UNIVERSITE ABDELMALEK ESSAADI
FACULTE DES SCIENCES
TETOUAN

جامعة عبد المالك السعدي
كلية العلوم
تطوان

Apprentissage profond pour la vision par ordinateur

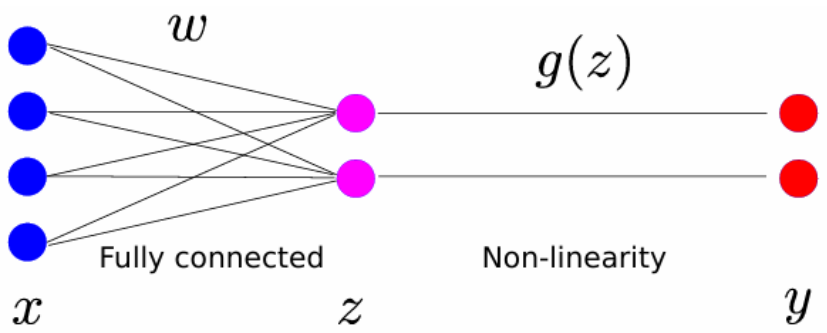
Presented by Pr. Abdellatif El Ouissari

Convolutional Neural Networks

Introduction

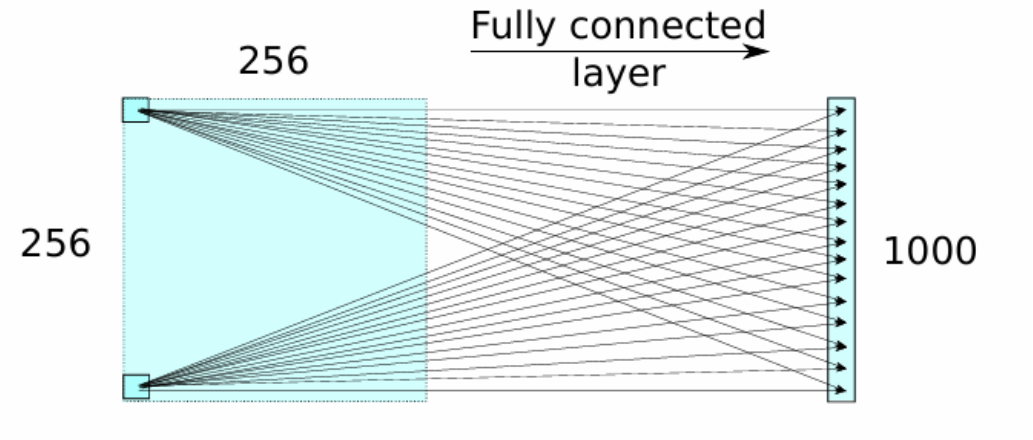
- Les réseaux neuronaux constituent un moyen très souple de modéliser des dépendances et des modèles complexes dans les données.

- Dans les leçons précédentes, nous avons vu les éléments suivants :
  - MLP : couches entièrement connectées, biais
  - Fonctions d'activation : sigmoïde, soft max, ReLU
  - Optimisation : descente de gradient, descente de gradient stochastique



Introduction

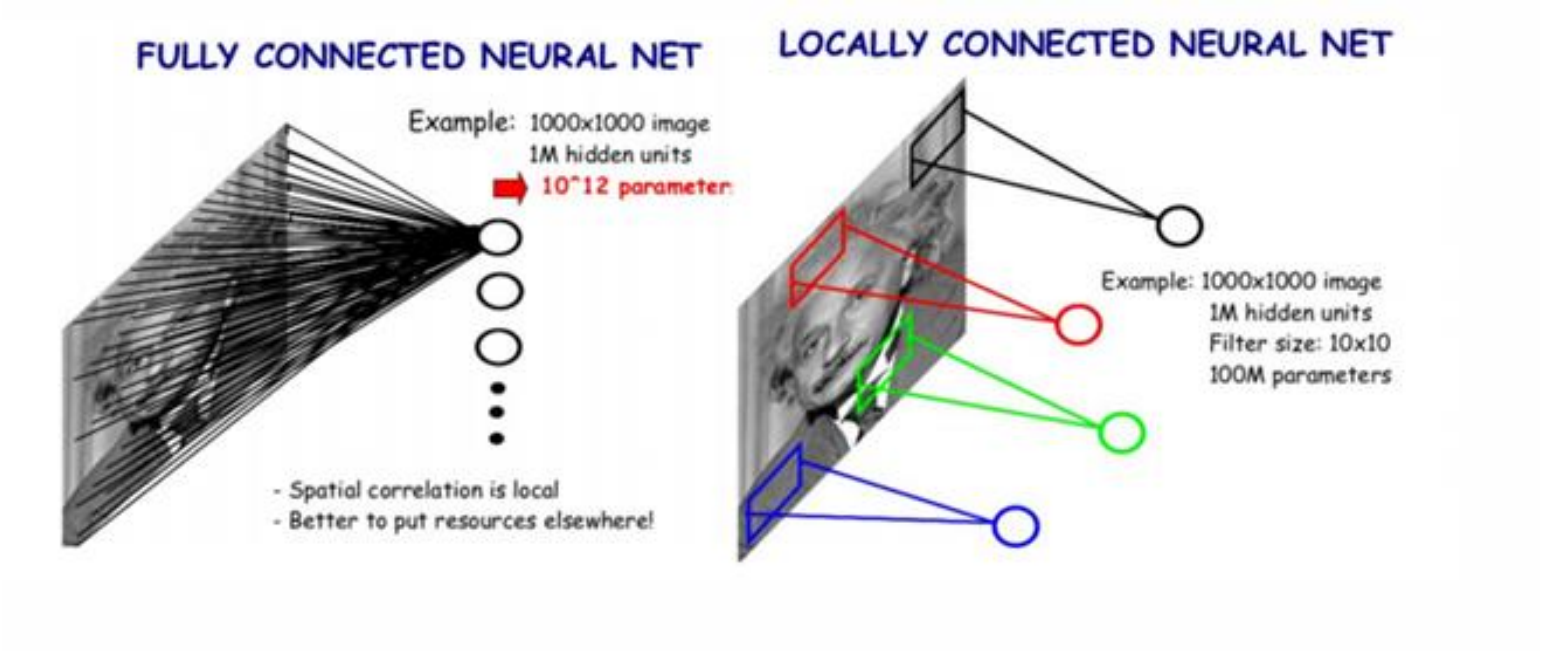
- Dans les MLP, chaque couche du réseau contient des couches entièrement connectées
- Malheureusement, cette approche présente de nombreux inconvénients



- Chaque unité cachée est connectée à chaque unité d'entrée
- Il y a une grande redondance dans ces poids :
  - Dans l'exemple ci-dessus, 65 millions de poids sont nécessaires.

Introduction

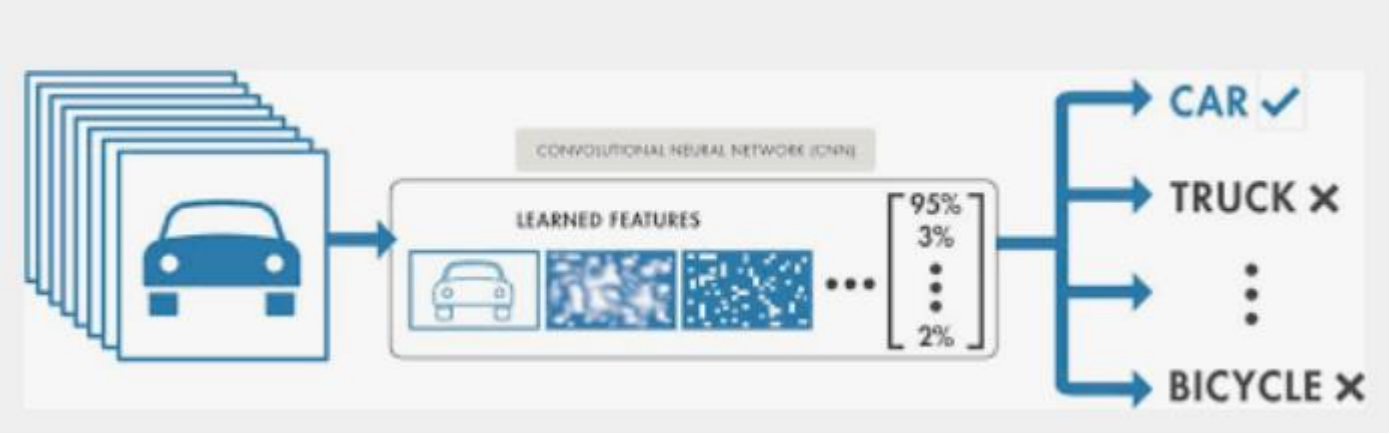
- Pour de nombreux types de données avec des structures topologiques en grille (par exemple, des images), il n'est pas nécessaire d'avoir autant de poids.
- Pour ces données, l'opération de convolution est souvent extrêmement utile.
- Réduit le nombre de paramètres à entraîner
  - L'entraînement est plus rapide
  - La convergence est plus facile : espace de paramètres plus petit



Introduction

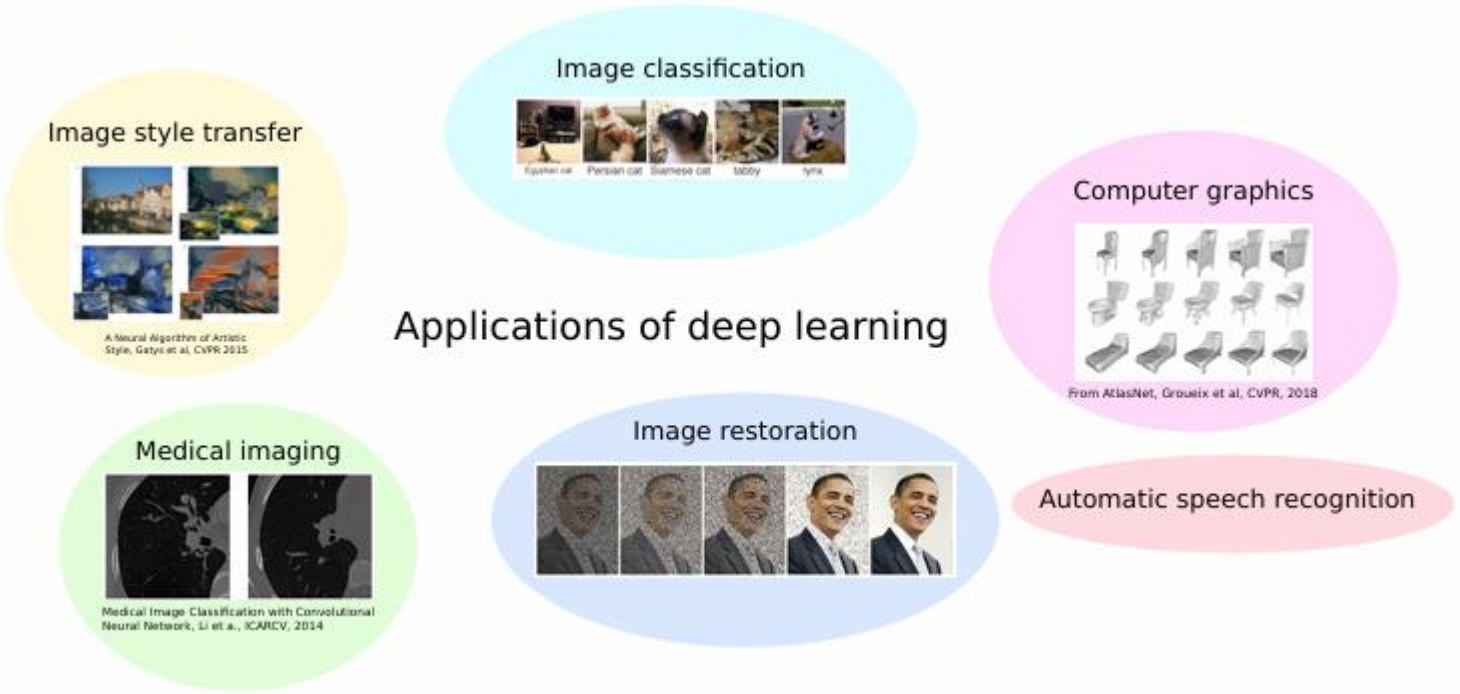
Un réseau de neurones avec des opérations de convolution est connu sous le nom de

Réseau de Neurones Convolutifs (CNN)



Introduction

- Depuis 2012, les CNN ont complètement révolutionné de nombreux domaines
- Les CNN produisent des résultats compétitifs/meilleurs pour la plupart des problèmes de traitement d'images et de vision par ordinateur.



Introduction

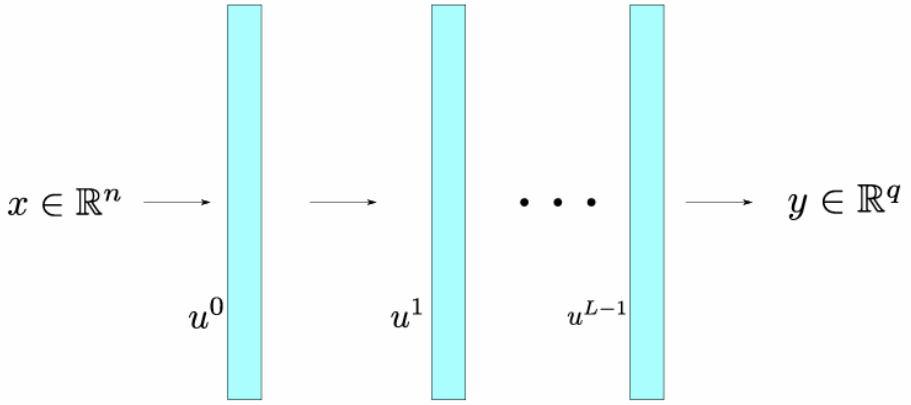
Notations

$x \in \mathbb{R}^n$: vecteur d'entrée

$y \in \mathbb{R}^q$: vecteur de sortie

u_ℓ : vecteur de caractéristiques à la couche ℓ

θ_ℓ : paramètres du réseau à la couche ℓ



Réseau neuronal à L couches

Introduction

•Un **Réseau de Neurones Convolutifs** (CNN) est simplement une concaténation de :

- Convolutions (filtres)
- Biais additifs
- Sous-échantillonnage (« Max-Pooling », etc.)
- Non-linéarités

Couches convolutives

Opérateur de convolution

Soit f et g deux fonctions intégrables. L'opérateur de convolution $*$ prend en entrée deux fonctions de ce type, et produit une autre fonction $h = f * g$, qui est définie en tout point $t \in \mathbb{R}$ comme :

$$\boldsymbol{h(t) = (f * g)(t) = \int_{-\infty}^{+\infty} f(\delta) g(t - \delta) d\delta}$$

- Intuitivement, la fonction h est définie comme le produit intérieur entre f et une version décalée de g

Couches convolutives

Dans de nombreuses applications pratiques, en particulier pour les **CNN**, nous utilisons l'opérateur de convolution discrète, qui agit sur des fonctions discrétisées ;

Opérateur de convolution discret

Soit f_n et g_n deux séries sommables, avec $n \in \mathbb{Z}$. L'opérateur de convolution discrète est défini comme suit :

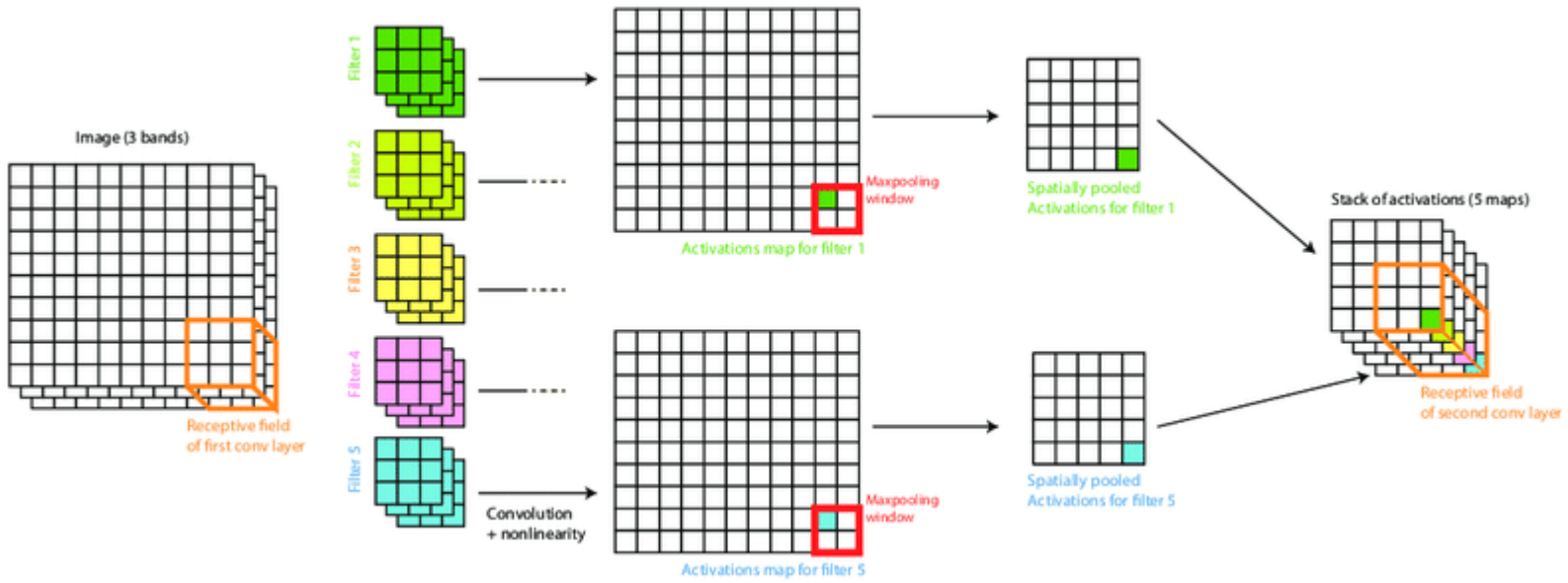
$$(f * g)(n) = \sum_{i=-\infty}^{+\infty} f(i)g(n - i)$$

Couches convolutives

Intuitivement, la fonction h est définie comme le produit intérieur entre f et une version décalée de g

En pratique, le **filtre** a un faible support spatial, de l'ordre de 3 × 3, ou 5 × 5

Par conséquent, seul un petit nombre de paramètres doit être entraîné (9 ou 25 pour ces filtres).



Couches convolutives

Propriétés de la convolution

1. **Associativité** : $(f * g) * h = f * (g * h)$
2. **Commutativité** : $f * g = g * f$
3. **Bilinéarité** : $(\alpha f) * (\beta g) = \alpha\beta(f * g), pour\ (\alpha, \beta) \in R \times R$
4. **Équivariance par translation** : $(f * (g + \tau))(t) = (f * g)(t + \tau)$

Couches convolutives

Associativité, commutativité

- L'associativité+commutativité implique que l'on peut effectuer la convolution dans n'importe quel ordre
- Il n'y a pas d'intérêt à avoir deux ou plusieurs convolutions consécutives.
  - Ceci est vrai en fait pour toute application linéaire

Équivariance à la translation

- L'équivariance implique que la convolution de toute entrée décalée $(f + \tau) * g$ contient la même information que $f * g$.
- Ceci est utile, puisque nous voulons détecter des objets n'importe où dans l'image.

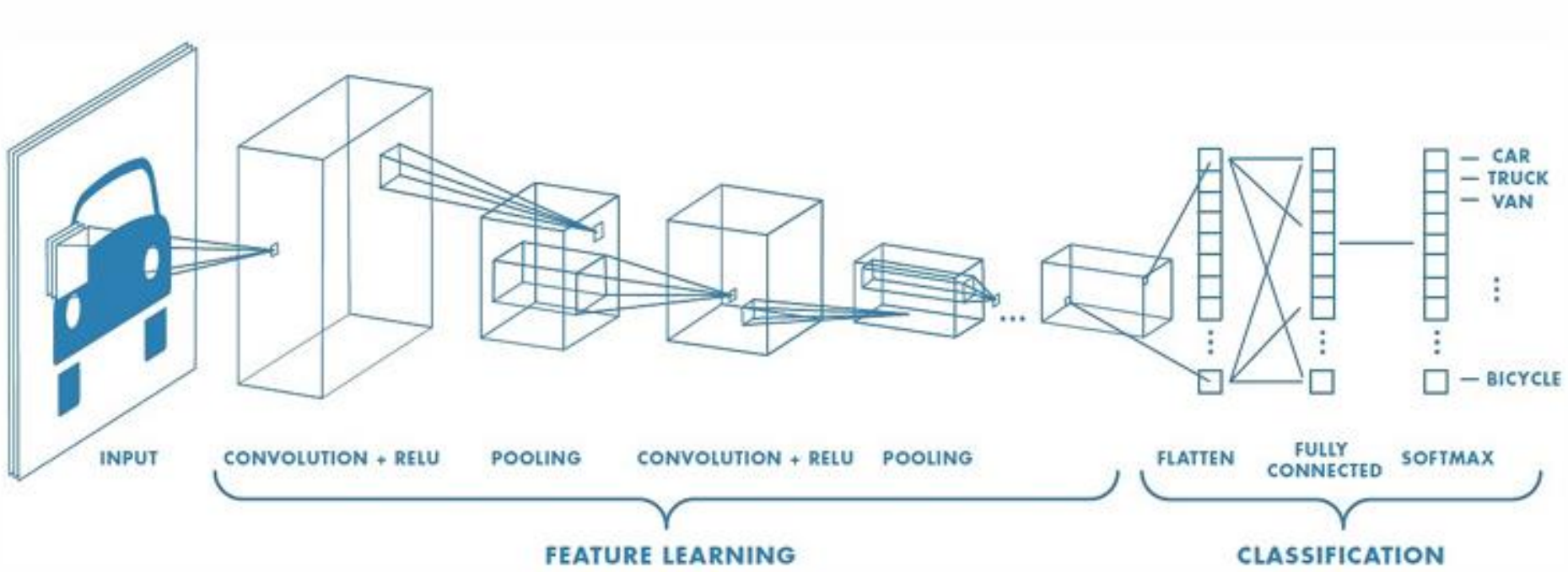
Couches convolutives - Convolution 2D

- Le plus souvent, nous allons travailler avec des images
- Par conséquent, nous avons besoin d'un opérateur de convolution 2D : il est défini d'une manière très similaire à la convolution 1D.

Opérateur de convolution 2D

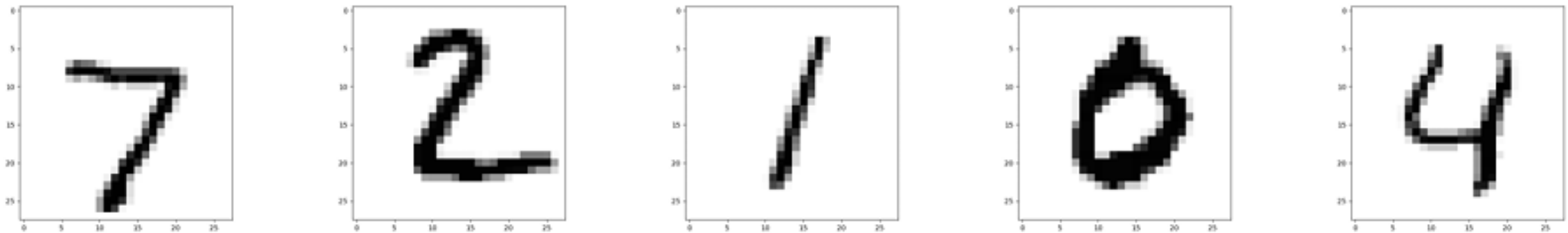
$$(\boldsymbol{f} * \boldsymbol{g})(\boldsymbol{s}, \boldsymbol{t}) = \sum_{\boldsymbol{i}=-\infty}^{+\infty} \sum_{\boldsymbol{j}=-\infty}^{+\infty} \boldsymbol{f}(\boldsymbol{i}, \boldsymbol{j}) \boldsymbol{g}(\boldsymbol{s} - \boldsymbol{i}, \boldsymbol{t} - \boldsymbol{j})$$

MLP pour les images



Reconnaissance de chiffres

Il s'agit de reconnaître de façon automatique des chiffres écrits à la main.

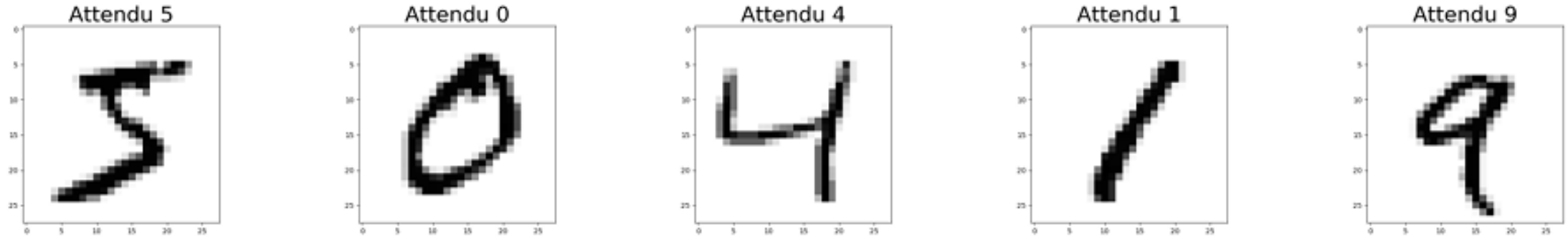


C'est l'un des succès historiques des réseaux de neurones qui permet par exemple le tri automatique du courrier par lecture du code postal.

Données

Base MNIST

Un élément essentiel de l'apprentissage automatique est de disposer de données d'apprentissage nombreuses et de qualité. Une telle base est la base MNIST dont voici les premières images.



Une donnée est constituée d'une image et du chiffre attendu.

Base MNIST

- La base est formée de 60000 données d'apprentissage et de 10000 données de test.
- Chaque donnée est de la forme : [une image, le chiffre attendu].
- Chaque image est de taille 28×28 pixels, chaque pixel contenant un des 256 niveaux de gris (numérotés de 0 à 255).

Ces données sont accessibles très simplement avec tensorflow/keras :

```
from tensorflow.keras.datasets import mnist
(X_train_data,Y_train_data),(X_test_data,Y_test_data) = mnist.load_data()
```

Noter que l'on peut afficher une image à l'aide de matplotlib par la commande :

```
plt.imshow(X_train_data[i], cmap='Greys')
```

suivie de `plt.show()`.

Traitement des données

Pour une utilisation par un réseau de neurones nous devons d’abord transformer les données.

Traitement des données

Donnée d'entrée. En entrée du réseau de neurones, nous devons avoir un vecteur. Au départ chaque image est un tableau de taille 28×28 ayant des entrées entre 0 et 255. Nous la transformons en un vecteur de taille $784 = 28^2$ et nous normalisons les données dans l'intervalle $[0, 1]$ (en divisant par 255).

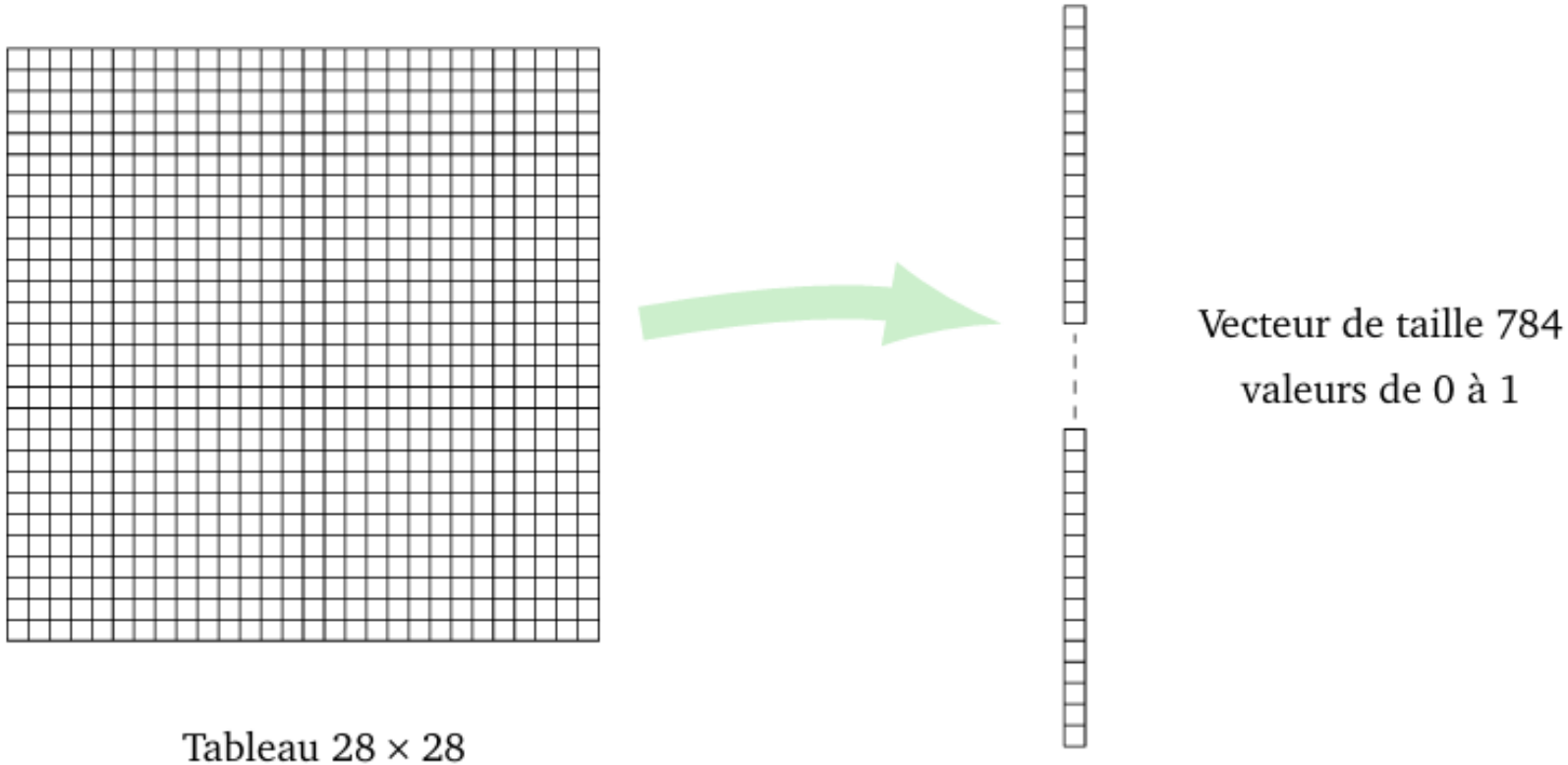


Tableau 28×28
valeurs de 0 à 255

Vecteur de taille 784
valeurs de 0 à 1

Traitement des données

Ainsi, une entrée X est un « vecteur-image », c'est-à-dire un vecteur de taille 784 représentant une image.

Donnée de sortie. Notre réseau de neurones ne va pas renvoyer le chiffre attendu, mais une liste de 10 probabilités. Ainsi chaque chiffre doit être codé par une liste de 0 et de 1.

- 0 est codé par (1,0,0,0,0,0,0,0,0,0),
- 1 est codé par (0,1,0,0,0,0,0,0,0,0),
- 2 est codé par (0,0,1,0,0,0,0,0,0,0),
- ...
- 9 est codé par (0,0,0,0,0,0,0,0,0,1).

Traitement des données

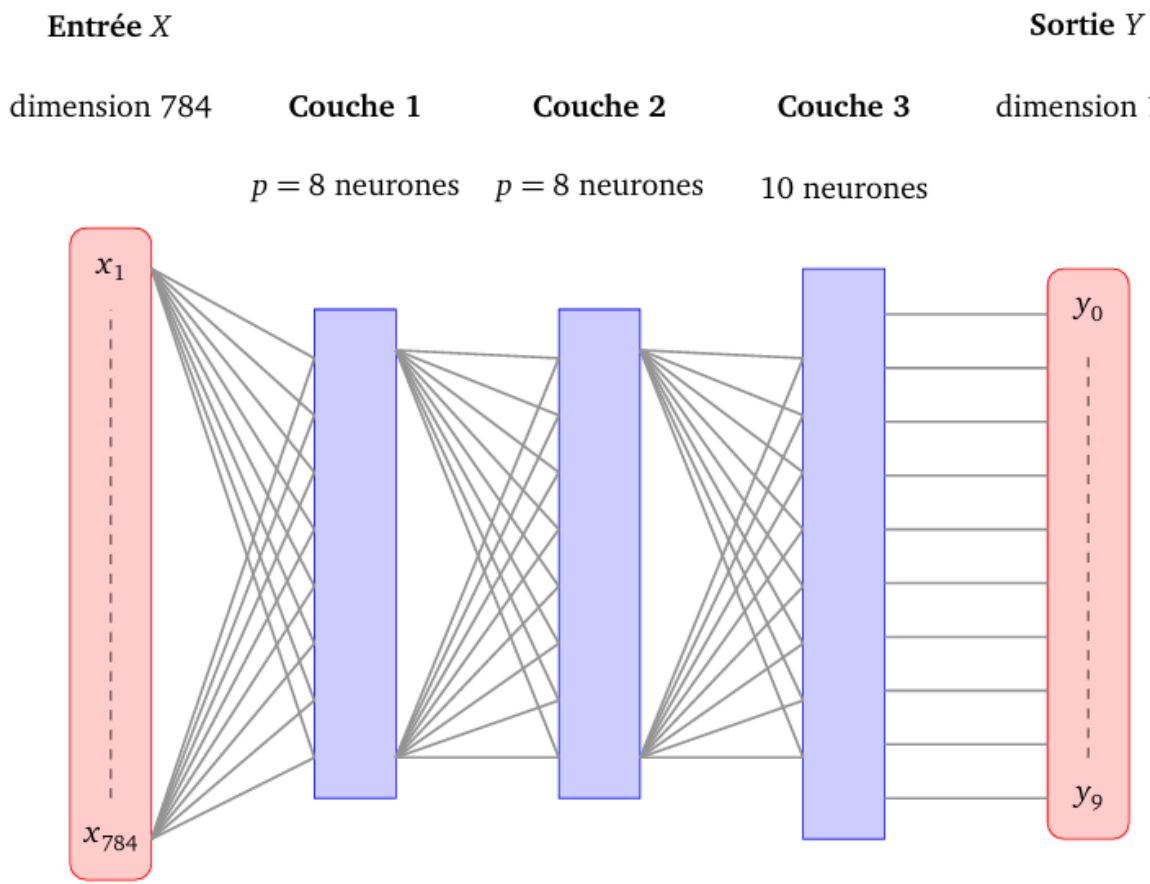
Fonction. Nous cherchons une fonction $F : \mathbb{R}^{784} \to \mathbb{R}^{10}$, qui à un vecteur-image associe une liste de probabilités, telle que $F(X_i) \simeq Y_i$ pour nos données transformées (X_i, Y_i), $i = 1, \ldots, 60\,000$.
Par exemple la fonction F, évaluée sur un vecteur-image X peut renvoyer

$$F(X) = (0.01, 0.04, 0.03, 0.01, 0.02, 0.22, 0.61, 0.02, 0.01, 0.01, 0.02).$$

Dans ce cas, le nombre le plus élevé est 0.61 au rang 6, cela signifie que notre fonction F prédit le chiffre 6 avec une probabilité de 61%, mais cela pourrait aussi être le chiffre 5 qui est prédit à 22%. Les autres chiffres sont peu probables.

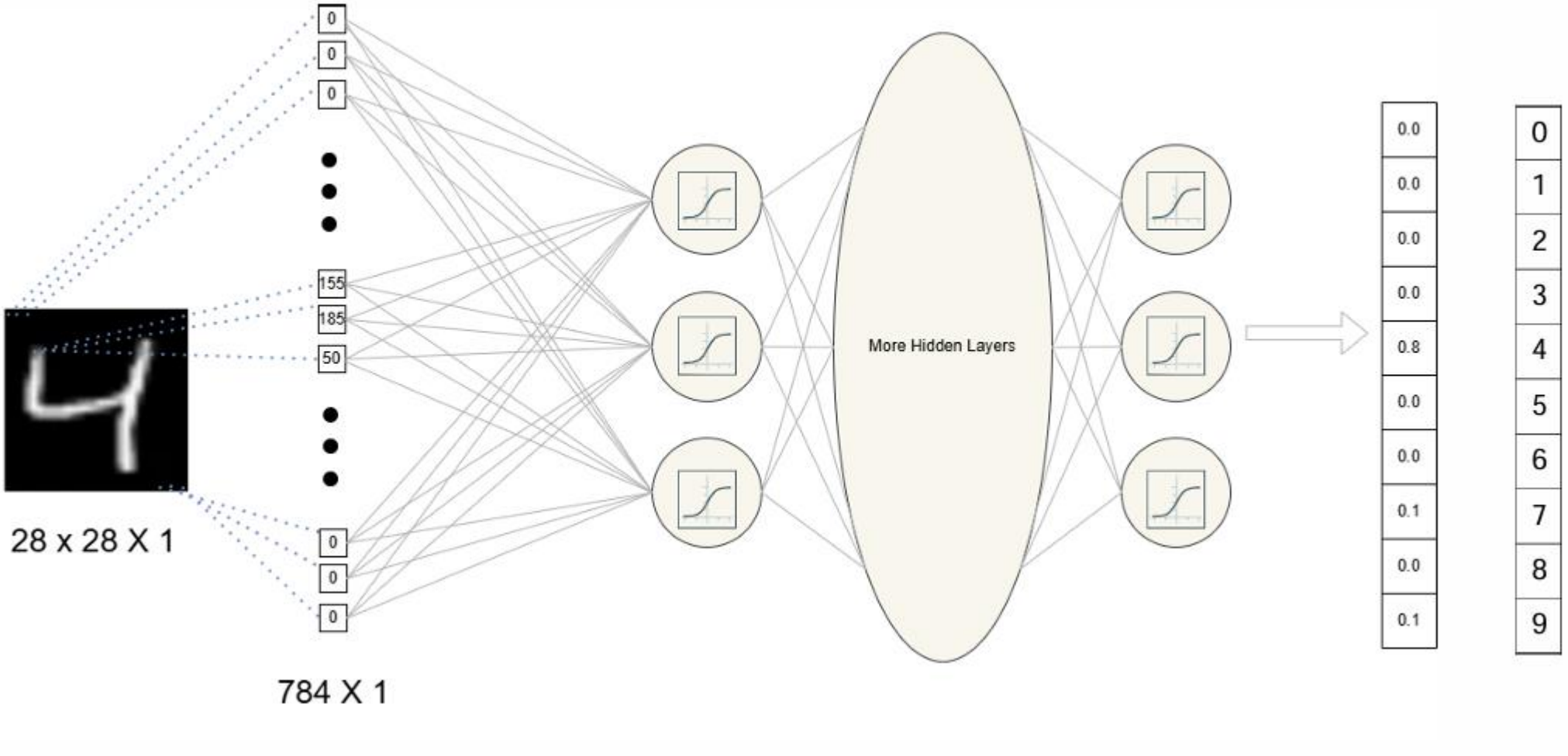
Réseau

On va construire un réseau de neurones qui produira une fonction $F : \mathbb{R}^{784} \to \mathbb{R}^{10}$. L'architecture est composée de 3 couches. En entrée nous avons un vecteur de taille 784. La première et la seconde couches sont composées chacune de $p = 8$ neurones. La couche de sortie est formée de 10 neurones, un pour chacun des chiffres.



Exemple

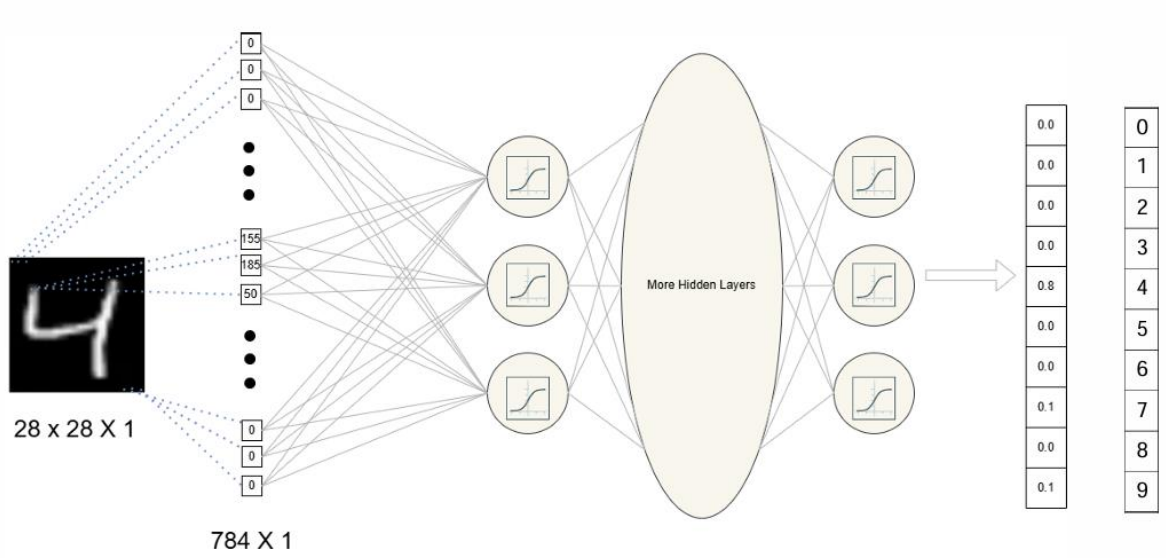
MLP pour les images



MLP pour les images

Chaque pixel est connecté à chaque neurone de la première couche :

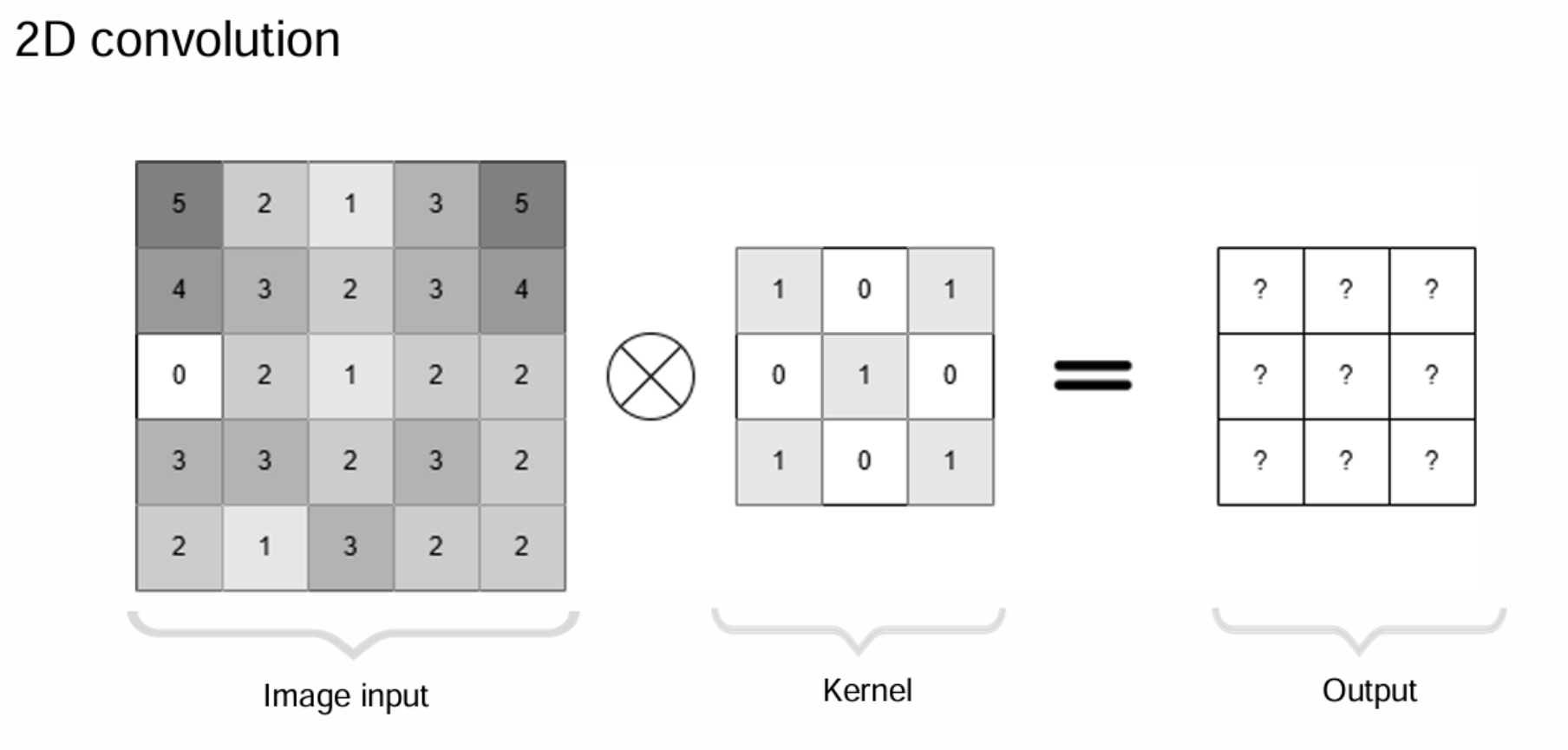
- Nombre élevé de paramètres !
- Certaines connexions ne sont pas pertinentes
- L'algorithme est trop dépendant du pixel : difficile à optimiser avec un ML



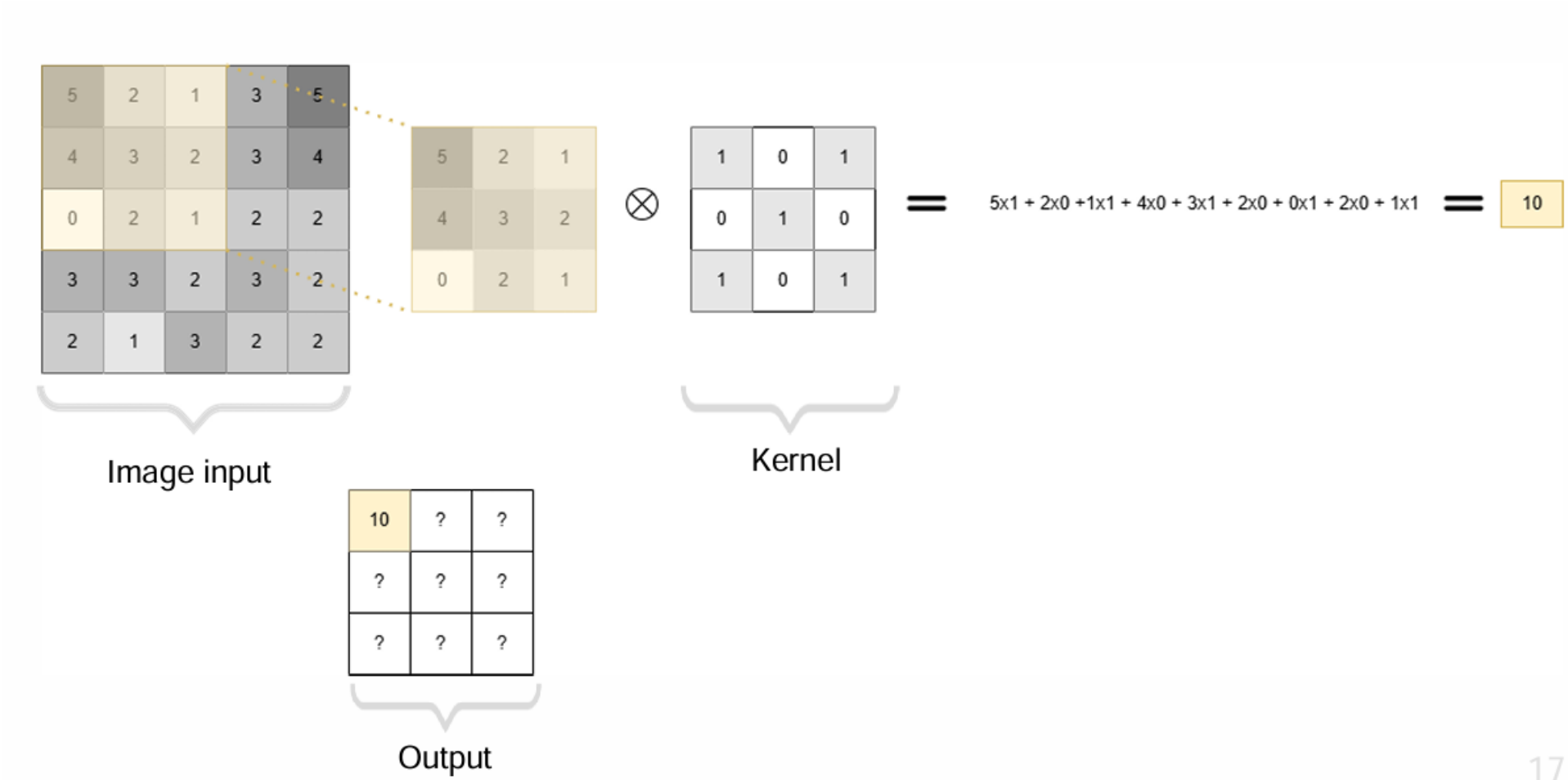
CNN

C'est là que viennent les CNN.

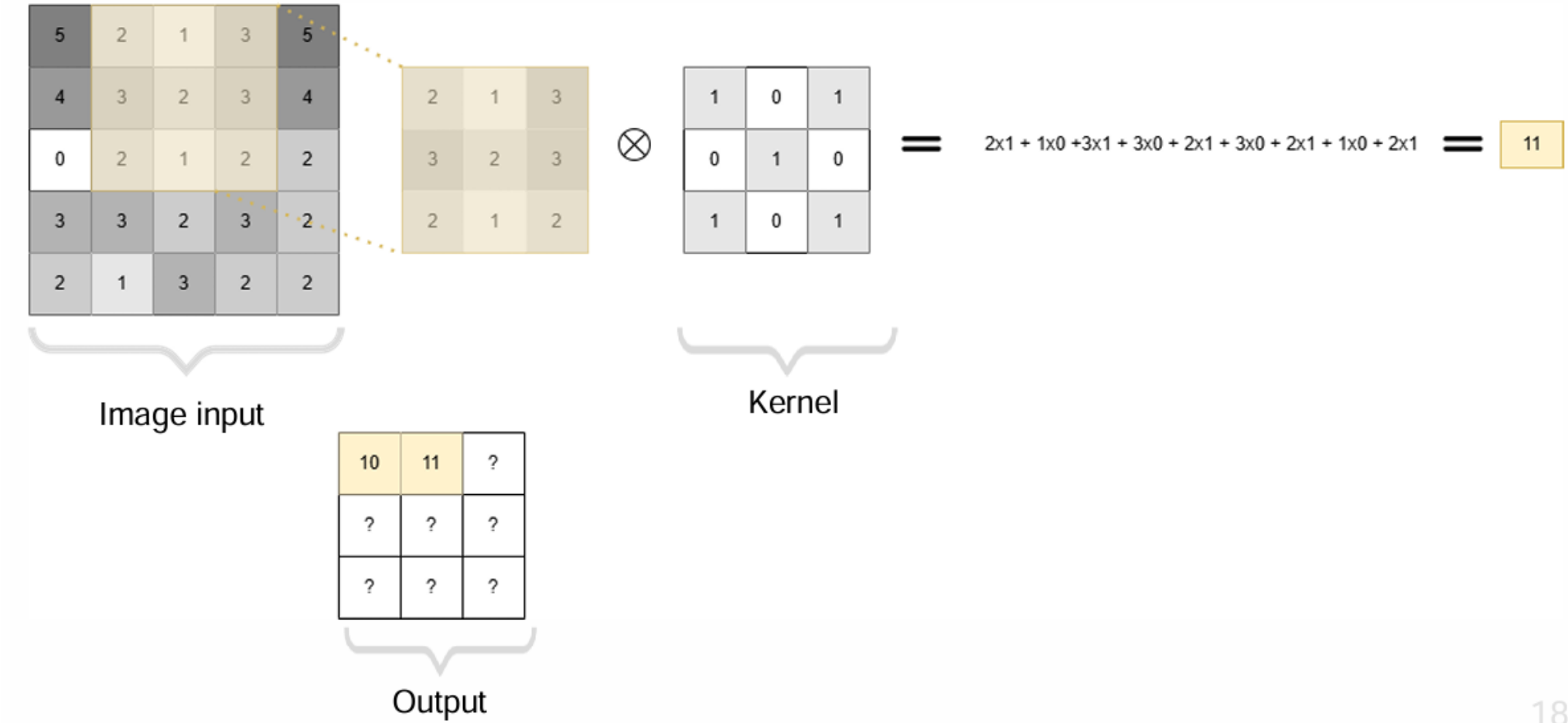
2D Convolution



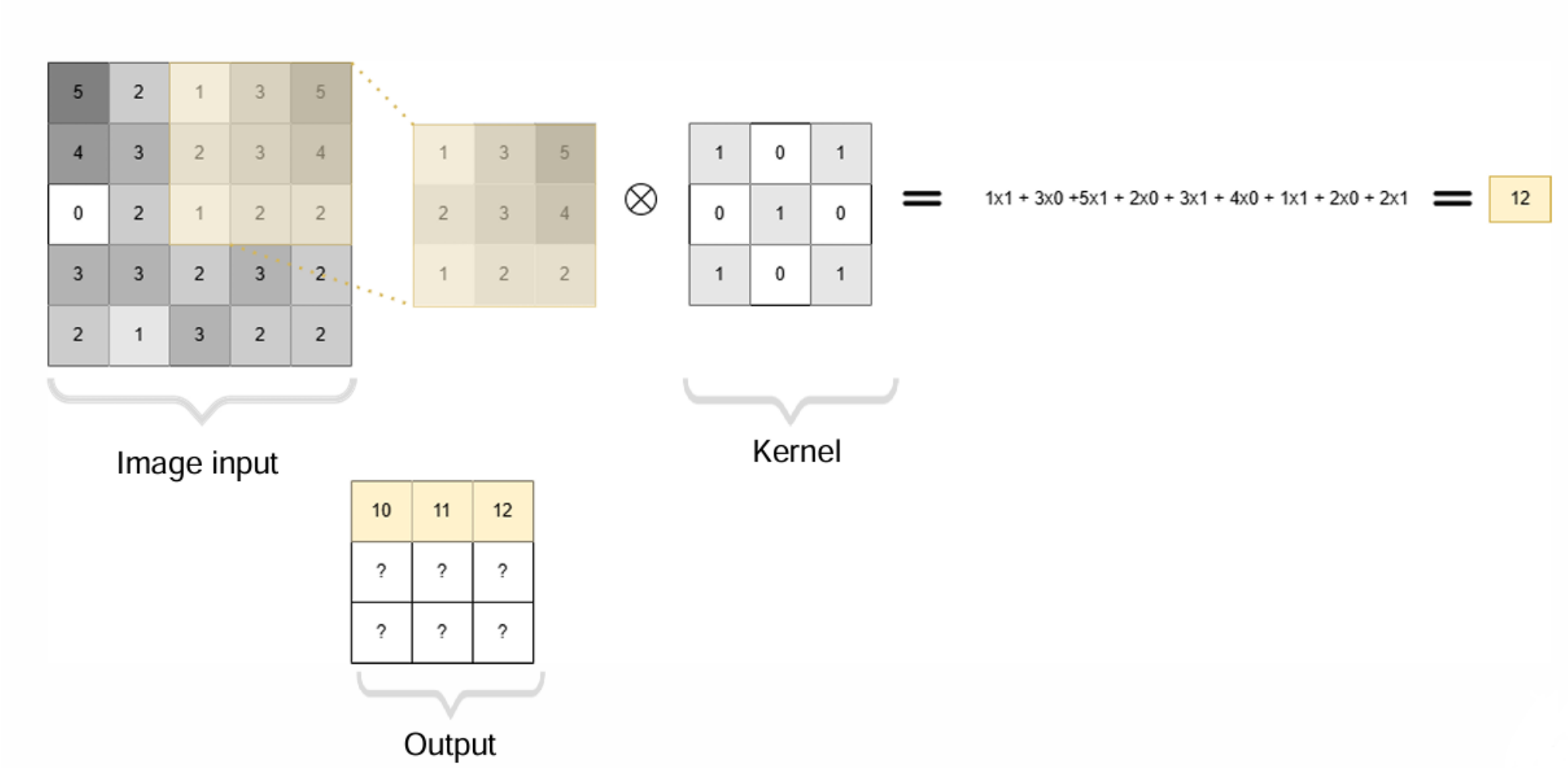
Principe de convolution



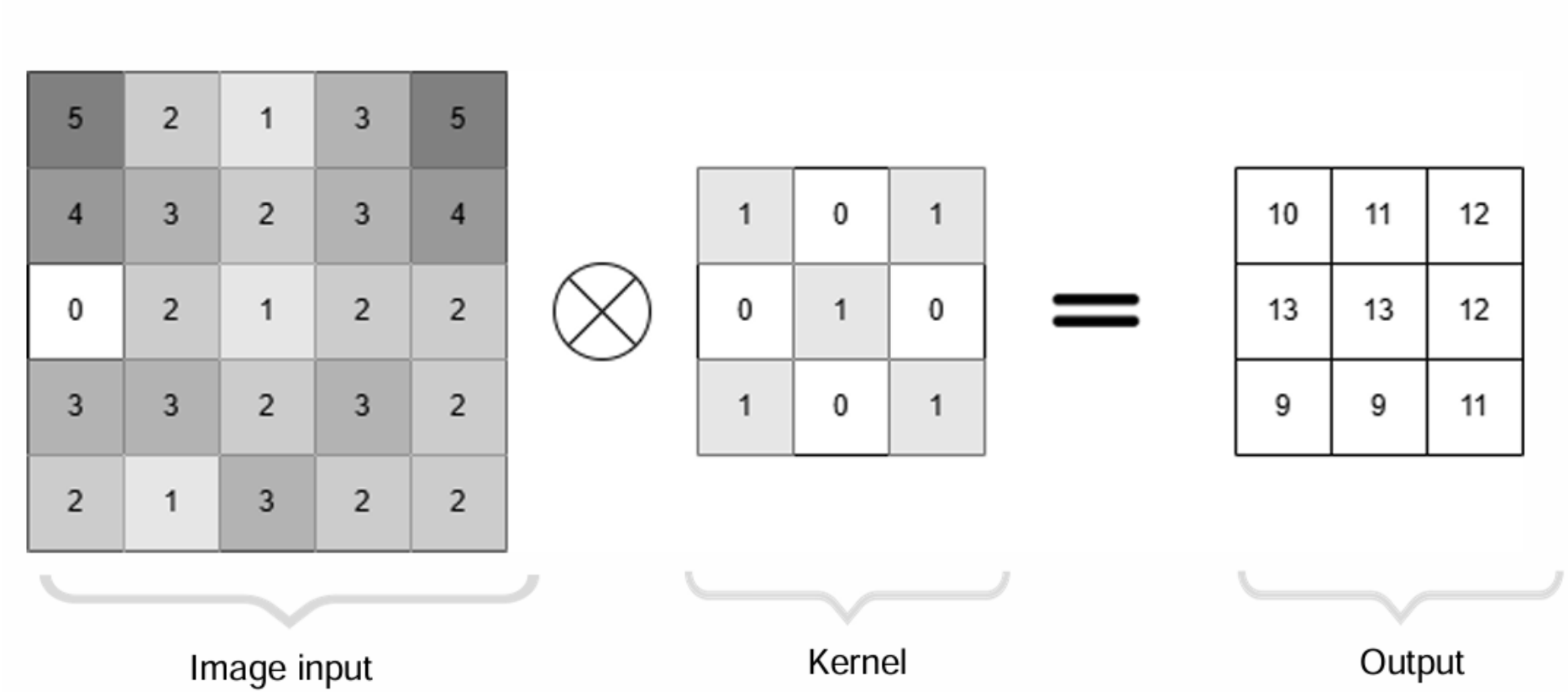
Principe de convolution



Principe de convolution



Principe de convolution



Principe de convolution

(a) Lenna (b) Horizontal edge (c) Vertical edge

Figure 4: The Lenna image and the effect of different convolution kernels.

Flou

On obtient une image floue par application du motif :

$$M = \frac{1}{9}\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}.$$

Image originale

Flou

Piqué

C'est en quelque sorte le contraire du flou ! Le motif est :

$$M = \begin{pmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{pmatrix}.$$

Voici un exemple avec l'image originale à gauche et l'image transformée à droite qui a l'air plus nette que l'originale !

Image originale

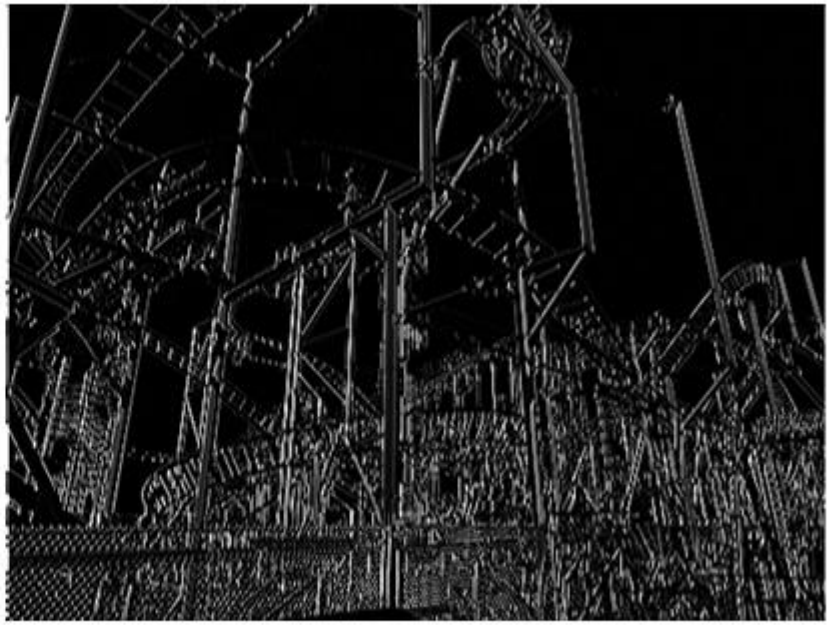
Piqué

Verticales

Le motif suivant renforce les lignes verticales :

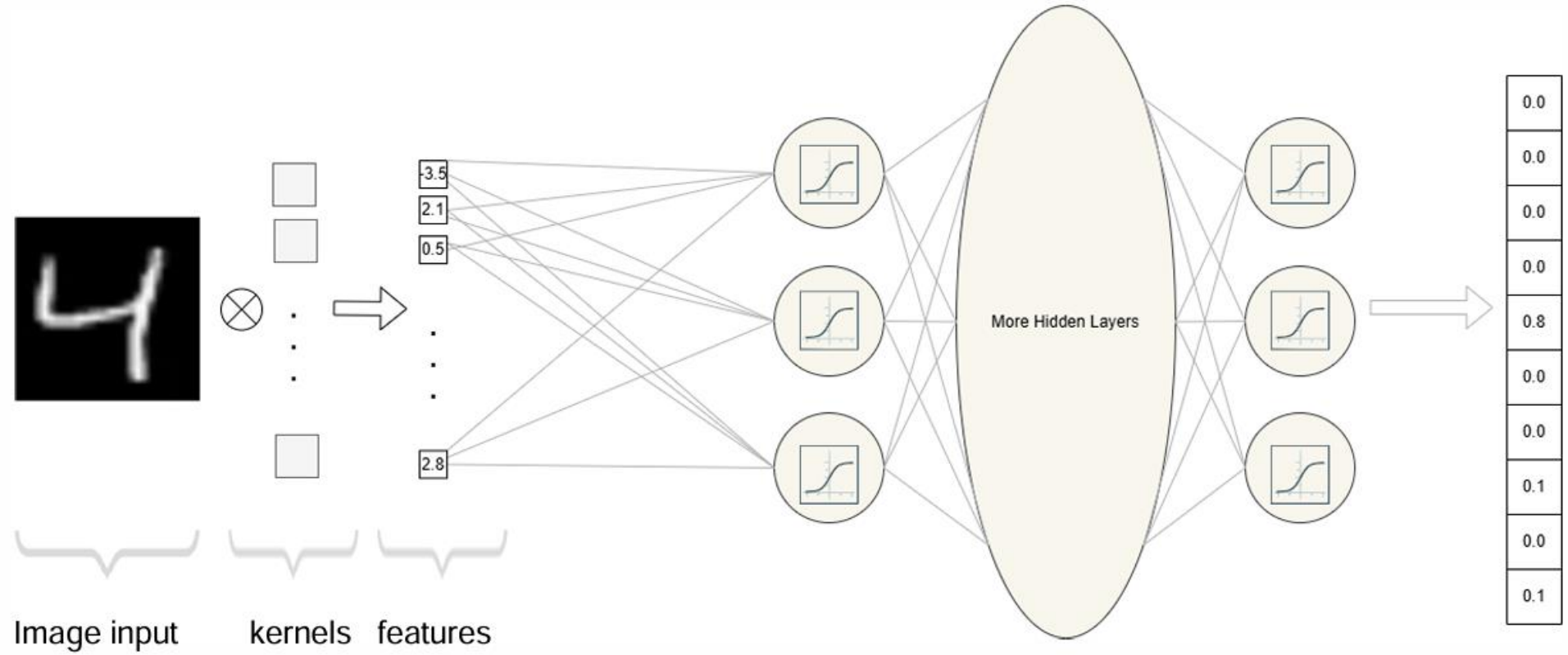
$$M = \begin{pmatrix} -1 & 2 & -1 \\ -1 & 2 & -1 \\ -1 & 2 & -1 \end{pmatrix}.$$

Image originale

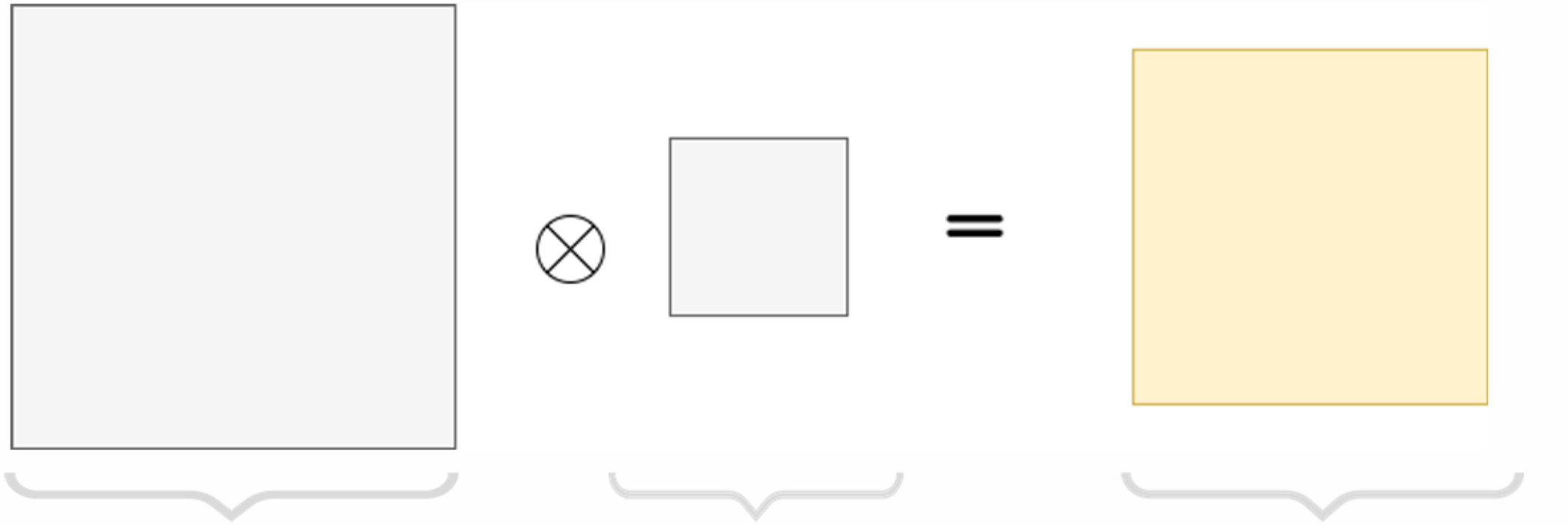


Verticales

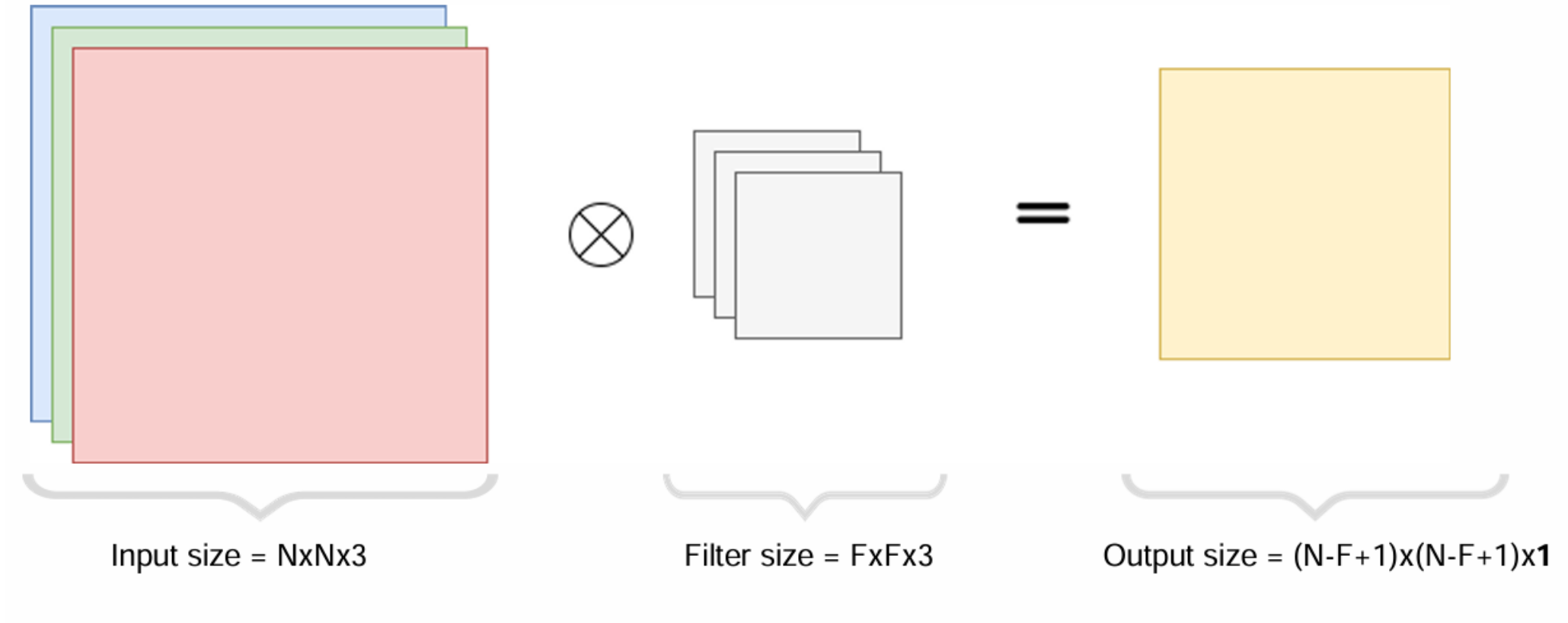
CNN : une méthode pour améliorer les entrées d'images dans les MLP



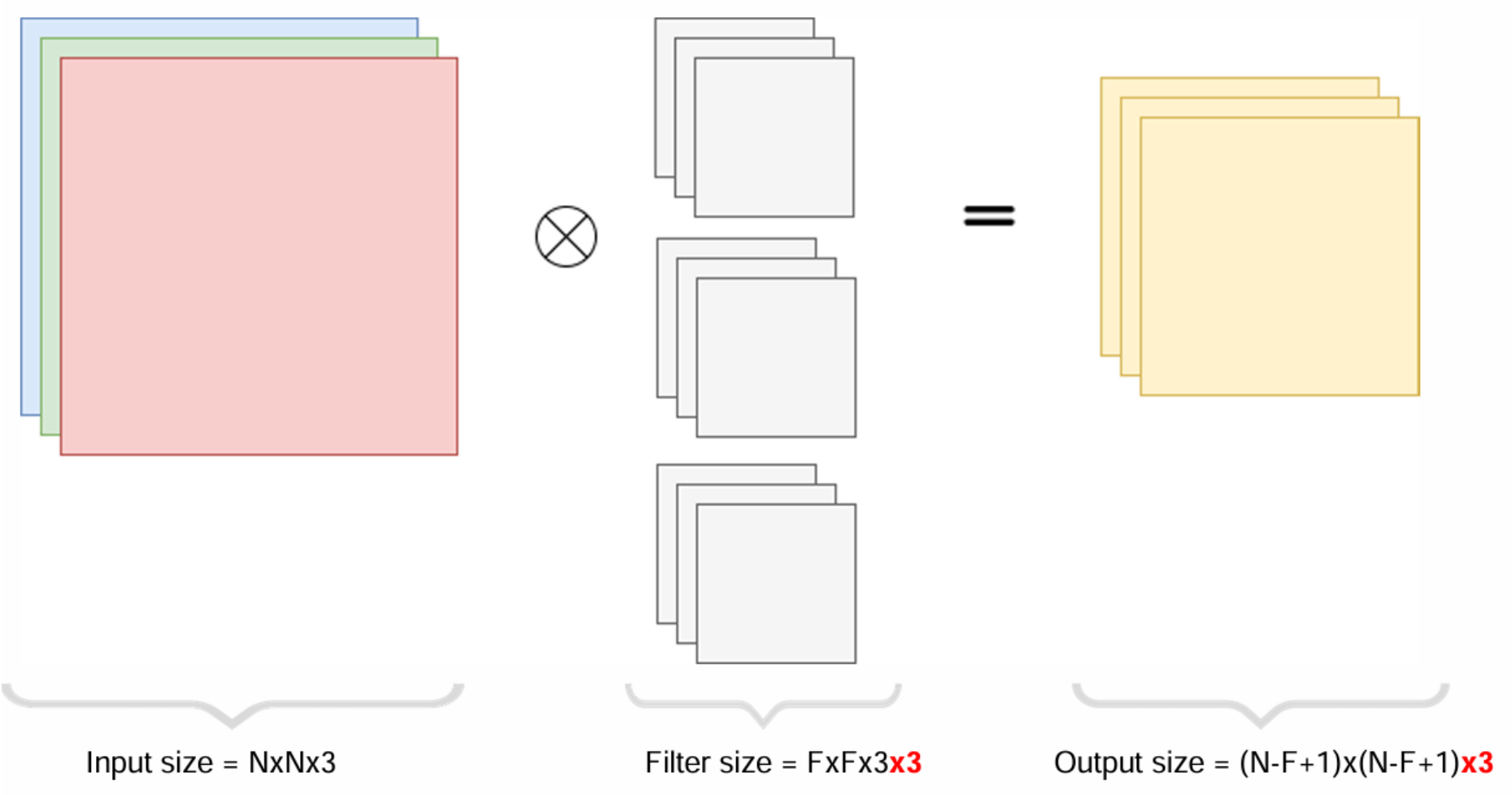
Dimensions



Dimensions

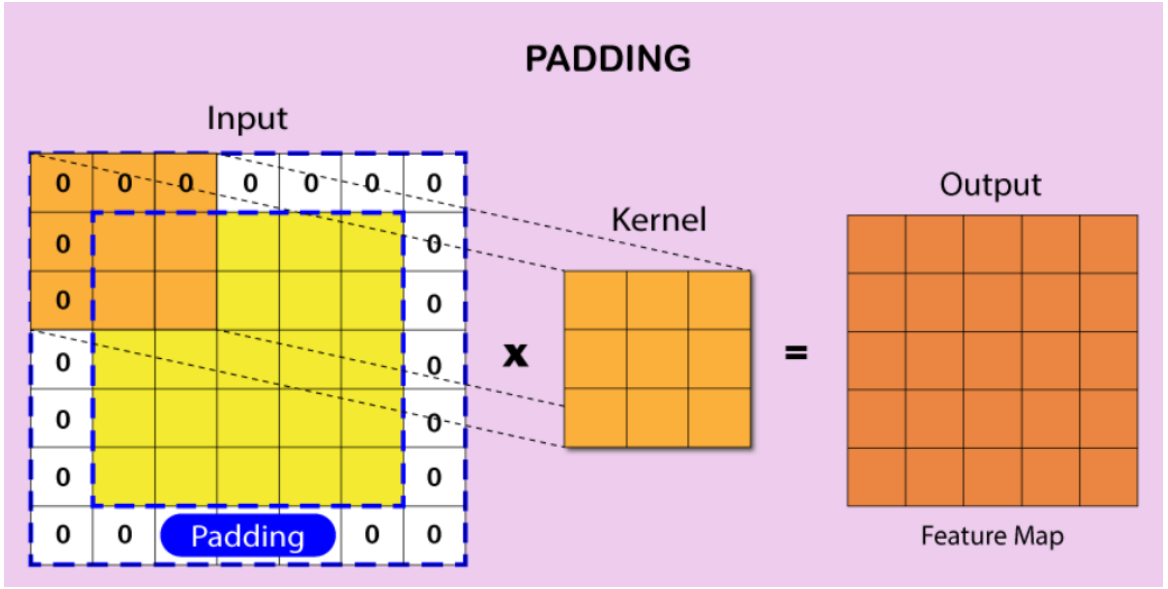


Dimensions


Input size = NxNx3
Filter size = FxFx3x3
Output size = (N-F+1)x(N-F+1)x3

Paramètres de convolution

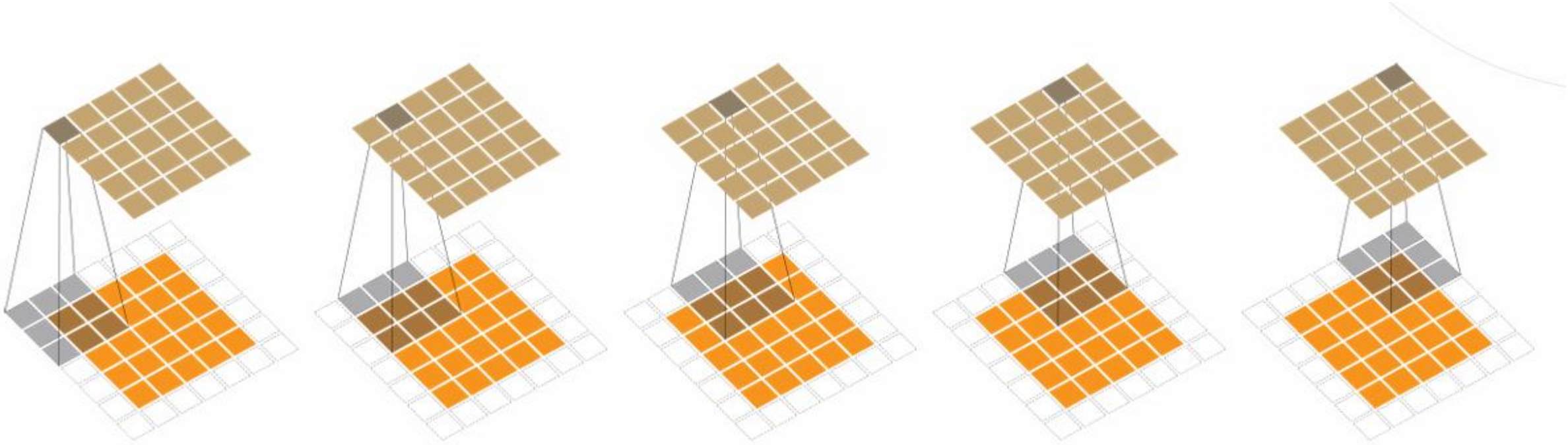
Paramètres d'une couche convolutive : **Padding**



Paramètres de convolution: Padding

Le padding est utilisé pour rendre la dimension de la sortie égale à celle de l'entrée en ajoutant des zéros au cadre d'entrée de la matrice. Le remplissage permet au noyau d'occuper plus d'espace pour couvrir l'image et il est précis pour l'analyse des images. Grâce au remplissage, les informations sur les bords des images sont également préservées de la même manière qu'au centre de l'image.

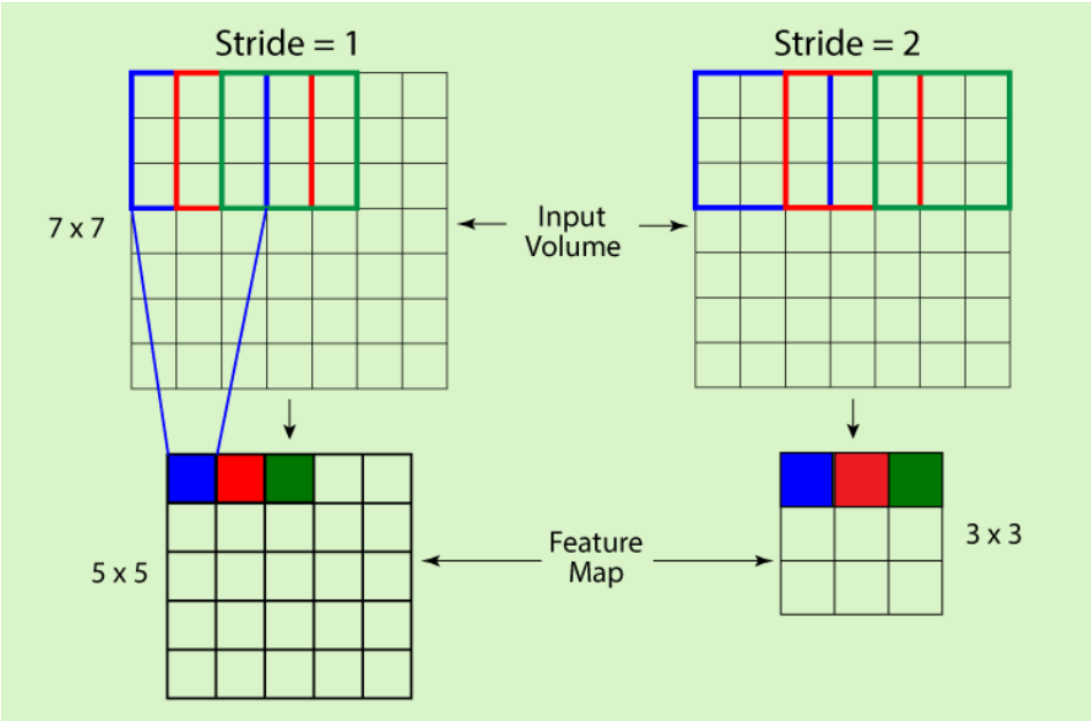
Paramètres de convolution: Padding



Input size = N
Filter size = F
Padding = P
Output size = N-F+2P+1

Paramètres de convolution: Stride

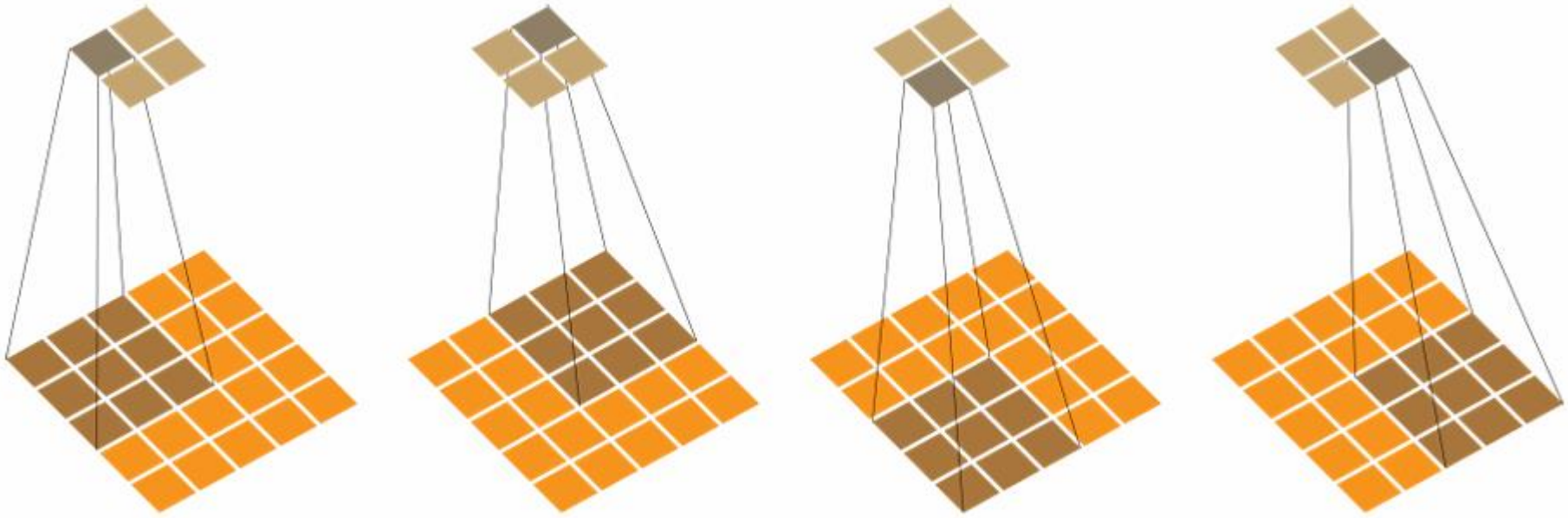
Paramètres d'une couche convolutive : **Stride**



Paramètres de convolution: Stride

Stride contrôle la façon dont le filtre convolue sur l'entrée, c'est-à-dire le nombre de pixels qui se déplacent sur la matrice d'entrée. Si stride est fixé à 1, le filtre se déplace sur 1 pixel à la fois et si stride est fixé à 2, le filtre se déplace sur 2 pixels à la fois. Plus la valeur de stride est élevée, plus le résultat sera petit et vice versa.

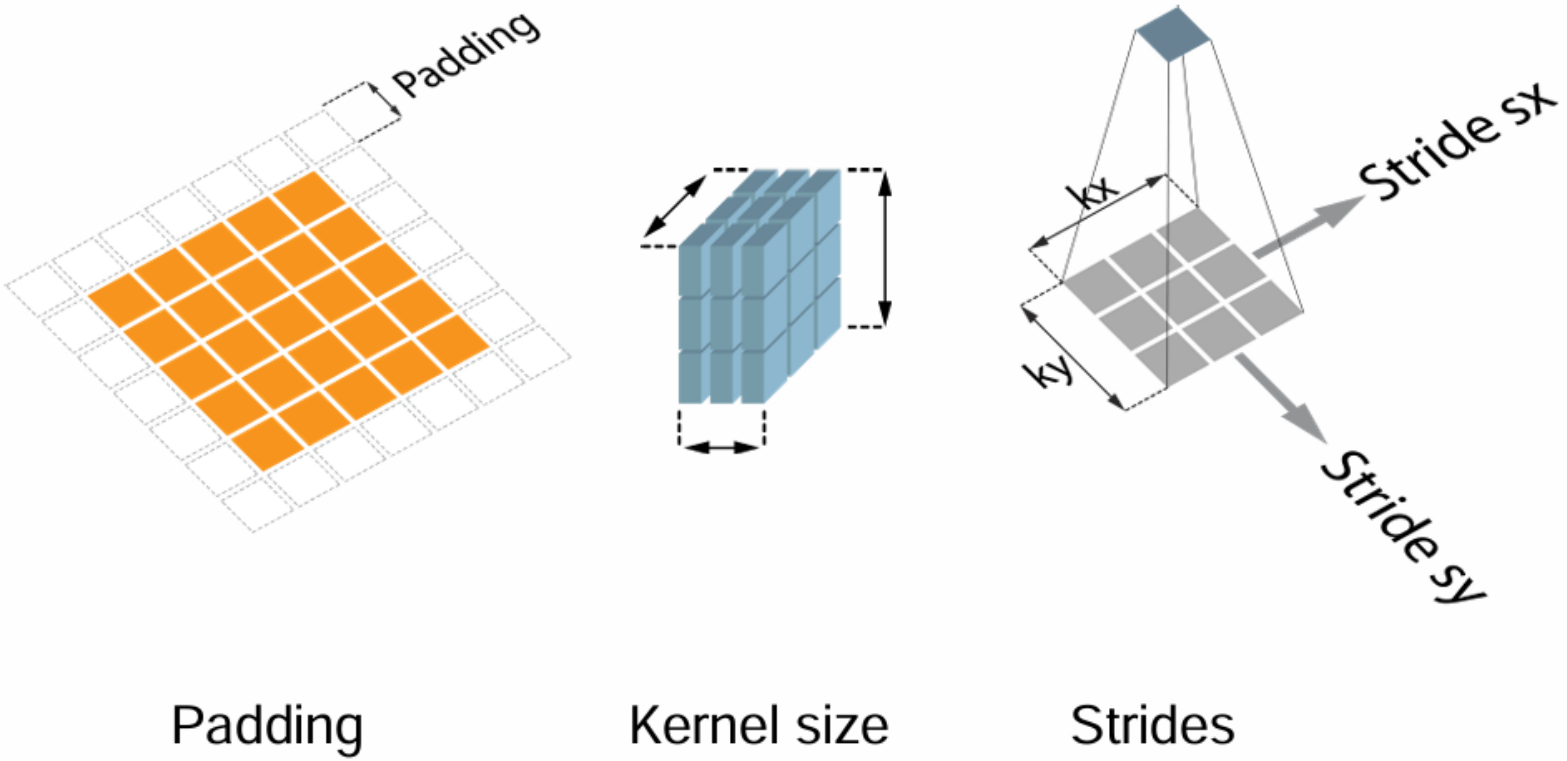
Paramètres de convolution: Stride



Input size = N
Filter size = F
Padding = P
Stride = S

Output size $= \frac{N-F+2P}{S} + 1$

Paramètres de convolution



Padding Kernel size Strides

Noyaux de convolution

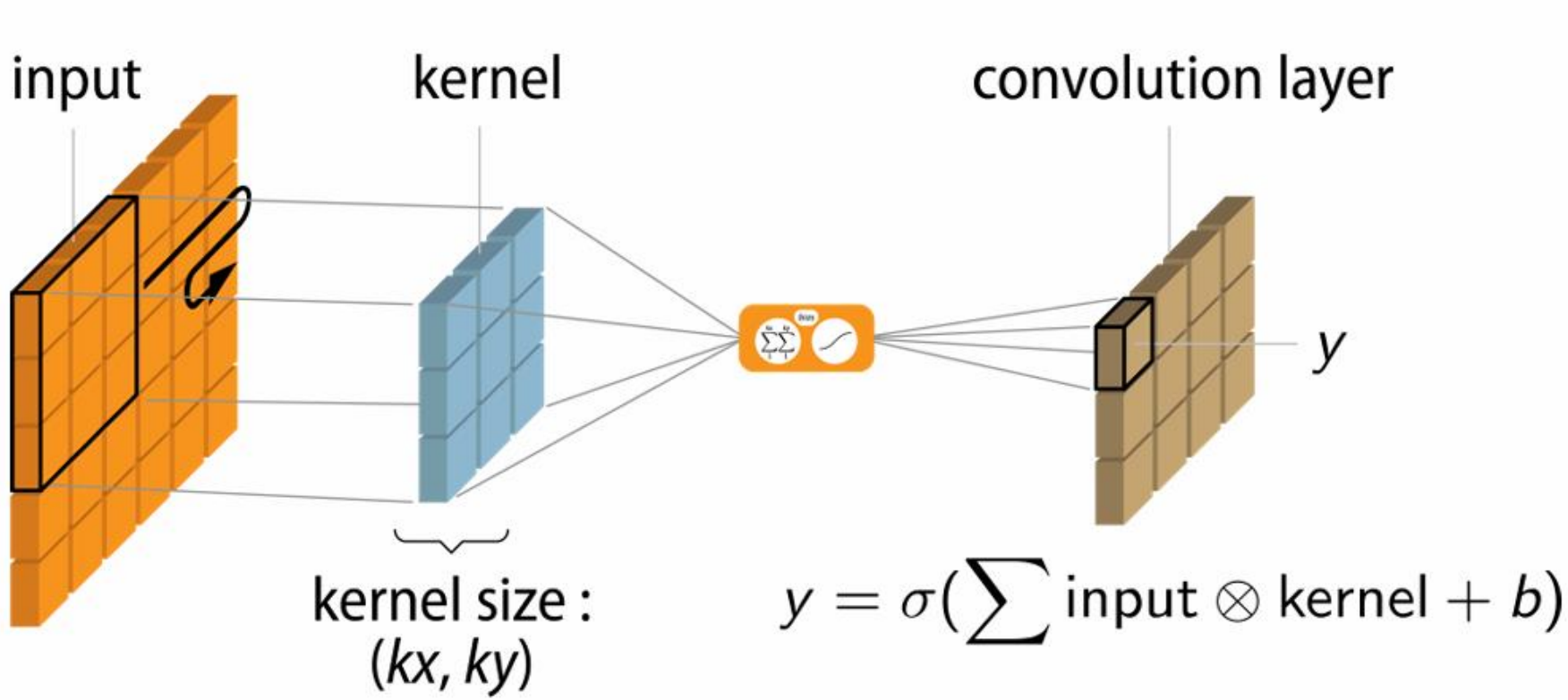
Les noyaux de convolution sont d'excellents détecteurs de caractéristiques
MAIS :

- Il est difficile d'énumérer toutes les caractéristiques.
- Et si certaines caractéristiques dépassent l'imagination humaine ?

Idée : Apprendre les noyaux de convolution à la place.

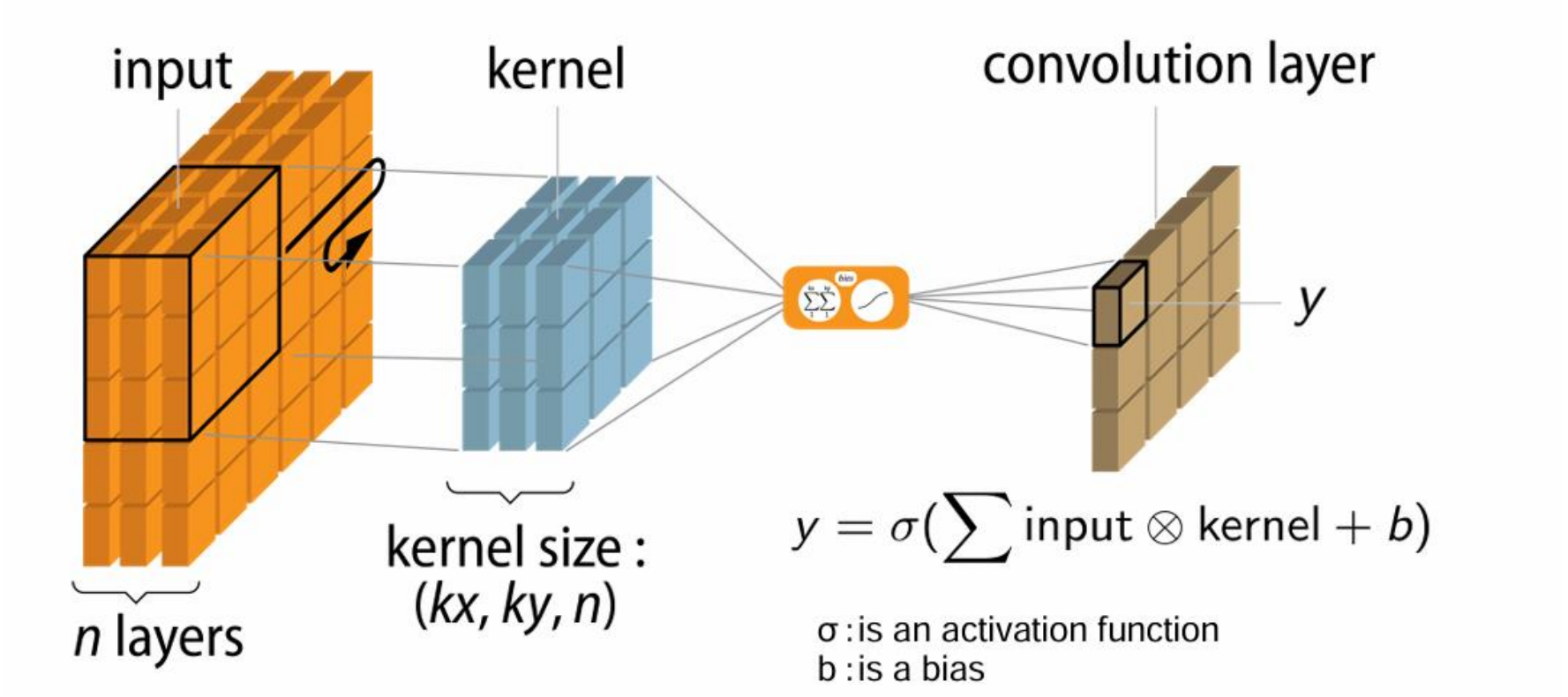
= Entraîner un modèle capable d'apprendre quels sont les noyaux à utiliser !

Couches convolutives : entrée à 1 canal



Nombre de paramètres pour une couche convolutive : **kx . ky + 1**

Couches convolutives : 3 canaux d'entrée



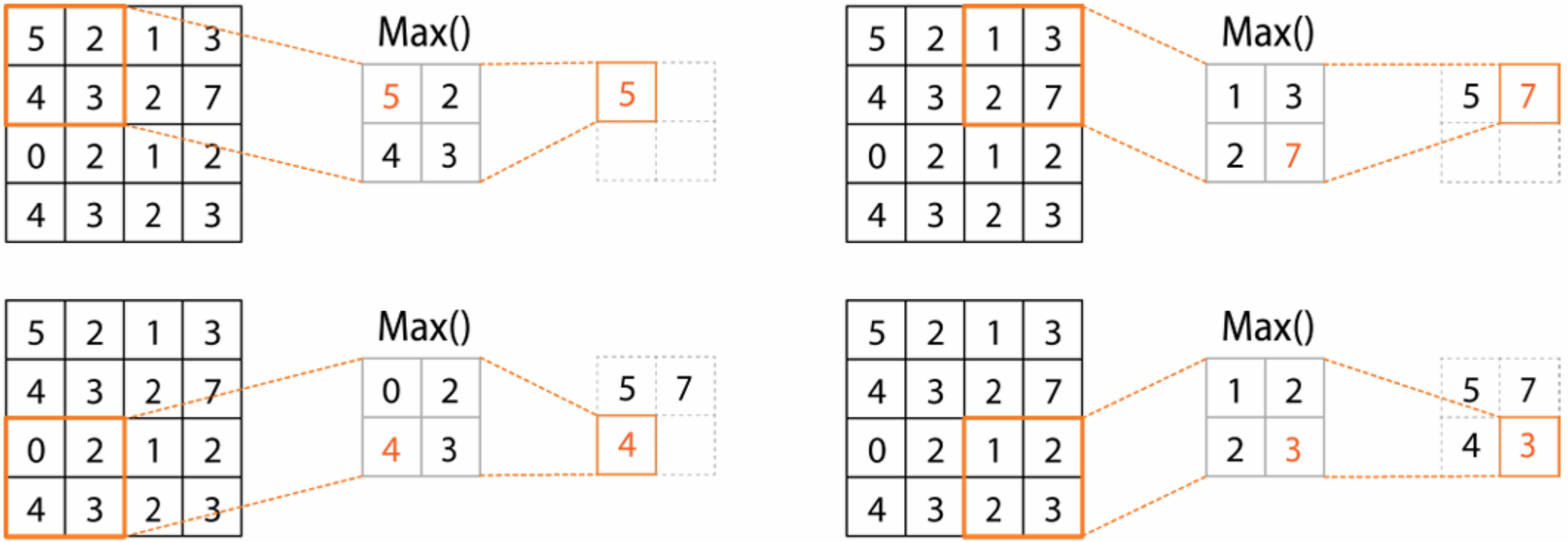
Si nous voulons générer **n** couches convolutives, nous avons besoin de **n** noyaux convolutifs,

Max-pooling and Average pooling

Max-pooling : il sélectionne le maximum d'éléments de la carte. La couche de regroupement maximal qui en résulte contient les caractéristiques importantes de la carte. C'est l'approche la plus courante car elle donne de meilleurs résultats.

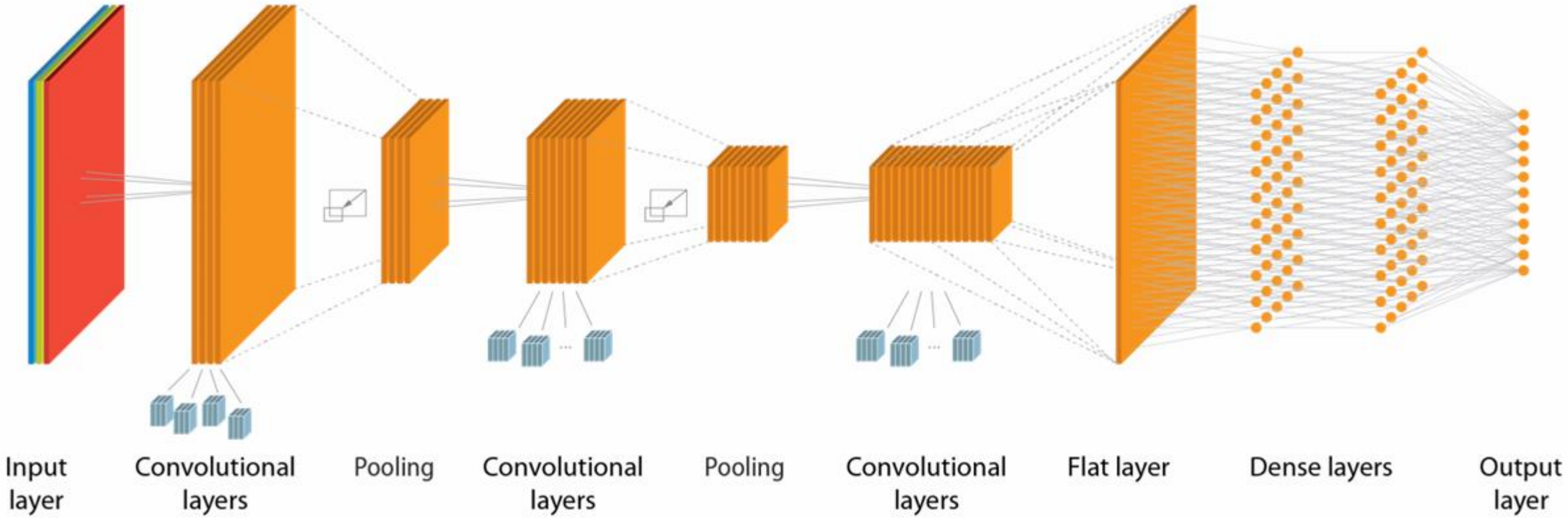
Average pooling : elle implique le calcul de la moyenne pour chaque parcelle de la carte.

Max Pooling



Il est possible de définir la taille de la fenêtre, le padding et les strides.
Par défaut, les strides correspondent à la taille de la fenêtre.
Une fenêtre (2,2) génère une image deux fois plus petite.

CNN


Input layer
Convolutional layers
Pooling
Convolutional layers
Pooling
Convolutional layers
Flat layer
Dense layers
Output layer

Application

FIN