Apprentissage statistique : modélisation décisionnelle et apprentissage profond (RCP209)

Réseaux convolutifs

Nicolas Audebert
nicolas.audebert@lecnam.net
http://cedric.cnam.fr/vertigo/Cours/ml2/

Département Informatique
Conservatoire National des Arts & Métiers, Paris, France

20 avril 2023

Plan du cours

Nombre de paramètres dans les réseaux entièrement connectés

Perceptron multi-couche à une couche cachée :

$$\mathbf{h} = \sigma(\mathbf{W}\mathbf{x}+)$$

$$\mathbf{y} = \mathbf{W}^{y}\mathbf{h} + y)$$

- Entrées : observations $\mathbf{x} \in \mathbb{R}^{d}$
- Couche cachée : n neurones
- Sortie : dimensionalité d' (généralement, $d' << d$)

Nombre de poids

Pour la couche cachée : $n_{\text{paramètres}} = d \cdot n + n$

- Matrice de poids $\mathbf{W}(d \times n)$
- Vecteur de biais $(1, \times n)$

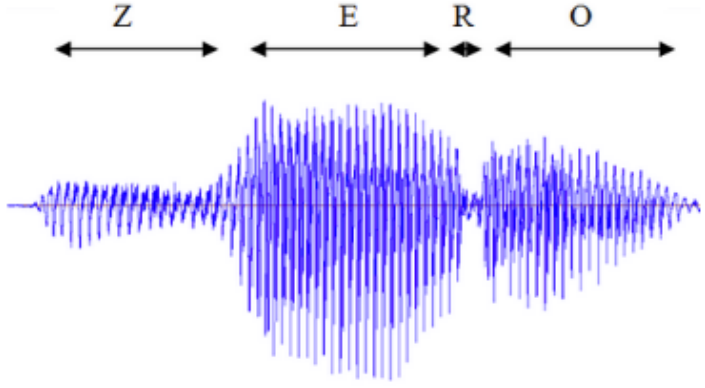
Pour la couche de sortie : $n_{\text{paramètres}} = n \cdot d' + d'$

Application numérique : image $(28, 28)$, 10 classes, 500 neurones :

$\Longrightarrow$ $28 \times 28 \times 500 + 500 = 392500$ paramètres ($\approx$ 1,5Mo)

$\Longrightarrow$ $500 \times 10 + 10 = 5010$ paramètres

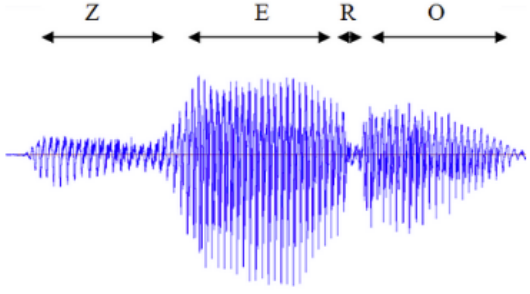
Passage à l'échelle

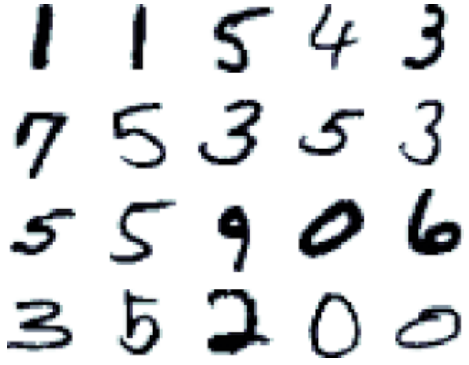


Perceptron multi-couche

- Signal audio : échantillonnage 16 kHz $\Longrightarrow$ 16 000 points par seconde !
  - 500 neurones, 3 secondes d'audio $\Longrightarrow$ 24 000 000 paramètres ($\approx$ 91Mo)
- Image de webcam 480p : $852 \times 480 \times 3$ scalaires par image !
  - 500 neurones $\Longrightarrow$ 614 880 000 paramètres ($\approx$ 2,3Go)

Structure et a priori sur les signaux





- Les composantes d'un signal ne sont pas indépendantes !
  - Signaux 1D (son, séries temporelles) : structure locale temporelle
  - Signaux 2D (images, radar, etc.) : structure locale spatiale

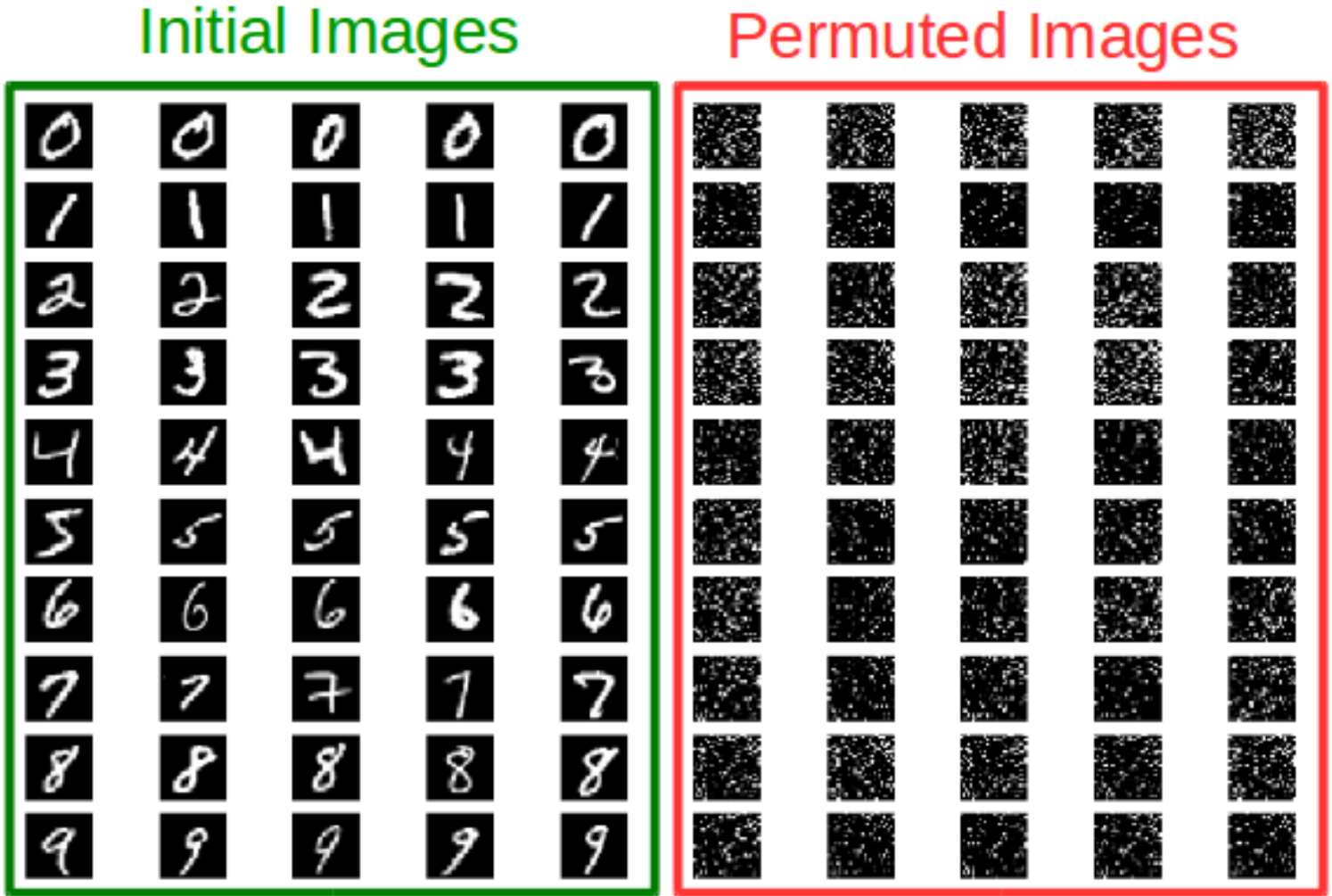
Structure des données

- Perceptron multi-couche : réseau entièrement connecté

$\Longrightarrow$ toutes les entrées sont connectées à tous les neurones cachés

$\Longrightarrow$ représentation vectorielle de l'entrée : l'ordre des dimensions est arbitraire !

$\Longrightarrow$ Mêmes performances avec les images de MNIST ou leurs versions mélangées !



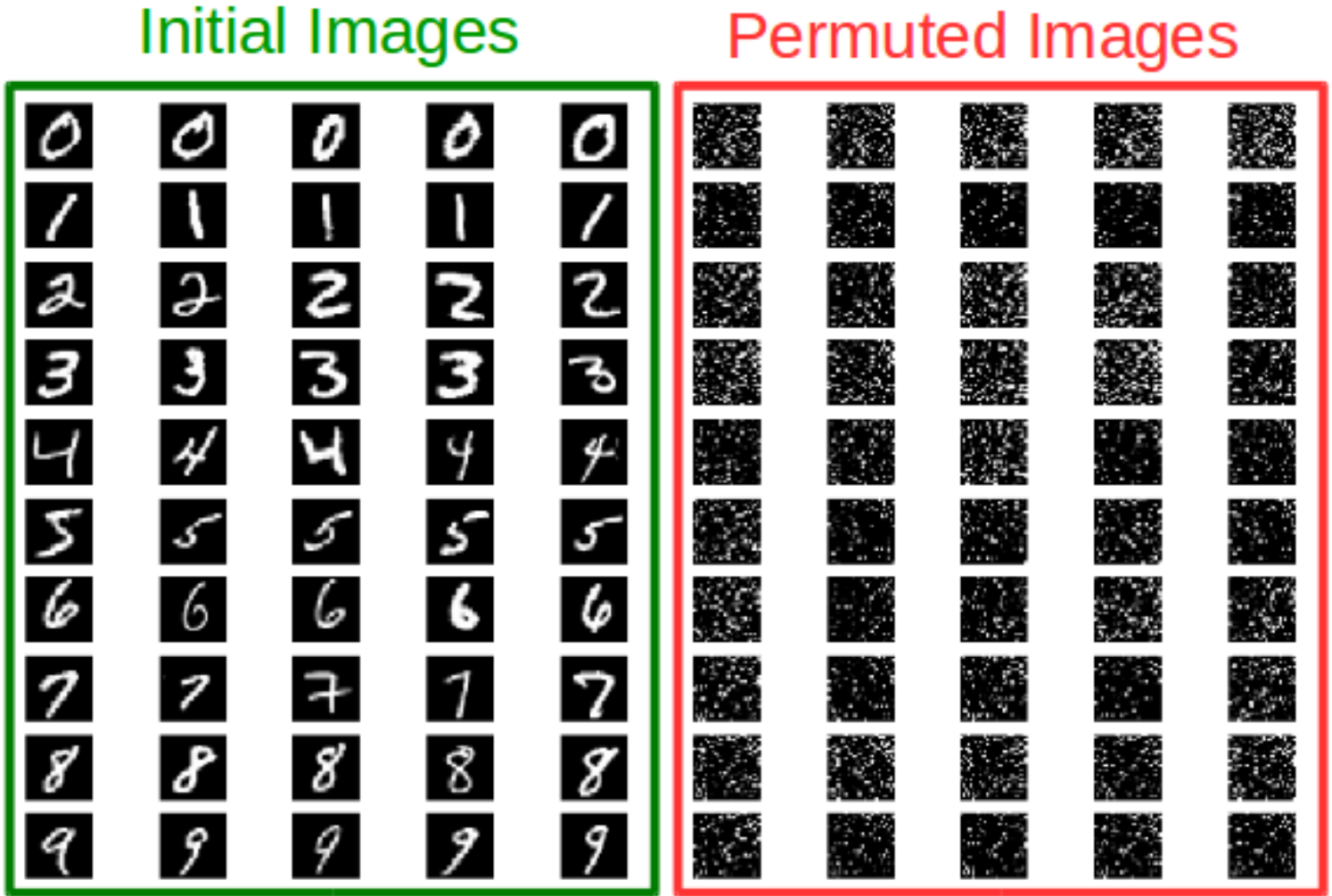
Structure des données

- Perceptron multi-couche : réseau entièrement connecté

$\Longrightarrow$ toutes les entrées sont connectées à tous les neurones cachés

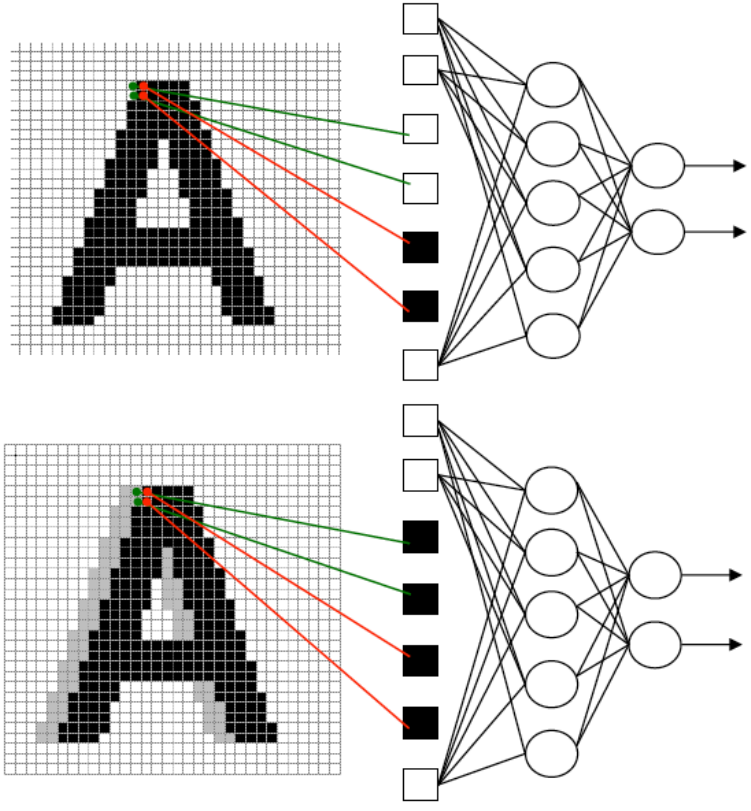
$\Longrightarrow$ représentation vectorielle de l'entrée : l'ordre des dimensions est arbitraire !

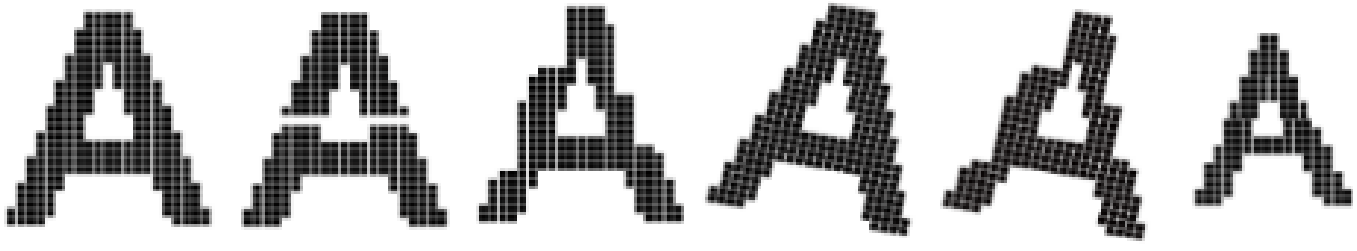
$\Longrightarrow$ Mêmes performances avec les images de MNIST ou leurs versions mélangées !



A priori sur les données

- **Invariance et stabilité**
  - Petites déformations → représentations similaires
  - Fortes déformations → représentations peu similaires
- Équivariance/invariance à la translation : déplacer un objet dans l'image ne change pas sa nature, donc ne devrait pas changer ses activations.
- Vrai également pour des petites transformations (échelle, rotation, *warping*...)

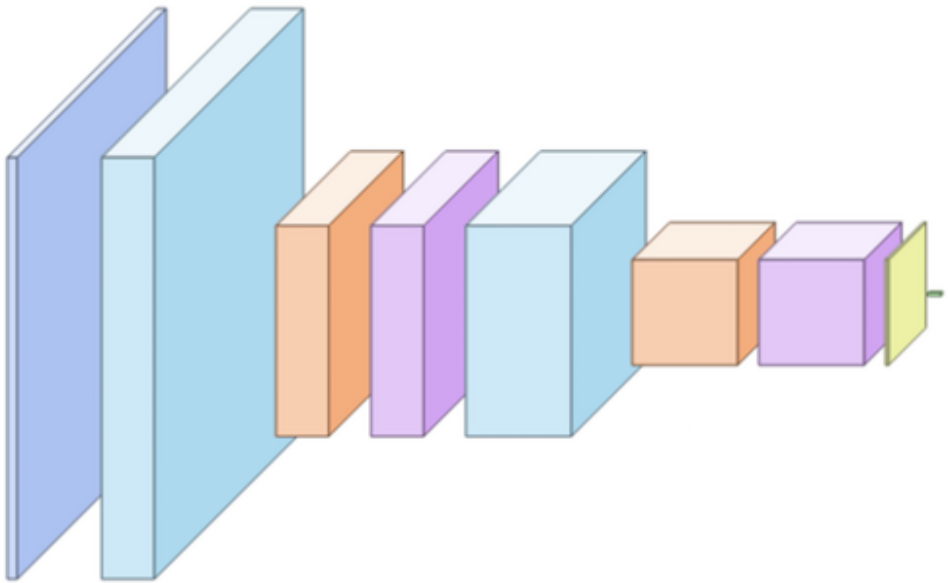




Réseaux de neurones convolutifs

Objectif : dépasser les limitations précédemment décrites.

- Faible nombre de paramètres optimisables
- Poids plus important donné à la structure locale du signal
- Intègre une invariance à des petites déformations locales
- Tous les paramètres peuvent être appris par rétropropagation ($\Longrightarrow$ couches différentiables)



Plan du cours

Définition générale

Soit $f : \mathbb{R} \to \mathbb{R}$ une fonction réelle (signal) et $h : \mathbb{R} \to \mathbb{R}$ une fonction réelle (filtre).

Opérateur de convolution

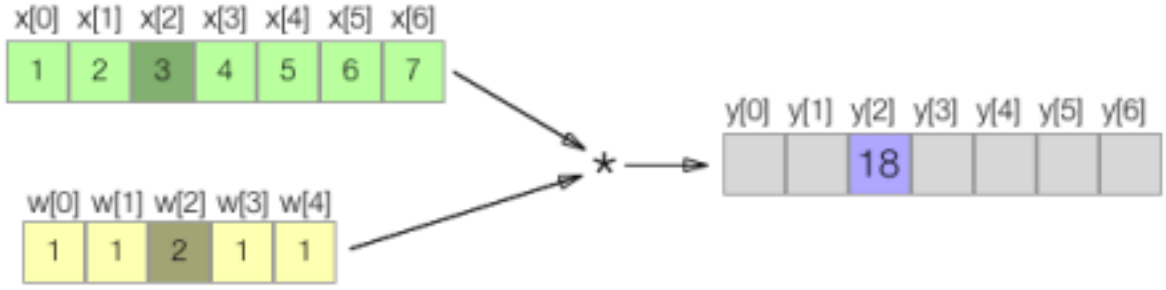
La convolution entre f et h est définie comme étant la fonction notée $(f * h)$ telle que :

$$(f * h)(x) = \int_{-\infty}^{+\infty} f(x - t)h(t)dt$$

Propriétés :

- Commutativité : $(f * h) = (h * f)$
- Bilinéarité : $f * (g + \lambda h) = (f * g) + \lambda(f * h)$
- Associativité : $f * (g * h) = (f * g) * h$

Convolution discrète unidimensionnelle



- Convolution discrète d'un signal f par un filtre à réponse impulsionnelle finie $\mathbf{h}$:
- Signal $f[i]$, $i \in \{1; N\}$
- Filtre $\mathbf{h}[i]$, $i \in \{1; d\}$ (d impair pour simplifier)
- Signal filtré (convolué) : $f'[i]$, $i \in \{1; N\}$

Convolution discrète 1D

Opérateur $T_h : f \rightarrow f' = T(f) = f * h$

$$T_h(f)[i] = (f * h)[i] = \sum_{n=-\frac{d-1}{2}}^{\frac{d-1}{2}} f[i-n] \cdot h[n]$$

Convolution en deux dimensions

- Convolution 2D discrète par un filtre fini $\mathbf{h}$: $T : f \rightarrow f' = T[f] = f * h$
- Signal $f[i,j]$, $i \in \{1; N\}^2$
- Filtre $\mathbf{h}[i]$, $i \in \{1; p\} \times \{1; q\}$ (p, q impairs pour simplifier)

$$f'(i,j) = (f \star h)(i,j) = \sum_{n=-\frac{p-1}{2}}^{\frac{p+1}{2}} \sum_{m=-\frac{q-1}{2}}^{\frac{q+1}{2}} f[i-n, m-j] \cdot h[n,m]$$

Example avec un filtre 3×3

$$\begin{aligned} f'[i,j] &= w_1 f[i-1,j-1] + w_2 f[i-1,j] + w_3 f[i-1,j+1] \\ &+ w_4 f[i,j-1] + w_5 f[i,j] + w_6 f[i,j+1] \\ &+ w_7 f[i+1,j-1] + w_8 f[i+1,j] + w_9 f[i+1,j+1] \end{aligned}$$

$$h = \begin{pmatrix} w_9 & w_8 & w_7 \\ w_6 & w_5 & w_4 \\ w_3 & w_2 & w_1 \end{pmatrix} \qquad g = \begin{pmatrix} w_1 & w_2 & w_3 \\ w_4 & w_5 & w_6 \\ w_7 & w_8 & w_9 \end{pmatrix}$$

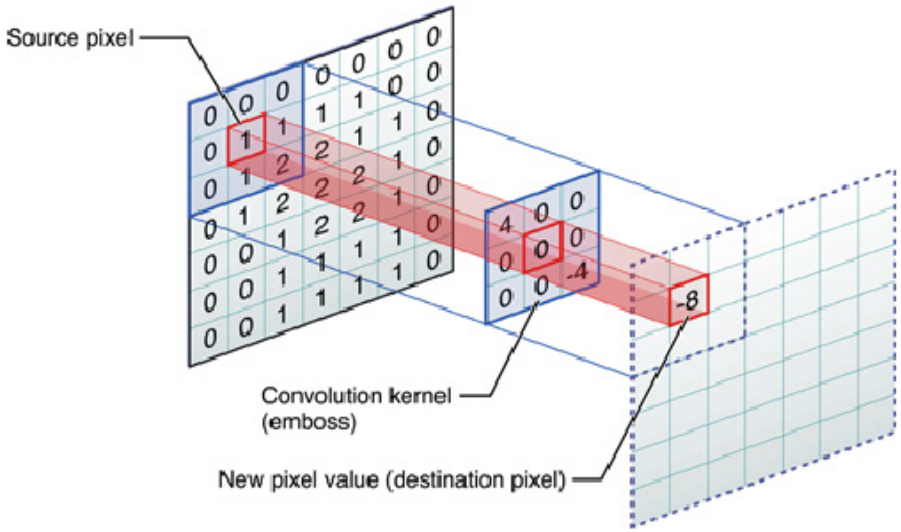
- **Calcul en pratique**
  1. Application d'une symétrie centrale au filtre : $h[n,m] \Rightarrow h'[n,m] = h[-n,-m]$
  2. Pour un pixel (i,j), somme pondérée entre le voisinage et les coefficients du filtre : $\sum_{n,m} f[i+n, j+m] \cdot g[n,m]$

Corrélation croisée ou convolution ?

- Convolution 2D : $f'[i,j] = (f * h)[i,j] = \sum_{n=-\frac{p-1}{2}}^{+\frac{p+1}{2}} \sum_{m=-\frac{q-1}{2}}^{+\frac{q+1}{2}} f[i-n, m-j] \cdot h[n,m]$
- Corrélation croisée : $f'[i,j] = (f \star h)[i,j] = \sum_{n=-\frac{p-1}{2}}^{+\frac{p+1}{2}} \sum_{m=-\frac{q-1}{2}}^{+\frac{q+1}{2}} f[i+n, m+j] \cdot h[n,m]$

Corrélation croisée $\approx$ convolution, sans la symétrisation du masque de poids du filtre :

$$h = \begin{pmatrix} -4 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 4 \end{pmatrix} \Rightarrow g = \begin{pmatrix} 4 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & -4 \end{pmatrix}$$

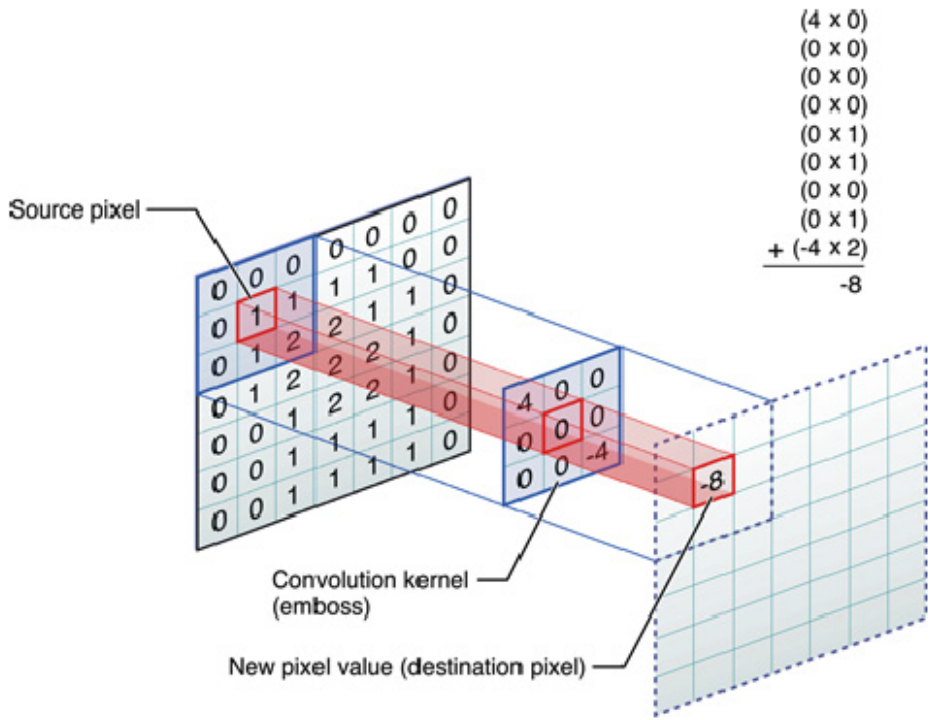


Corrélation croisée ou convolution ?

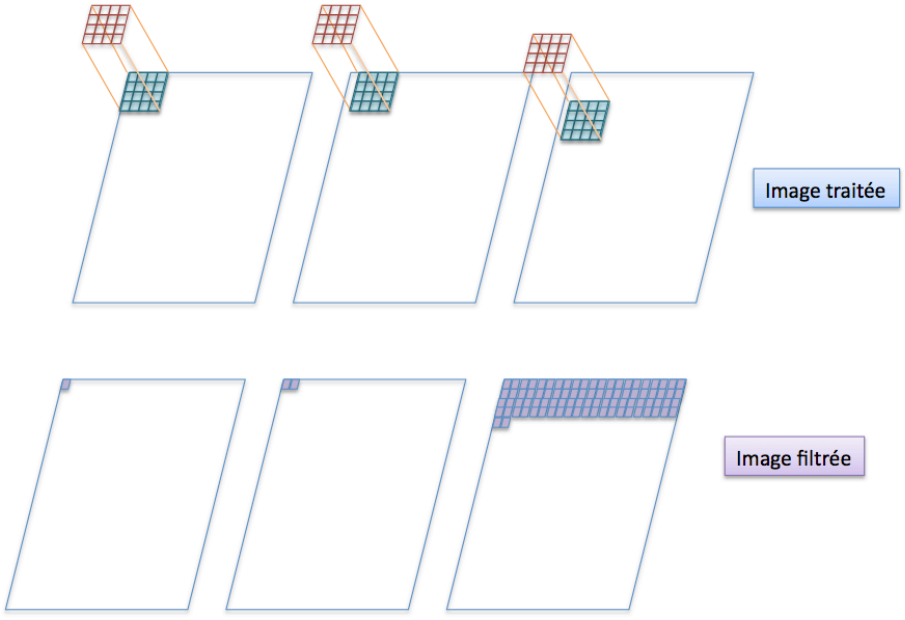
- Convolution 2D : $f'[i,j] = (f * h)[i,j] = \sum_{n=-\frac{p-1}{2}}^{+\frac{p+1}{2}} \sum_{m=-\frac{q-1}{2}}^{+\frac{q+1}{2}} f[i-n, m-j] \cdot h[n,m]$
- Corrélation croisée : $f'[i,j] = (f \star h)[i,j] = \sum_{n=-\frac{p-1}{2}}^{+\frac{p+1}{2}} \sum_{m=-\frac{q-1}{2}}^{+\frac{q+1}{2}} f[i+n, m+j] \cdot h[n,m]$

Corrélation croisée $\approx$ convolution, sans la symétrisation du masque de poids du filtre :

$$h = \begin{pmatrix} -4 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 4 \end{pmatrix} \Rightarrow g = \begin{pmatrix} 4 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & -4 \end{pmatrix}$$



Interprétation de la corrélation croisée



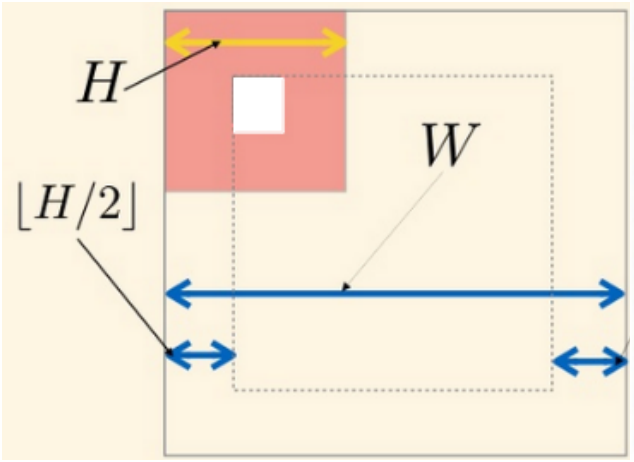
- Corrélation croisée : produit scalaire entre chaque région de l'image I et le filtre $\mathbf{h}$:

$$(I \star h)[i,j] = I[i-d : i+d, j-d : j+d] \cdot \mathbf{h} = \sum_{p,q} I[i,j] \cdot h[i,j]$$

- L'entrée de l'opération de filtrage est une image 2D, la sortie est une "carte d'activation"(image filtrée).

Padding

- Pour un filtre $\mathbf{h}$ de dimensions (w, h), que faire le long des bords de l'image ?
  - il y a $\lfloor \frac{h}{2} \rfloor$ (ou $\lfloor \frac{w}{2} \rfloor$) pixels qui sont dans le filtre mais hors de l'image
- Option 1 : convolution "valide" $\Longrightarrow$ on ne calcule pas la convolution sur ces pixels $\Longrightarrow$ la sortie a des dimensions réduites par rapport à l'image
- Option 2 : convolution "identique"ou "complète" $\Longrightarrow$ ajouter du remplissage (*padding*) avec des valeurs arbitraires : zéro, recopie, symétrie, etc.



0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	1	2	5	6	3	6	7	3	0	0
0	0	2	3	4	6	7	5	1	8	4	0	0
0	0	8	7	6	5	7	6	3	3	4	0	0
0	0	2	3	5	6	7	8	2	7	3	0	0
0	0	4	5	3	2	1	6	8	7	2	0	0
0	0	1	4	5	3	2	6	7	8	1	0	0
0	0	2	3	4	5	6	8	9	2	1	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0

2	2	2	3	4	6	7	5	1	8	4	4	4
1	1	1	1	2	5	6	3	6	7	3	3	3
1	1	1	1	2	5	6	3	6	7	3	3	7
3	2	2	3	4	6	7	5	1	8	4	4	8
7	8	8	7	6	5	7	6	3	3	4	4	3
3	2	2	3	5	6	7	8	2	7	3	3	7
5	4	4	5	3	2	1	6	8	7	2	2	7
4	1	1	4	5	3	2	6	7	8	1	1	8
3	2	2	3	4	5	6	8	9	2	1	1	2
3	2	2	3	4	5	6	8	9	2	1	1	2
3	1	1	4	5	3	2	6	7	8	1	1	2

Convolution à pas (*strided convolution*)

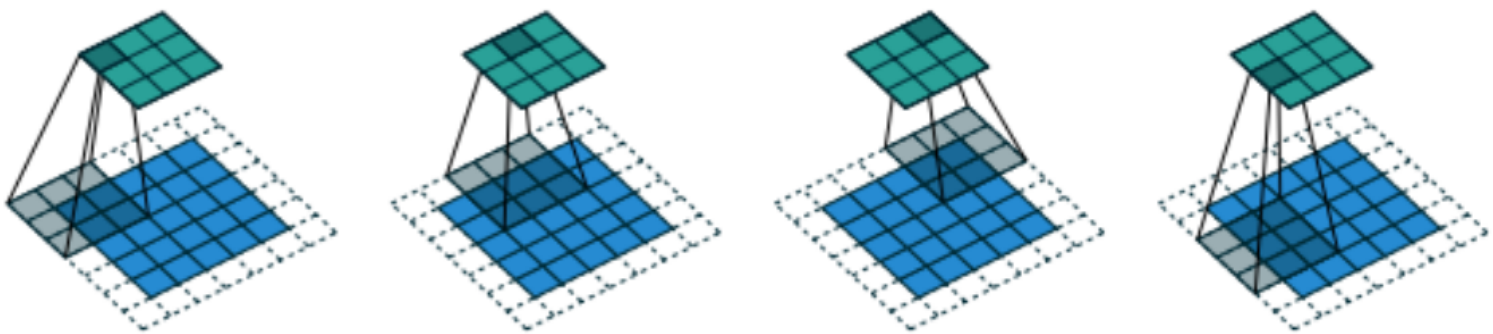
Formule classique de la convolution :

$$T_h(f)[i,j] = (f \star h)[i,j] = \sum_{n=-\frac{d-1}{2}}^{\frac{d-1}{2}} \sum_{m=-\frac{d-1}{2}}^{\frac{d-1}{2}} f[n-i, m-j]h[n,m]$$

Formule de la convolution à pas :

$$T_h(f)[i,j] = (f \star h)[i,j] = \sum_{n=-\frac{d-1}{2}}^{\frac{d-1}{2}} \sum_{m=-\frac{d-1}{2}}^{\frac{d-1}{2}} f[n-s\cdot i, m-s\cdot j]h[n,m]$$

- La convolution à pas s décale de s la fenêtre (région de l'image) observée $\approx$ sous-échantillonnage de l'image filtrée ;
- La convolution classique est un cas particulier de la convolution à pas avec $s = 1$.



Convolution d'une image (5×5) par un filtre (3×3), avec un *padding* de 1 pixel et un pas de 2.

Exemple de filtre convolutif : calcul des gradients d'une image

Pour une image I, on appelle G_x et G_y les gradients horizontaux et verticaux définis par :

$$\vec{G}(x,y) = (G_x, G_y) = \left(\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y}\right)$$

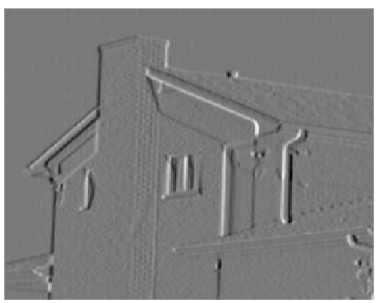
Filtres de Sobel

La dérivée s'approxime par différences finies : $\frac{\partial I}{\partial x}[i,j] \approx \frac{1}{2}(I[i+1,j] - I[i-1,j])$
En notant :

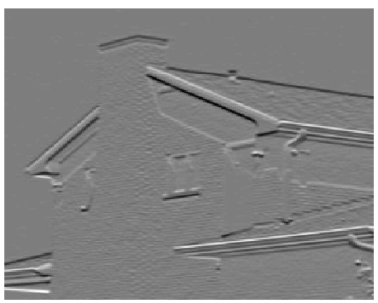
$$M_x = \frac{1}{4} \cdot \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \quad M_y = \frac{1}{4} \cdot \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

alors $G_x \approx I \star M_x$ et $G_y \approx I \star M_y$.

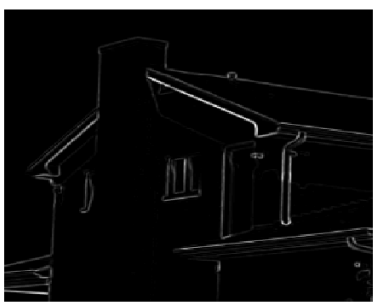
Image source



$I \star G_x$



$I \star G_y$



$I_e = \|\vec{G}\|$

Plan du cours

Pooling

Définition

Agrégation d'un ensemble de valeurs $\mathbf{x} = \{x_1, x_2, \ldots, x_N\}$ en un unique scalaire représentatif $\tilde{\mathbf{x}}$.

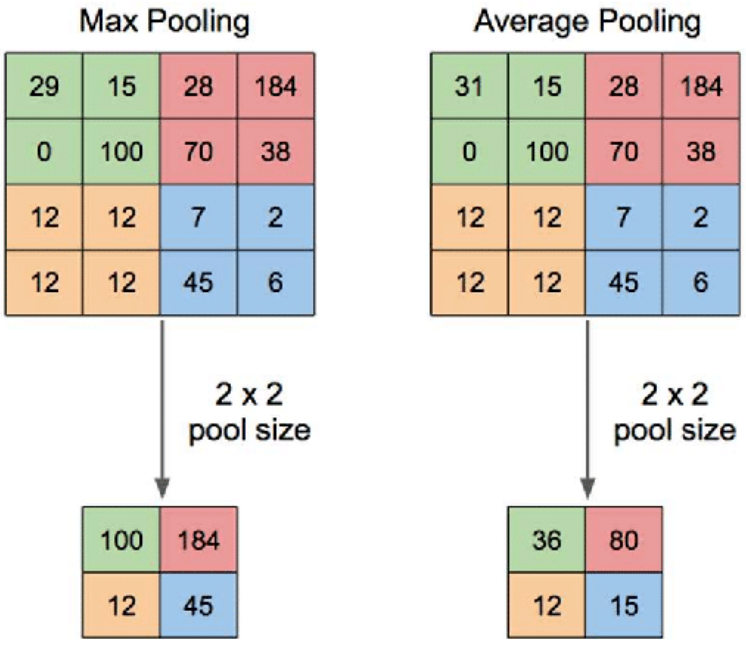
Exemples de fonctions de *pooling* :

- Max pooling : $\tilde{\mathbf{x}} = \max_{i=\{1,\ldots,N\}} x_i$
- Average pooling : $\tilde{\mathbf{x}} = \frac{1}{N} \sum_{i=1}^{N} x_i$
- ℓ_p pooling : $\tilde{\mathbf{x}} = \left(\frac{1}{N} \sum_{i=1}^{N} x_i^p \right)^{\frac{1}{p}}$

Intérêt du pooling

- Description sommaire des statistiques d'un échantillon $\Longrightarrow$ réduction de dimension
- Invariant à la position des valeurs (permutation des valeurs $\Longrightarrow$ même résultat)

Échantillonnage des cartes filtrées



Pooling spatial

- Agrégation sur une image (ou une carte) des valeurs, région par région.
- Entrée du pooling : carte ($\sim$ image), sortie : carte réduite.
- L'agrégation se fait de manière locale $\Longrightarrow$ le pooling a un "champ réceptif"restreint à un voisinage.

Paramètres :

- Fonction de pooling ($\min, \max, \text{avg}, \ldots$)
- Taille du voisinage local, pas du pooling (*stride*)

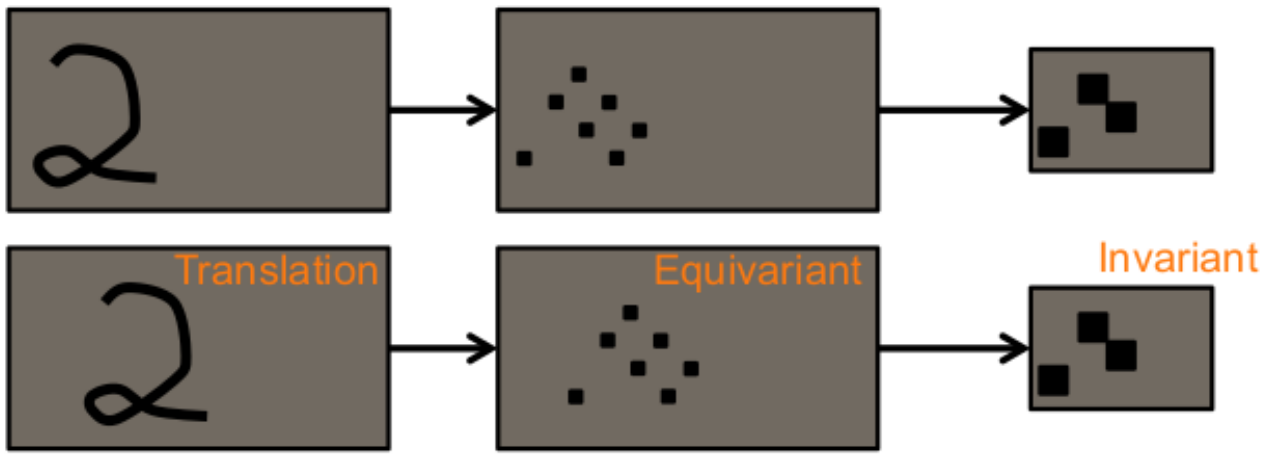
Spatial pooling : de l'équivariance à l'invariance

Équivariance de f par rapport à une transformation g : $f(g(x)) = g(f(x))$.

- Exemple : f convolution, g translation.

Invariance de f par rapport à une transformation g : $f(g(x)) = f(x)$.

- Exemple : f *max-pooling*, g translation (petite).



Invariance à la translation selon $\overrightarrow{T} = (t_x, t_y)$ si :

- $\overrightarrow{T} \not\Rightarrow$ fait apparaître un nouveau max
- $\overrightarrow{T} \not\Rightarrow$ sort le max de la région considérée

Ici, carte 5×5, max-pooling 3×3 : $\max = 15$,
Invariance OK : si $(t_x, t_y) \in \pm 1$ **px**

$$C = \begin{bmatrix} 11 & -5 & 1 & -2 & 0 \\ 1 & 3 & 0 & 0 & 5 \\ 8 & 4 & 15 & -10 & 4 \\ 8 & 6 & 5 & 3 & 7 \\ 3 & 0 & -2 & 9 & 3 \end{bmatrix}$$

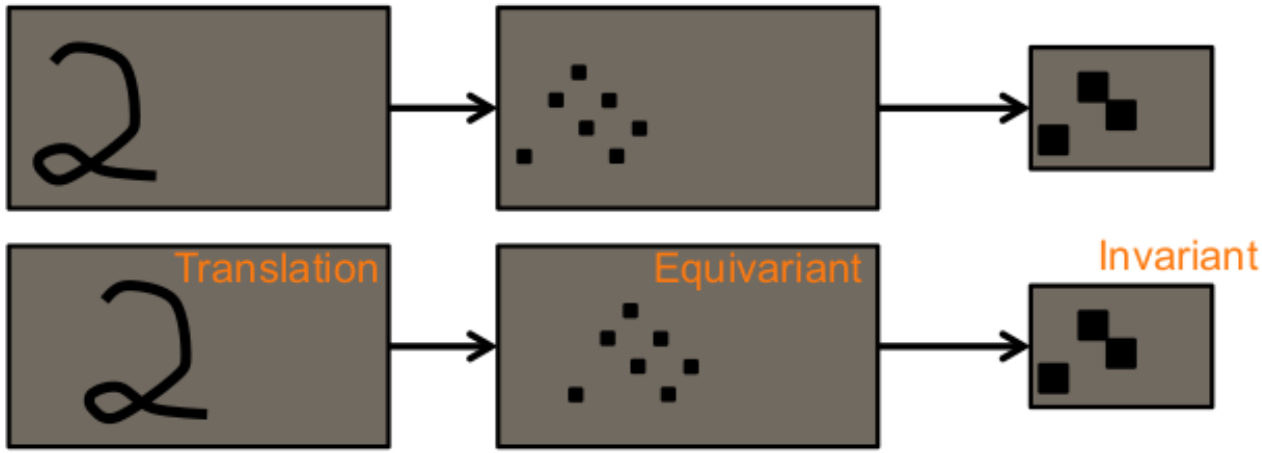
Spatial pooling : de l'équivariance à l'invariance

Équivariance de f par rapport à une transformation g : $f(g(x)) = g(f(x))$.

- Exemple : f convolution, g translation.

Invariance de f par rapport à une transformation g : $f(g(x)) = f(x)$.

- Exemple : f *max-pooling*, g translation (petite).



Invariance à la translation selon $\overrightarrow{T} = (t_x, t_y)$ si :

- $\overrightarrow{T} \not\Rightarrow$ fait apparaître un nouveau max
- $\overrightarrow{T} \not\Rightarrow$ sort le max de la région considérée

Ici, carte 5×5, max-pooling 3×3 : $\max = 15$,
Invariance KO : si $t_x = +1$ px

$$C = \begin{bmatrix} 11 & -5 & 1 & -2 & 0 \\ 1 & 3 & 0 & 0 & 5 \\ 8 & 15 & 4 & -10 & 4 \\ 8 & 6 & 5 & 3 & 7 \\ 3 & 0 & -2 & 9 & 3 \end{bmatrix}$$

Plan du cours

Couche convolutive

- Entrée : carte I ($\sim$ image) de dimension $(C_{\text{in}}, W_{\text{in}}, H_{\text{in}})$
- Paramètres : C_{out} (nombre de canaux de sortie), k_w, k_h taille des filtres.
- Sortie : cartes d'activation de dimension $(C_{\text{out}}, W_{\text{out}}, H_{\text{out}})$

La j^{e}carte d'activation de sortie s'obtient en sommant chaque canal d'entré, filtrée par le noyau $\mathbf{h}_{i,j}$, plus un biais :

$$o_j = \sum_{i=1}^{C_{\text{in}}} I \star \mathbf{h}_{i,j} + b_j$$

Il y a donc $n_{\text{params}} = \underbrace{C_{\text{out}} C_{\text{in}} k_w k_h}_{\text{filtres}} + \underbrace{C_{\text{out}}}_{\text{biais}}$ paramètres.

À noter

En général :

- On ajoute du *padding* de sorte à ce que $W_{\text{out}} = W_{\text{in}}$ et $H_{\text{out}} = H_{\text{in}}$.
- Les filtres sont carré : $k_w = k_h$.

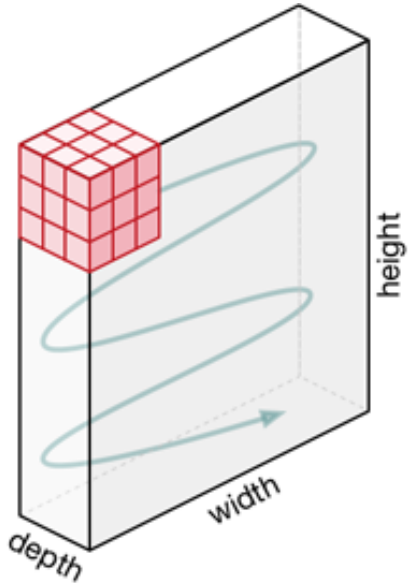
La convolution est une opération linéaire, la couche convolutive est une opération affine (avec le biais).

Couche convolutive sur les tenseurs

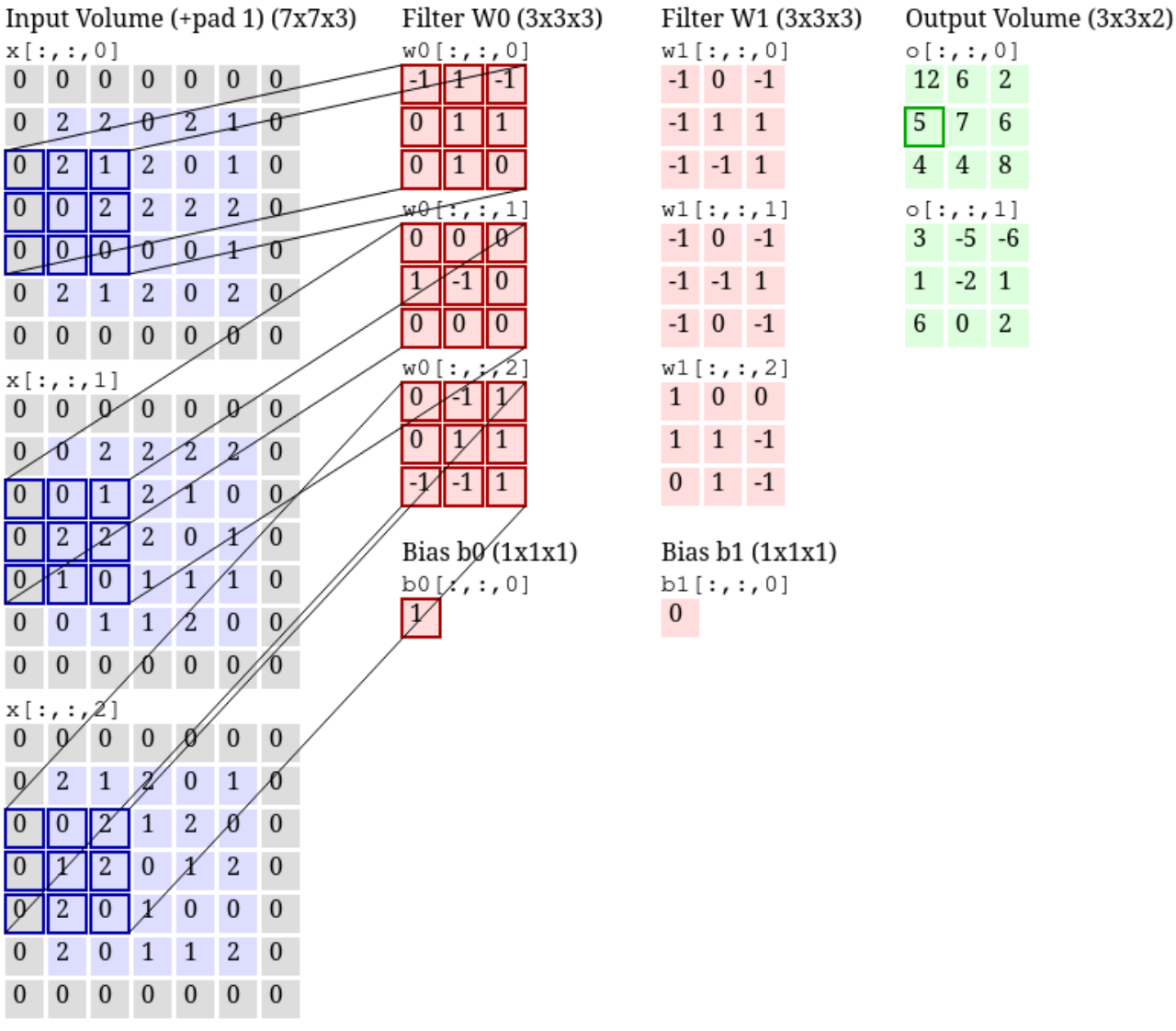
En pratique, cette définition de la convolution revient à réaliser une convolution 3D en prenant en compte tous les canaux d'entrée avec des filtres de dimension $(C_{\text{in}}, k_w, k_h)$:

$$T_h[i,j] = (f \star h)[i,j] = \sum_{k=1}^{K} \sum_{n=-\frac{d-1}{2}}^{\frac{d-1}{2}} \sum_{m=-\frac{d-1}{2}}^{\frac{d-1}{2}} f[i-n, m-j, k]\, h[n, m, k] + b$$

Le filtrage selon la troisième dimension ("profondeur") permet d'apprendre les corrélations entre les cartes d'activation pour différents filtres.



Illustration



Rétropropagation dans une couche convolutive

Algorithme de rétropropagation : que vaut le gradient de la fonction de coût par rapport aux paramètres de la couche convolutive ?

- Paramètres : matrice de poids du filtre $(k \times k)$: $\mathbf{h} = \begin{pmatrix} h_{11} & h_{12} & \dots & h_{1k} \\ h_{21} & h_{22} & \dots & h_{2k} \\ \vdots & \vdots & \dots & \vdots \\ h_{k1} & h_{k2} & \dots & h_{kk} \end{pmatrix}$
- Pour une seule convolution : $\mathbf{s} = \mathbf{x} \star \mathbf{h}$

Gradients

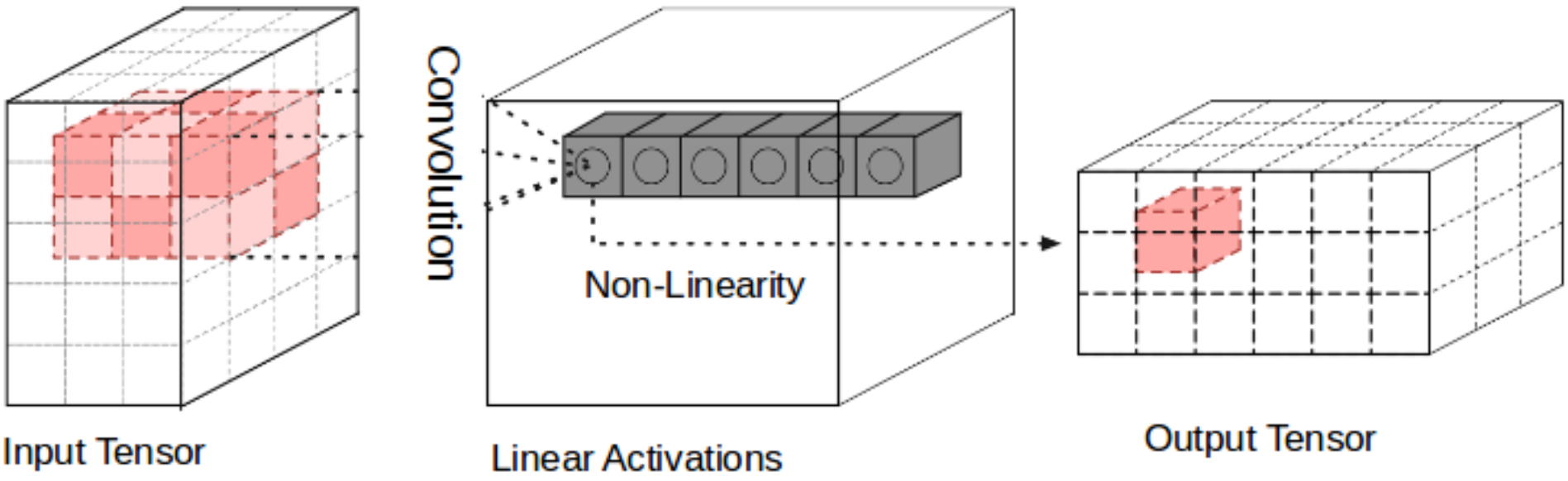
$$\frac{\partial \mathcal{L}}{\partial \mathbf{h}} = \mathbf{x} \star \frac{\partial \mathcal{L}}{\partial \mathbf{s}}$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{x}} = \mathbf{h}^T \circledast \frac{\partial \mathcal{L}}{\partial \mathbf{s}}$$

où $\star$ représente la corrélation croisée et $\circledast$ la corrélation croisée complète.

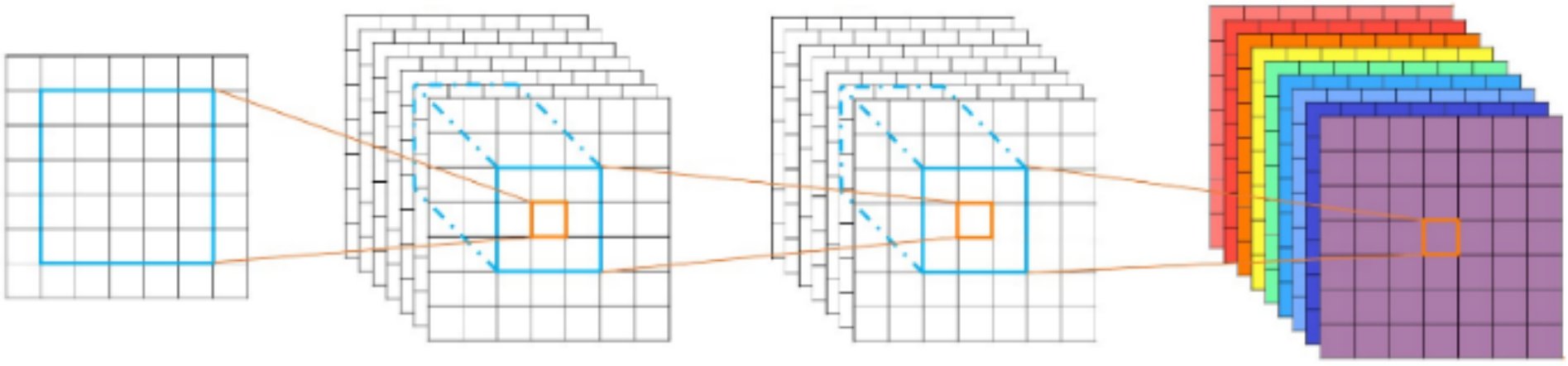
$\Longrightarrow$ la *backward pass* pour une couche convolutive est encore une couche convolutive !

Non-linéarité et couche convolutive



- Couche convolutive : tenseur en entrée $\rightarrow$ tenseur en sortie
  1. Convolution : opération de filtrage linéaire/affine
  2. Application d'une fonction d'activation non-linéaire élément par élément

Hiérarchie de convolutions



Empilement des couches convolutives

- Chaque couche convolutive applique un filtrage affine + activation non-linéaire
- L'empilement des couches convolutive créé une **hiérarchie de convolutions**
- L'empilement en profondeur permet de modéliser des fonctions décisionnelles plus complexes
  - Comme pour les perceptrons, la non-linéarité est indispensable, sinon le modèle hiérarchique est équivalent au modèle "plat".

Exemple de hiérarchie convolutive : détection de contours

Image niveaux de gris $I(W \times H \times 1)$

$G_x^2 \approx$ filtre 1

$G_y^2 \approx$ filtre 2

1. 1re couche : convolution à deux filtres

$$M_x = \frac{1}{4} \cdot \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \quad M_y = \frac{1}{4} \cdot \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

   - non-linéarité quadratique : $\sigma(z) = z^2$
   - $\Rightarrow$ Sortie : tenseur $(W \times H \times 2) \approx (G_x^2, G_y^2)$

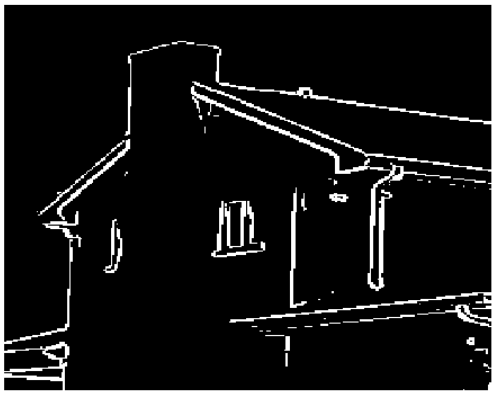
2. 2e couche : convolution avec un seul filtre 1×1 : $[+1, +1]$
   - Pour chaque pixel : $S = G_x^2 + G_y^2$
   - Activation non-linéaire : échelon $\sigma(z) = H(z - t)$ avec un seuil t

$\Longrightarrow$ Détection de contours !

Exemple de hiérarchie convolutive : détection de contours

G_x^2

G_y^2



Sortie

1. 1[re]couche : convolution à deux filtres

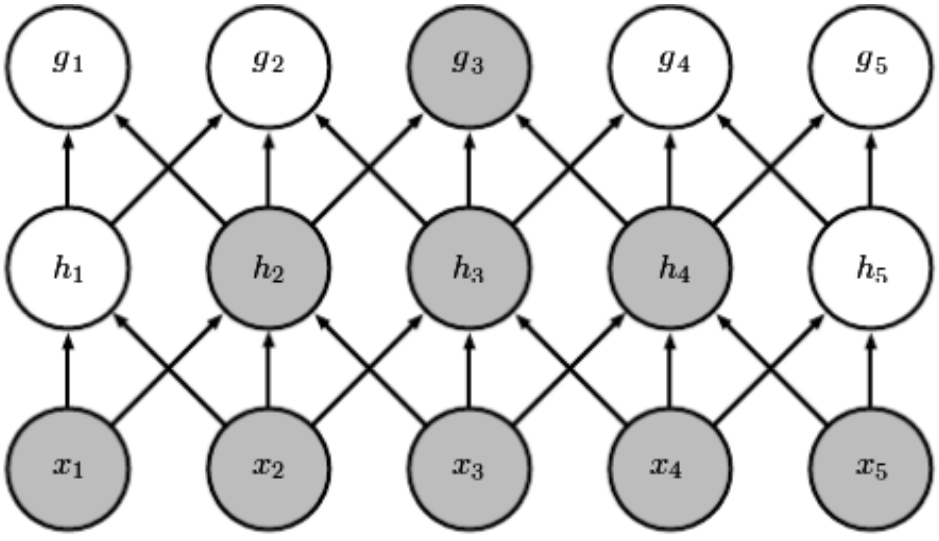
$$M_x = \frac{1}{4} \cdot \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \quad M_y = \frac{1}{4} \cdot \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

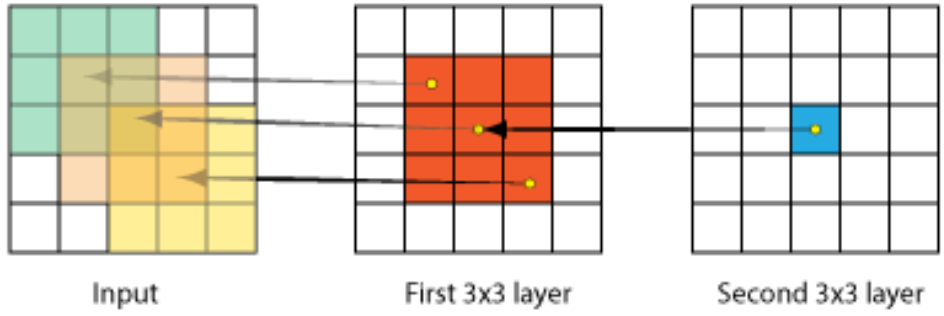
   - non-linéarité quadratique : $\sigma(z) = z^2$
   - $\Rightarrow$ Sortie : tenseur $(W \times H \times 2) \approx (G_x^2, G_y^2)$

2. 2[e]couche : convolution avec un seul filtre 1×1 : $[+1, +1]$
   - Pour chaque pixel : $S = G_x^2 + G_y^2$
   - Activation non-linéaire : échelon $\sigma(z) = H(z - t)$ avec un seuil t

$\Longrightarrow$ Détection de contours !

Champ réceptif





Le champ réceptif correspond à l'étendue spatiale dans l'image de départ du voisinage dont les pixels ont contribué à produire une activation donnée.

- Empilement de deux convolutions 3×3 : même "champ réceptif"qu'une seule convolution 5×5 sur l'image de départ

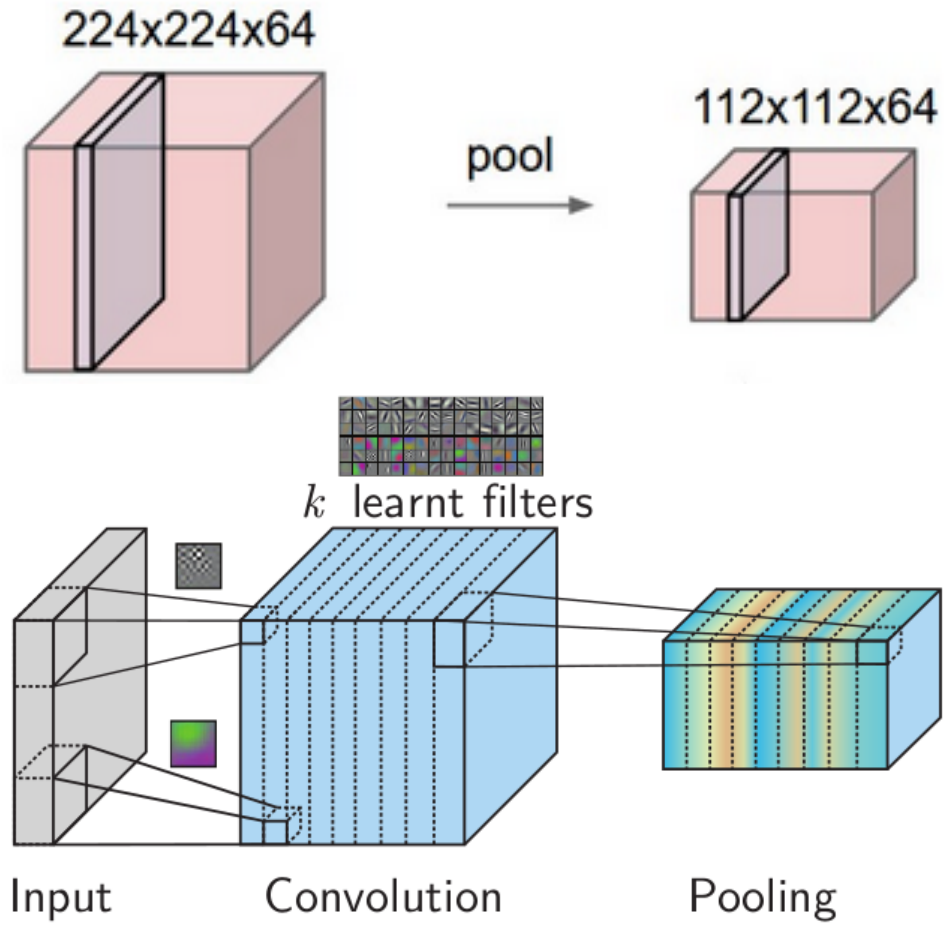
Les CNN empilent des couches convolutives pour combiner des activations ($\sim$ cartes filtrées) de plus en plus complexe, tout en augmentant le champ réceptif.

$\Longrightarrow$ connectivité locale pour chaque couche mais indirectement globale en considérant tout le réseau

Échantillonnage dans les réseaux convolutifs

En général, le *pooling* se fait canal par canal dans les tenseurs d'activation, de façon indépendante $\Longrightarrow$ *spatial pooling*.

- Bloc de base des modèles convolutifs : n couches convolutives + *maxpooling*
- Entrée : tenseur (C, W, H)/sortie : tenseur $(C', \frac{W}{k}, \frac{H}{k})$ avec k le pas du *pooling*



Rétropropagation dans une couche de *pooling*

pooling $\Longrightarrow$ pas de paramètres donc seul le gradient par rapport à l'entrée est utile.
$\mathbf{s} = \text{pooling}(\mathbf{x})$ (entrée $\mathbf{x}$, sortie $\mathbf{s}$, sur un voisinage $k \times k$)

Gradients du *maxpooling*

$$\mathbf{s}(i,j) = \max_{p,q} \mathbf{x}(p,q)$$

Gradient non-nul seulement pour la position maximale de chaque région :

$$\frac{\partial \mathcal{L}}{\partial \mathbf{x}}(i,j) = \begin{cases} \frac{\partial \mathcal{L}}{\partial \mathbf{x}}(i,j) & \text{si} (i,j) = \arg\max_{p,q} \mathbf{x}(p,q) \\ 0 & \text{sinon} \end{cases}$$

Gradients de l'*average pooling*

$$\mathbf{s}(i,j) = \frac{1}{k^2} \sum_{p,q} \mathbf{x}(p,q)$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{x}}(i,j) = \frac{1}{k^2} \sum_{p,q} \frac{\partial \mathcal{L}}{\partial \mathbf{s}}(p,q)$$

Réseau de neurones convolutif (*Convolutional neural network*)

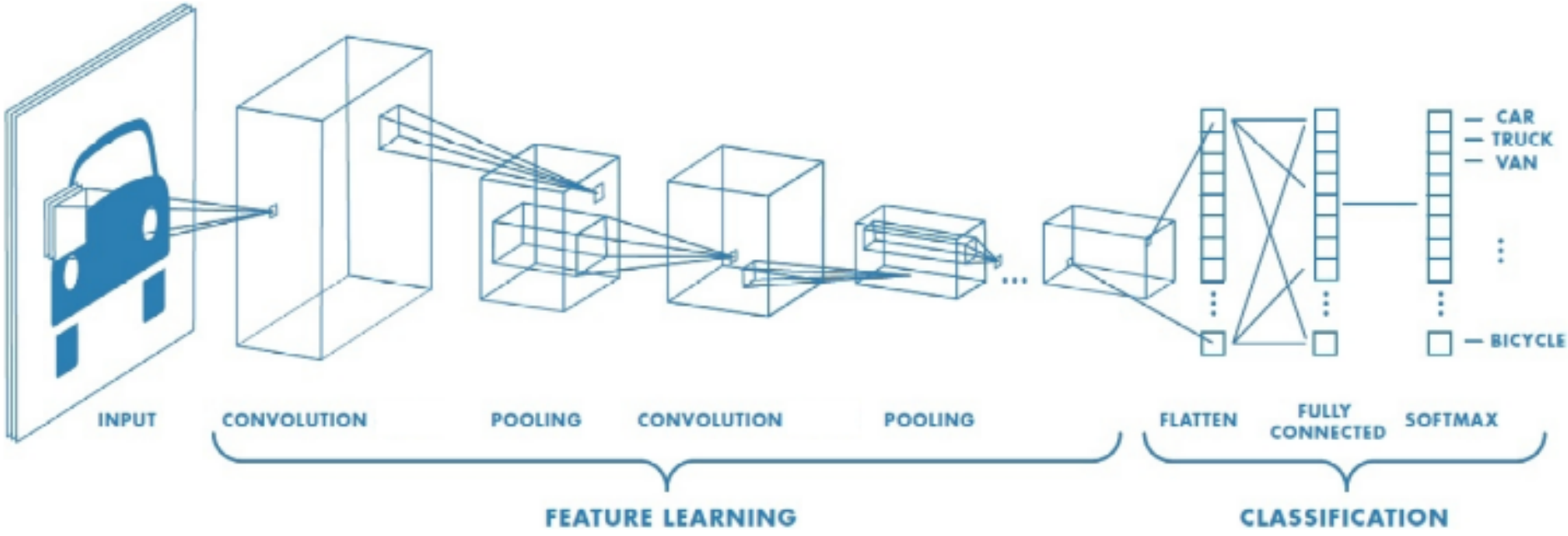
Construction d'un réseau convolutif

- Empilement de blocs convolution+*pooling*

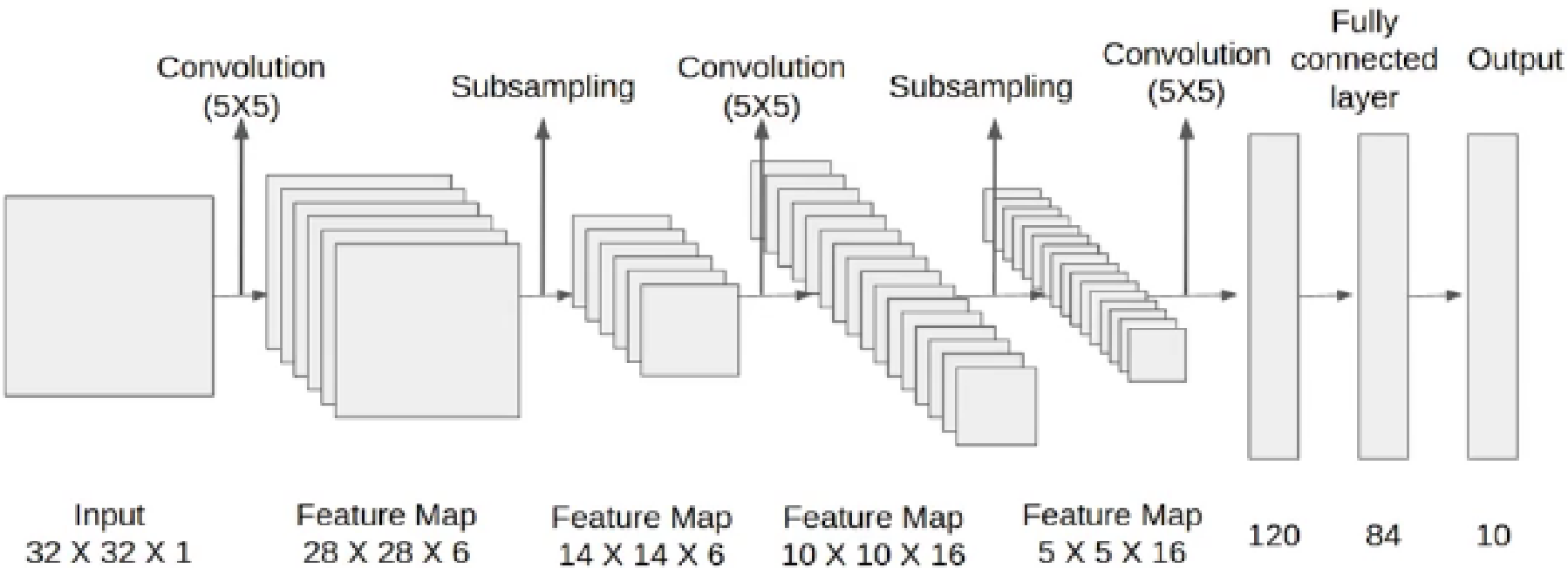
$\Longrightarrow$ réduction de la dimension spatiale

- Couches entièrement connectées (perceptron) pour projeter les activations vers l'espace de sortie

$\rightarrow$ si nécessaire, le tenseur d'activations est aplati avant la partie entièrement connectée



Exemple : LeNet-5 LeCun et al. 1998



5 “couches”avec des paramètres :

1. Convolution 5×5 : $32 \times 32 \times 1 \rightarrow 28 \times 28 \times 6$
2. Convolution 5×5 : $14 \times 14 \times 6 \rightarrow 10 \times 10 \times 16$
3. Convolution 5×5 : $10 \times 10 \times 16 \rightarrow 1 \times 1 \times 120$
4. Couche entièrement connectée : $120 \rightarrow 84$
5. Couche entièrement connectée ($\sim$ régression logistique) : $84 \rightarrow 10$

Bibliographie I

 LeCun, Yann et al. (nov. 1998). "Gradient-Based Learning Applied to Document Recognition". In : *Proceedings of the IEEE* 86.11, p. 2278-2324. ISSN : 0018-9219. DOI : 10.1109/5.726791.