# Introduction aux Réseaux de Neurones

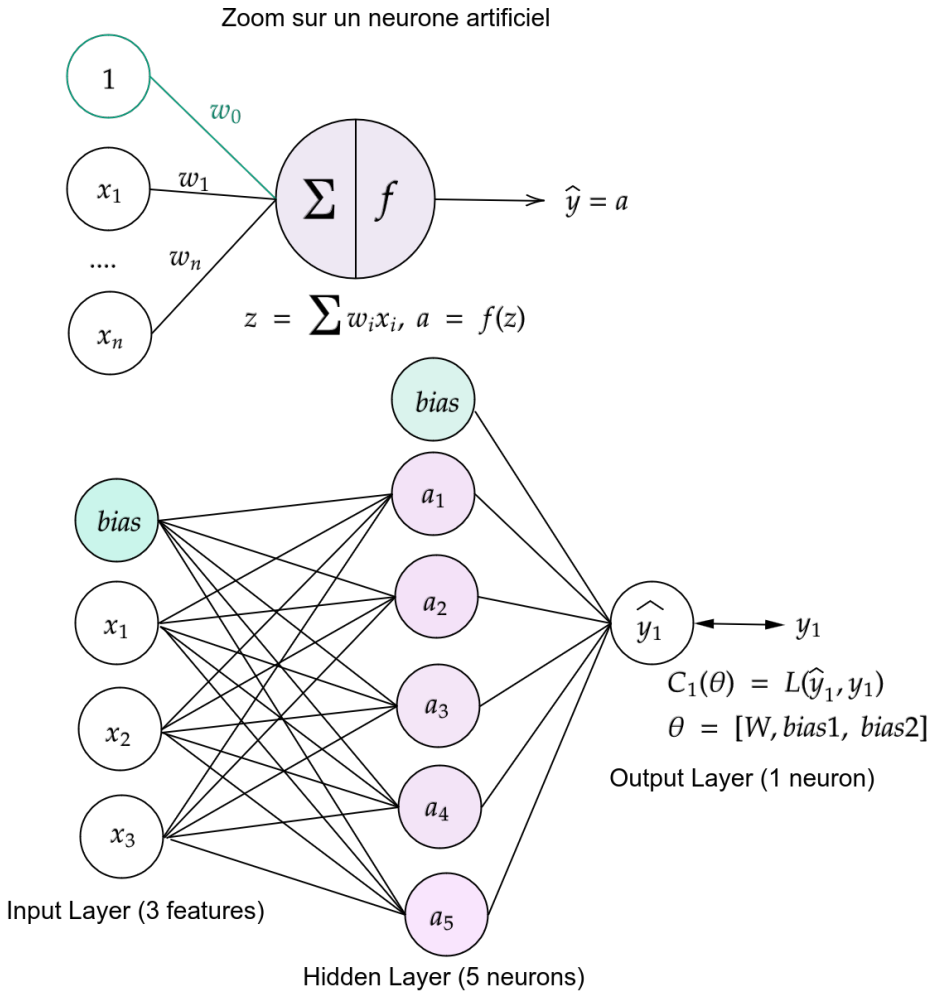
Master 1 GBM

Sara El Bouch

April 3, 2025

## Objectif de ce cours:

- Un neurone, c'est quoi ? Les fonctions d'activation ?
- Fonctions coûts (Classification/Régression) & Algorithme de Descente de Gradient
- Algorithme de Rétropropagation du Gradient.



# Modèle d'imitation mathématique

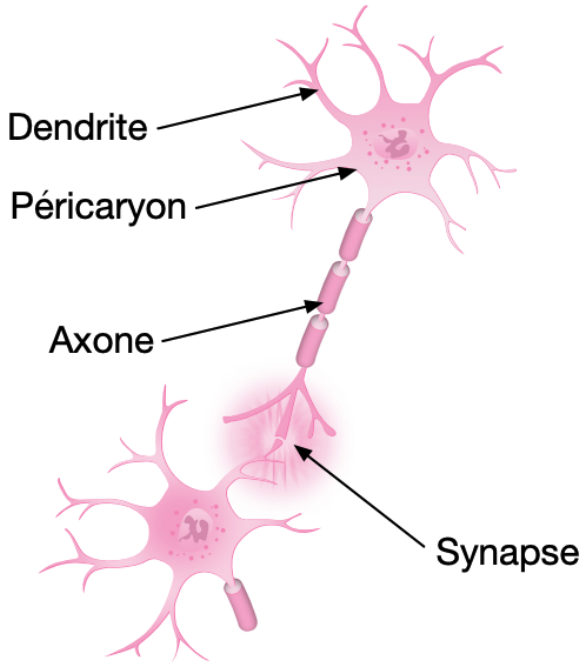



Figure: Neurone

# Perceptron

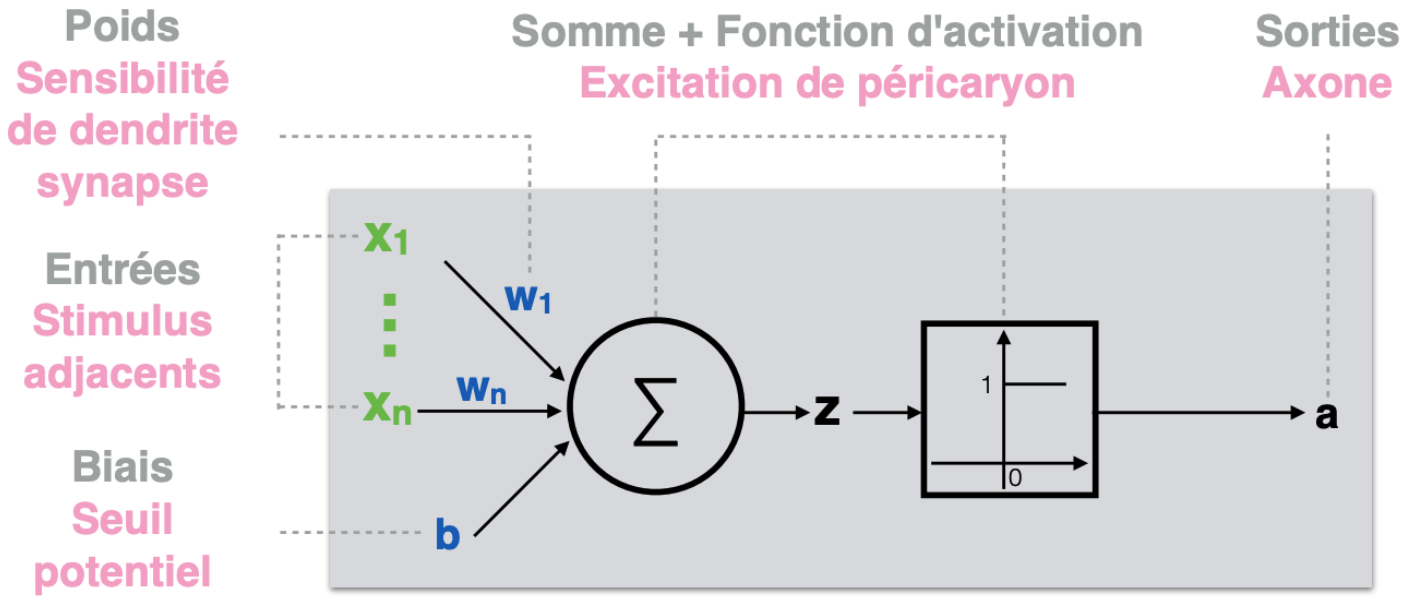



Figure: Le Perceptron

# Non-linéarités et Fonctions d'activations

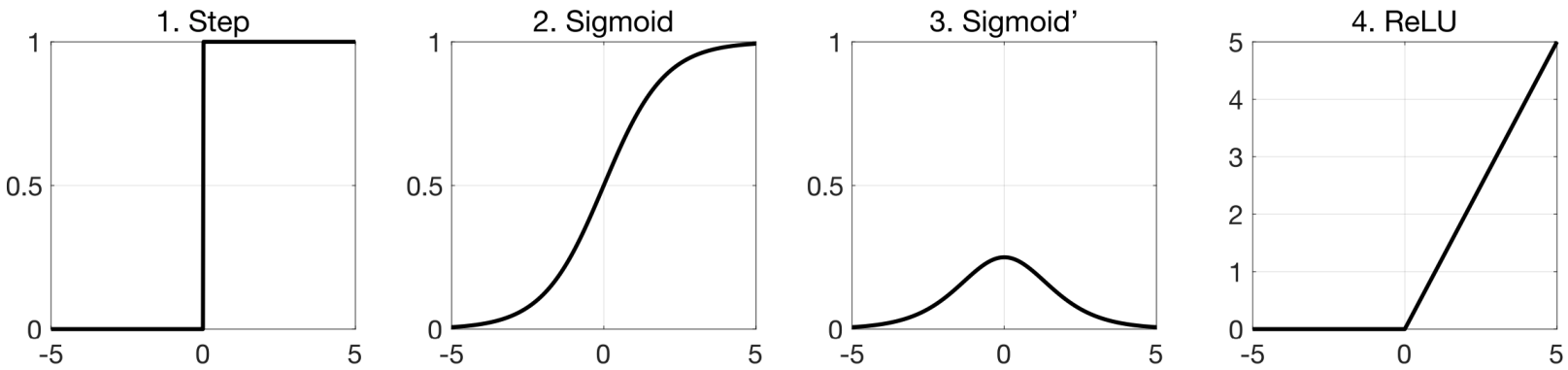



Figure: Fonctions d'activations

# Perceptron multi-couches (MLP)

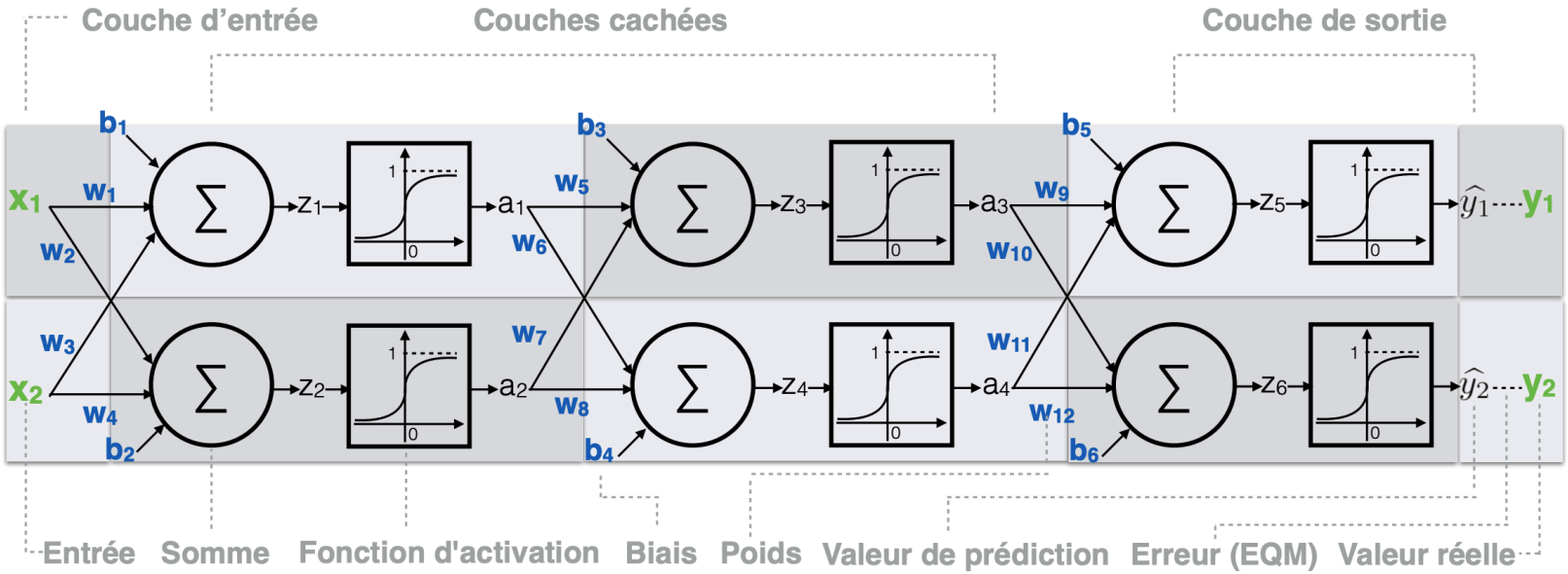



Figure: Zoom sur la structure d'un perceptron multi-couches

# Descente du gradient

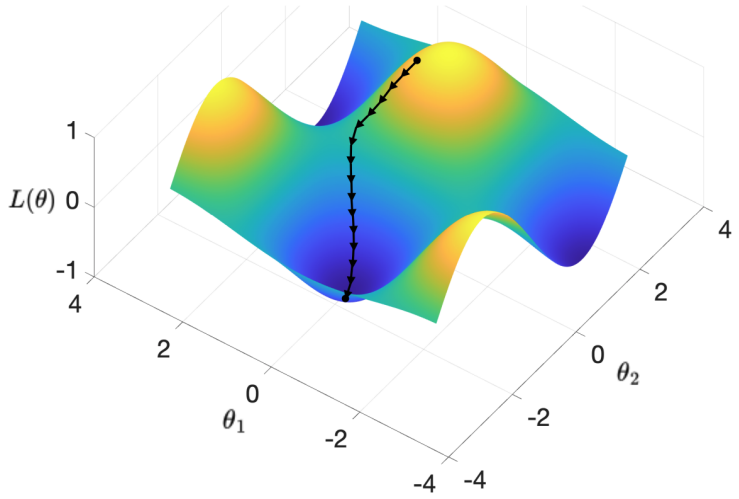
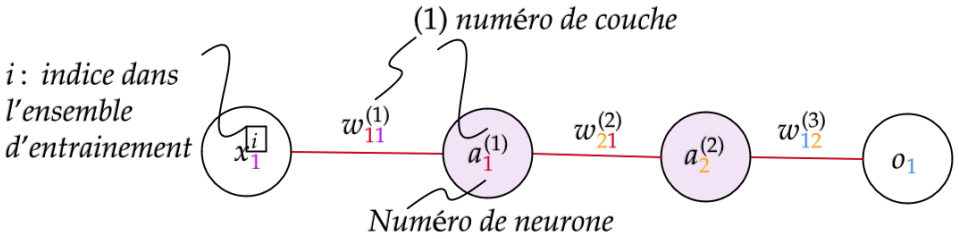


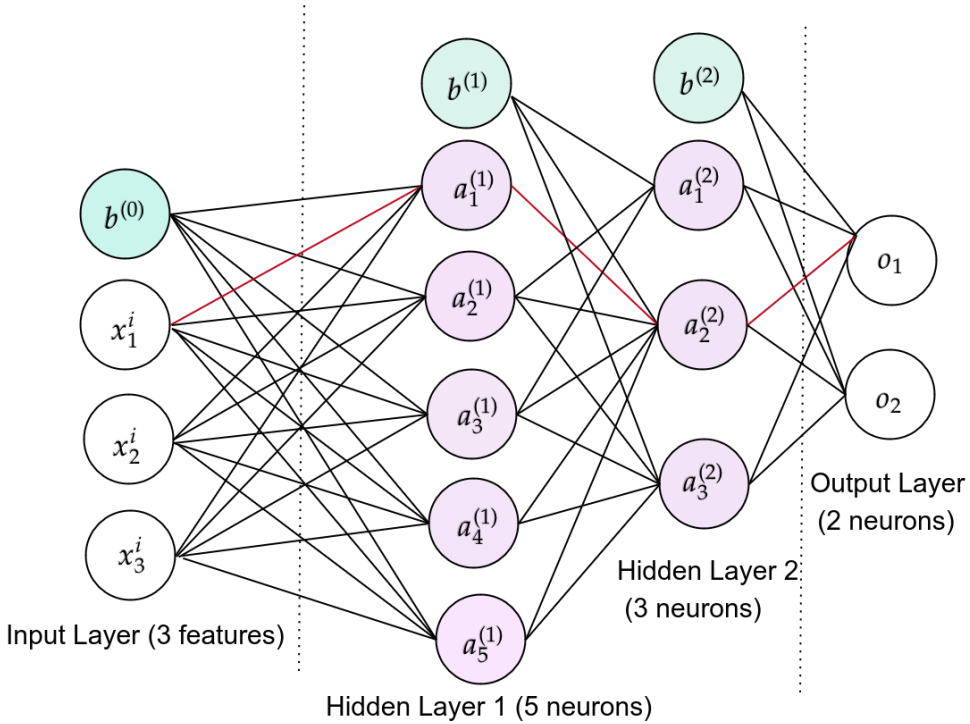

Figure: $L(\theta)$ est la fonction coût, ($\theta_1, \theta_2$ pour simplifier).

$$\theta^{(n+1)} = \theta^{(n)} - \eta \nabla L(\theta) \tag{1}$$

- $\theta^{(n+1)}$ est la position suivante, $\theta^{(n)}$ est la position actuelle.
- $\eta$: taux d'apprentissage.
- $\nabla L(\theta)$ est la direction du gradient de l'augmentation la plus rapide.
- -: la direction opposée de l'augmentation

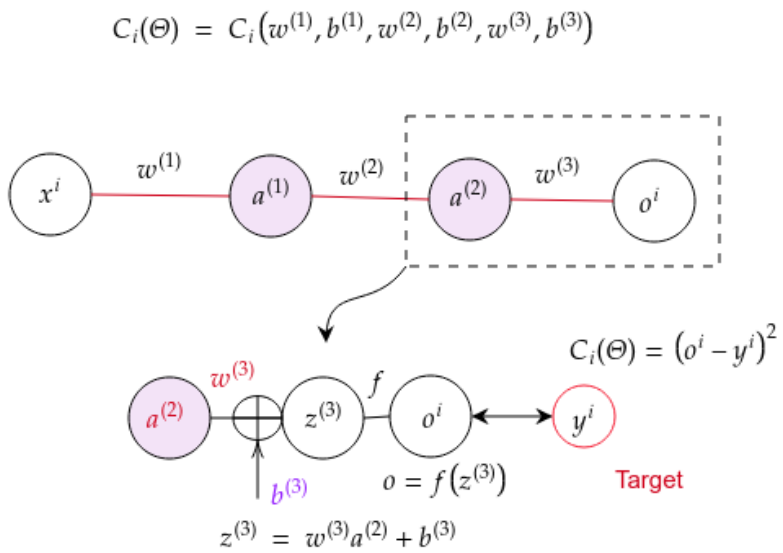
# Perceptron multi-couches (MLP)




- Profondeur $L$, $L = 3$ ( 2 couches cachées + 1 couche de sortie)

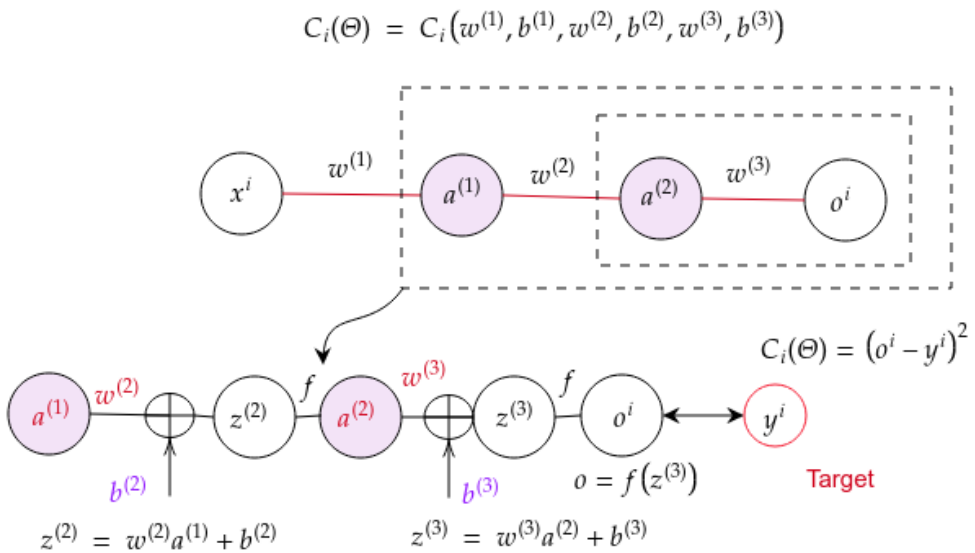
# Rétropropagation du Gradient




$$\frac{\partial C_i}{\partial w^{(3)}} = \frac{\partial C_i}{\partial o} \times \frac{\partial o}{\partial z^{(3)}} \times \frac{\partial z^{(3)}}{\partial w^{(3)}} = 2(o-y) \times f^{'}(z^{(3)}) \times a^{(2)} \quad (2)$$

$$\frac{\partial C_i}{\partial b^{(3)}} = \frac{\partial C_i}{\partial o} \times \frac{\partial o}{\partial z^{(3)}} \times \frac{\partial z^{(3)}}{\partial b^{(3)}} = 2(o-y) \times f^{'}(z^{(3)}) \times 1 \quad (3)$$

$$\frac{\partial C_i}{\partial a^{(2)}} = \frac{\partial C_i}{\partial o} \times \frac{\partial o}{\partial z^{(3)}} \times \frac{\partial z^{(3)}}{\partial a^{(2)}} = 2(o-y) \times f^{'}(z^{(3)}) \times w^{(3)} \quad (4)$$



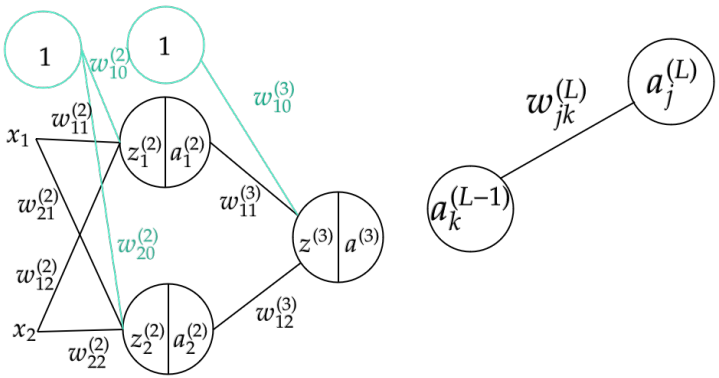

$$\frac{\partial C_i}{\partial w^{(2)}} = \frac{\partial C_i}{\partial a^{(2)}} \frac{\partial a^{(2)}}{\partial w^{(2)}} = \frac{\partial C_i}{\partial a^{(2)}} \frac{\partial a^{(2)}}{\partial z^{(2)}} \frac{\partial z^{(2)}}{\partial w^{(2)}} \tag{5}$$

$$\frac{\partial C_i}{\partial b^{(2)}} = \frac{\partial C_i}{\partial a^{(2)}} \frac{\partial a^{(2)}}{\partial b^{(2)}} = \frac{\partial C_i}{\partial a^{(2)}} \frac{\partial a^{(2)}}{\partial z^{(2)}} \frac{\partial z^{(2)}}{\partial b^{(2)}} \tag{6}$$

$$\frac{\partial C_i}{\partial a^{(1)}} = \frac{\partial C_i}{\partial a^{(2)}} \frac{\partial a^{(2)}}{\partial a^{(1)}} = \frac{\partial C_i}{\partial a^{(2)}} \frac{\partial a^{(2)}}{\partial z^{(2)}} \frac{\partial z^{(2)}}{\partial a^{(1)}} \tag{7}$$

$\frac{\partial a^{(2)}}{\partial z^{(2)}} = f'(z^{(2)})$.

# Exemple




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

1. Initialisation aléatoire des paramètres du réseaux $\Theta$.
2. Calcul de la fonction coût $C(\Theta) = \frac{1}{2}(a^{(3)} - y)^2$.
3. On pose $\delta_i^{(L)} = \frac{\partial C}{\partial z_i^{(L)}}$.
   - $\delta_1^{(3)} = a_1^{(3)} \times (1 - a_1^{(3)}) \times (a_1^{(3)} - y)$
   - $\delta_1^{(2)} = a_1^{(2)} \times (1 - a_1^{(2)}) \times \delta_1^{(3)} \times w_{11}^{(3)}$
   - $\delta_2^{(2)} = a_2^{(2)} \times (1 - a_2^{(2)}) \times \delta_{(3)} \times w_{12}^{(3)}$

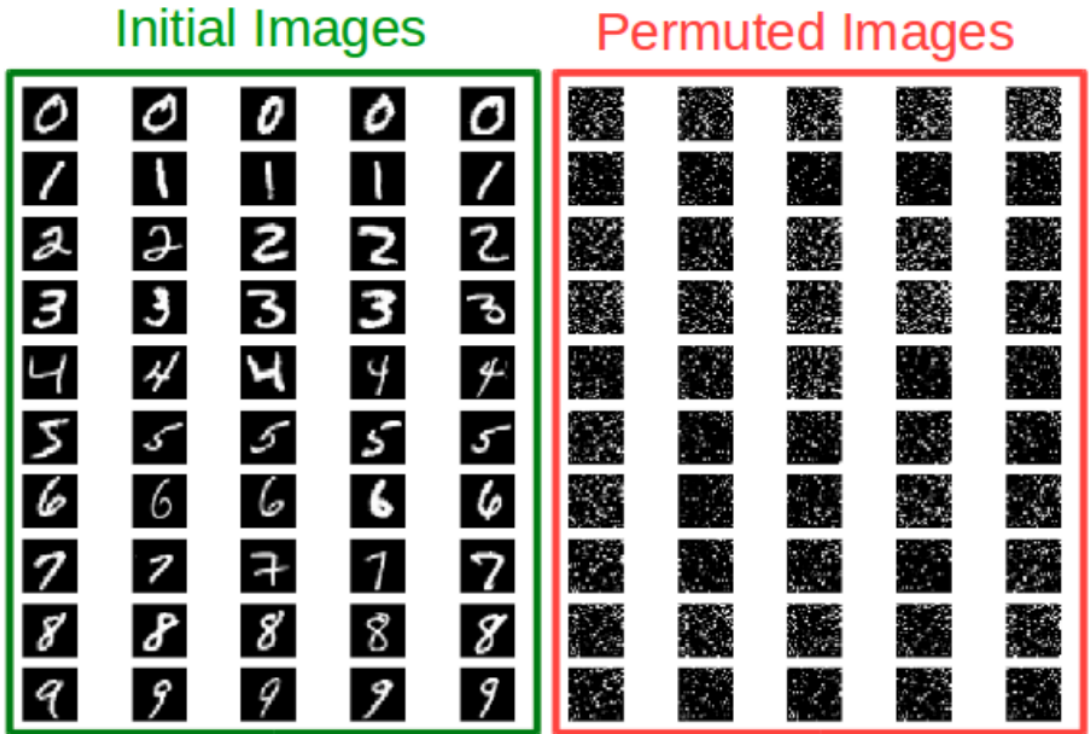
$$\delta_1^{(2)} = \frac{\partial C}{\partial z_1^{(2)}} = \frac{\partial C}{\partial z_1^{(3)}} \times \frac{\partial z_1^{(3)}}{\partial z_1^{(2)}}$$
$$= \delta_1^{(3)} \frac{\partial w_{11}^{(3)} \times a_1^{(2)} + w_{12}^{(3)} a_2^{(2)}}{\partial z_1^{(2)}}$$

4. Mettre à jour les poids du modèle

$$w_{11}^{(3)} = w_{11}^{(3)} - \eta\delta_1^{(3)} a_1^{(2)}, \quad w_{11}^{(2)} = w_{11}^{(2)} - \eta\delta_1^{(2)} x_1, \quad w_{21}^{(2)} = w_{11}^{(2)} - \eta\delta_2^{(2)} x_1$$
$$w_{12}^{(3)} = w_{12}^{(3)} - \eta\delta_1^{(3)} a_2^{(2)}, \quad w_{12}^{(2)} = w_{12}^{(2)} - \eta\delta_1^{(2)} x_2, \quad w_{22}^{(2)} = w_{12}^{(2)} - \eta\delta_2^{(1)} x_2$$

# Perceptron multicouches

- Toutes les entrées sont connectées à tous les neurones cachés.
- Représentation vectorielle de l'entrée: l'ordre des dimension est arbitraire.
- Même performances avec les images de MNIST ou leur versions mélangées !

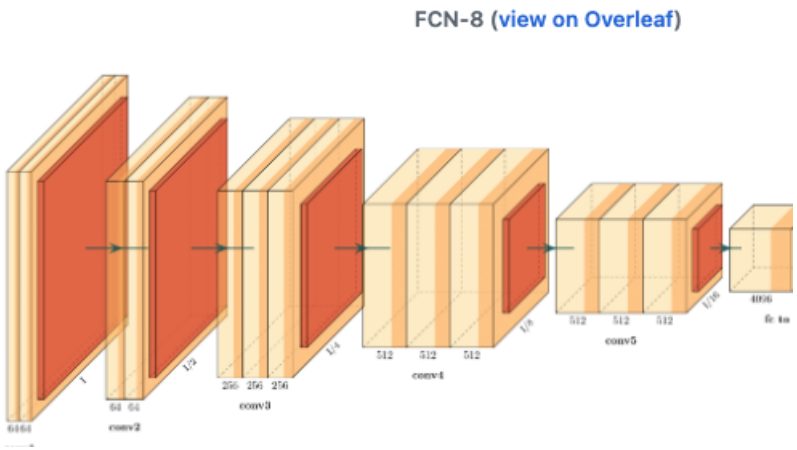


# A priori sur les données

- Invariance et stabilité
  - Petites déformations → représentations similaires
  - Fortes déformations → représentations peu similaires
- Invariance à la translation: déplacer un objet dans l'image ne change pas sa nature donc ne devrait pas changer ses activations.

**Objectif de ce cours:**

- Faible nombre de paramètres optimisables
- Poids plus important donnée à la structure **locale** de l'entrée
- intégrer une **invariance** à de petites déformations locales



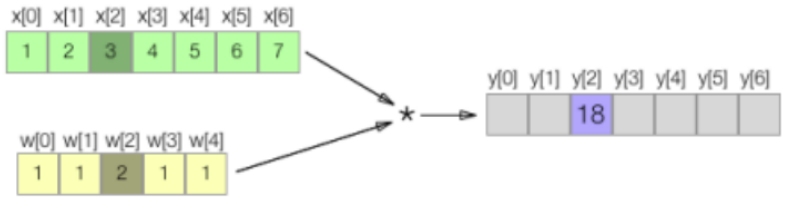
Soit $f : \mathbb{R} \to \mathbb{R}$ un signal, et $h$ une fonction réelle (un filtre). La convolution entre $f$ et $h$ est définie comme étant:

$$f * h(x) = \int h(t) f(x - t) dt \tag{8}$$

# Convolution discrète en 1D

Soit $f$ signal discret $\{f(i)\}_{i=1}^{N}$ et $\{h(i)\}_{i=1}^{d}$ la réponse impulsionnelle finie de dimension $d$ (impair pour simplifier). La convolution discrète est définie par:

$$(f * h)(i) = \sum_{i=-\frac{d-1}{2}}^{\frac{d-1}{2}} h(n) f(i-n) \tag{9}$$




**Remarque** En traitement du signal, la convolution et la corrélation ne sont pas identiques. La convolution est une corrélation avec le filtre pivoté à 180 degrés. Dans le contexte des réseaux de neurones, où les coefficients des filtres sont appris, cette distinction peut être ignorée. Comme c'est courant dans la littérature d'apprentissage automatique, dans ce document, convolution et corrélation sont considérées comme équivalentes.

# Convolution 2D

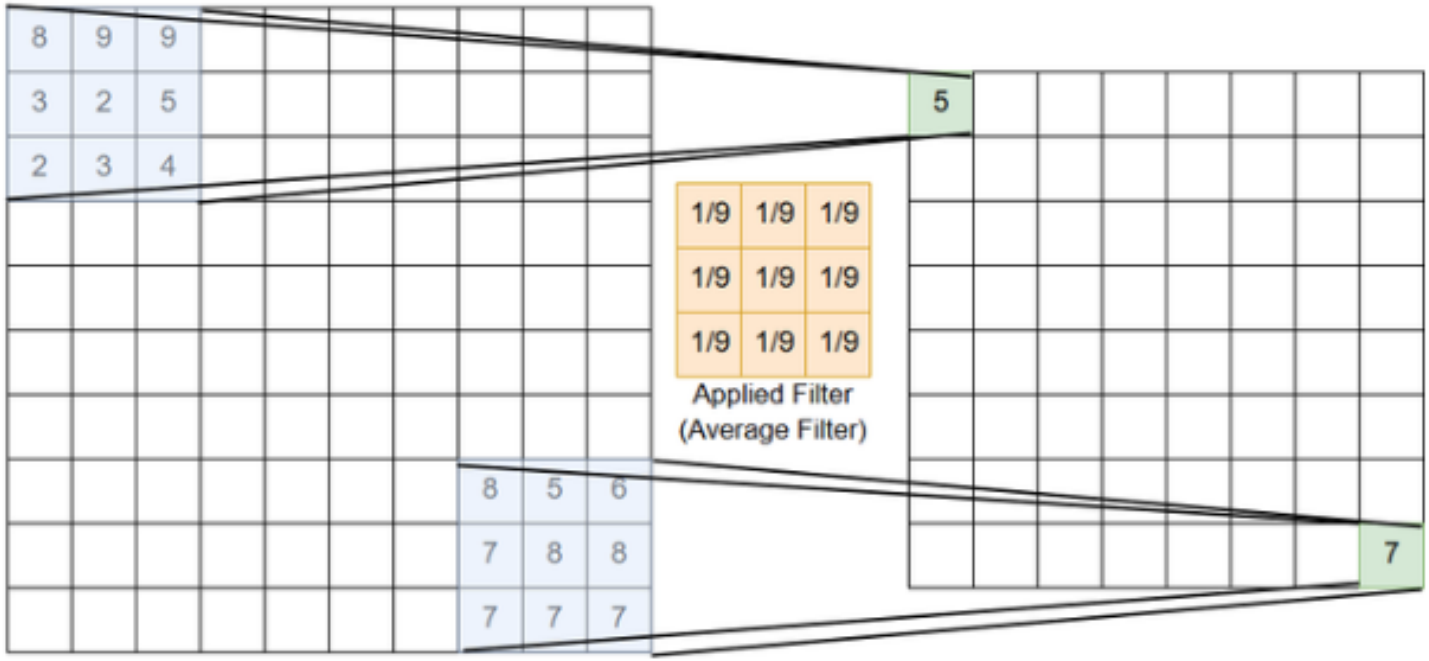



Figure: Filtrage d'une image de taille 10 par une filtre de taille $3 \times 3$, la sortie a une taille $8 \times 8$ plus petite que l'entrée

En général pour une image de taille $n \times m$ convoluée avec un filtre de taille $a \times b$, la sortie a une taille $(n - a + 1 \times m - b + 1)$.

# Padding

Le remplissage (Padding) est un paramètre important du filtrage convolutif. Un remplissage de p signifie qu'aux bords gauche, droit, supérieur et inférieur de l'entrée, p colonnes (rangées) sont ajoutées à l'entrée. Dans l'image on applique un padding $p = 2$ avec des zéros (au milieu) et avec des values aléatoires (à droite).

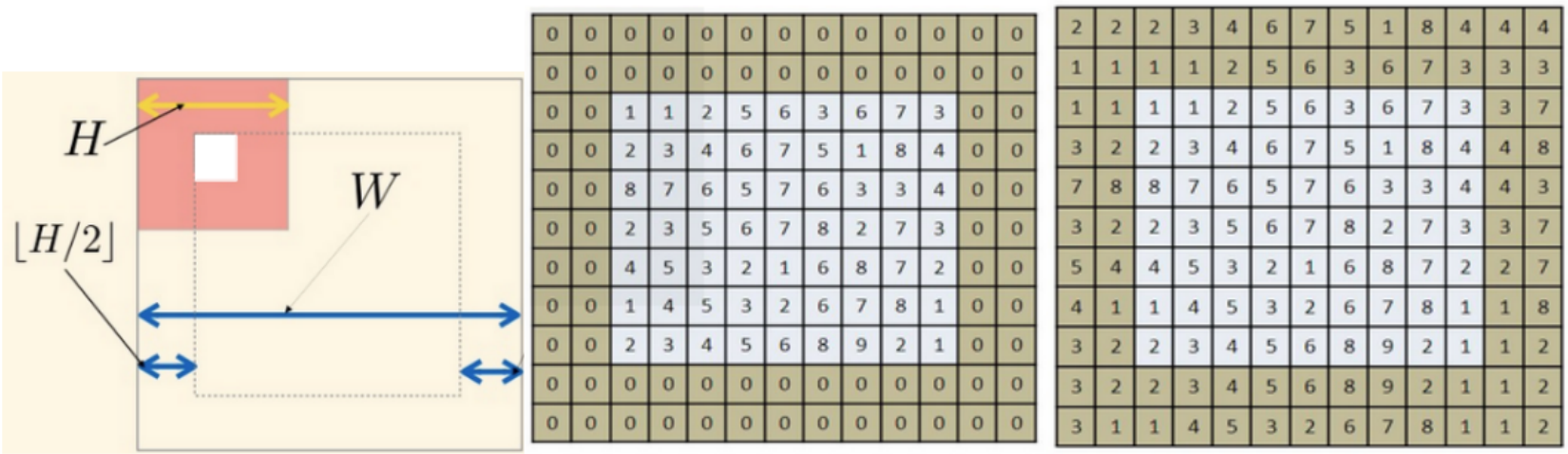



Figure: Remarque important, si on veut préserver la taille de l'image

# Convolution à pas $s$ (strided convolution)

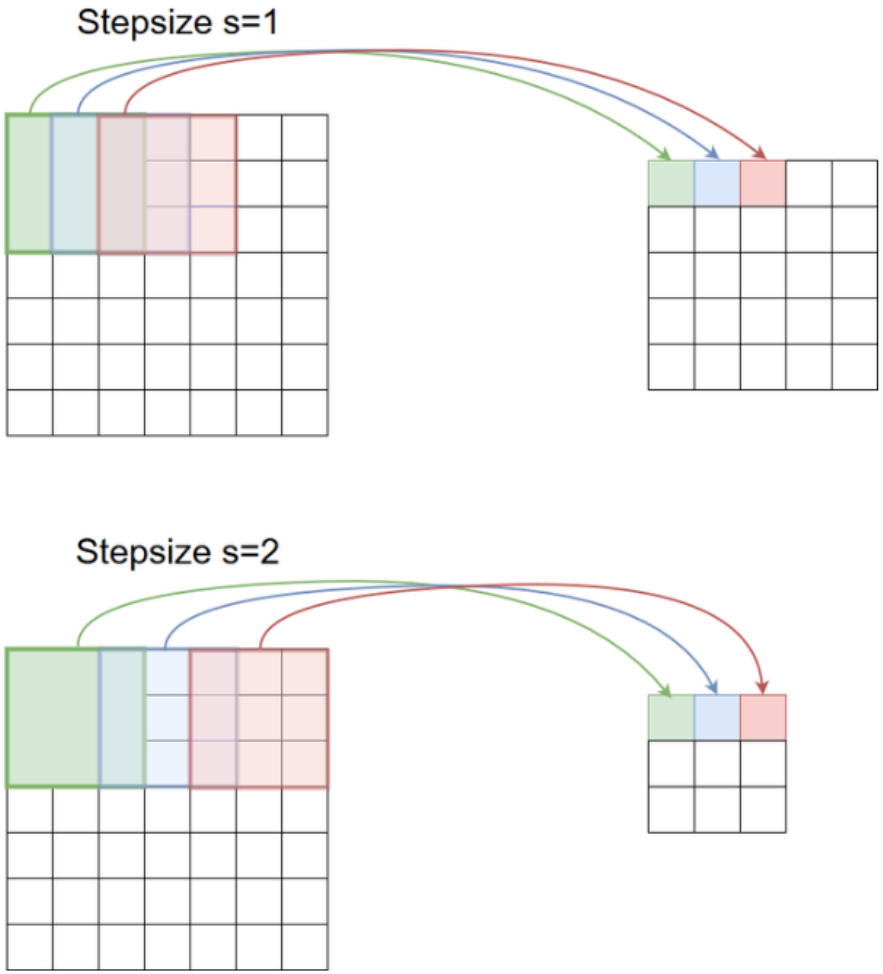
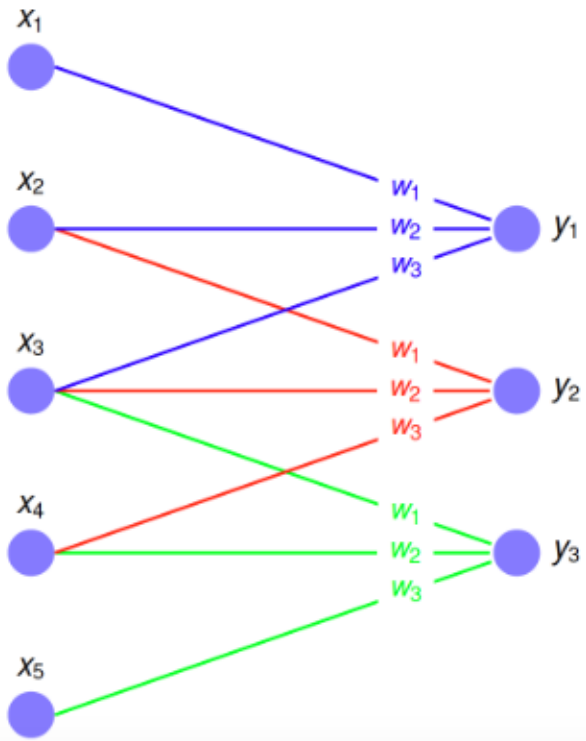



Figure: Le filtre n'a pas besoin d'être convolué par pas de 1 à travers l'entrée. Des pas plus grands produisent une sortie proportionnellement plus petite. Dans l'image ci-dessous, le filtrage avec un pas de 2 est montré sous le filtrage de la même entrée avec un pas de 1.

# Réseau convolutionnel



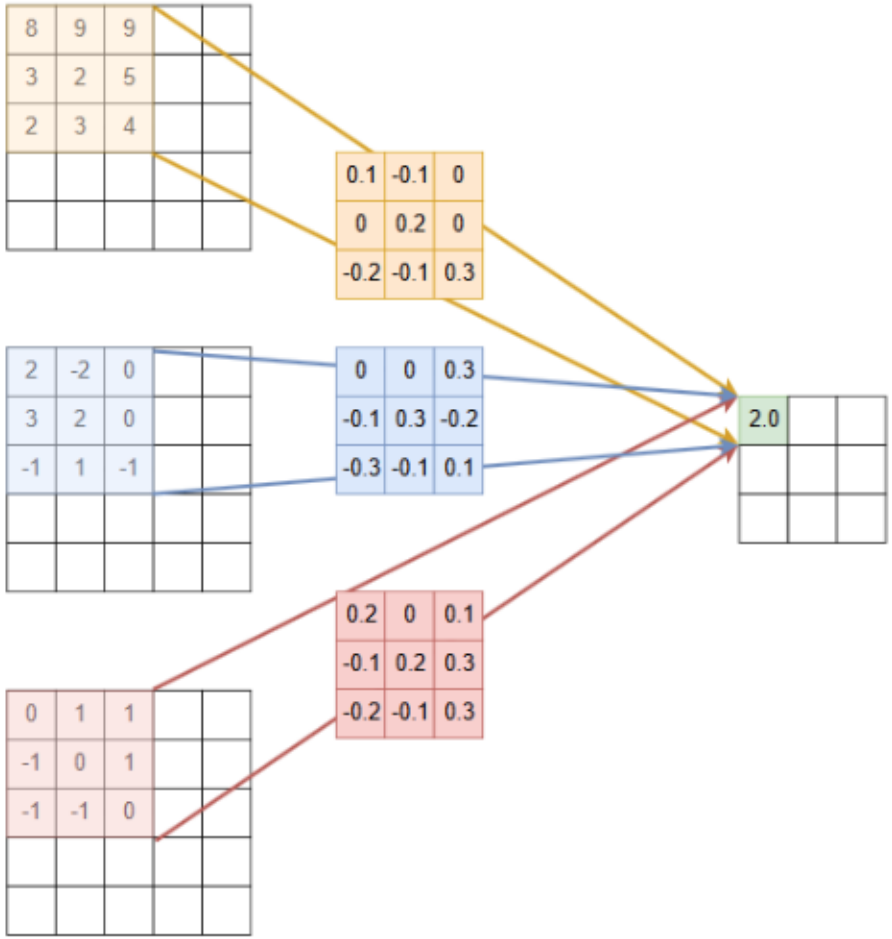

$$y_i = g\left(\sum_{j=1}^{3} w_j \cdot x_{i+j-1}\right)$$

with shared weight-vector

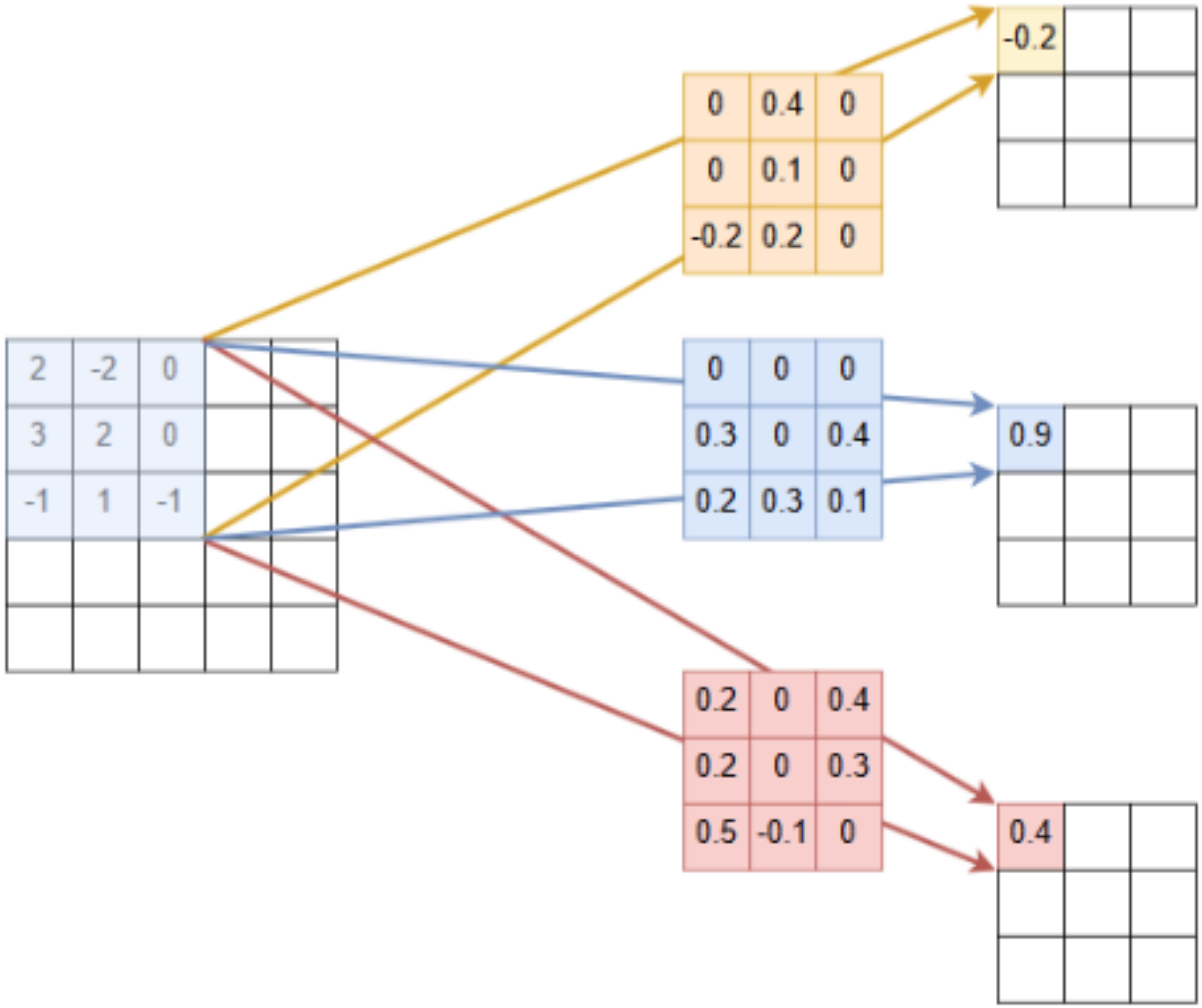
$$\mathbf{w} = [w_1, w_2, w_3]$$

Les poids (bleus, rouge, vert) sont les mêmes ! (contrairement aux réseaux denses MLP). Comme pour les perceptrons, la non-linéarité $g$ est indispensable.

# Plusieurs canaux en entrée



# Plusieurs cartes de caractéristiques (feature maps)

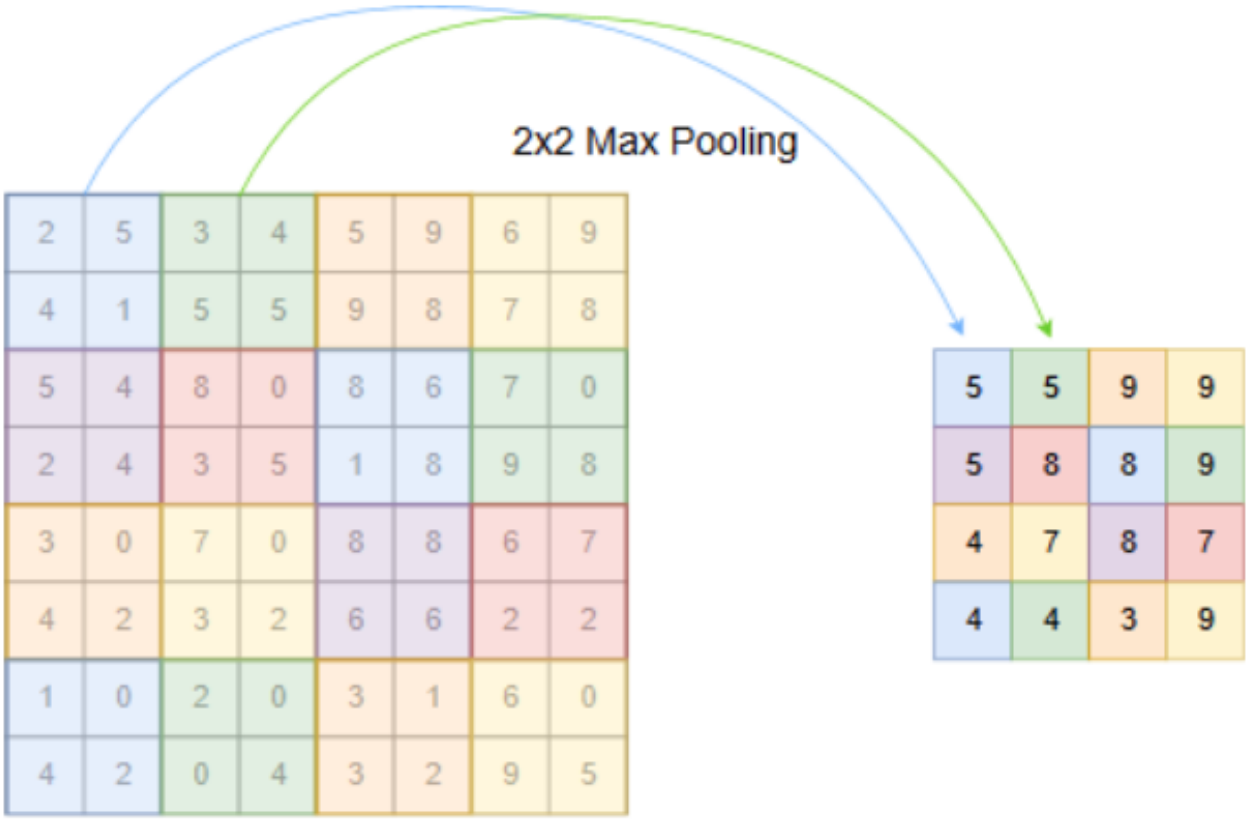


# Echantillonnage (Pooling)

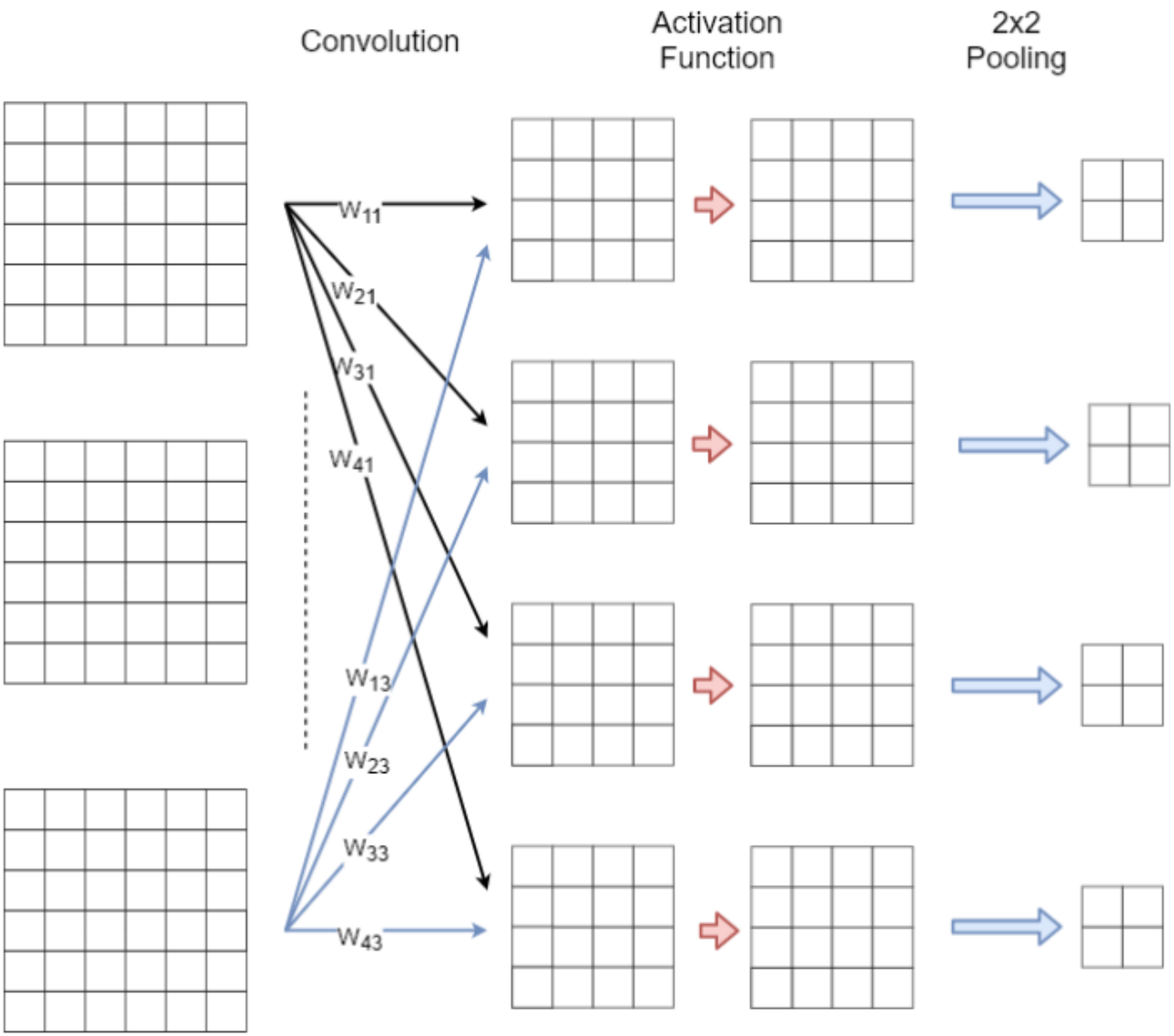
Les couches convolutives extraient des caractéristiques spatiales de leur entrée. Les filtres (ensembles de poids partagés) définissent quelles caractéristiques sont extraites. Pendant la phase d'entraînement, les filtres sont appris de sorte qu'après l'apprentissage, ils représentent des motifs qui apparaissent fréquemment dans les données d'entraînement.
Les avantages du pooling sont:

- Description sommaire des statistiques d'un échantillon $\rightarrow$ réduction de dimension
- elles fournissent un certain degré d'invariance au décalage, dans le sens où si dans deux entrées une certaine caractéristique n'apparaît pas exactement à la même position, mais dans des positions proches, elles produisent la même sortie après pooling.

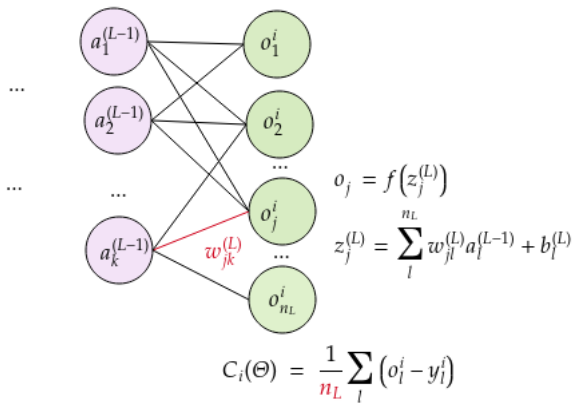
# Exemple: Max Pooling



# Succession de convolution et pooling



# Pour aller plus loin




$$C = \frac{1}{N} \sum_{i=1}^{N} C_i, \nabla \mathbf{C} = \begin{bmatrix} \vdots \\ \frac{\partial C}{\partial w_{jk}^{(L)}} \\ \frac{\partial C}{\partial b_j^{(L)}} \\ \vdots \\ \frac{\partial C}{\partial w_{jk}^{(L-1)}} \\ \vdots \end{bmatrix}$$

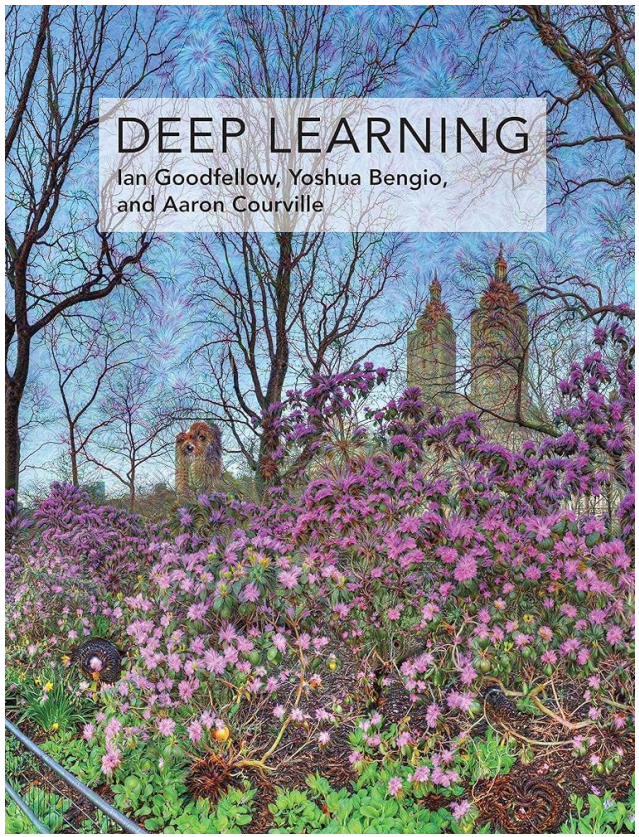
$$\frac{\partial C_i}{\partial w_{jk}^{(L)}} = \frac{\partial C_i}{\partial o_j^i} \times \frac{\partial {o_j}^i}{\partial z_j^{(L)}} \times \frac{\partial z_j^{(L)}}{\partial w_{jk}^{(L)}}$$

$$\frac{\partial C_i}{\partial a_k^{(L-1)}} = \sum_{j=1}^{n_L} \frac{\partial C_i}{\partial o_j^i} \times \frac{\partial o_j^i}{\partial z_j^{(L)}} \times \frac{\partial z_j^{(L)}}{\partial a_k^{(L-1)}}$$

- Problème numérique du gradient (évanouissement, explosion).
- L'algorithme du gradient stochastique, et ses variantes.
- Régularisation: EarlyStopping, pruning, Weight decay, Data Augmentation.

# References

Deep learning:
`https://www.deeplearningbook.org/`




Online courses
`https://www.coursera.org/learn/neural-networks-deep-learning-fr`
(Réseaux de neurones et Deep Learning Andrew Ng).
Youtube Videos: 3Blue1Brown