Convolutional networks

Disclaimer

- Large part of those slides came from Fidle IN2P3
- Fidle.Contact@grenoble.cnrs.fr
- https://gricad-gitlab.univ-grenoble-alpes.fr/talks/fidle

- Next session in Nov 2023

About Fidle

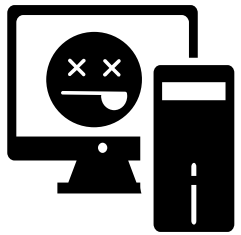
This repository contains all the documents and links of the **Fidle Training** .
Fidle (for Formation Introduction au Deep Learning) is a 2-day training session
co-organized by the Formation Permanente CNRS and the Resinfo/SARI and DevLOG CNRS networks.

The objectives of this training are :

- Understanding the **bases of Deep Learning** neural networks

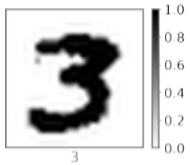
24 M pixels
(r,v,b) 3x8 bits

3 x 24 M neurons ?!



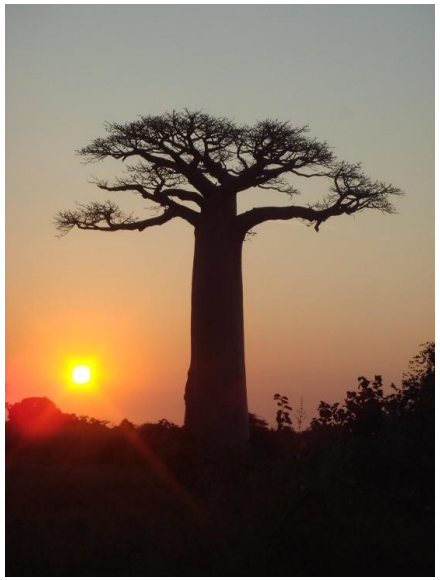
Why convolutional Neural Networks (CNN)&?

For a fully connected layer of (only) ±1000 neurons, we would need to



0.0008 M pixels
28x28, 8 bits

➡ 785.000 params

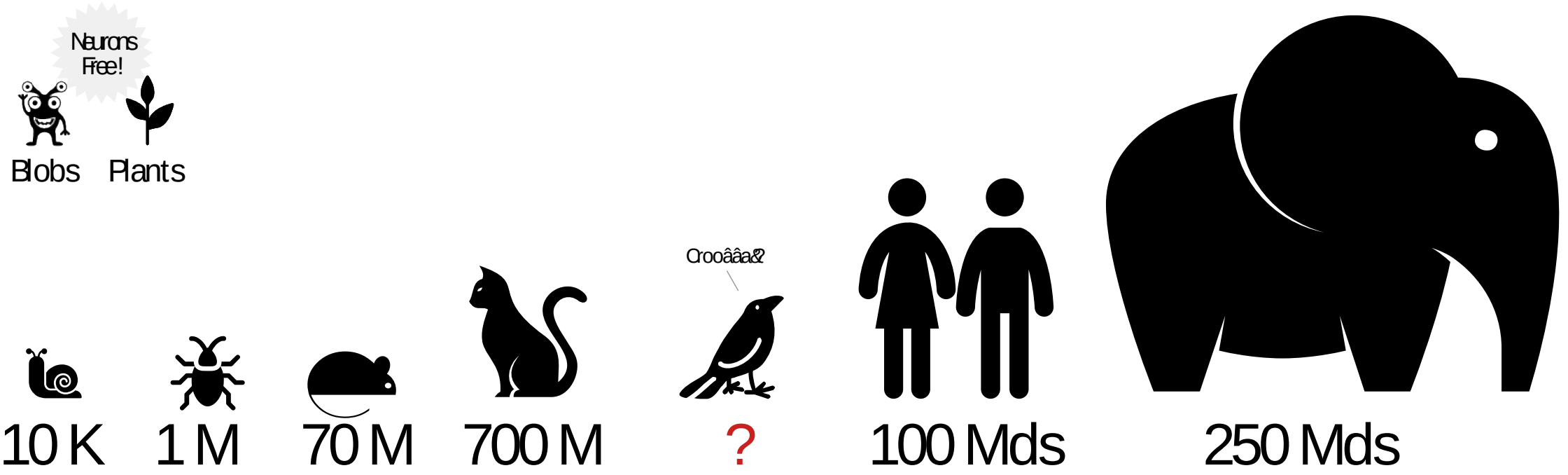


24 M pixels
(r,v,b) 3x8 bits

~~➡~~ 72. 10E9 params...

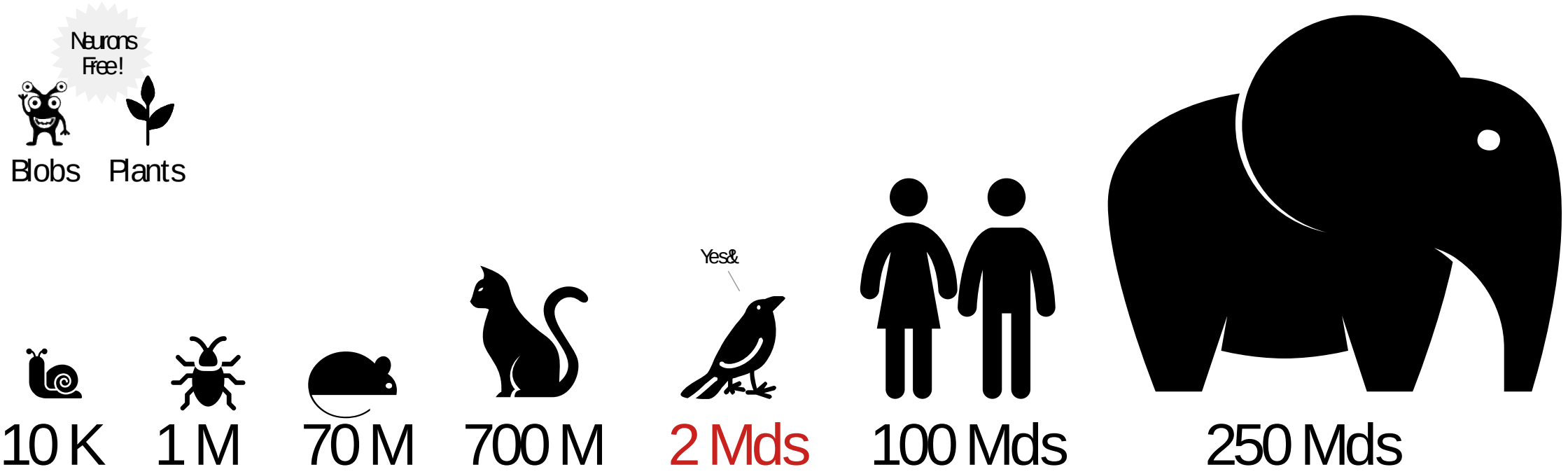
Short Digression&

One neuron is **good**... but more than one is **better** !

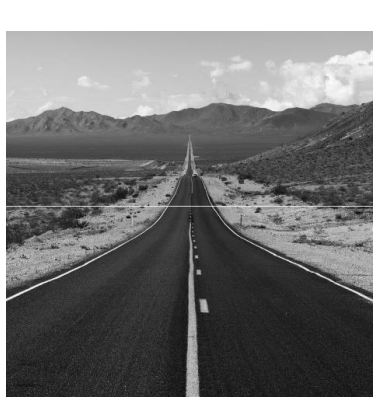


Short Digression&

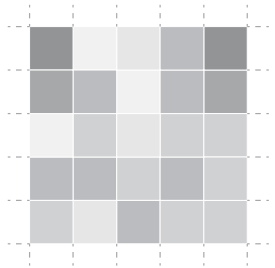
One neuron is **good**... but more than one is **better** !



Principle of convolutions



By Jan Kroon, from Pexels.com



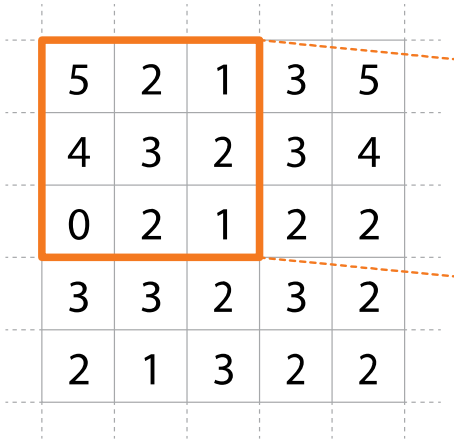


Image piece

5	2	1
4	3	2
0	2	1

X

$\otimes$

Kernel 3x3

1	0	1
0	1	0
1	0	1

ω

$=$

10

y

$$y = 5x1 + 2x0 + 1x1 + 4x0 + 3x1 + 2x0 + 0x1 + 2x0 + 1x1 = 10$$

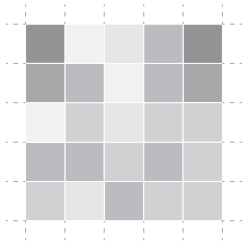
$$y = \sum_{i=1}^{n}\sum_{j=1}^{m} x_{i,j} \cdot \omega_{i,j} \quad \text{with} \begin{cases} n & \text{kernel width} \\ m & \text{kernel height} \end{cases}$$

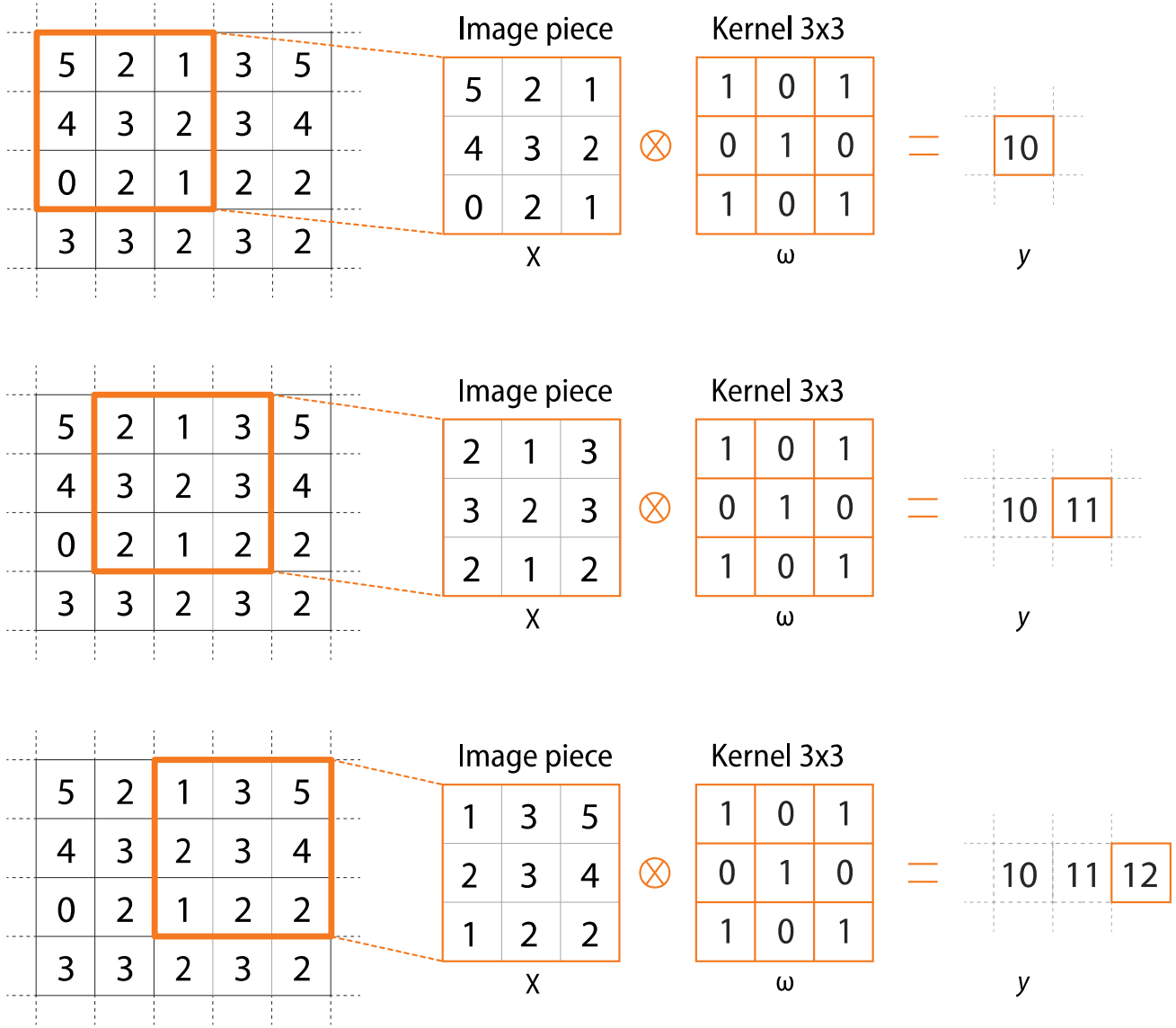
2D convolution

$\otimes$ Hadamard product

Principle of convolutions

By Jan Kroon, from Pexels.com

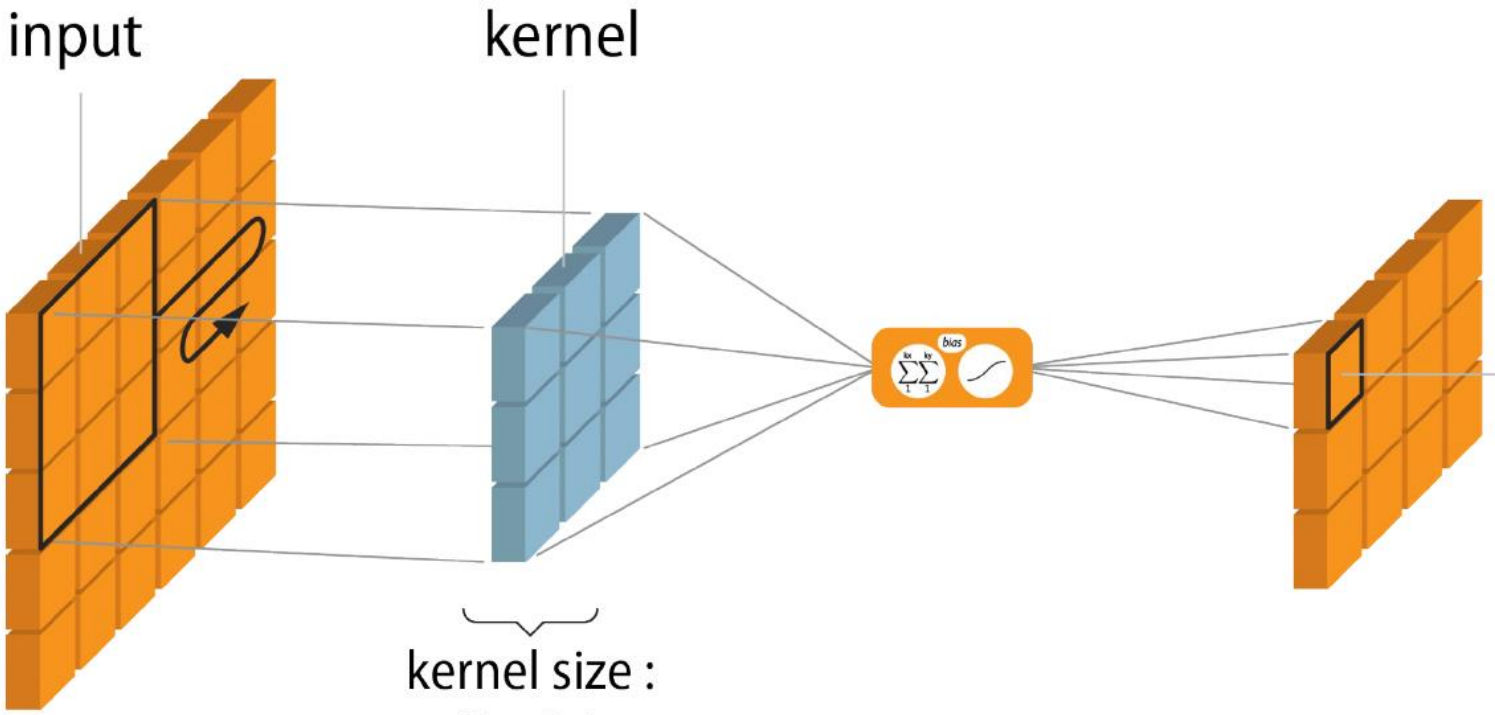




> We can perform convolutions in 1, 2, 3 ...or n-dimensional spaces!

⊗ is Hadamard product

Convolutional layer

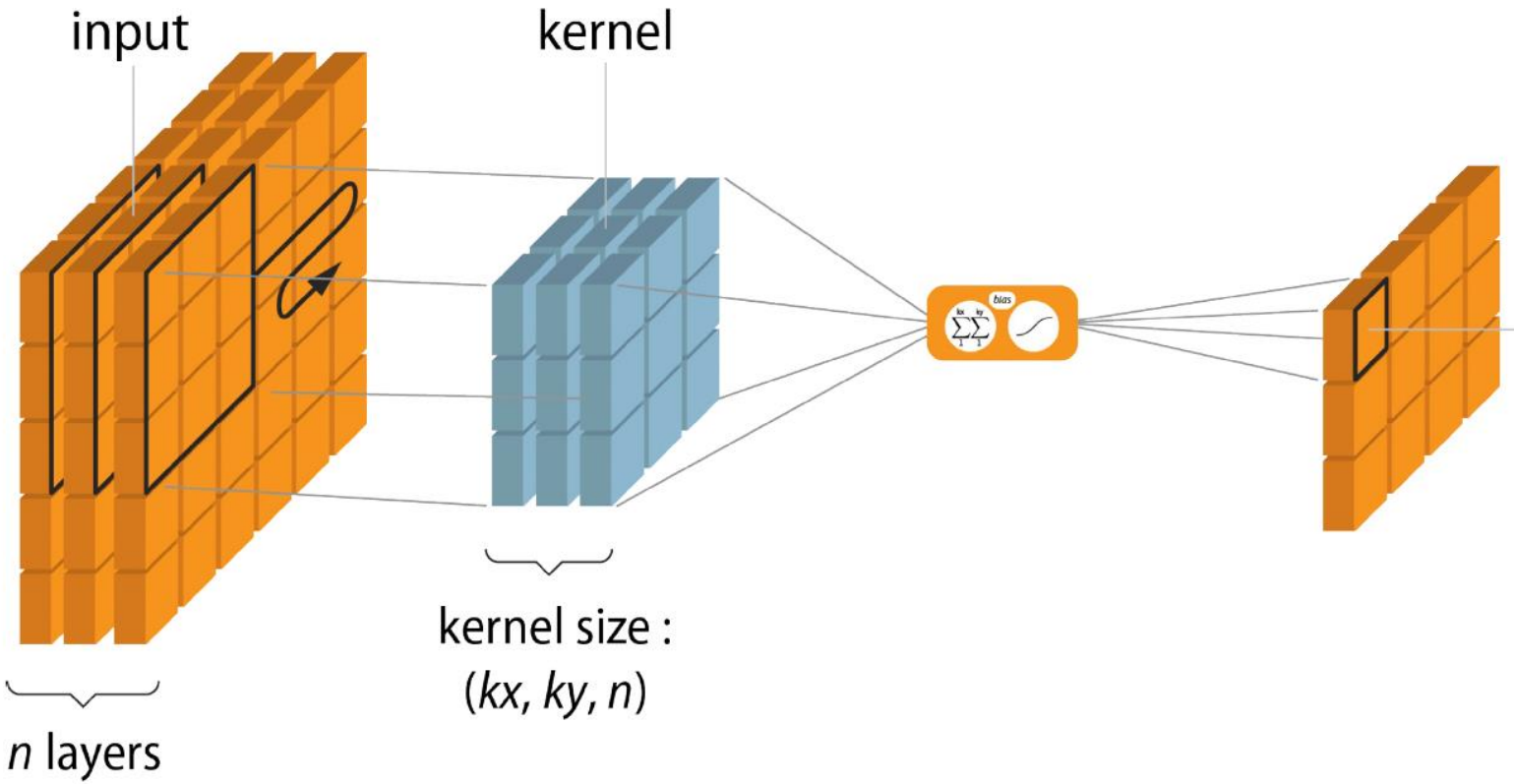


$$y = \sigma(\sum \text{input} \otimes \text{kernel} + b)$$

Where :
σ : is an activation function
b : is a bias

Number of parameters for a convolutional layer : $kx \,.\, ky + 1$

Convolutional layers

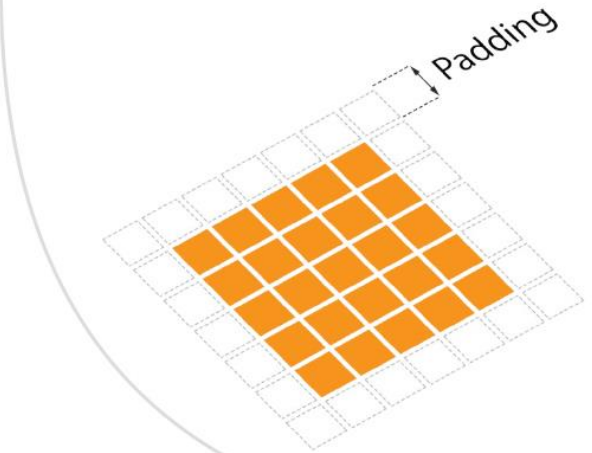


$$y = \sigma(\sum \text{input} \otimes \text{kernel} + b)$$

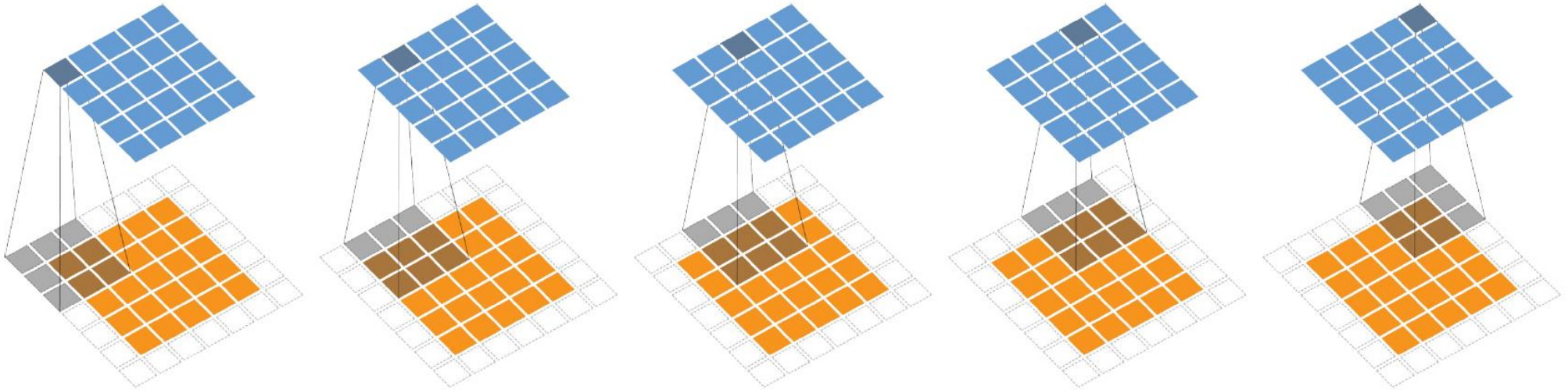
Where :
σ : is an activation function
b : is a bias

If we want to generate m convolutional layers, we will need m convolutional neurons

Convolution parameters

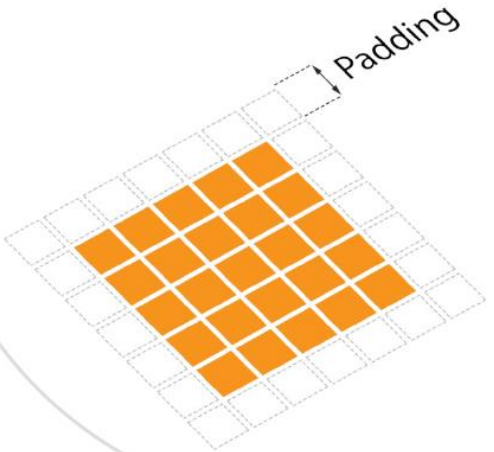


Parameters of a convolutional layer : padding

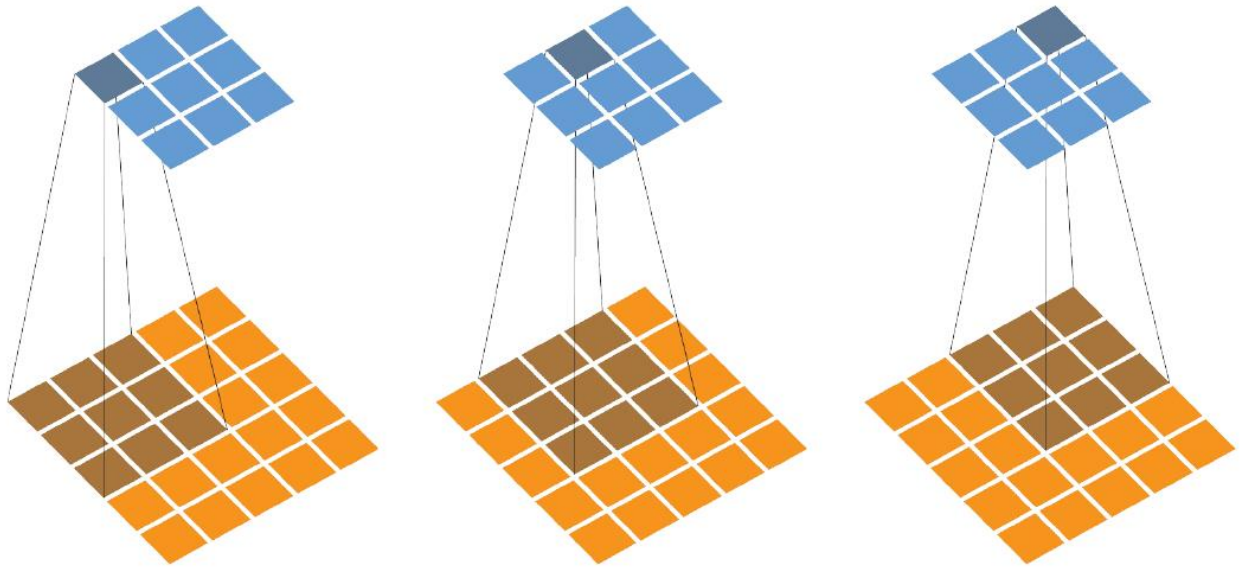


Keras : padding = 'same' $\Longrightarrow$ size is preserved

Convolution parameters

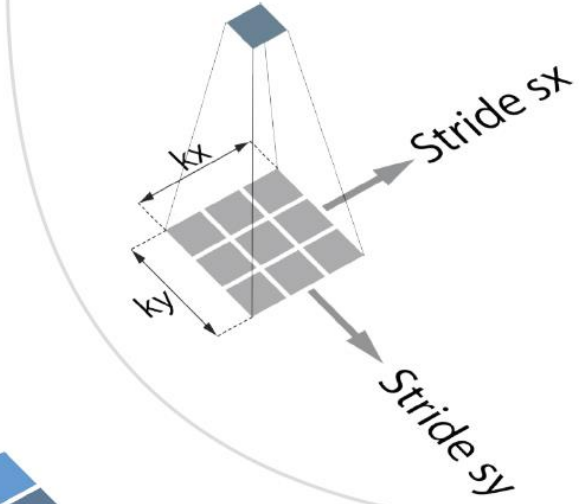


Parameters of a convolutional layer : padding

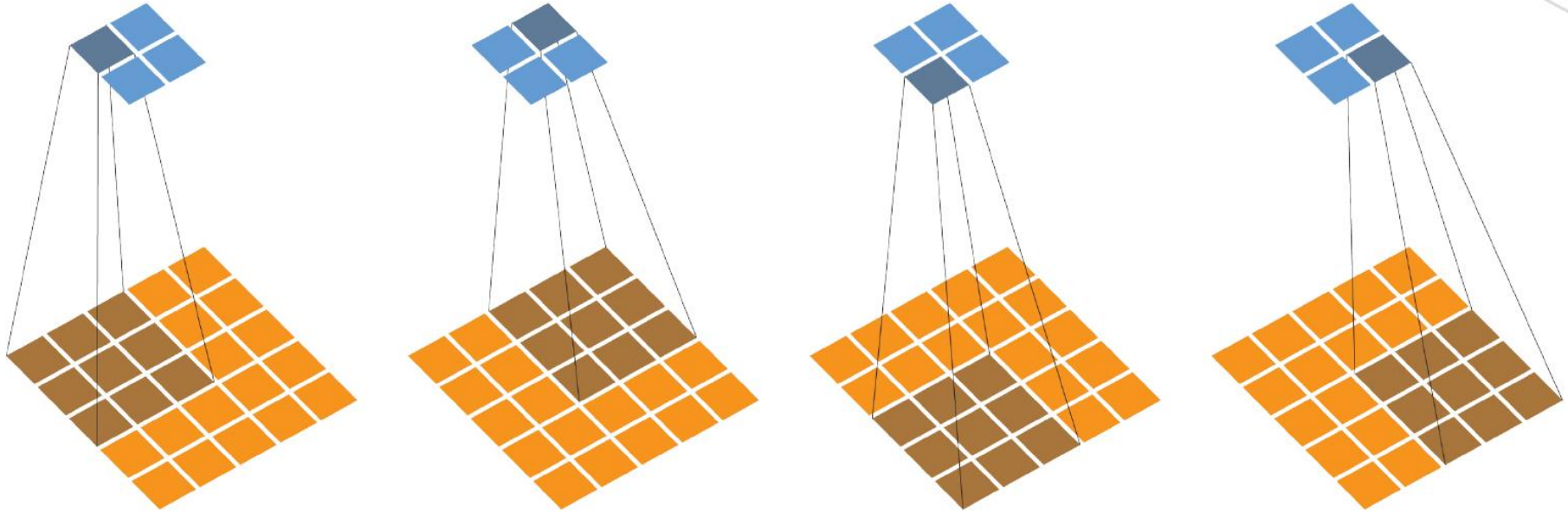


Keras : padding = 'valid' $\Rightarrow$ Size is not preserved (no padding)

Convolution parameters

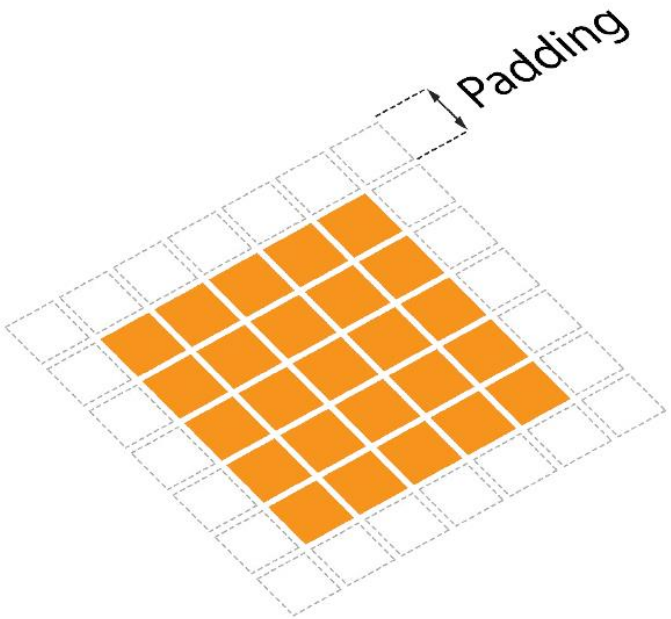


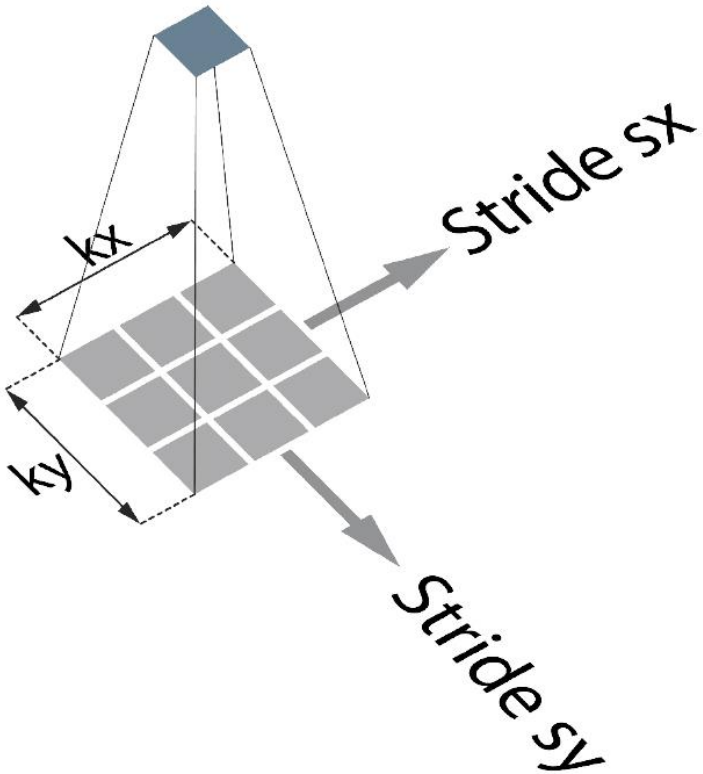
Parameters of a convolutional layer : Strides



strides = (sx,sy) $\Longrightarrow$ A strides=(2,2) will reduce by 2 the output size

Convolution parameters





padding

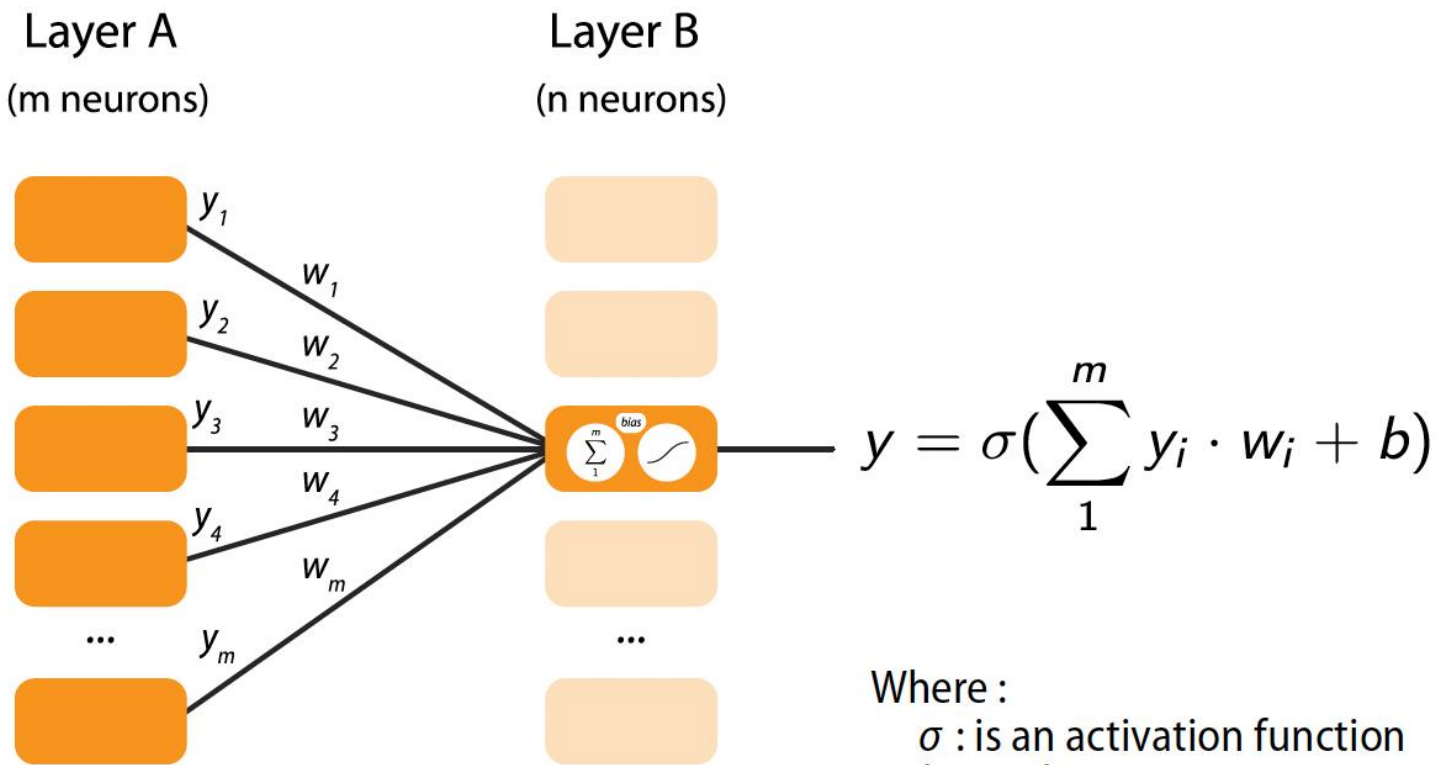
'same' 'valid'

kernel size

strides

Number of parameters

For a fully connected layer :

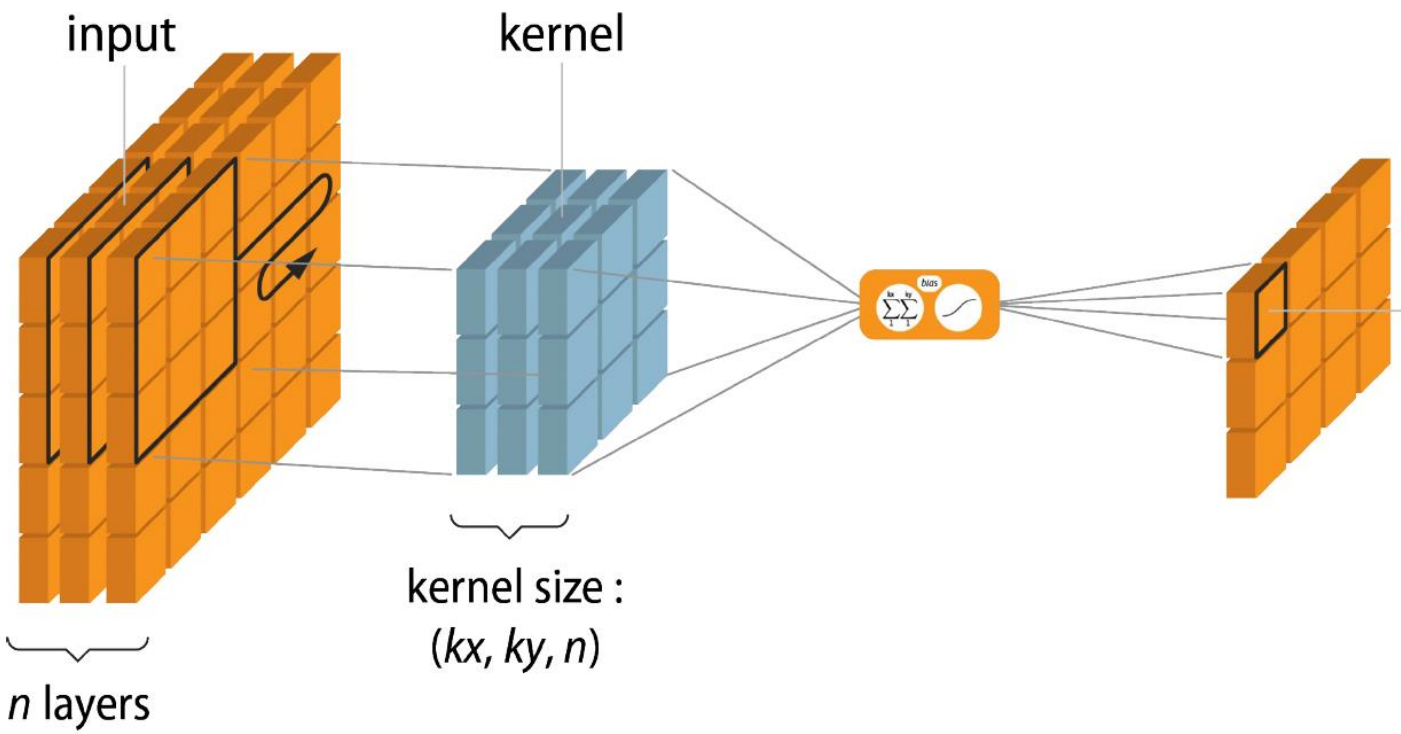


Where :
σ : is an activation function
b : is a bias

Number of parameters for a DNN layer : $n\,(m+1)$

Number of parameters

For a convolutional layer :



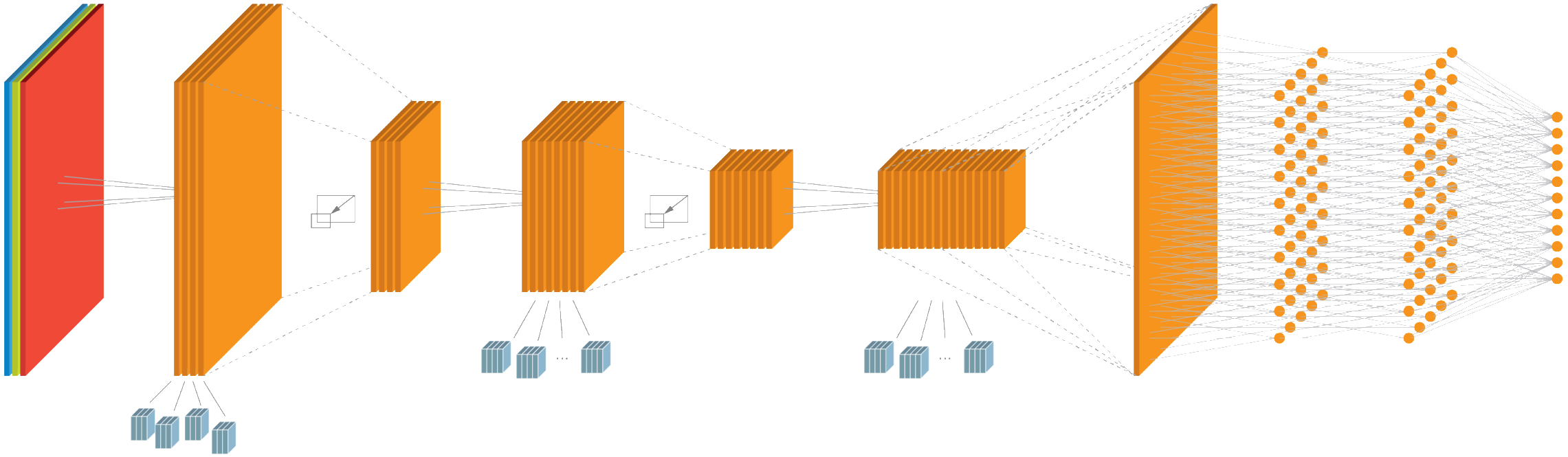
$$y = \sigma(\sum \text{input} \otimes \text{kernel} + b)$$

Where :
σ : is an activation function
b : is a bias

Number of parameters for a convolutional layer : $n.kx.Ky + 1$

If we want to generate m convolutional layers, we will need m convolutional neurons, so, number of parameters is : $m.(n.kx.ky + 1)$

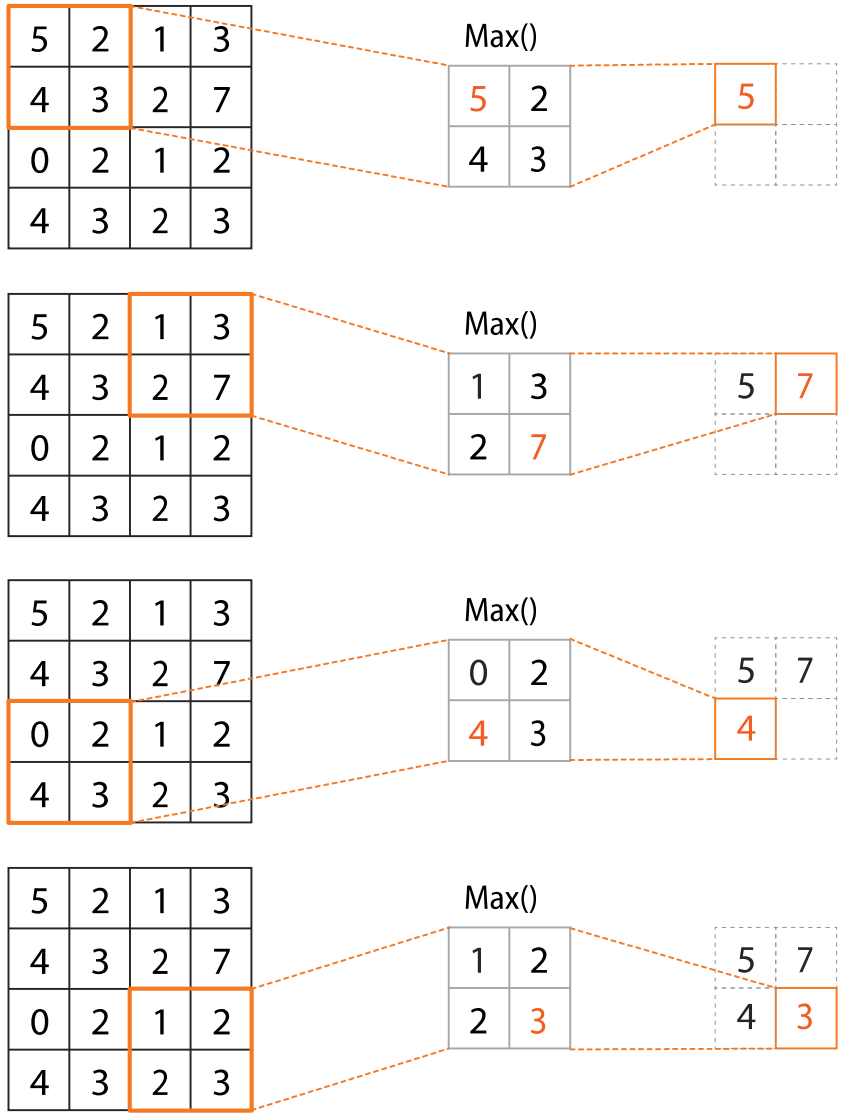
Convolutional Neural Networks (CNN)



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

Convolutional Neural Networks (CNN)

Principle of Max Pooling :

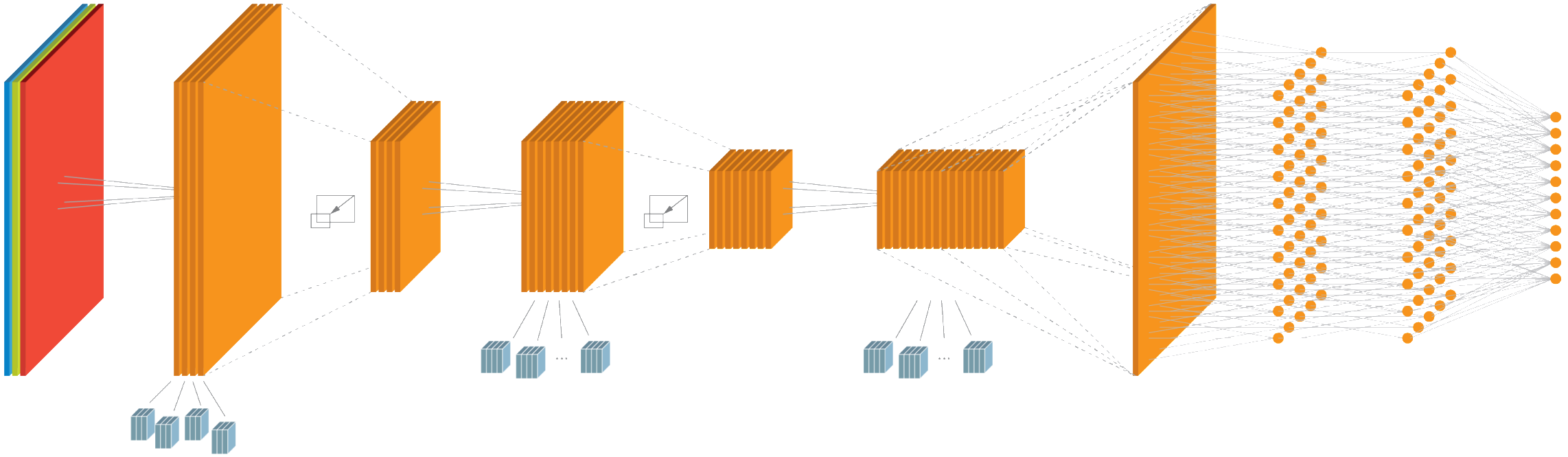


It is possible to set the **window size, padding** mode and **strides**.

By default, the strides correspond to the size of the window.

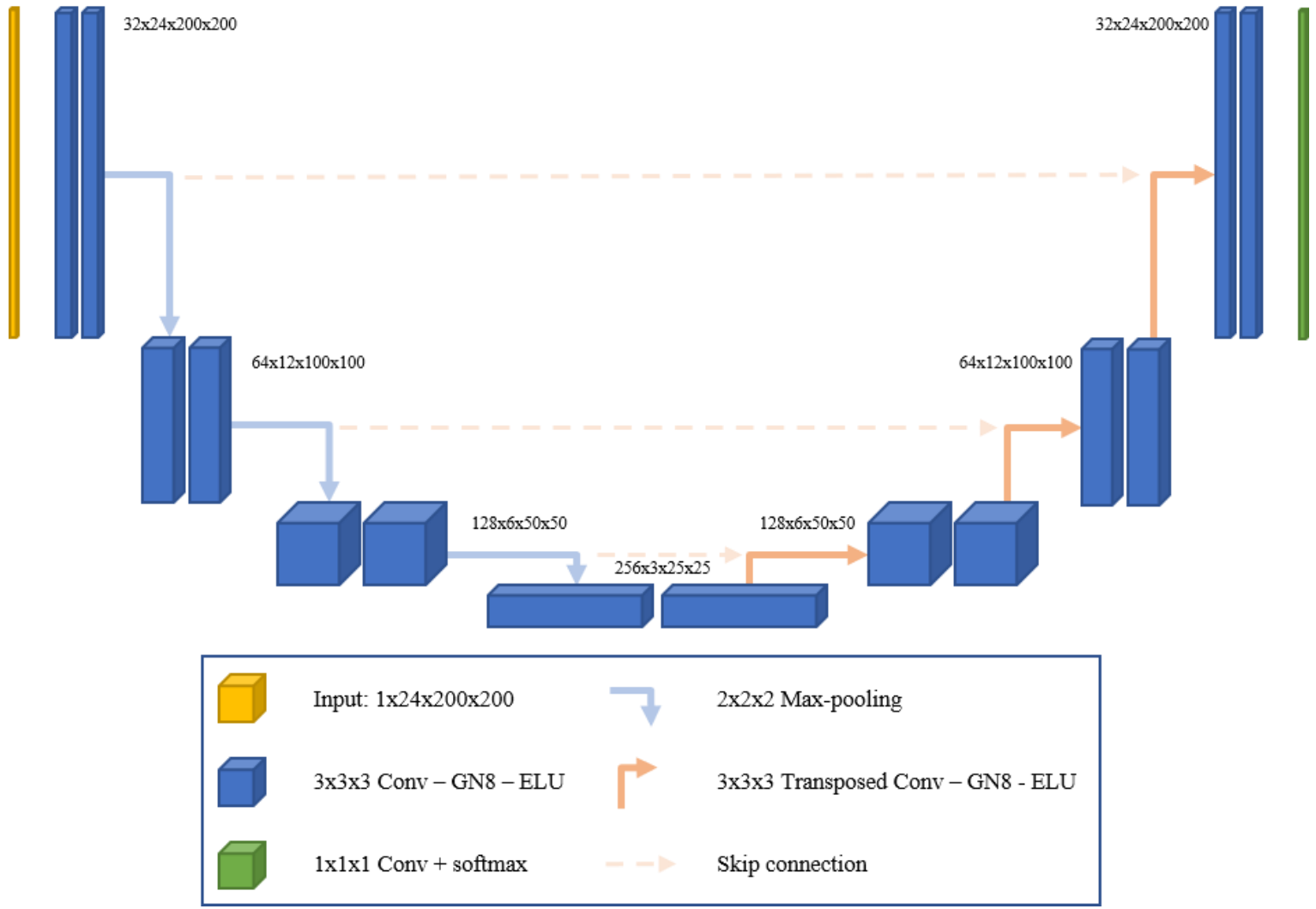
A window (2,2) generates an image twice as small.

Convolutional Neural Networks (CNN)



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

U-NET


32x24x200x200
32x24x200x200
64x12x100x100
64x12x100x100
128x6x50x50
128x6x50x50
256x3x25x25
Input: 1x24x200x200
2x2x2 Max-pooling
3x3x3 Conv – GN8 – ELU
3x3x3 Transposed Conv – GN8 - ELU
1x1x1 Conv + softmax
Skip connection