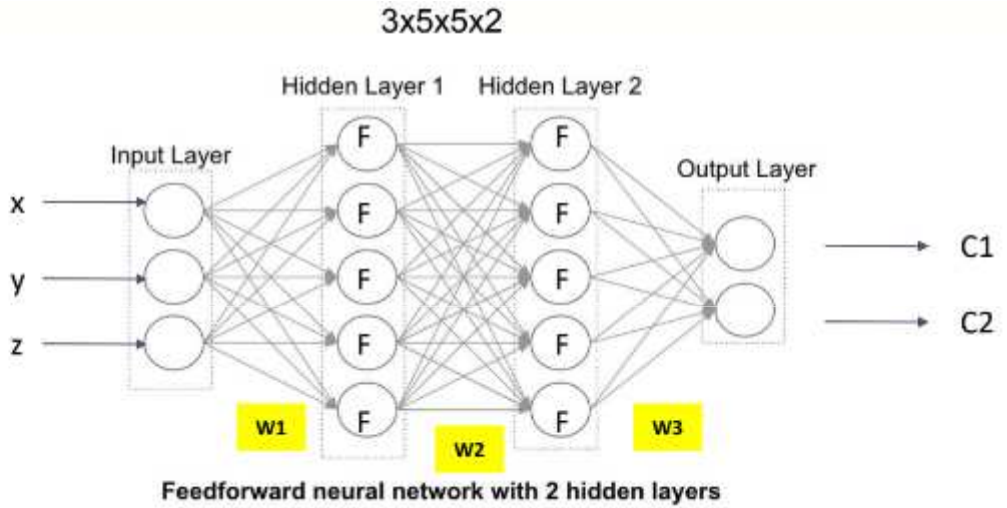
Chapitre III: Réseaux de neurones convolutifs: CNN

MP2SDD

A.U: 2023-2024

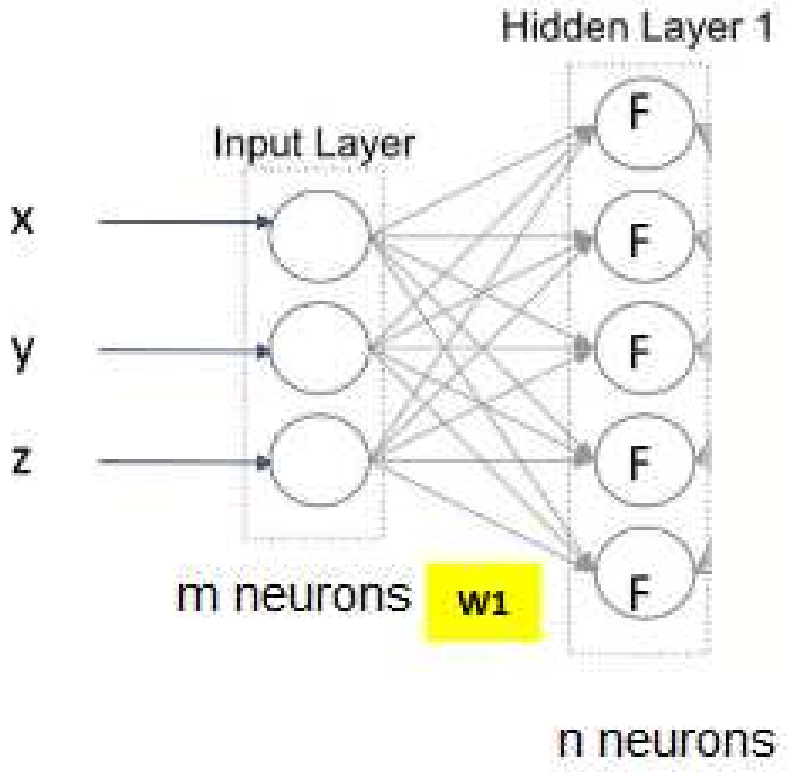
ANN: Rappel



Feedforward neural network with 2 hidden layers

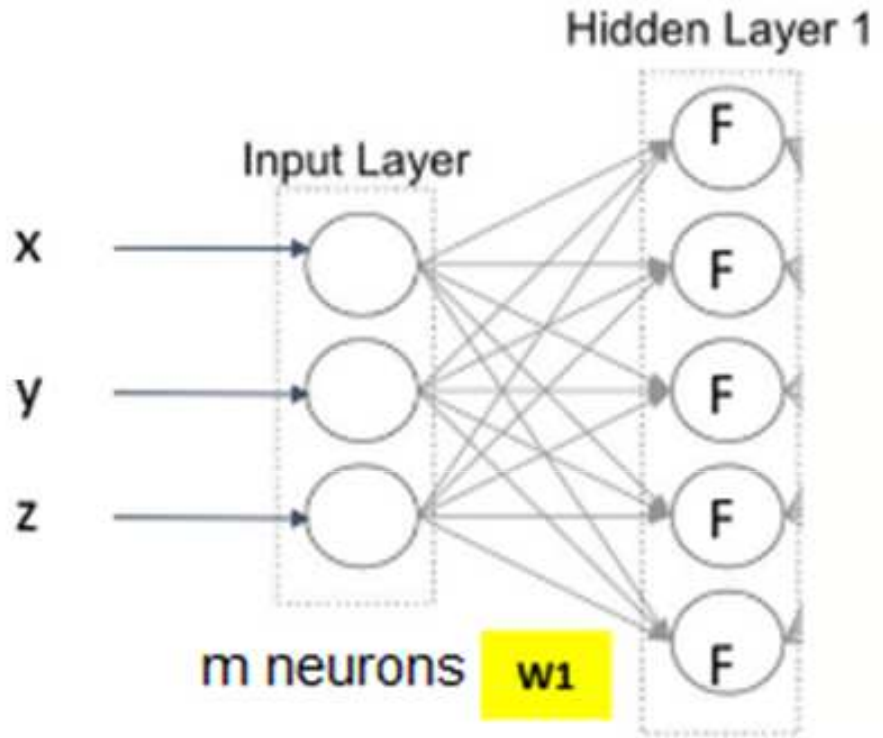
[x,y,z]: Input Vector
W1: Weight Matrix of Input Layer
W2: Weight Matrix of Hidden Layer
W3: Weight Matrix of Output Layer
F: Activation Function
C1: Class Output 1
C2: Class Output 2

ANN: nombre de paramètres



size(Input Layer)=m
size(Hidden Layer 1)=n
shape(w1)=(n,m)
size(w1)=nxm

ANN appliqué sur une image



Input : image couleur (RVB) de dimension (256,256)

reshape image to a vector

size(Input vector)= 256x256x3=196608

m= 196608

soit n=100000

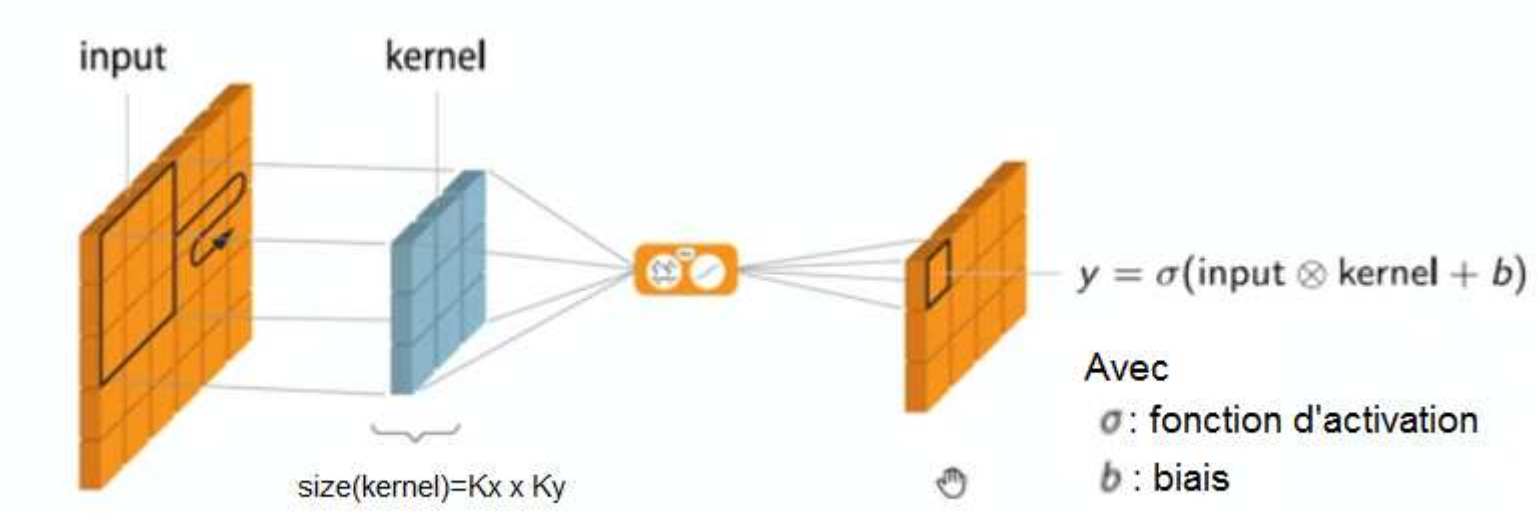
nbre de paramètres=196608 x 100000

Trop de paramètres !!!

Convolution

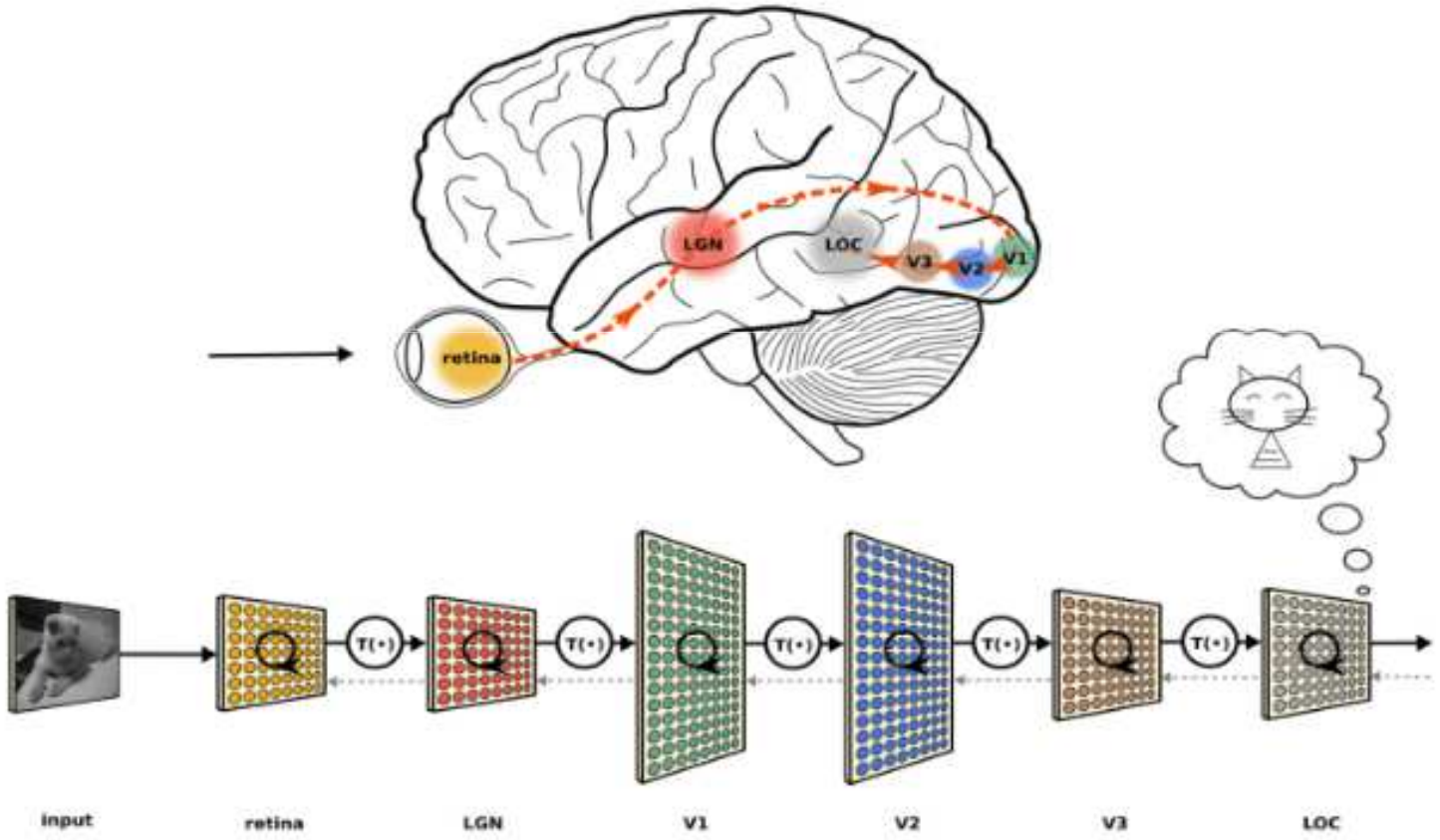
- Hypothèses:
  - les pixels proches ont presque la **même valeur**
  - Avons-nous vraiment besoin de neurones séparés pour chaque pixel ?
  - **Réutilisons les paramètres** de notre réseau

Idée de base derrière la convolution

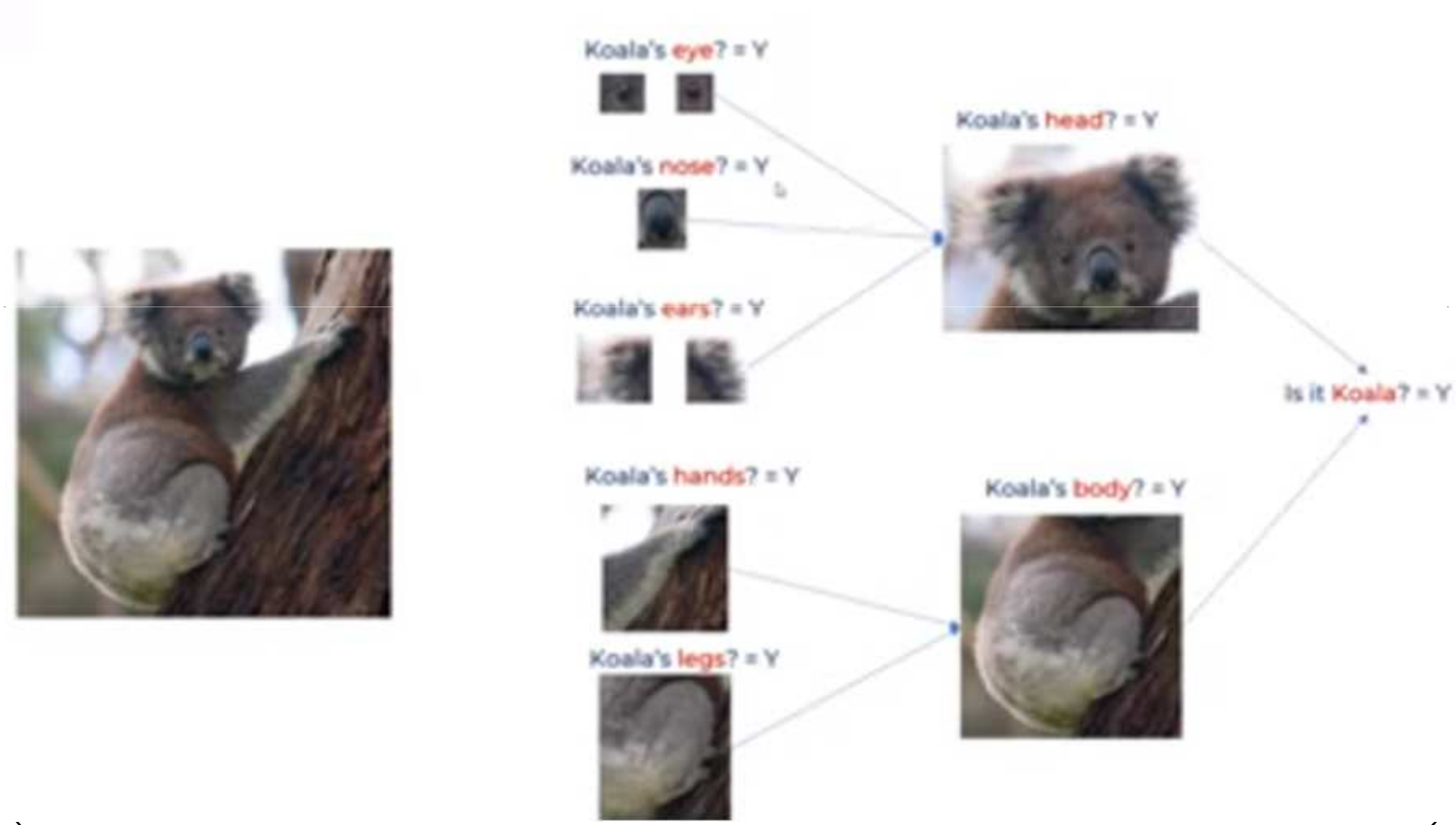


Nombre de paramètres = Kx x Ky +1

Inspiration du cortex visuel



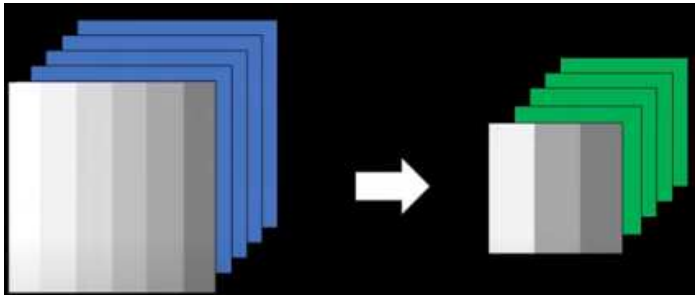
Inspiration du cortex visuel

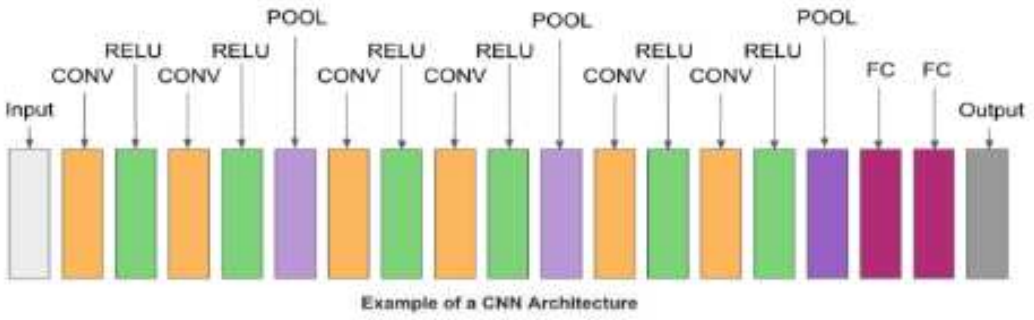


Architecture du CNN

CNN définit des couches spécifiques

- Couche convolutive (CONV)
  - Opérateur de convolution
  - Représentation linéaire (somme du produit)
- Couche ReLU (RELU)
  - Optimisation de la Descente de Dégradé
  - Introduire la représentation non linéaire
- Couche de pooling (POOL)
  - max pooling
  - Average pooling
- Couche complètement connectée (FC)
  - Émuler les classificateurs MLP
  - Tâche de classification



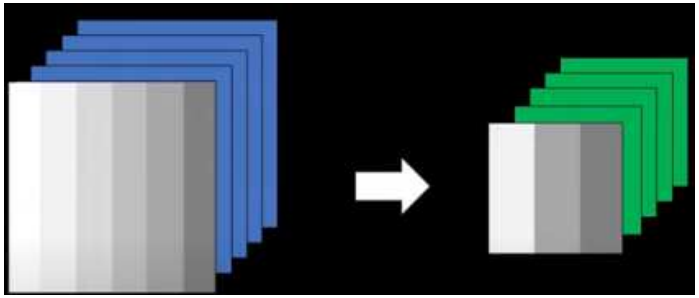


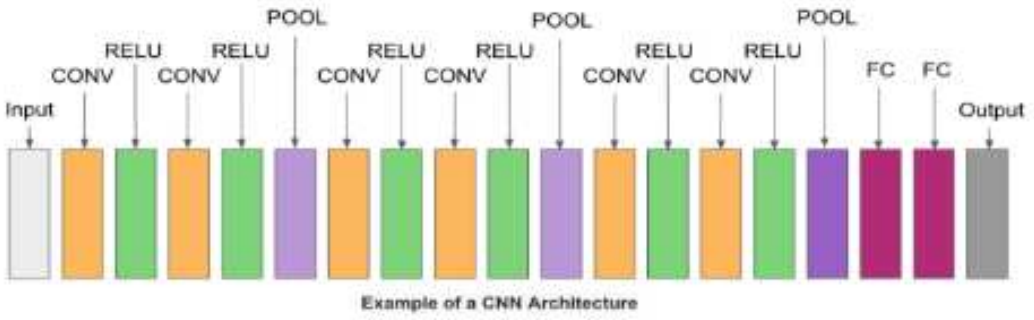
Example of a CNN Architecture

Architecture du CNN

CNN définit des couches spécifiques

- **Couche convolutive (CONV)**
  - Opérateur de convolution
  - Représentation linéaire (somme du produit)
- Couche ReLU (RELU)
  - Optimisation de la Descente de Dégradé
  - Introduire la représentation non linéaire
- Couche de pooling (POOL)
  - max pooling
  - Average pooling
- Couche complètement connectée (FC)
  - Émuler les classificateurs MLP
  - Tâche de classification

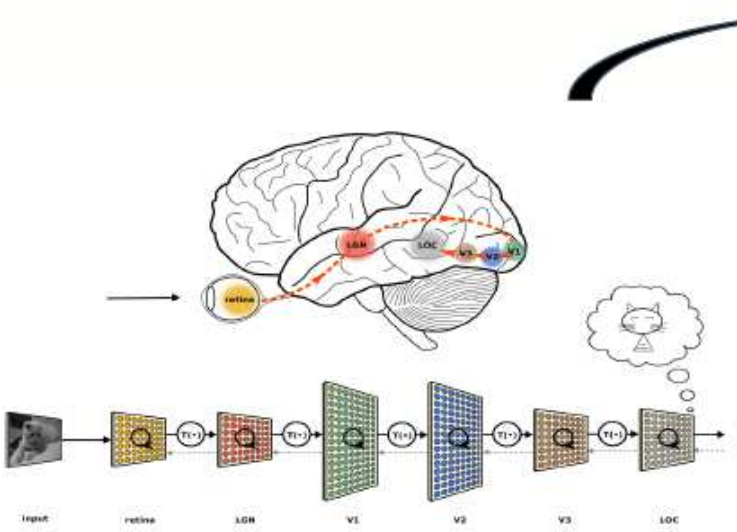


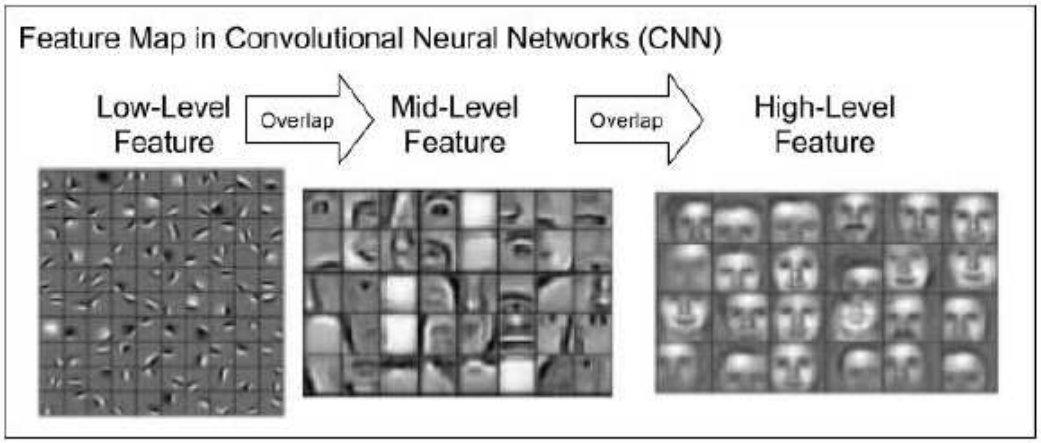


Example of a CNN Architecture

Couches de convolution

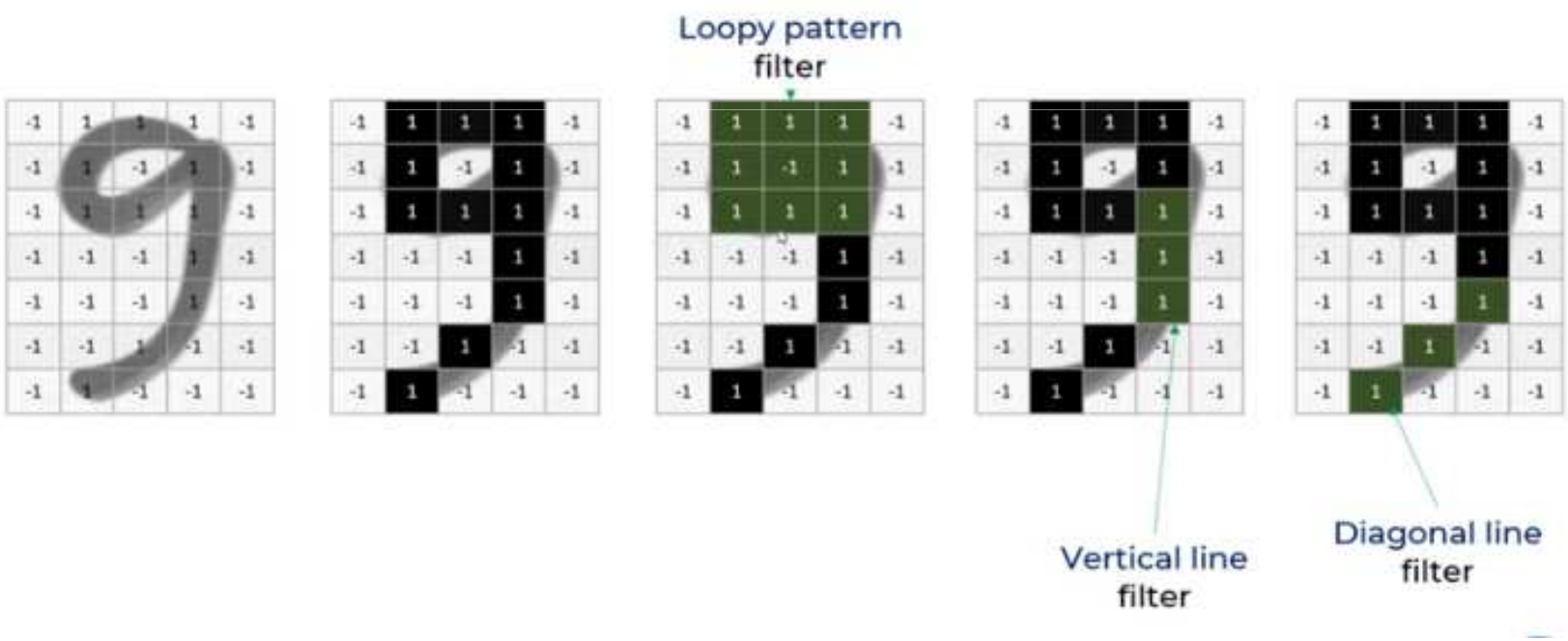
- Plusieurs couches pour extraire les « Features » d'une image:
  - ❑ features simples
  - ❑ feature complexe
  - ❑ features sophistiquées





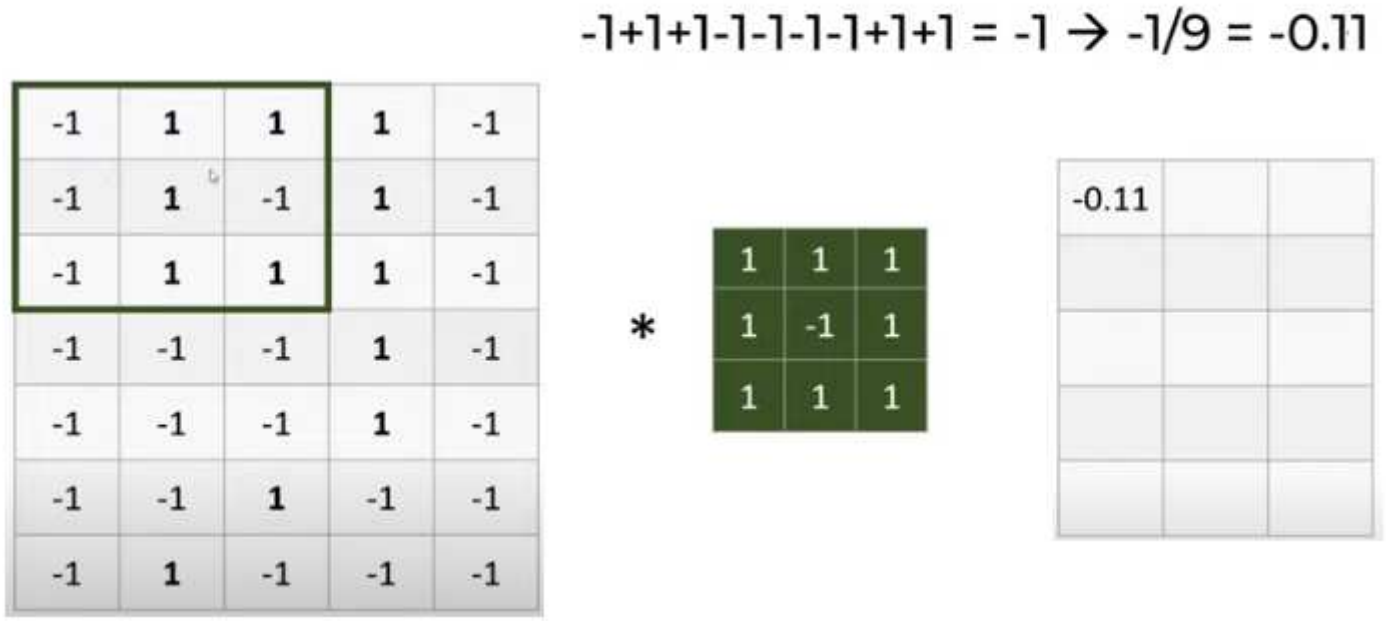
Couches de convolution

- Filtre: détecter les caractéristiques de l'image



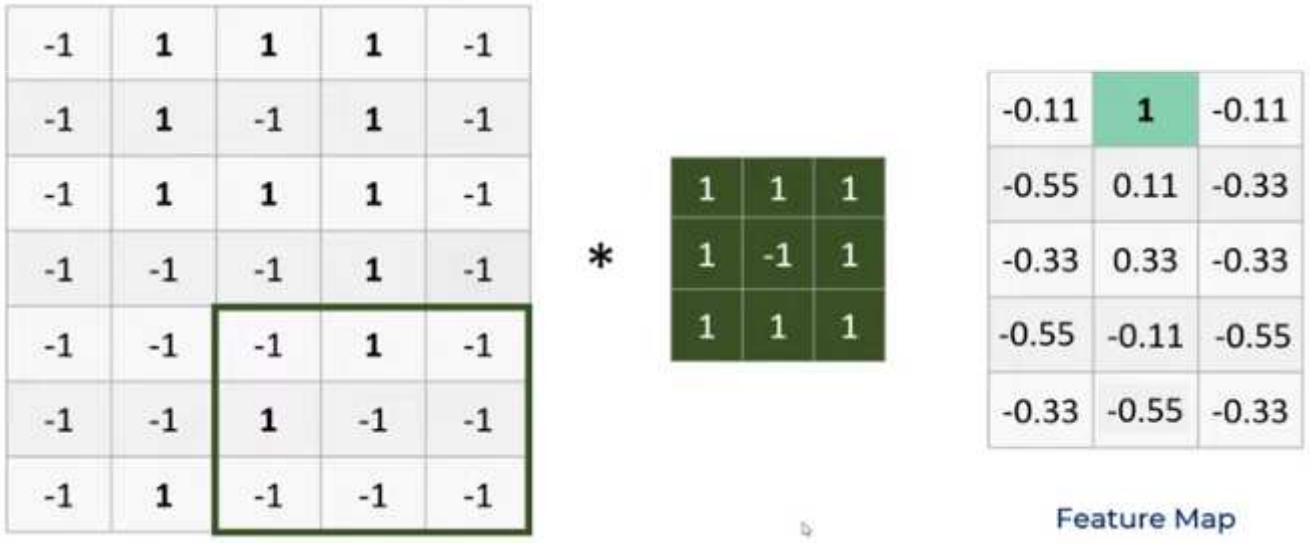
Couches de convolution

- Appliquant le premier filtre sur l'image d'origine

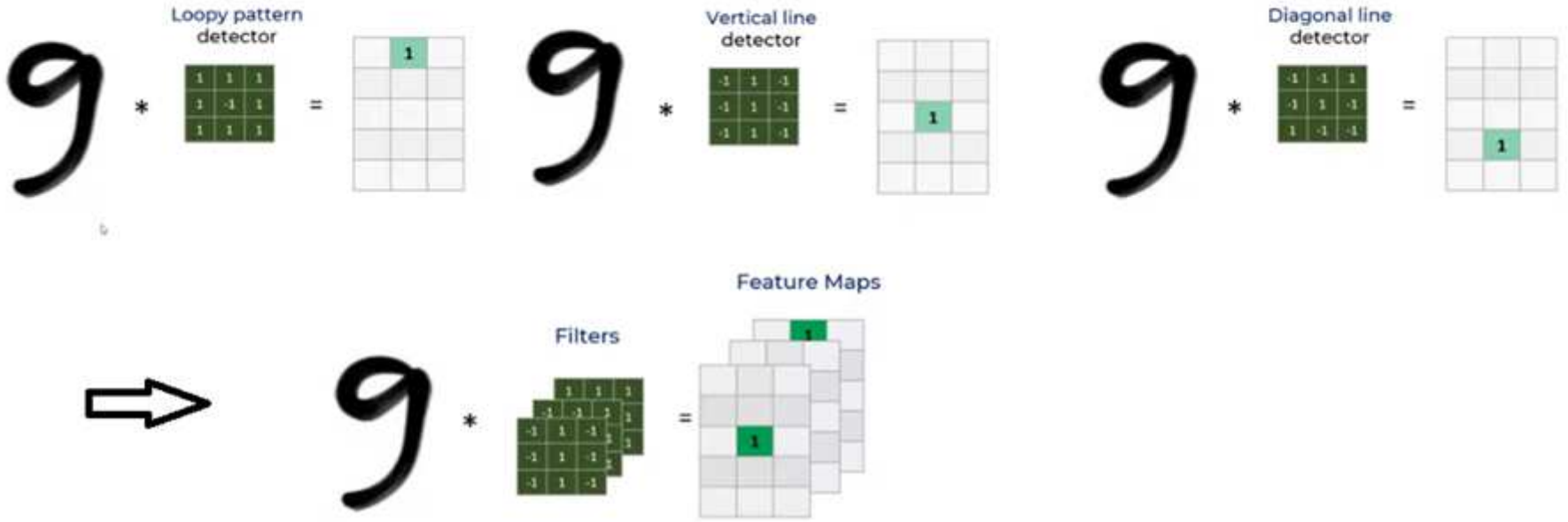


Couches de convolution

- Appliquant le premier filtre sur l'image d'origine



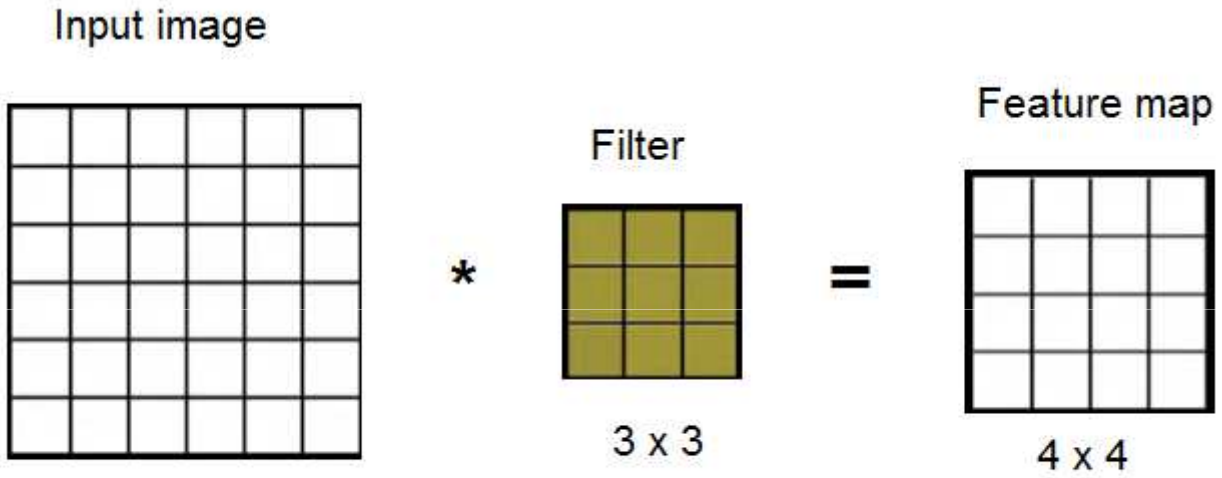
Couches de convolution



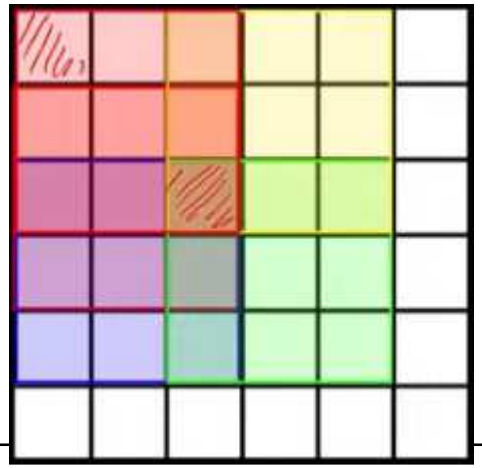
Couches de convolution

- Problèmes:

1.



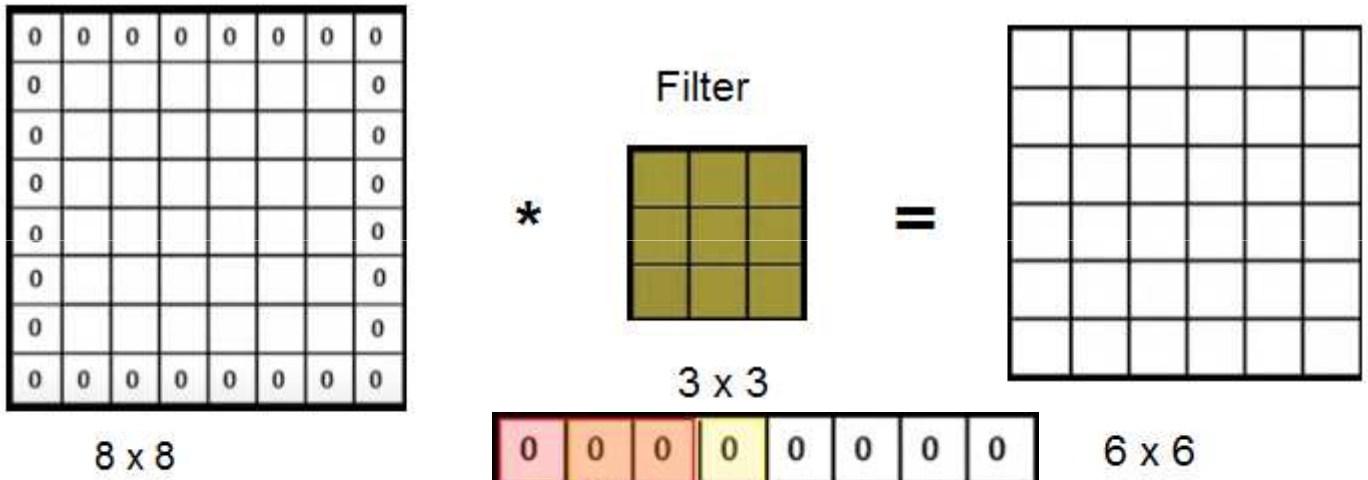
2. 6 x 6



Couches de convolution

- Padding

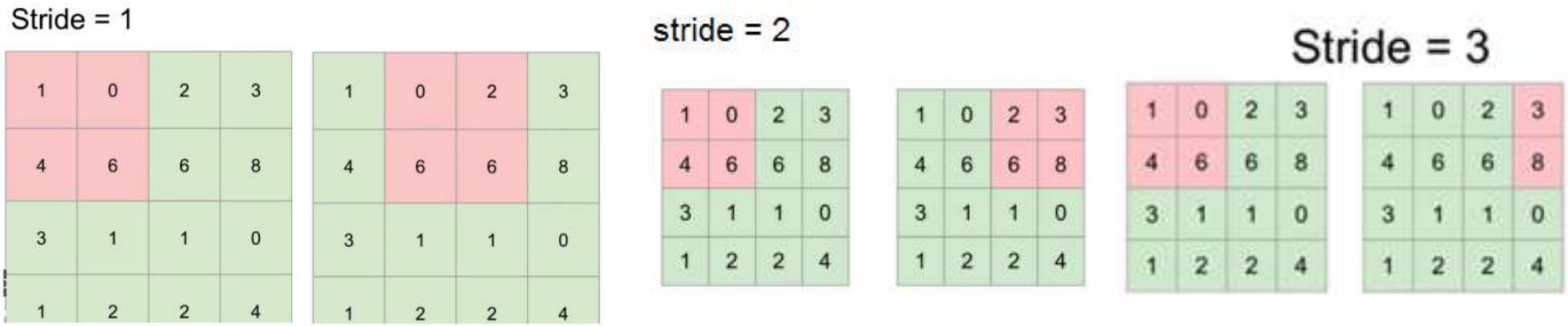
1



2

Couches de convolution

- 2 types de padding:
  - Same
  - Valid
- Stride



Couches de convolution

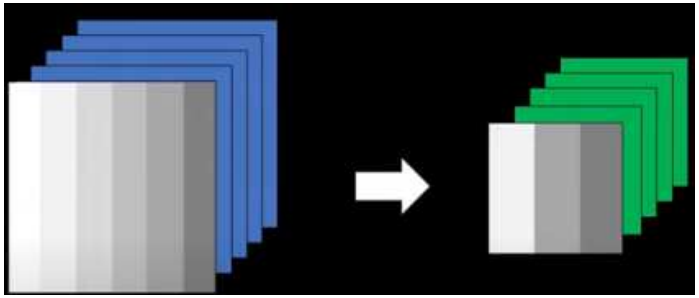
- Syntaxe:

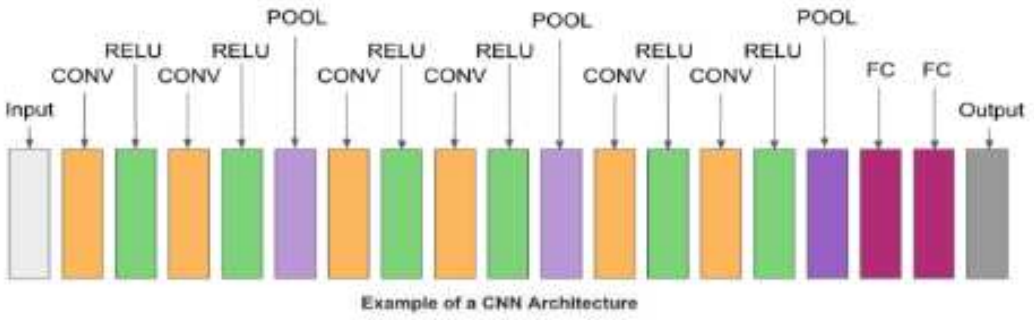
```
Conv2D(filters, kernel_size, strides=(1, 1), padding='valid',
data_format=None, dilation_rate=(1, 1), groups=1,
 activation=None, use_bias=True,
kernel_initializer='glorot_uniform', bias_initializer='zeros',
kernel_regularizer=None, bias_regularizer=None,
activity_regularizer=None, kernel_constraint=None,
bias_constraint=None, **kwargs)
```

Architecture du CNN

CNN définit des couches spécifiques

- Couche convolutive (CONV)
  - Opérateur de convolution
  - Représentation linéaire (somme du produit)
- **Couche ReLU (RELU)**
  - Optimisation de la Descente de Dégradé
  - Introduire la représentation non linéaire
- Couche de pooling (POOL)
  - max pooling
  - Average pooling
- Couche complètement connectée (FC)
  - Émuler les classificateurs MLP
  - Tâche de classification





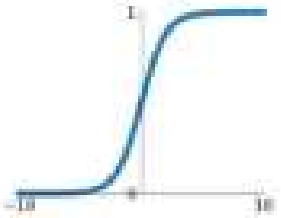
Example of a CNN Architecture

Couche ReLU

Activation Functions

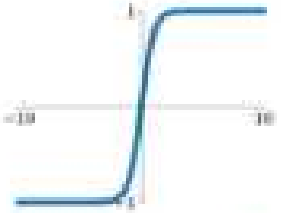
Sigmoid

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



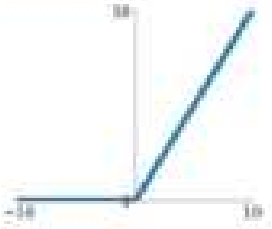
tanh

$$\tanh(x)$$



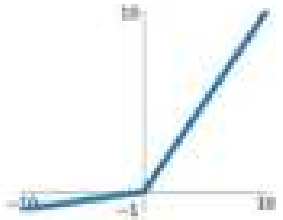
ReLU

$$\max(0, x)$$



Leaky ReLU

$$\max(0.1x, x)$$

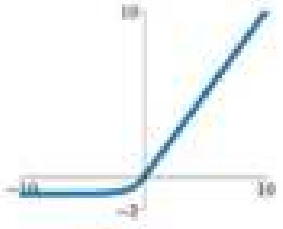


Maxout

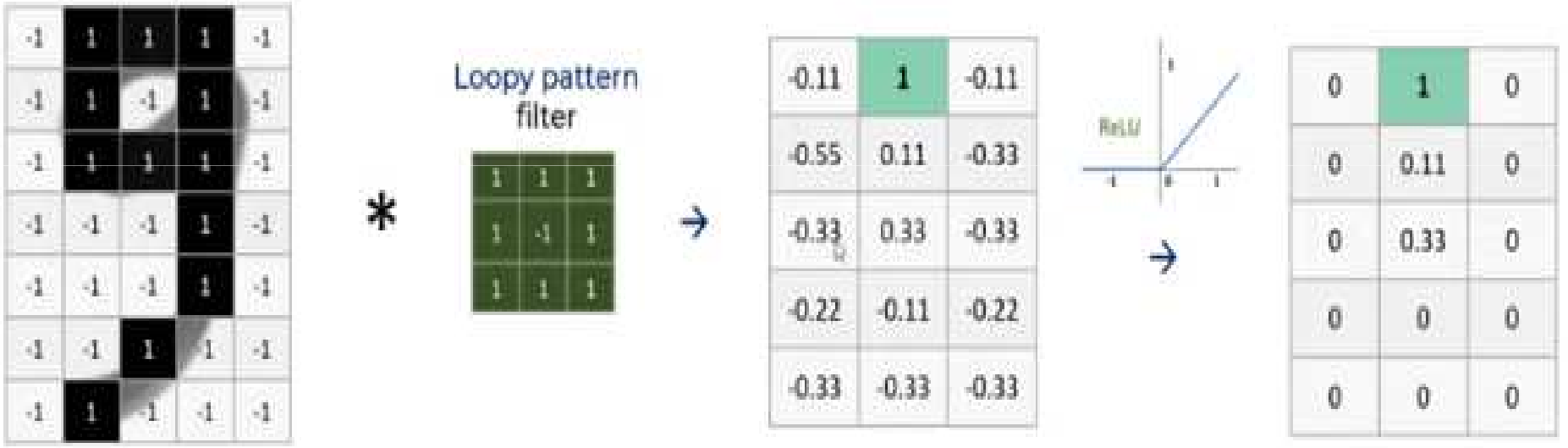
$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



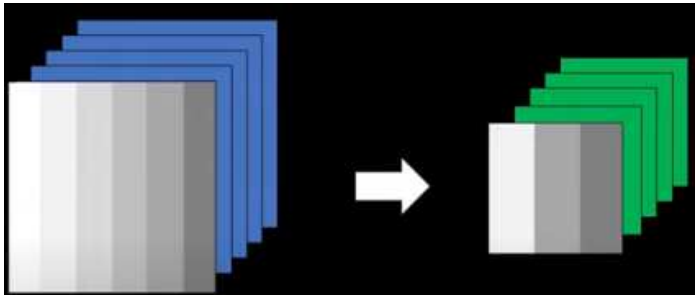
Couche ReLU

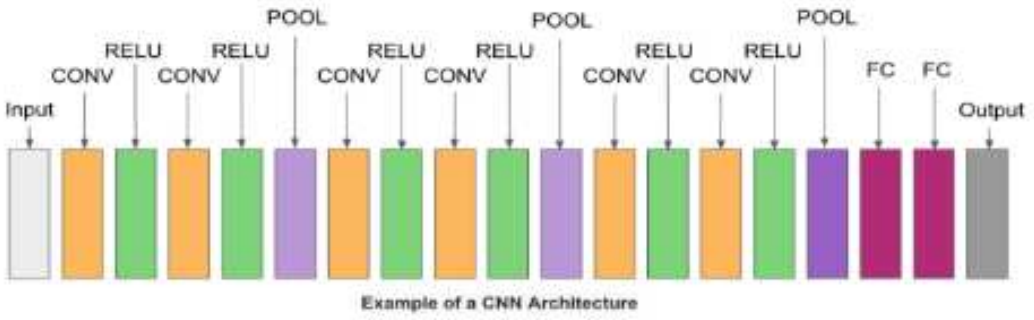

Loopy pattern
filter
-0.11 1 -0.11
-0.55 0.11 -0.33
-0.33 0.33 -0.33
-0.22 -0.11 -0.22
-0.33 -0.33 -0.33
ReLU
0 1 0
0 0.11 0
0 0.33 0
0 0 0
0 0 0

Architecture du CNN

CNN définit des couches spécifiques

- Couche convolutive (CONV)
  - Opérateur de convolution
  - Représentation linéaire (somme du produit)
- Couche ReLU (RELU)
  - Optimisation de la Descente de Dégradé
  - Introduire la représentation non linéaire
- **Couche de pooling (POOL)**
  - max pooling
  - Average pooling
- Couche complètement connectée (FC)
  - Émuler les classificateurs MLP
  - Tâche de classification





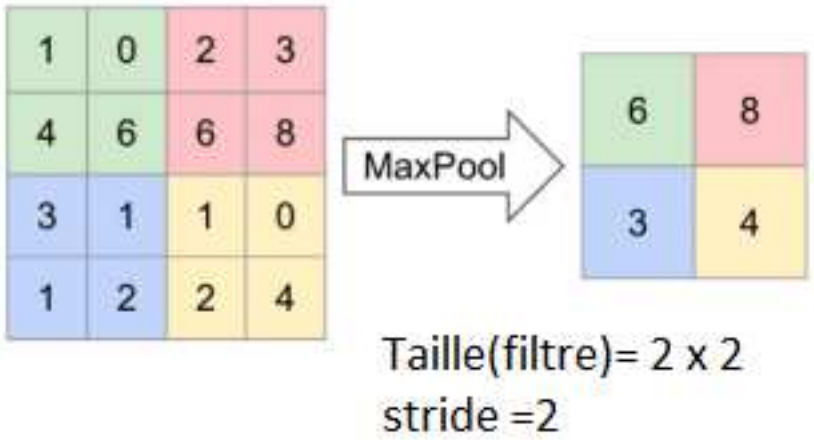
Example of a CNN Architecture

Couche de pooling

- Max pooling:
  - **Objectif**: réduire la taille de l'image tout en améliorant ses « features »
  - **Paramètres**: taille du filtre , stride, etc

MaxPooling2D(pool_size=(2, 2), strides=None, padding='valid', data_format=None, **kwargs)

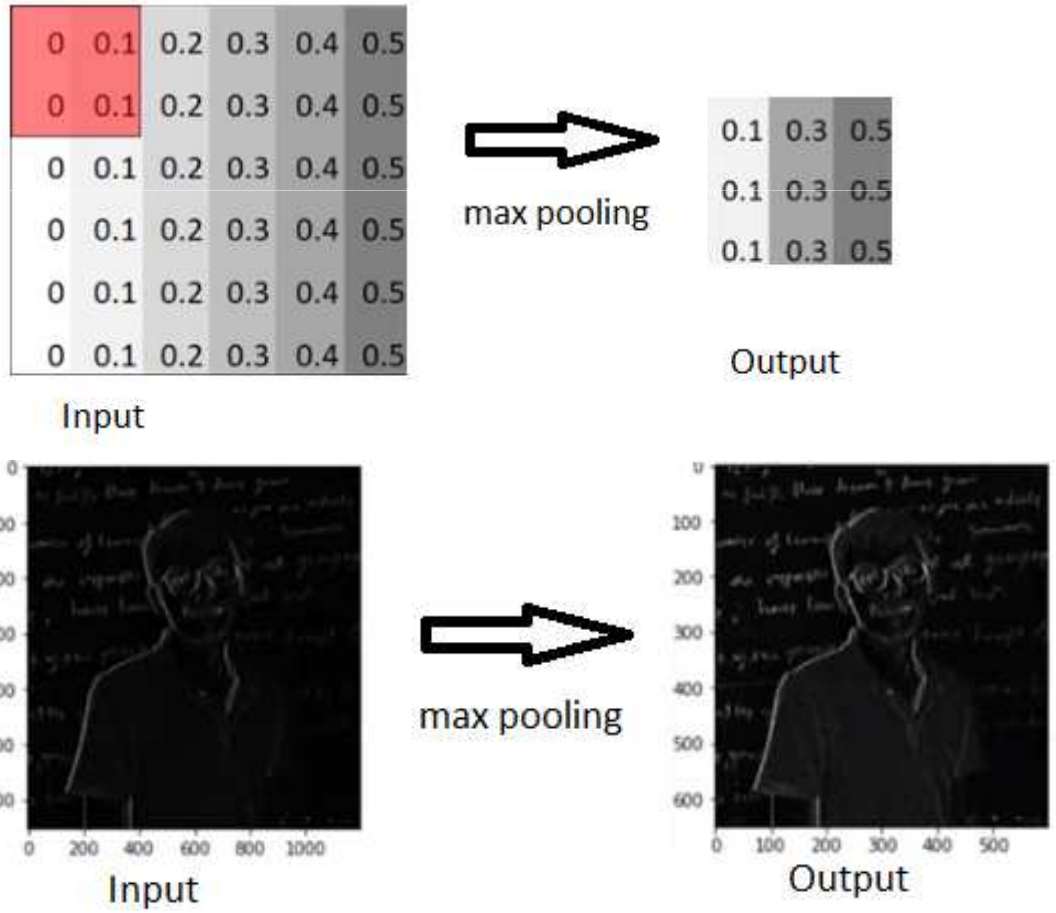
  - **Principe**: Appliquer le filtre et garder le maximum



Couche de pooling

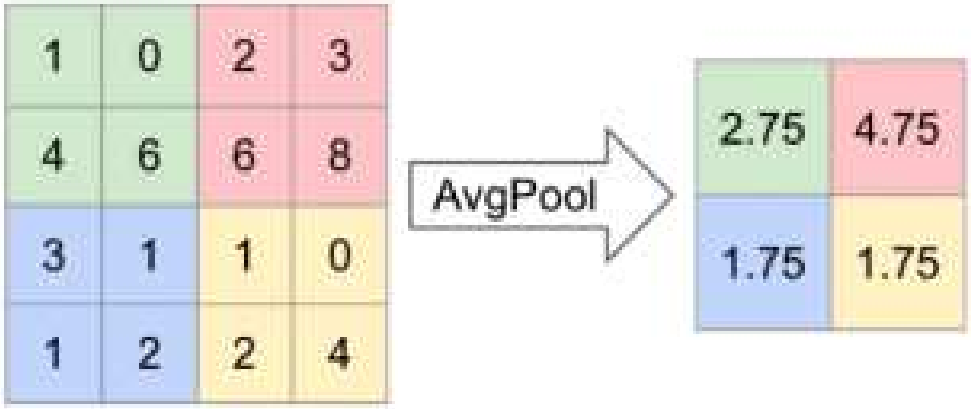
- Max pooling:

  Améliorer les « features »



Couche de pooling

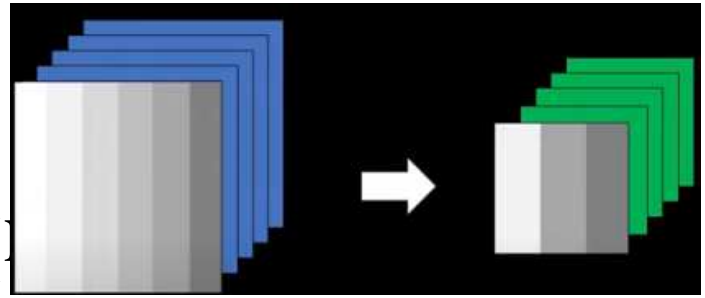
- Average pooling

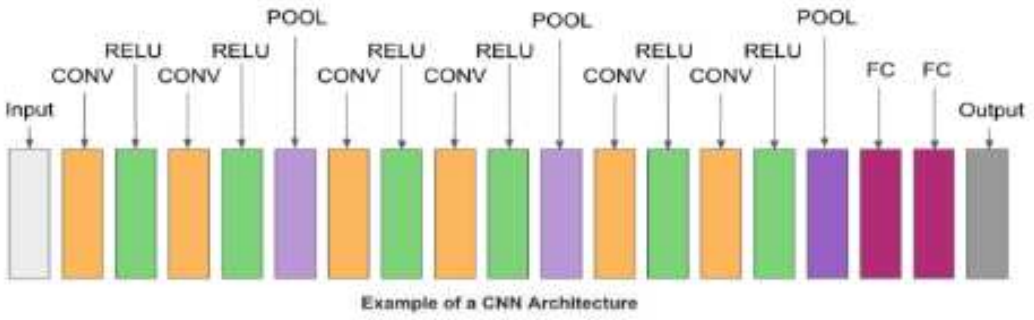


Architecture du CNN

CNN définit des couches spécifiques

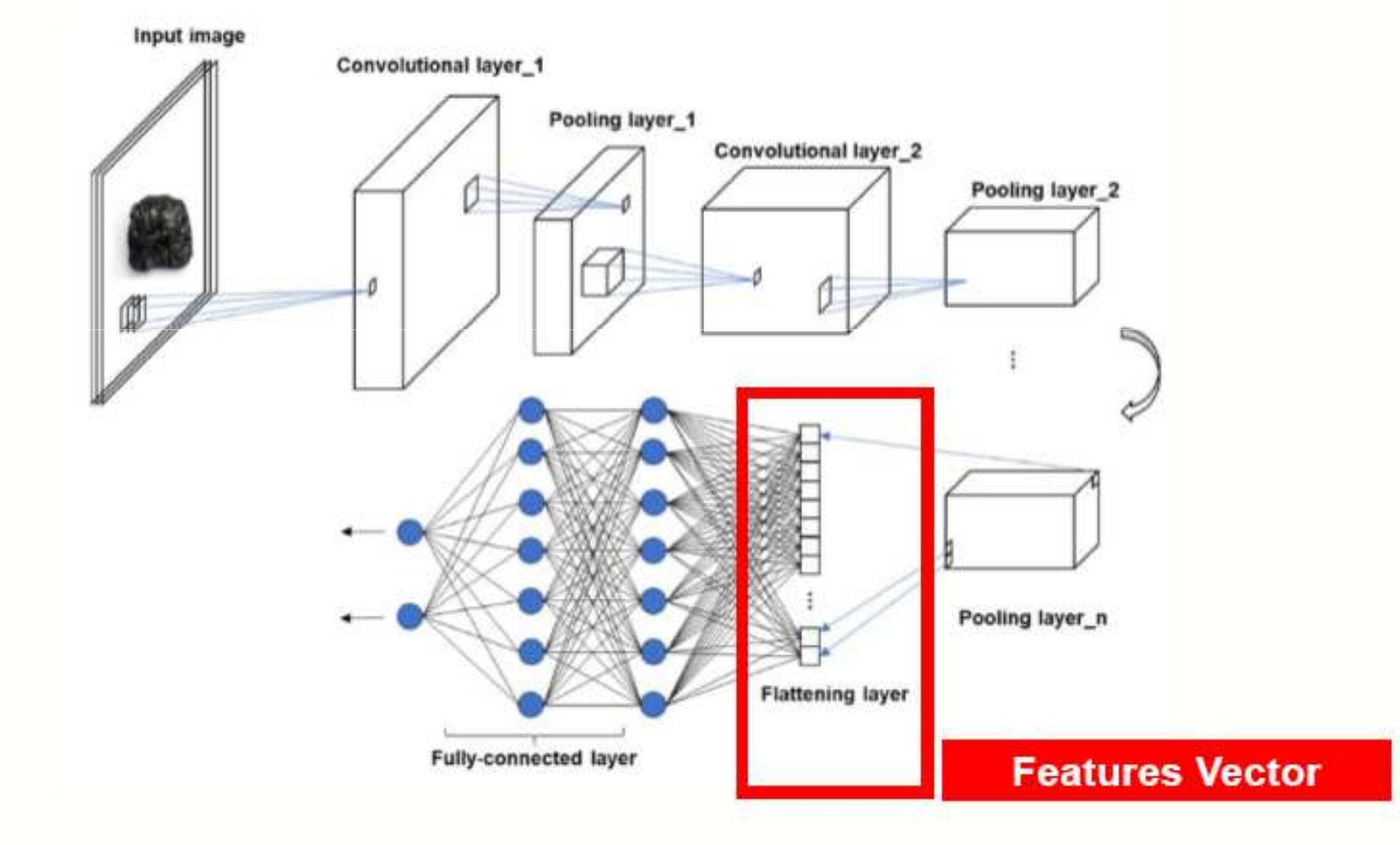
- Couche convolutive (CONV)
  - Opérateur de convolution
  - Représentation linéaire (somme du produit)
- Couche ReLU (RELU)
  - Optimisation de la Descente de Dégradé
  - Introduire la représentation non linéaire
- Couche de pooling (POOL)
  - max pooling
  - Average pooling
- **Couche complètement connectée (**
  - Émuler les classificateurs MLP
  - Tâche de classification





Example of a CNN Architecture

Couche complètement connectée (FC)



Architecture du CNN