P}(C = k \mid X_1 \ \dots \ X_{100})$ est maximale.

Modélisation : loi normale

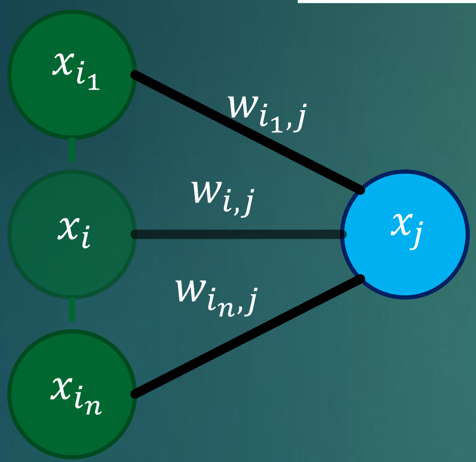
$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$



Résultat : 65,19%

Perceptron multicouche (MLP)

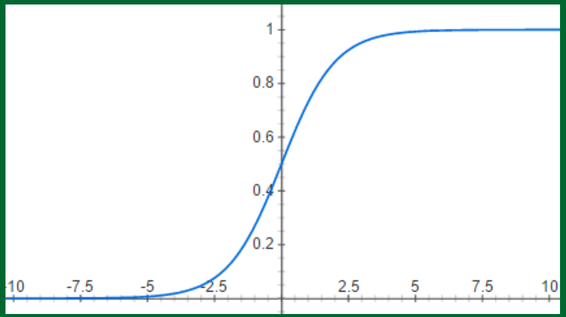
Un neurone



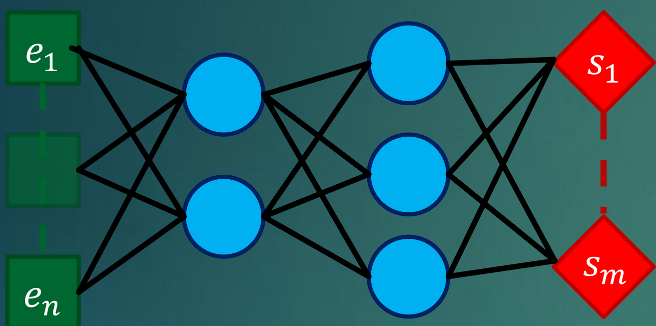
$$x_j = f(h_j)$$

$$h_j = \sum_{i \in pred(j)} w_{i,j} x_i$$

$$f(x) = sigm(x) = \frac{1}{1+e^{-x}}$$



Perceptron multicouche (MLP)



Obtention de e et de la sortie attendue a.

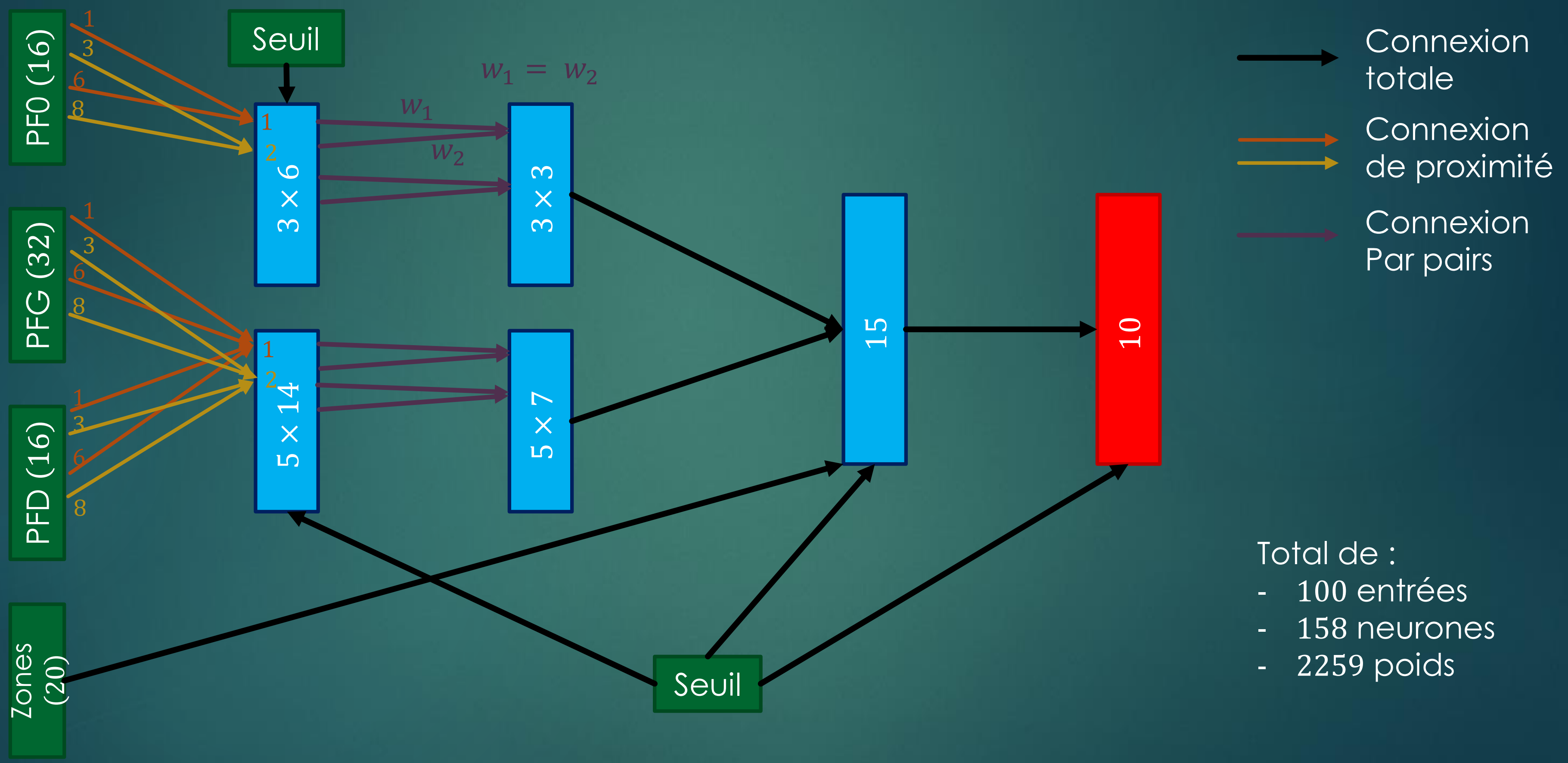
Mise à jour des valeurs des neurones jusqu’à obtenir s.

Calcul de E, mise à jour des poids pour faire diminuer E.

Erreur quadratique : $E = \frac{1}{2}\sum_{j=1}^{m}(s_j - a_j)^2$

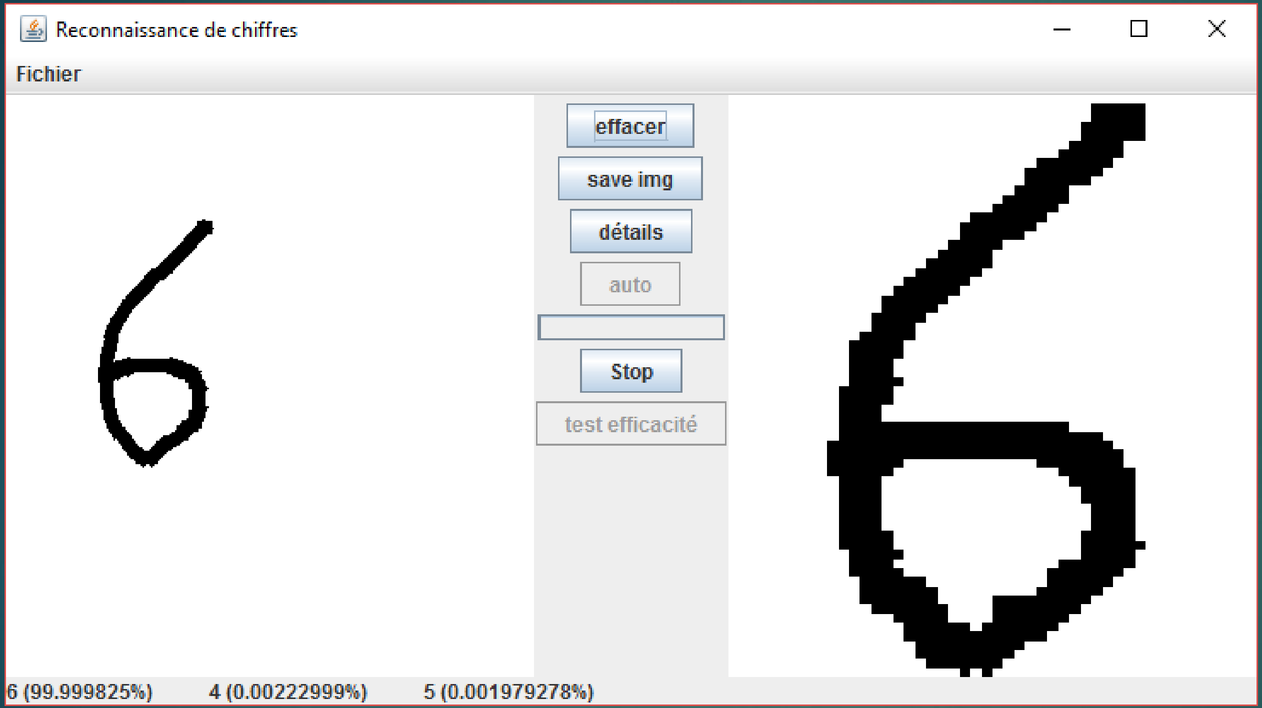
Mise à jour des poids : $w_{i,j} \leftarrow w_{i,j} - \alpha \frac{\partial E}{\partial w_{i,j}}$

Perceptron multicouche (MLP)



Total de :
- 100 entrées
- 158 neurones
- 2259 poids

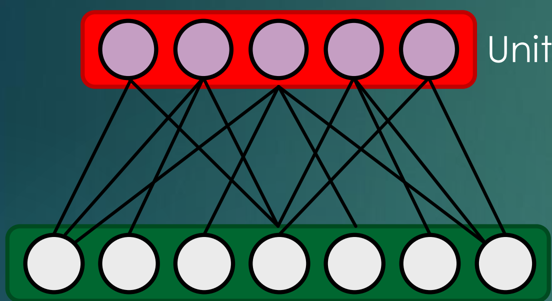
Perceptron multicouche (MLP)



Résultat :
99,76%
$e_Q = 2{,}07$

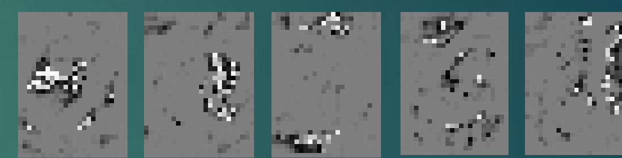
Deep Belief Network (DBN)

Machine de Boltzmann restreinte



Unités cachées

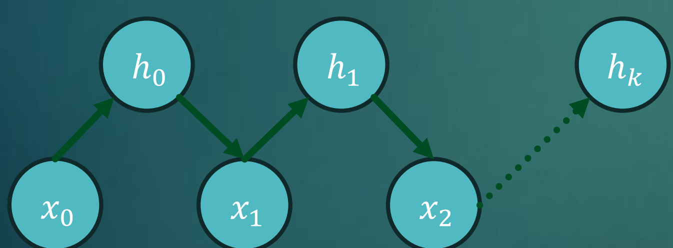
Unités visibles



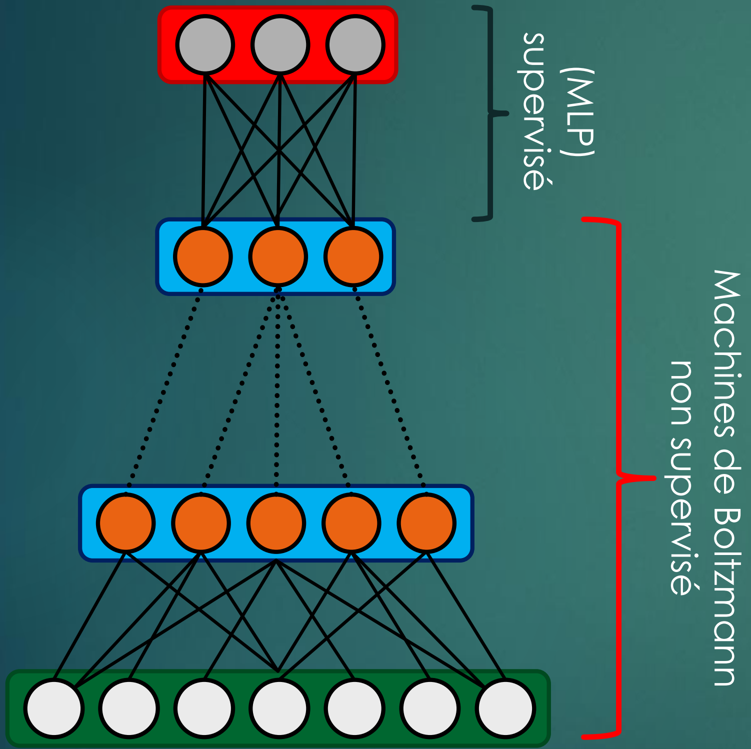
Mise à jour des unités visibles, calcul de celles cachées → (x_0, h_0).

Echantillonnage de Gibbs → (x_k, h_k).

Mise à jours des poids grâce aux deux vecteurs précédents.



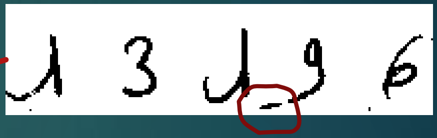
Deep Belief Network (DBN)



Résultat :
97,21%
$e_Q = 31{,}37$

Conclusion

enveloppes	Réseau bayésien : N	(MLP) : e_Q	(DBN) : e_Q
1	2	0,0036	0,84
2	2	0,033	3,2
3	2	0,021	0,00040
4	3	0,00054	0,18
5	erreur	erreur	erreur
6	0	3,3	3,8
7	2	1,0	2,5
8	2	1,9	2,1
9	2	0,000019	1,2
10	2	0,00035	2,3
11	0	0,00033	0,22
12	4	0,012	0,039
13	2	0,0056	2,0
14	1	0,000049	1,3
15	3	1,5	0,31
16	4	1,4	0,013
17	2	0,044	1,5
18	5	0,00032	0,00031
moyenne	$N = 2{,}2$	$N = 4{,}7 \quad e_Q = 0{,}012$	$N = 3{,}9 \quad e_Q = 0{,}28$



$N = 5$

$N = 4$

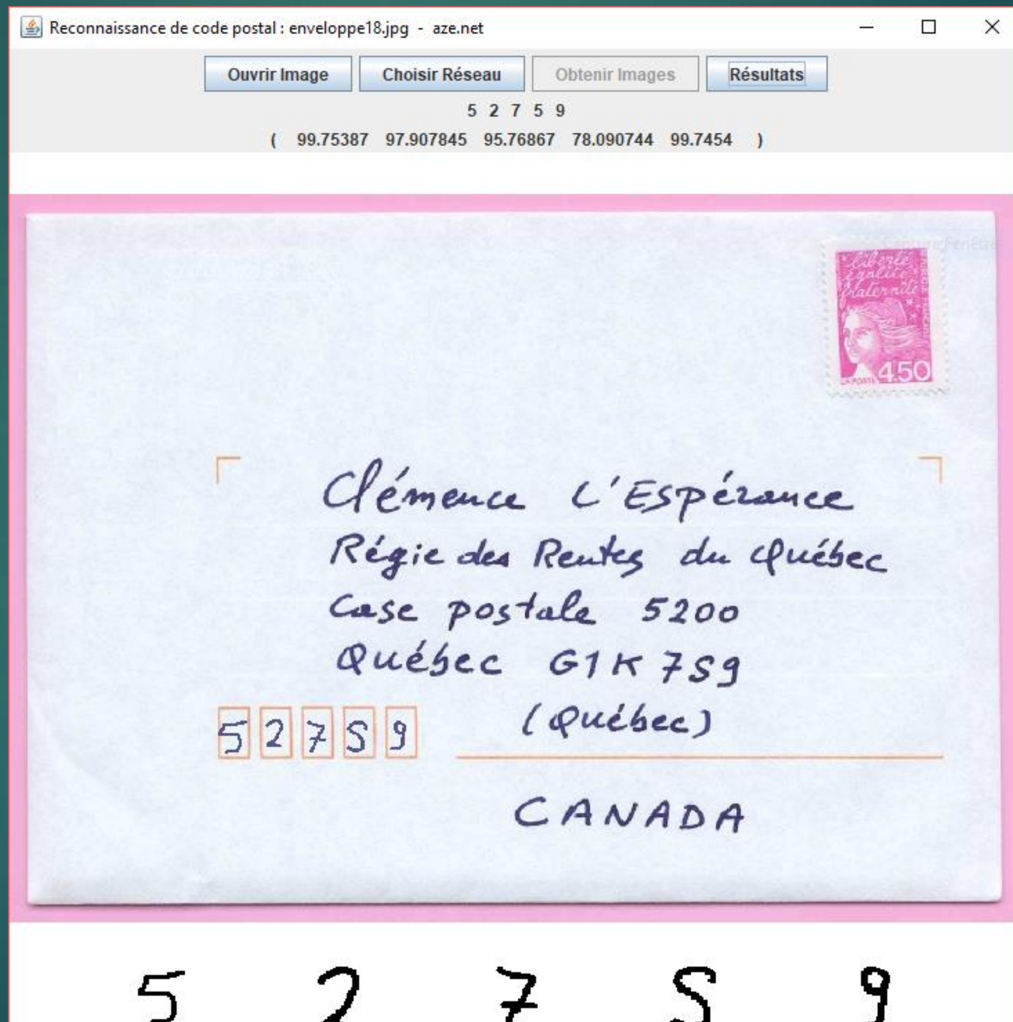
$N = 3$

$N = 2$

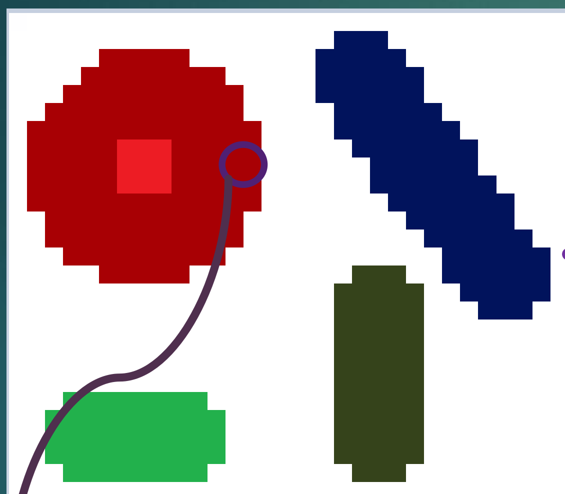
$N = 1$

$N = 0$

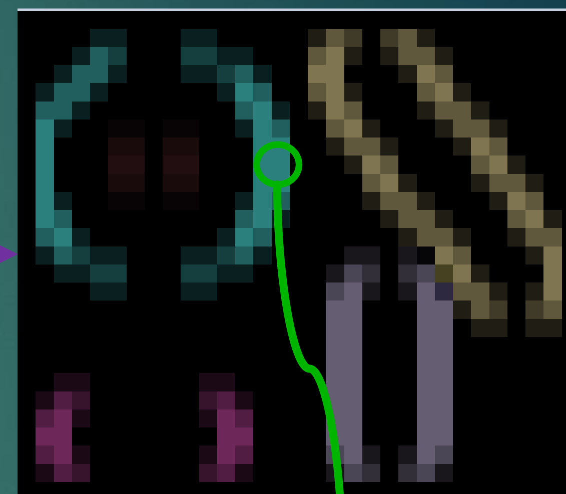
Programme final



Matrice de convolution



$$\frac{1}{4}\begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix}$$



$R : 168$

$V : \quad 0$

$B : \quad 4$

$$\frac{1}{4}\begin{pmatrix} -168 + 0 \times 168 + 255 \\ -2 \times 168 + 0 \times 168 + 2 \times 255 \\ -168 + 0 \times 168 + 255 \end{pmatrix}$$

...

$R : \quad 87$

$V : 255$

$B : 251$

Erreur (MLP)

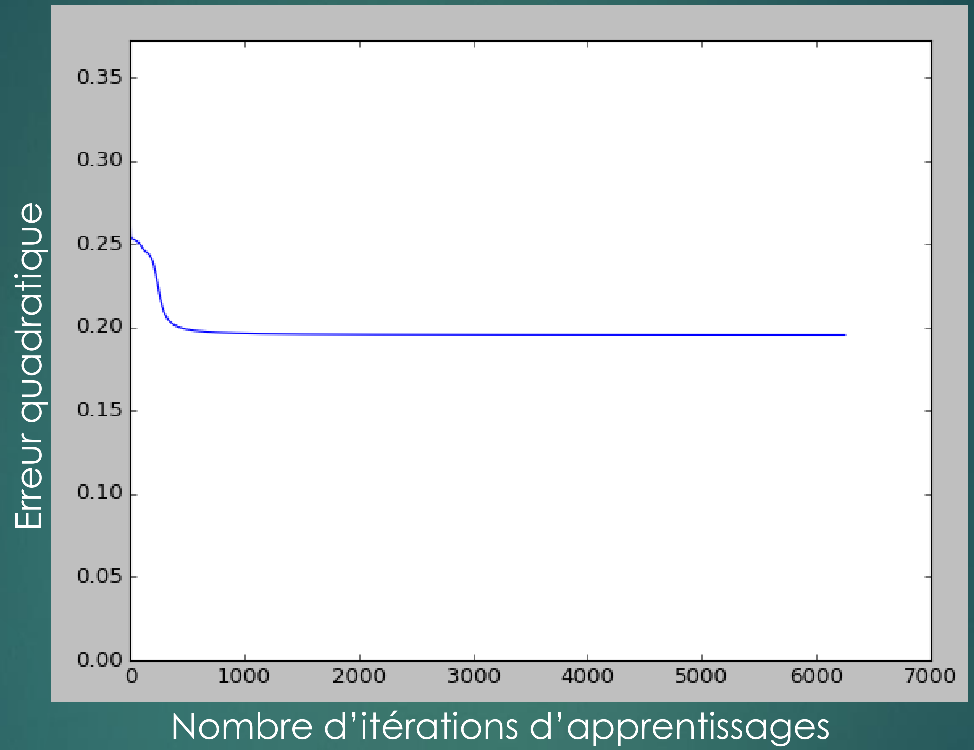
On cherche la dérivée de l'erreur par rapport à $w_{i,j}$: $\frac{\partial E}{\partial w_{i,j}} = \frac{\partial E}{\partial h_j} \frac{\partial h_j}{\partial w_{i,j}} = \delta_j x_i$ où $\delta_j = \frac{\partial E}{\partial h_j} = \frac{\partial E}{\partial x_j} \frac{\partial x_j}{\partial h_j}$

Pour la couche de sortie on a alors : $\delta_j = (x_j - a_j)\, f'(h_j)$

Sinon on a : $\delta_j = \left(\sum_{k \in succ(j)} \frac{\partial E}{\partial h_k} \frac{\partial h_k}{\partial x_j} \right) f'(h_j) = \left(\sum_{k \in succ(j)} w_{i,j} \delta_k \right) f'(h_j)$

En prenant $f = sigm$ on obtient : $f' = sigm' = sigm\,(1 - sigm)$ soit $f'(h_j) = x_j\,(1 - x_j)$

Inertie (moment)



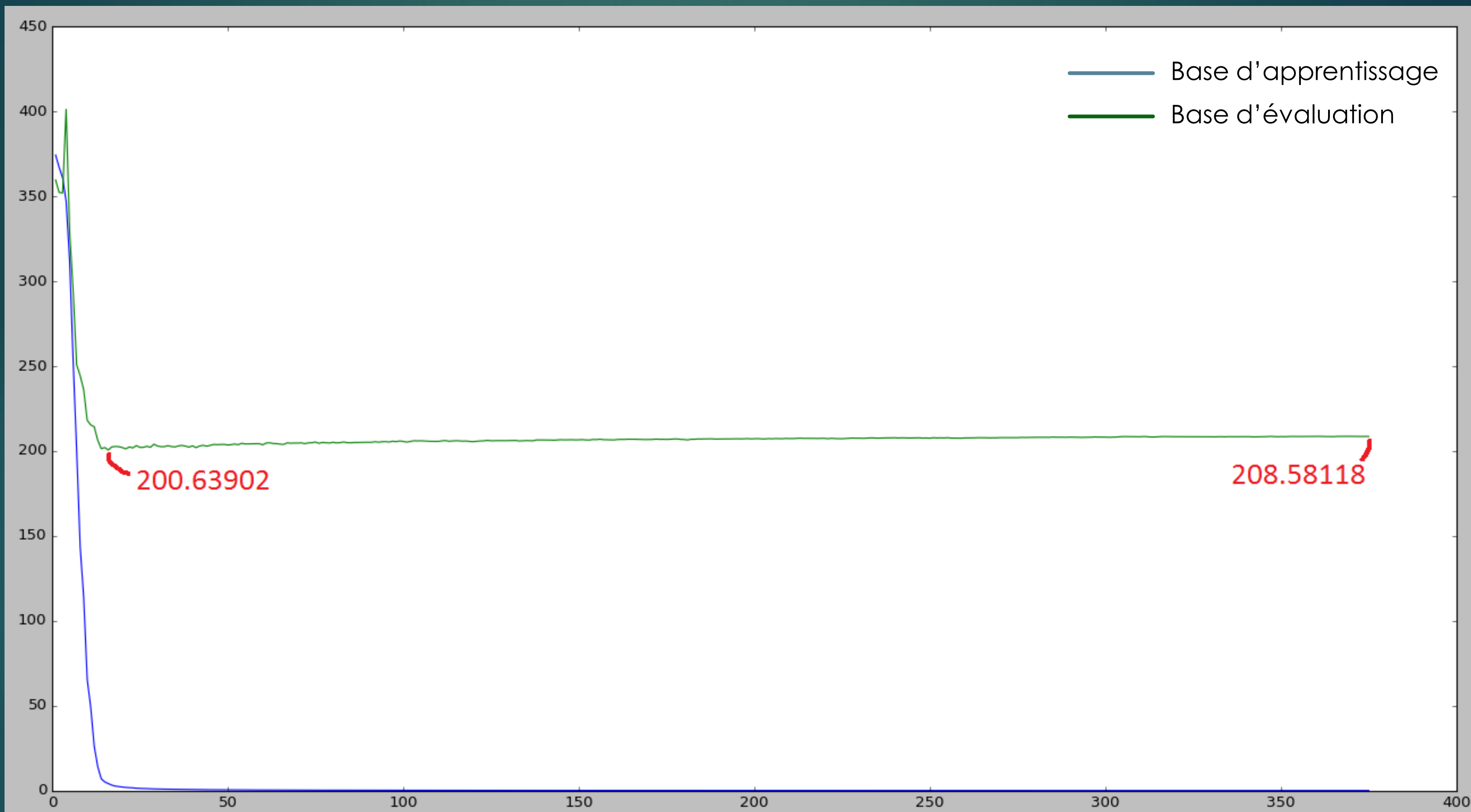
Solution :

$$w_{i,j}(t+1) \leftarrow w_{i,j}(t) + \Delta w_{i,j}(t) + \lambda\,\Delta w_{i,j}(t-1)$$

$$\lambda \in [0,1]$$

Surapprentissage

Erreur quadratique



Nombre d'itérations d'apprentissages
(en milliers)

Génération d'exemples