SAISON 23/24

Formation

# Introduction au Deep Learning

Séquence n°7

## “Réseaux convolutifs (CNN)”

FIDLE



v2.10


Formation

# Introduction au Deep Learning

## Questions and answers :

https://fidle.cnrs.fr/q2a

**Accompanied by :**
IA Support (dream) Team of IDRIS

**Directed by :**
Agathe, Baptiste et Yanis - UGA/DAPI
Thibaut, Kamel - IDRIS

FIDLE

Formation

# Introduction au Deep Learning

FIDLE https://fidle.cnrs.fr/listeinfo

Fidle information list

Agoria http://fidle.cnrs.fr/agoria New !

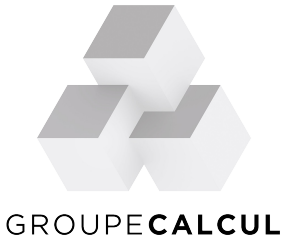
AI exchange list


Formation

# Introduction au Deep Learning


https://listes.services.cnrs.fr/wws/info/devlog

List of ESR* « Software developers » group




https://listes.math.cnrs.fr/wws/info/calcul

List of ESR* « Calcul » group

(*) ESR is Enseignement Supérieur et Recherche, french universities and public academic research organizations


Abonnez-vous !

https ://fidle.cnrs.fr/youtube
https://www.youtube.com/@CNRS-FIDLE

Vendredi 19 janvier, 10h00

http://www.idris.fr/panoramia.html
https://www.youtube.com/@IDRISCNRS


Le magazine de actualité en IA


All inclusive !

Python, modules, Keras, PyTorch
Datasets, notebooks, etc.

# New Fidle **environment** !


Classical installation 



6.5 GB → 2.7 GB
± 40 notebooks

New !

All inclusive !

Python, modules, Keras, PyTorch
Datasets, notebooks, etc.

# New Fidle **environment** !

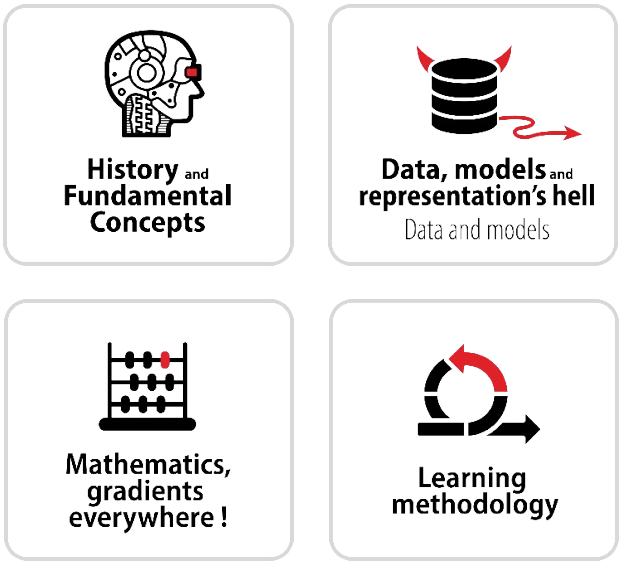

docker

Classical installation

6.5 GB ➜ 2.7 GB
± 40 notebooks

# Previously on Fidle

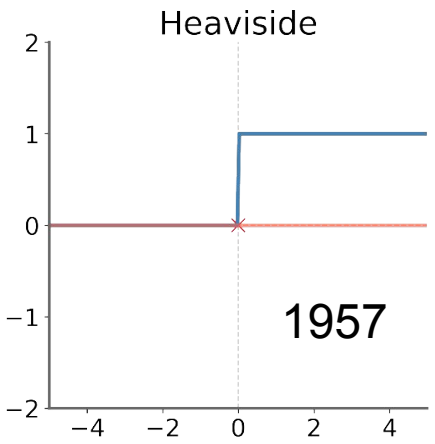
## Few little things and concepts to keep in mind

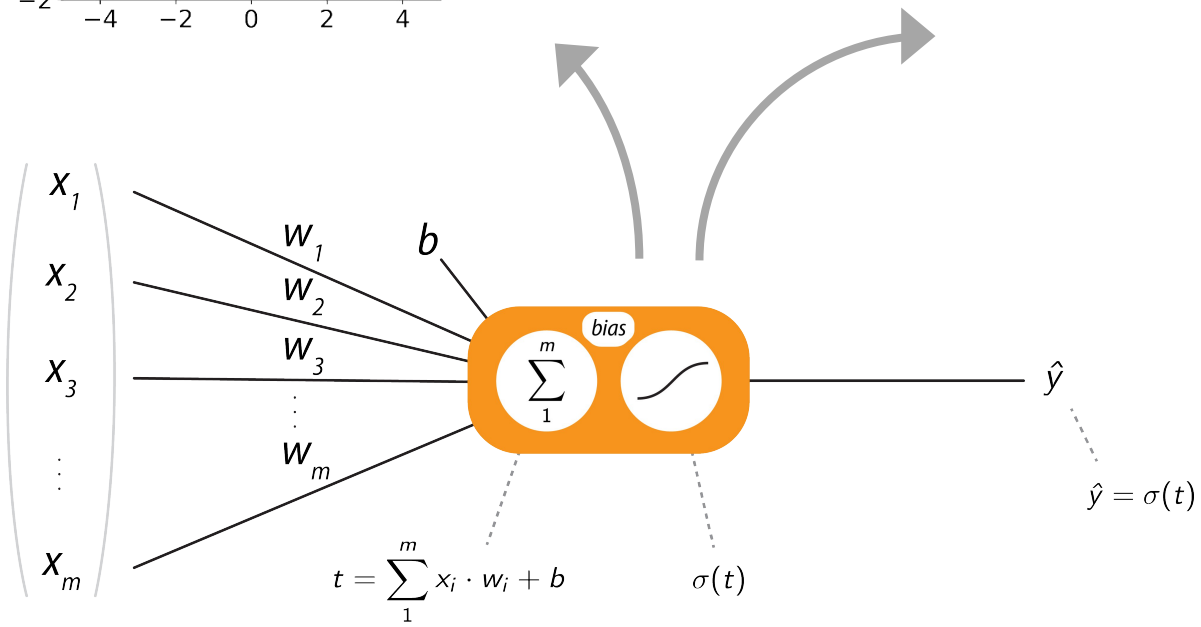



- Regression vs. Classification
- Data normalization
- Training and validation
- Epochs and Batchs
- Activation functions
- Loss function
- Optimization and gradient descent
- Underfiting / Overfiting
- Metrics
- Softmax and Argmax function
- Numpy shape

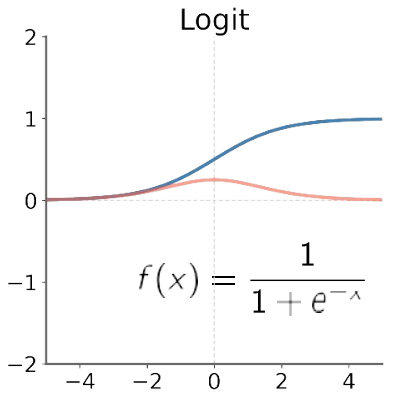
Easy :-)

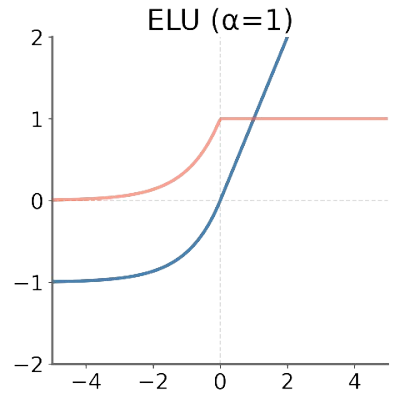
# Activation functions

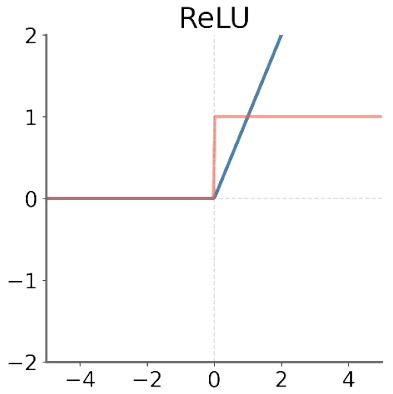


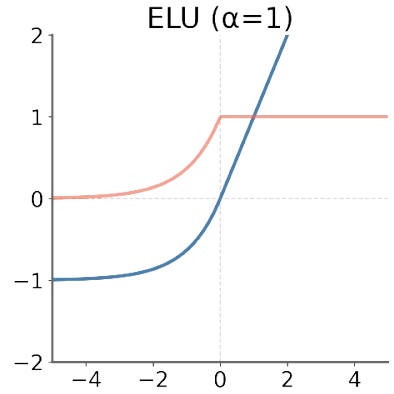



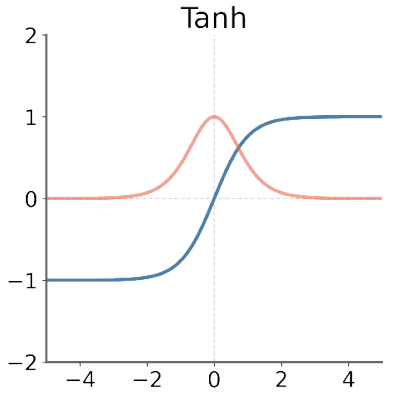

| Input | Bias / Weight | Activation function | Output |
|---|---|---|---|
| $X$ | $\Theta$, $b$ | $\sigma(t)$ | $\hat{y}$ |

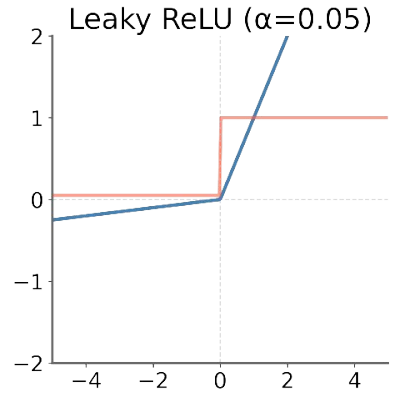


# DNN regression and classification

**Neurons**

$$y = \sigma\left(\sum_{1}^{m} y_i \cdot w_i + b\right)$$

Activation : ReLU, etc.

$$\sigma(x) = \max(0, x)$$

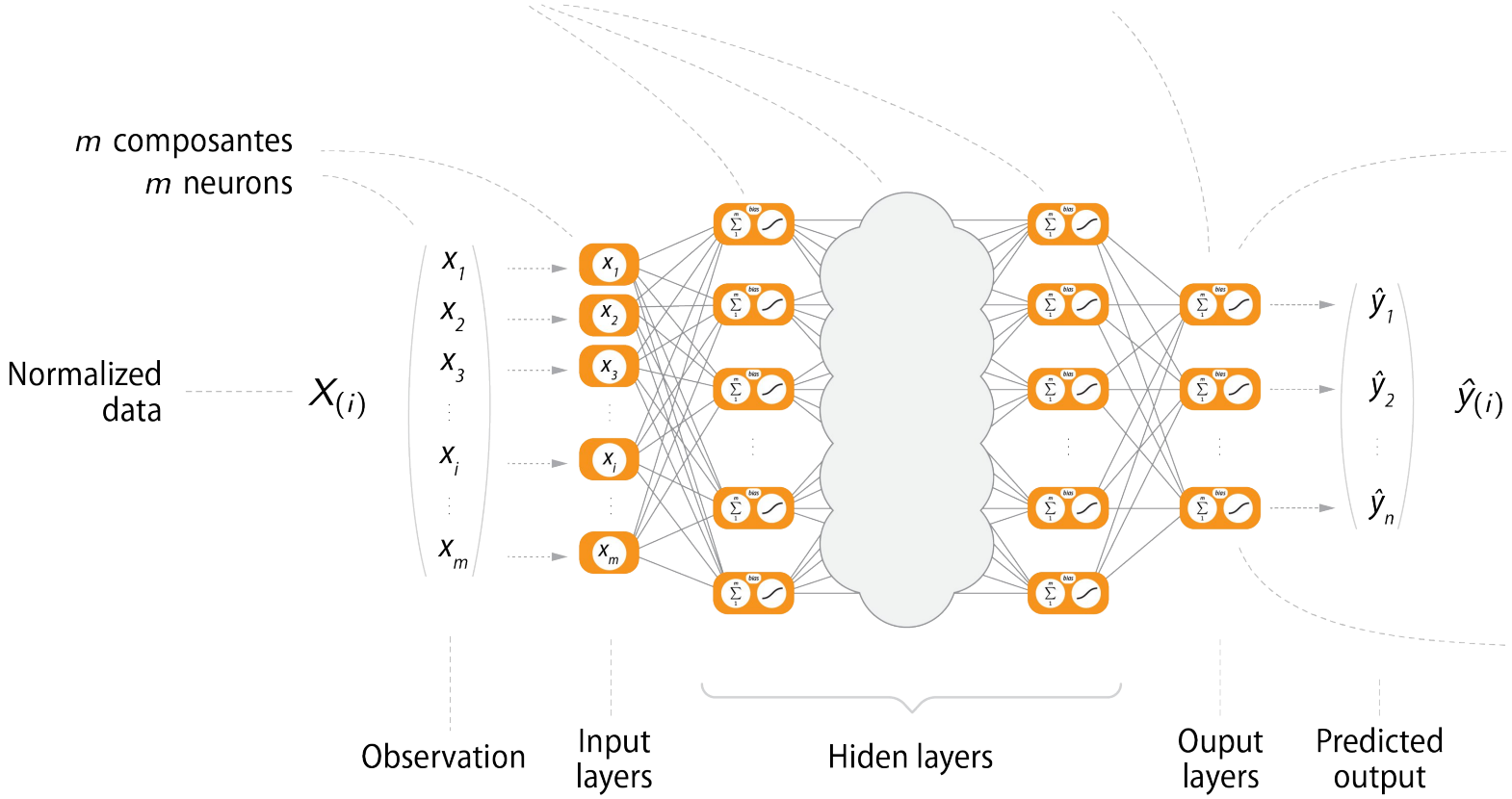
**Binary classification**

Activation : Sigmoid

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

Loss: Binary cross entropy

$$H(y, \hat{y}) = -\frac{1}{n}\sum_{i=1}^{n} y_i \cdot \log \hat{y}_i + (1 - y_i) \cdot \log (1 - \hat{y}_i)$$




**Muticlass classification**

Activation : Softmax

$$\sigma(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}}$$

Loss: Categorical cross entropy

$$H(y, \hat{y}) = -\sum_{i=1}^{n} y_i \cdot \log \hat{y}_i$$

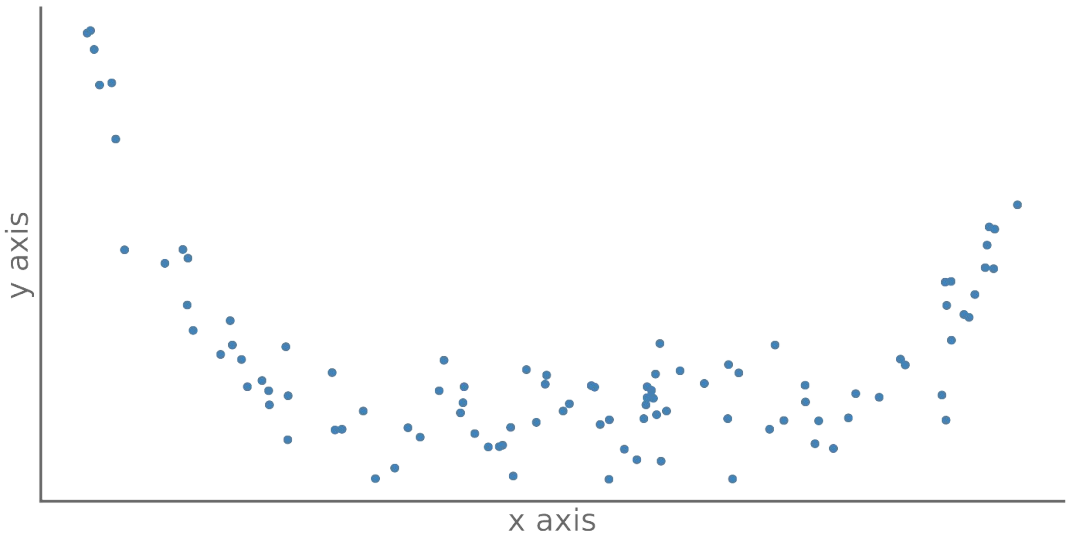
**Regression**

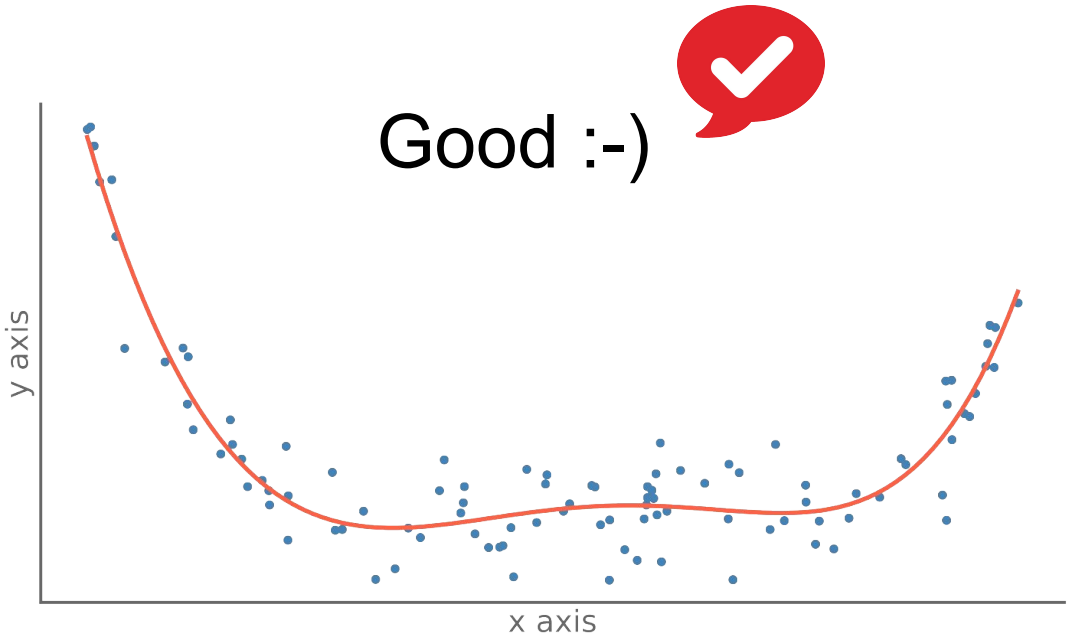
No activation

Loss : MSE, ...

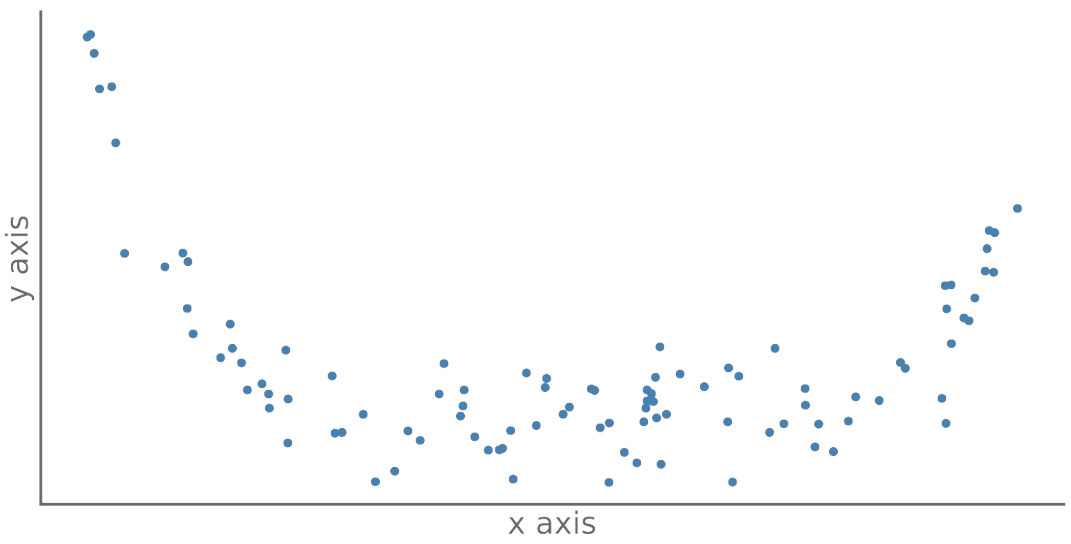
$$\frac{1}{n}\sum_{i=1}^{n}\left[\hat{y}^{(i)} - y^{(i)}\right]^2$$

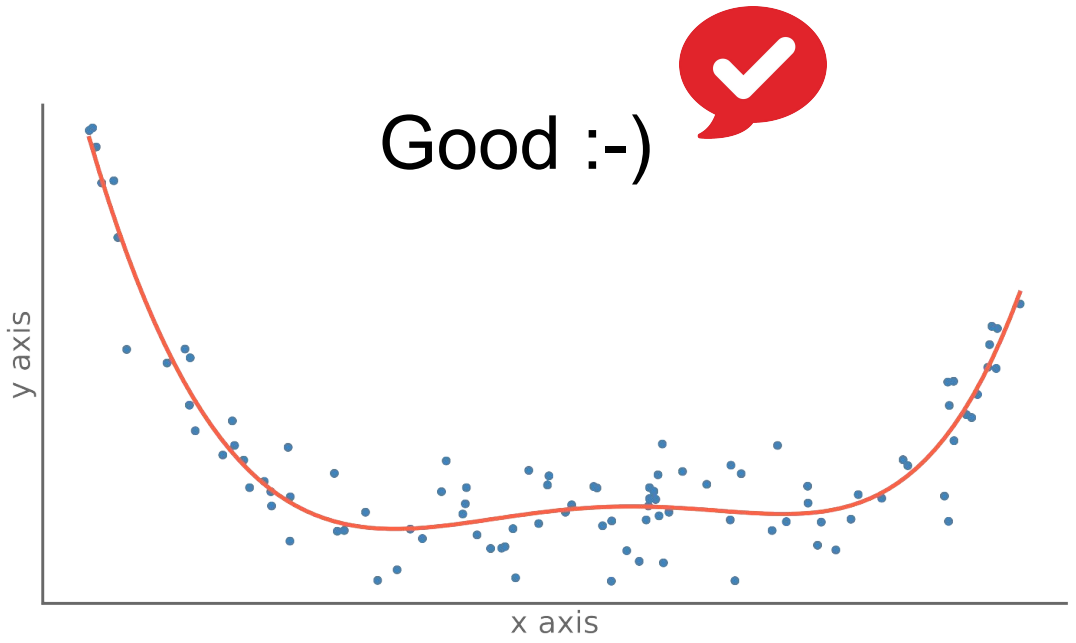
# {under | good | over} fiting

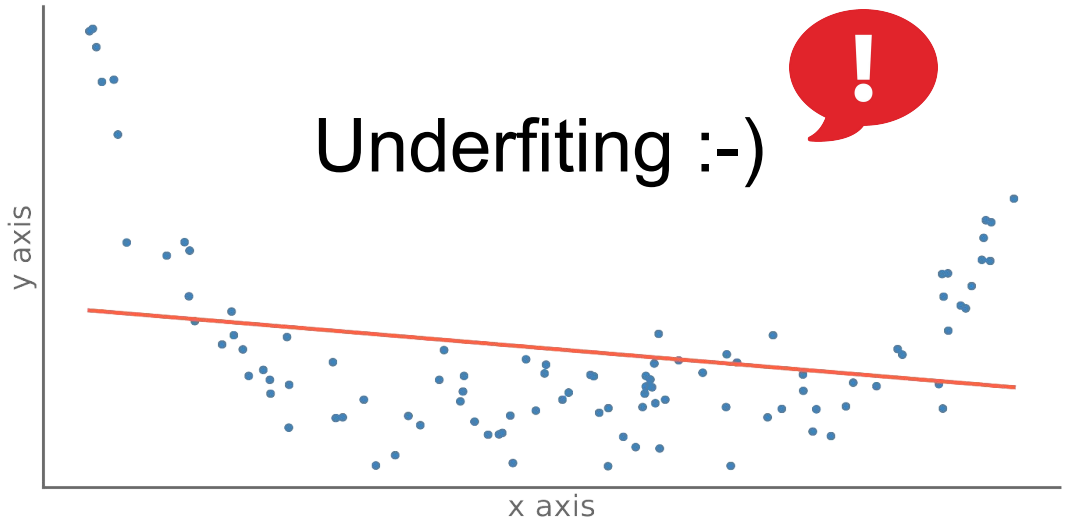


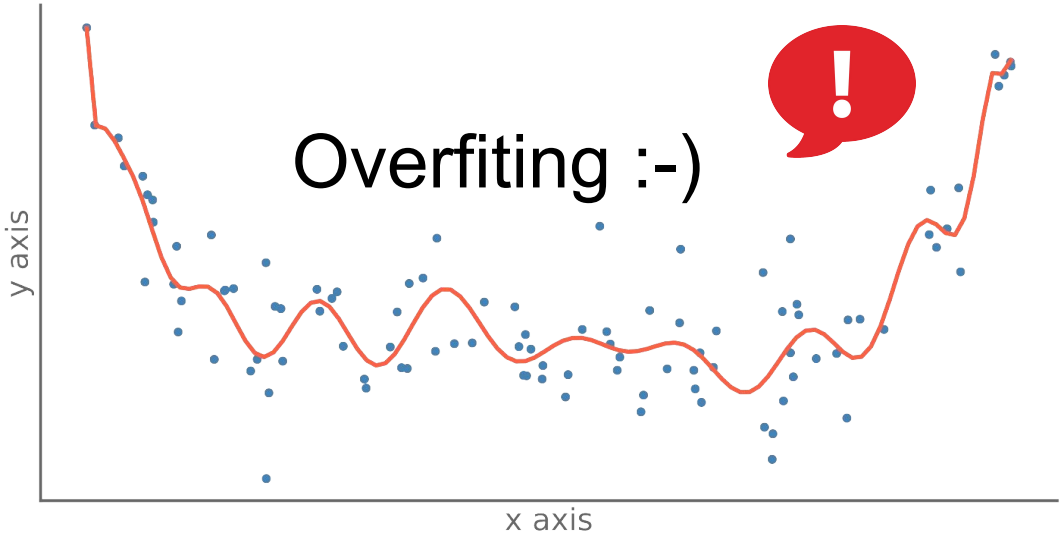



# {under | good | over} fiting



FIDLE

SAISON 23/24

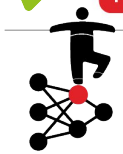
## Bases, Concepts et Enjeux

1 ✓

**History and Fundamental Concepts**

2 ✓ New !

**Data, models and representation's hell**
Data and models

3 ✓ New !

**Demonstration Illustration**
LLM / Text to Image

4 ✓

**AI, Law, Society and Ethics**

## L'IA comme un outil,

5 ✓ New !

**Mathematics, gradients everywhere !**

6 ✓ New !

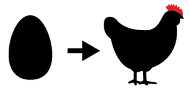
**Learning methodology**

7

**Convolutional models**
CNN

8

**Sparse (text) and sequences data**
Embedding, RNN

9

**«Attention is All You Need»**
Transformers

10

**Graph Neural Network**
GNN

11

**Autoencoder networks**
AE

12

**Variational Antoencoder**
VAE

13

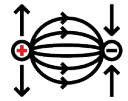
**Generative Adversarial Networks**
GAN

14

**Diffusion Model**
Text to image

15

**Deep Reinforcement Learning**
RL

16

**Physics-Informed Neural Networks**
PINNS

## Acteur de l'IA

17

**Learning faster and cheaper,**
Eco-Friendly

18

**Jean-Zay**
GPU acceleration

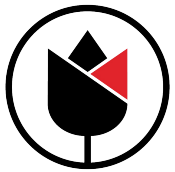
19 New !

**New models**
VLM, SM, Multimodal, ...

20 New !

**Case Study**
Experience feedback

7



**Convolutional models**

CNN

1 What is a **Convolutional Neuron Network** (CNN) ?

- Understanding what a CNN is
- Identify use cases

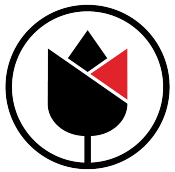
2 Example 1 : MNIST

- A very simple case

3 Example 2 : GTSRB

- A more elaborate case

7



**Convolutional models**

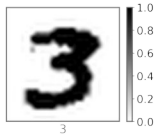
CNN

1 What is a **Convolutional Neuron Network** (CNN) ?
- Understanding what a CNN is
- Identify use cases

2 Example 1 : MNIST
- A very simple case

3 Example 2 : GTSRB
- A more elaborate case

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

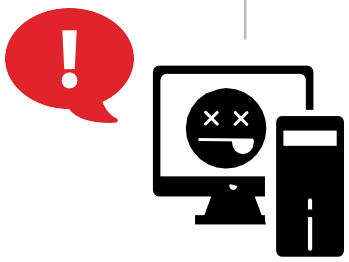


0.0008 M pixels
28x28, 8 bits

785 K parameters

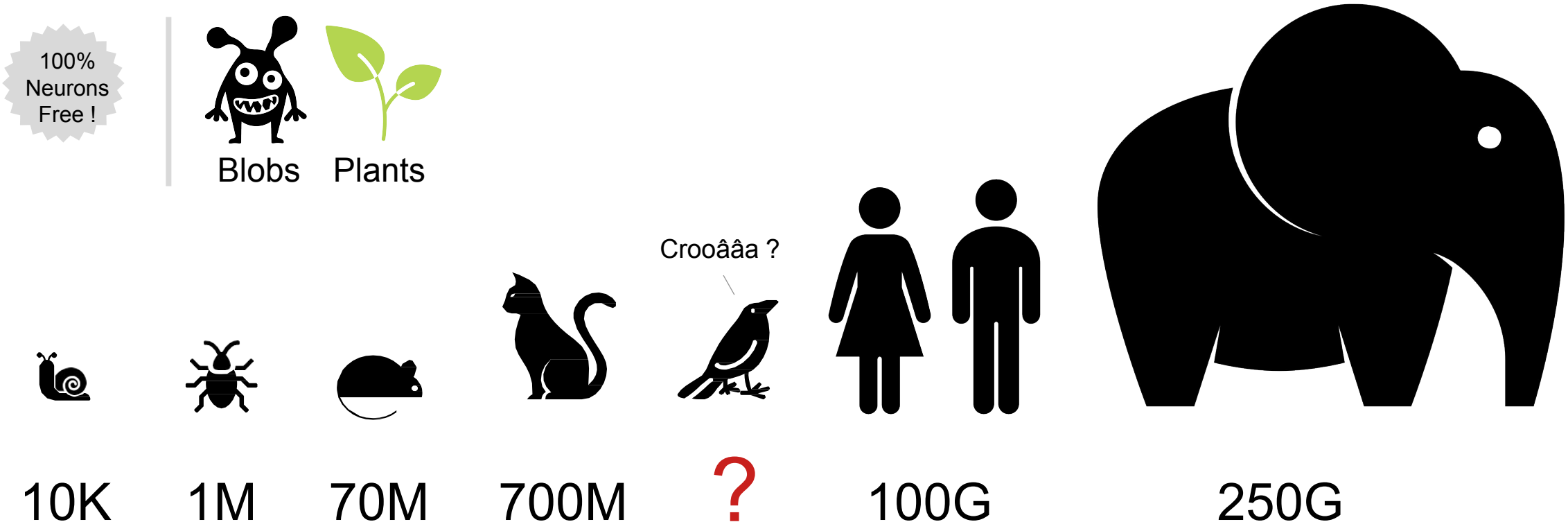
24 M pixels
(r,v,b) 3x8 bits

**72 G** parameters...



Short didgression !

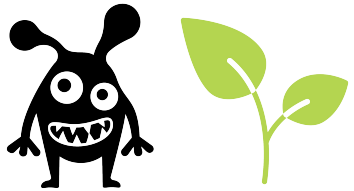
# One neuron is good... but more than one is better !



Short didgression !

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

100% Neurons Free !

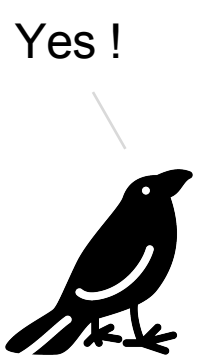






10K 1M 70M 700M 2G 100G 250G

Short didgression !

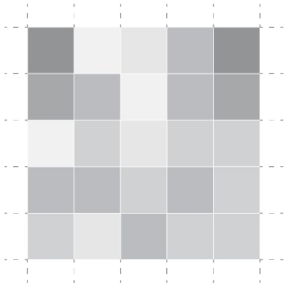



https://youtube.com/shorts/n1AyE3kBrNY

# Principle of convolutions

## 2D convolution

By Jan Kroon, from Pexels.com



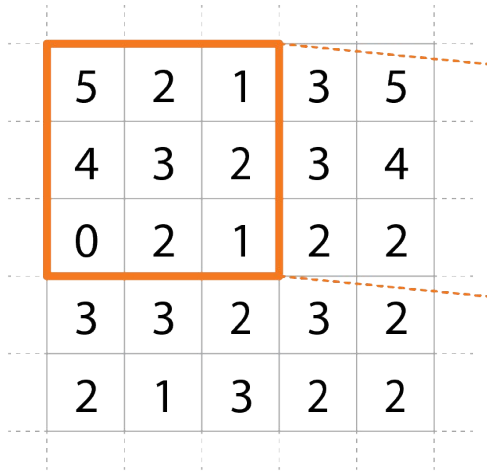



Image piece

| | | |
|---|---|---|
| 5 | 2 | 1 |
| 4 | 3 | 2 |
| 0 | 2 | 1 |

X

⊗

Kernel 3x3

| | | |
|---|---|---|
| 1 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 1 |

ω

=

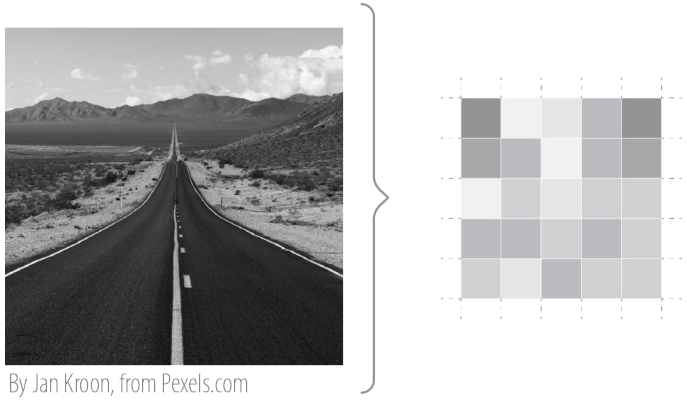
| |
|---|
| 10 |

y

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

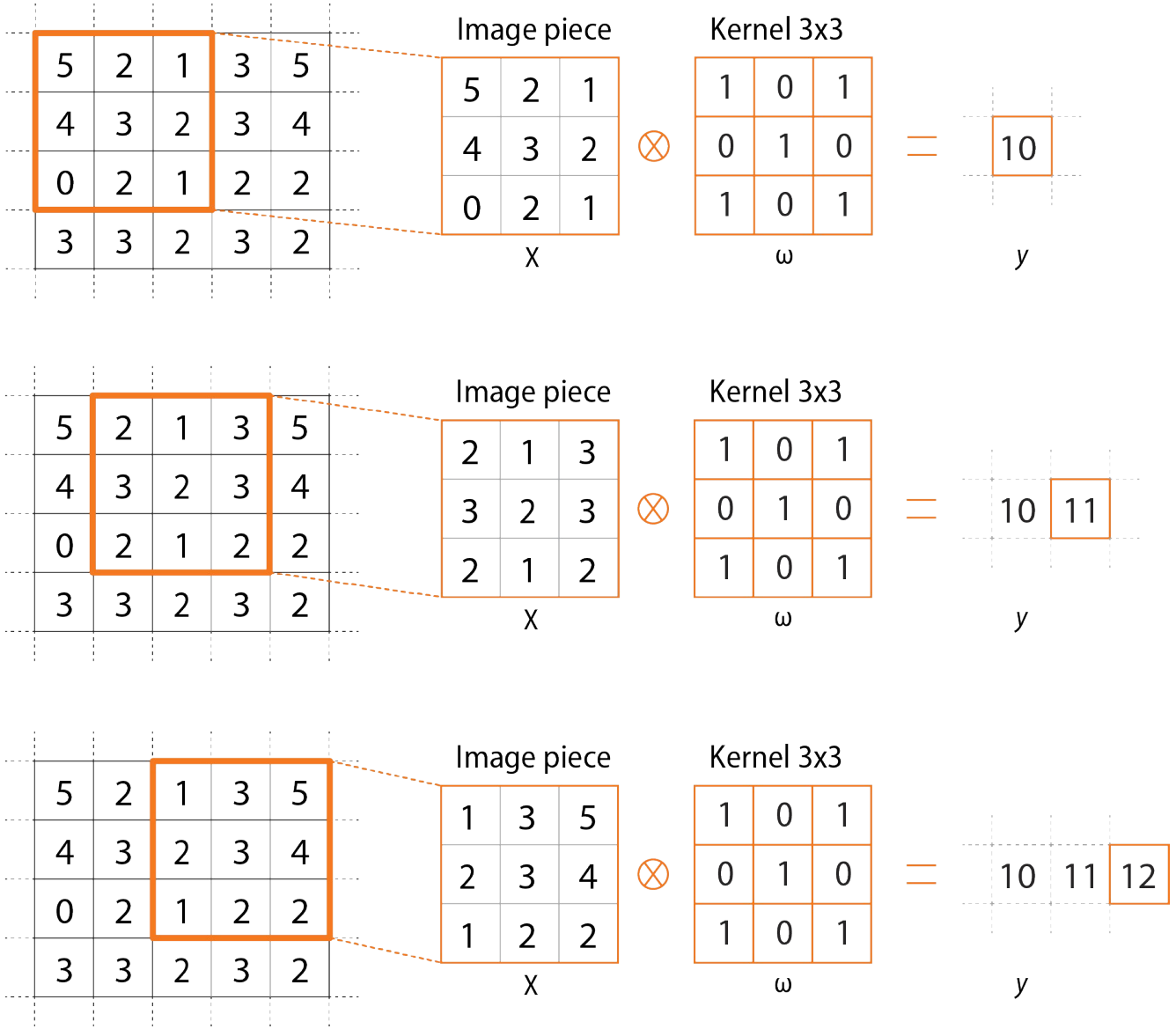
$$y = \sum_{i=1}^{n}\sum_{j=1}^{m} x_{i,j} \cdot \omega_{i,j} \quad \text{with} \begin{cases} n & \text{kernel width} \\ m & \text{kernel height} \end{cases}$$

⊗ is Hadamard product



By Jan Kroon, from Pexels.com

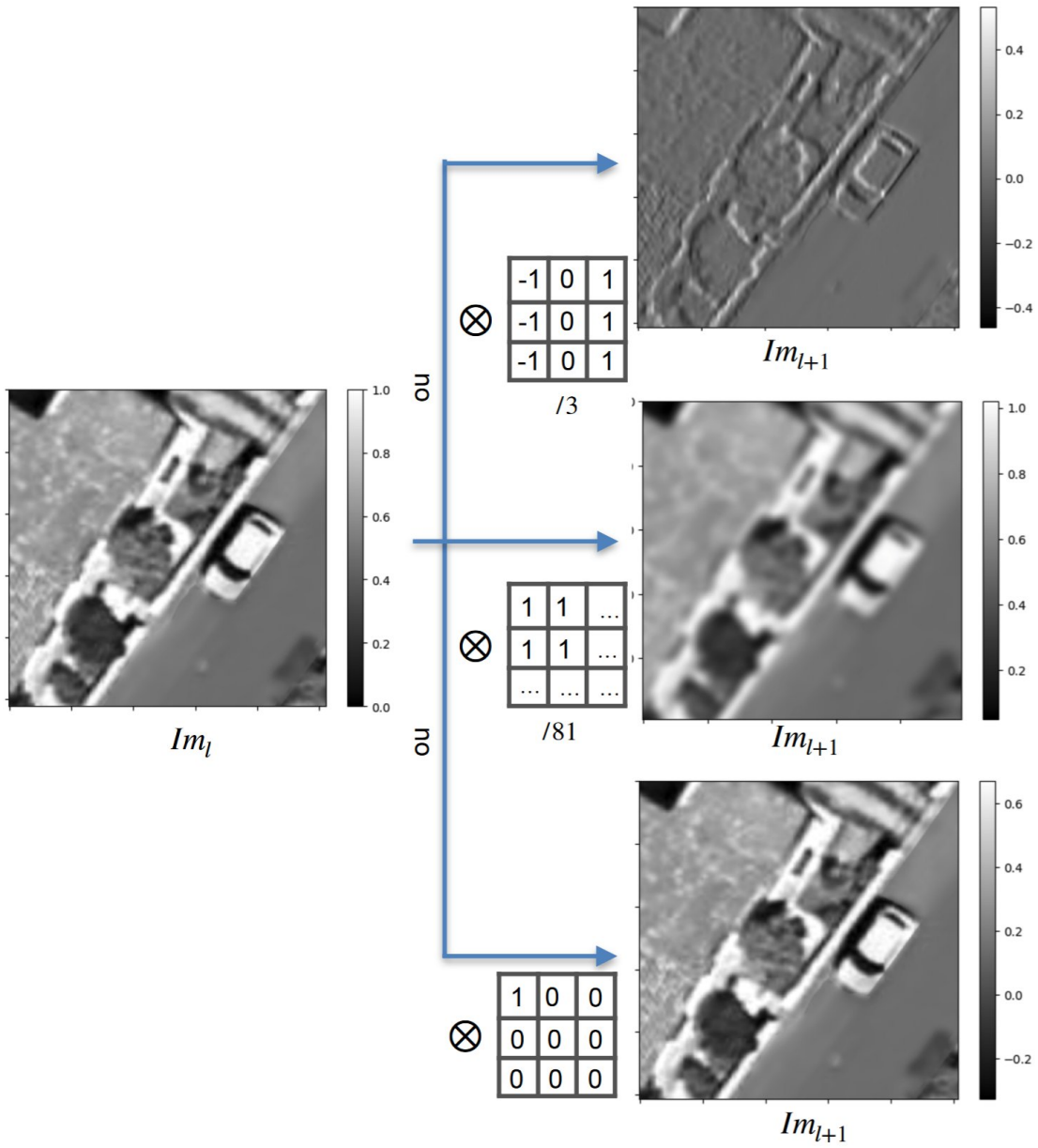





We can perform convolutions in 1, 2, 3 or n-dimensional spaces !

⊗ is Hadamard product

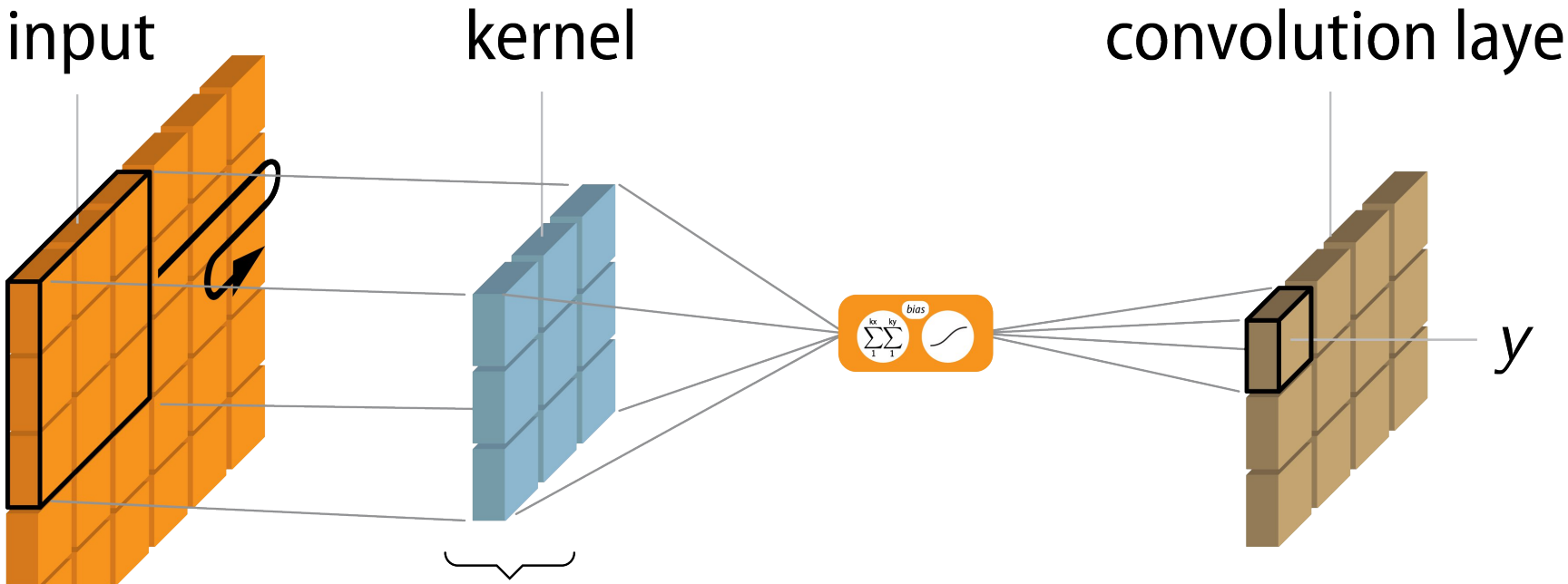
# Interest of convolutions




5

**Mathematics, gradients everywhere !**

# Convolutional layers



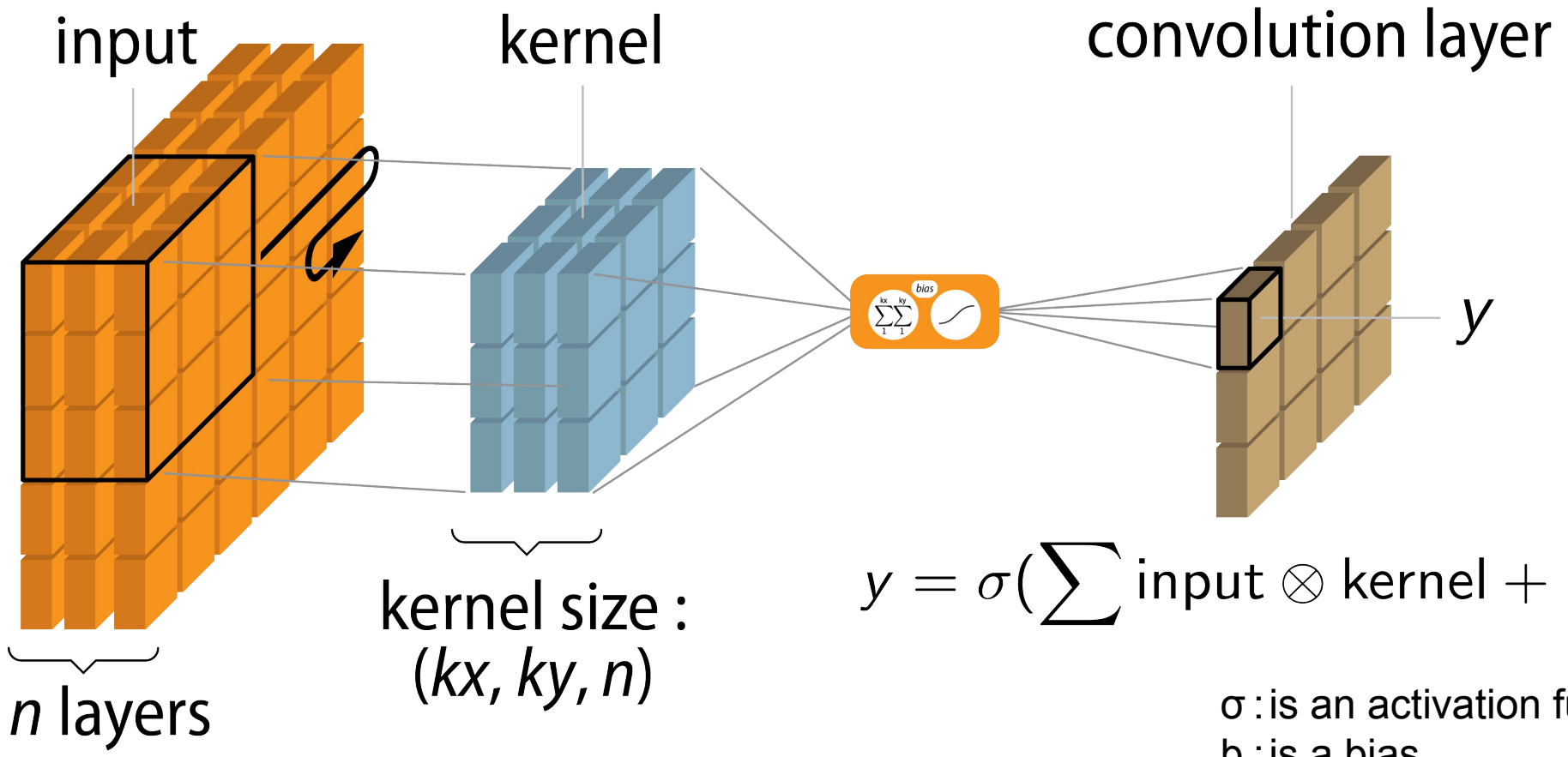

kernel size :
(*kx*, *ky*)

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

σ : is an activation function
b : is a bias

Number of parameters or a convolutional layer : $kx \cdot ky + 1$

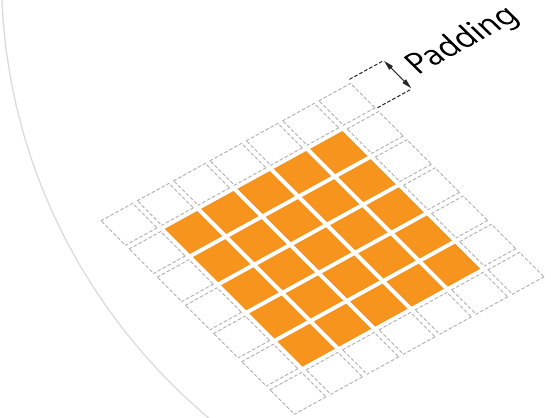
# Convolutional layers

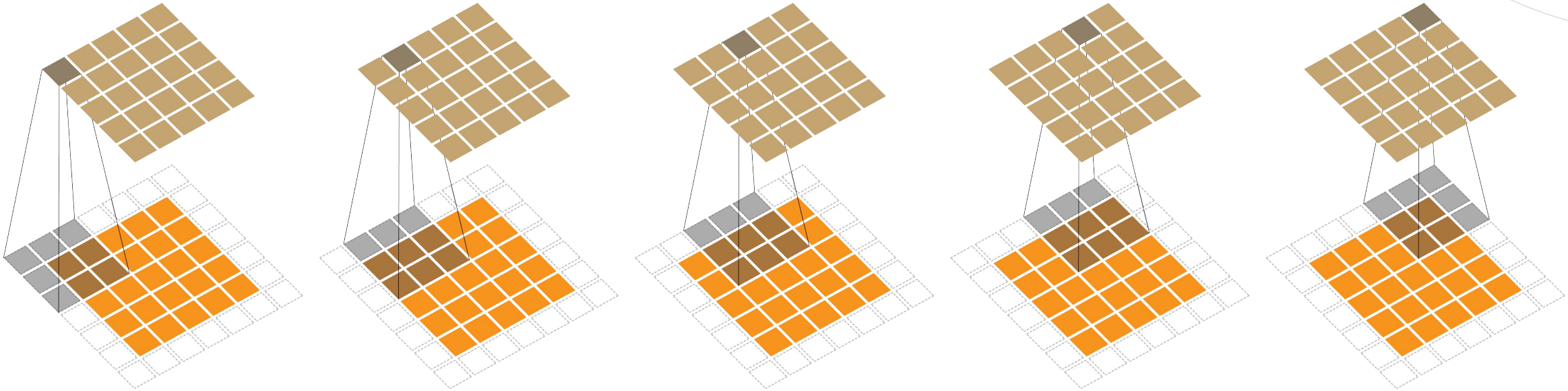



If we want to generate *m* convolutional layers,
we will need *m* convolutional neurons

# Convolution parameters

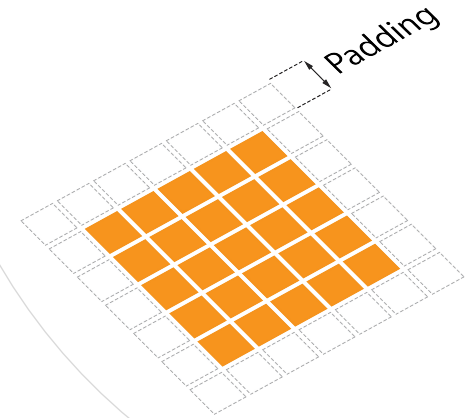
Parameters of a convolutional layer : **Padding**



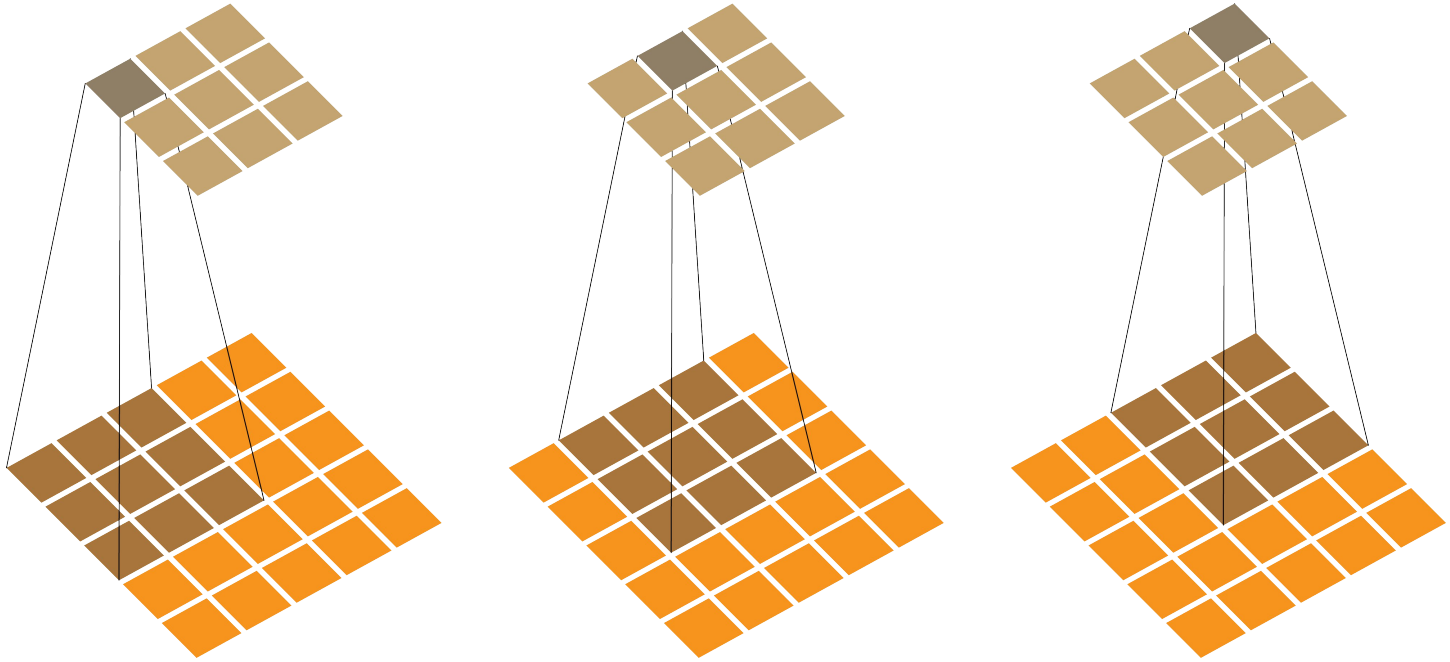



Good padding ➜ Size is preserved

# Convolution parameters



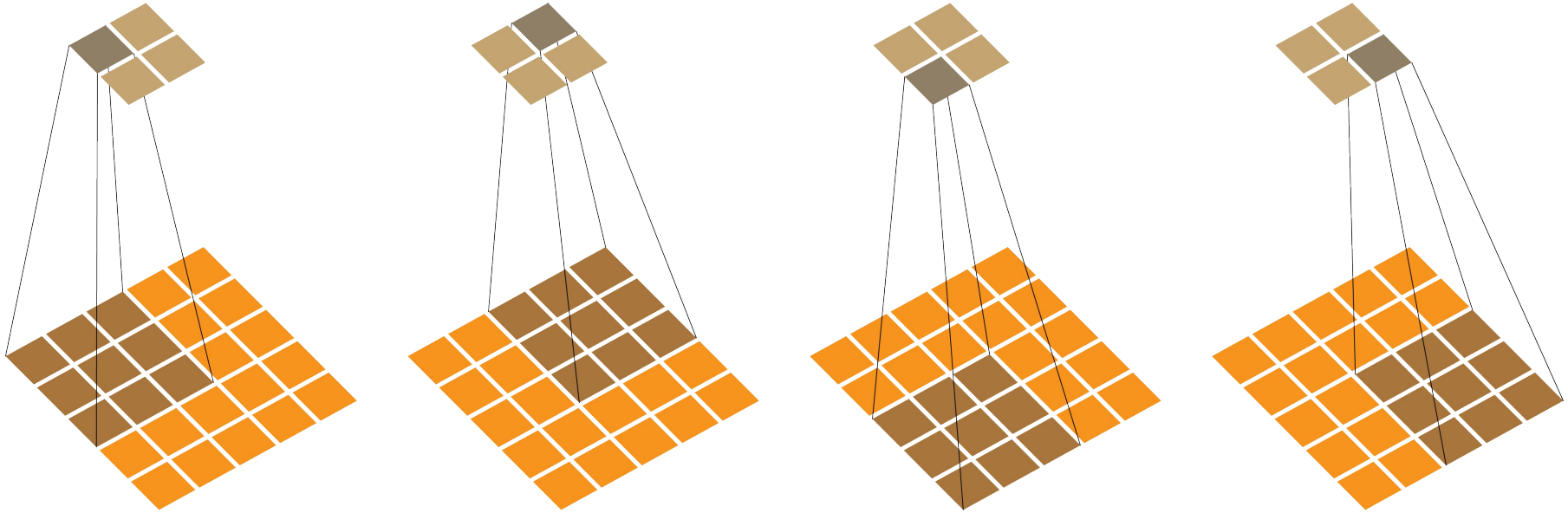

Parameters of a convolutional layer : **Padding**

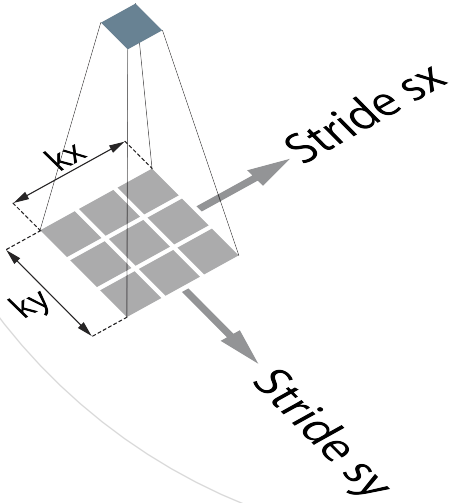


No padding ➜ Size is not preserved

# Convolution parameters

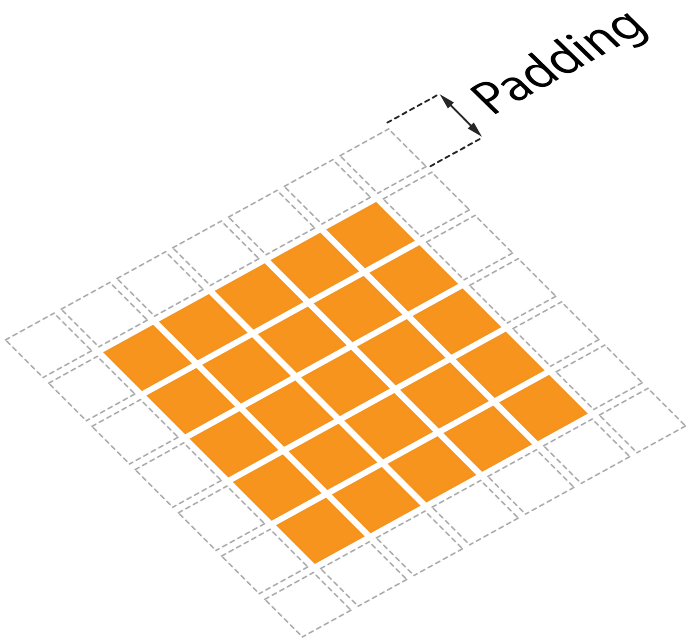
Parameters of a convolutional layer : **Strides**



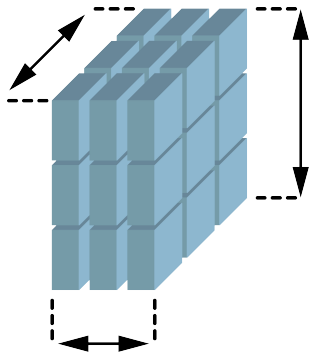



strides = (sx,sy) ➜ A strides=(2,2) will reduce by 2 the output size

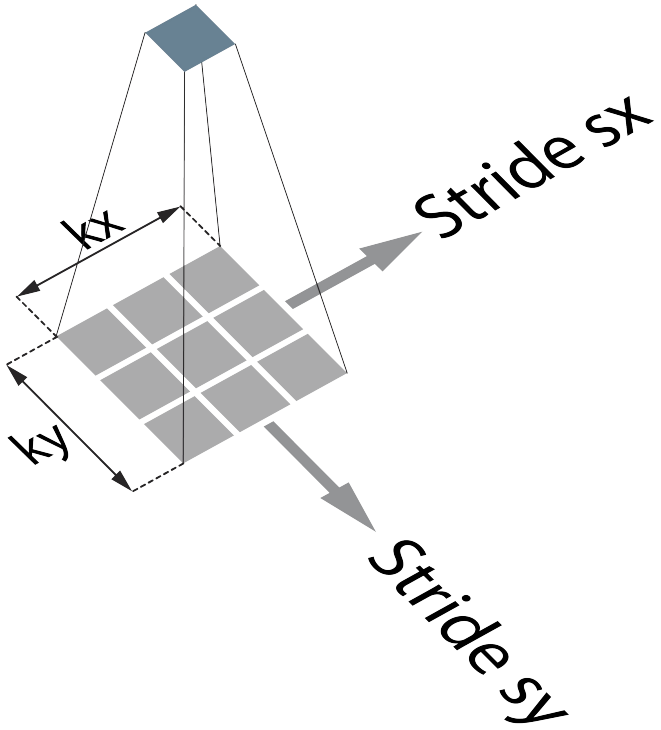
# Convolution parameters




Padding



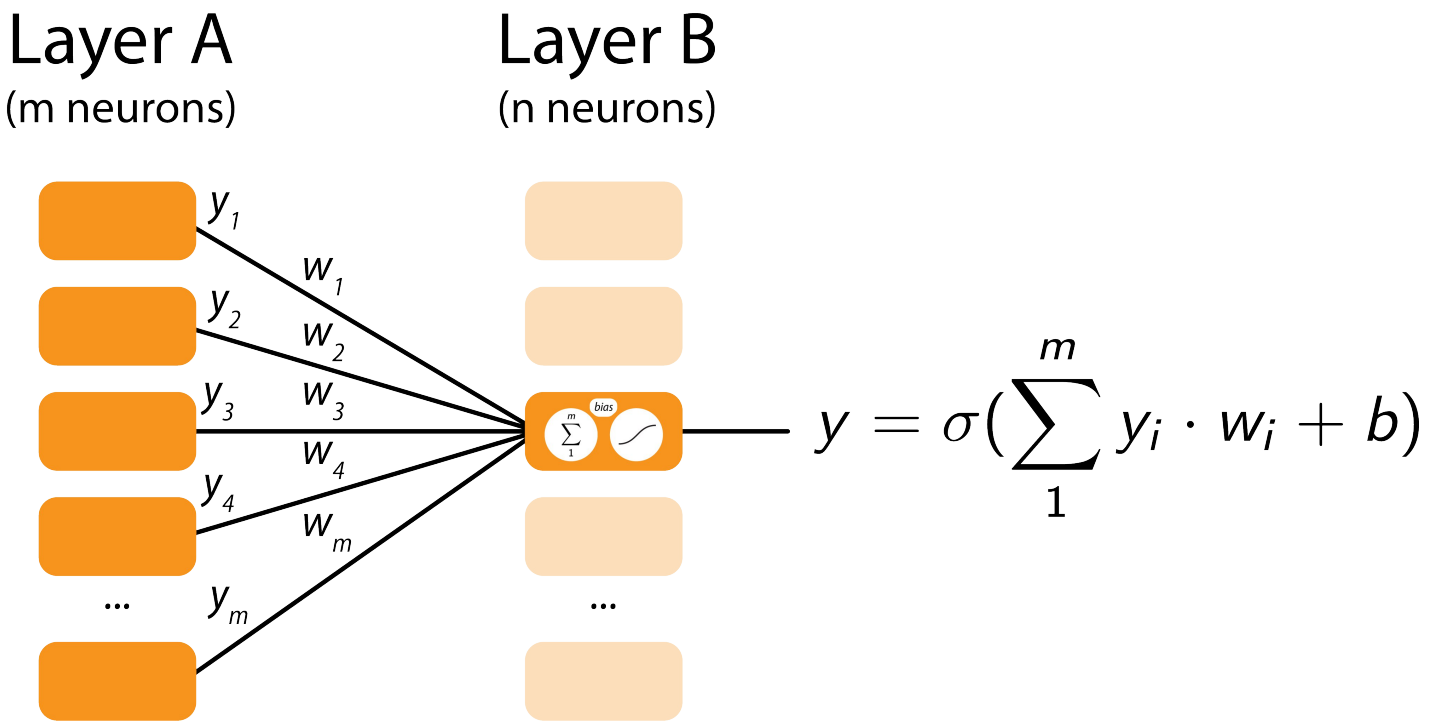
Kernel size




Strides

# Number of parameters

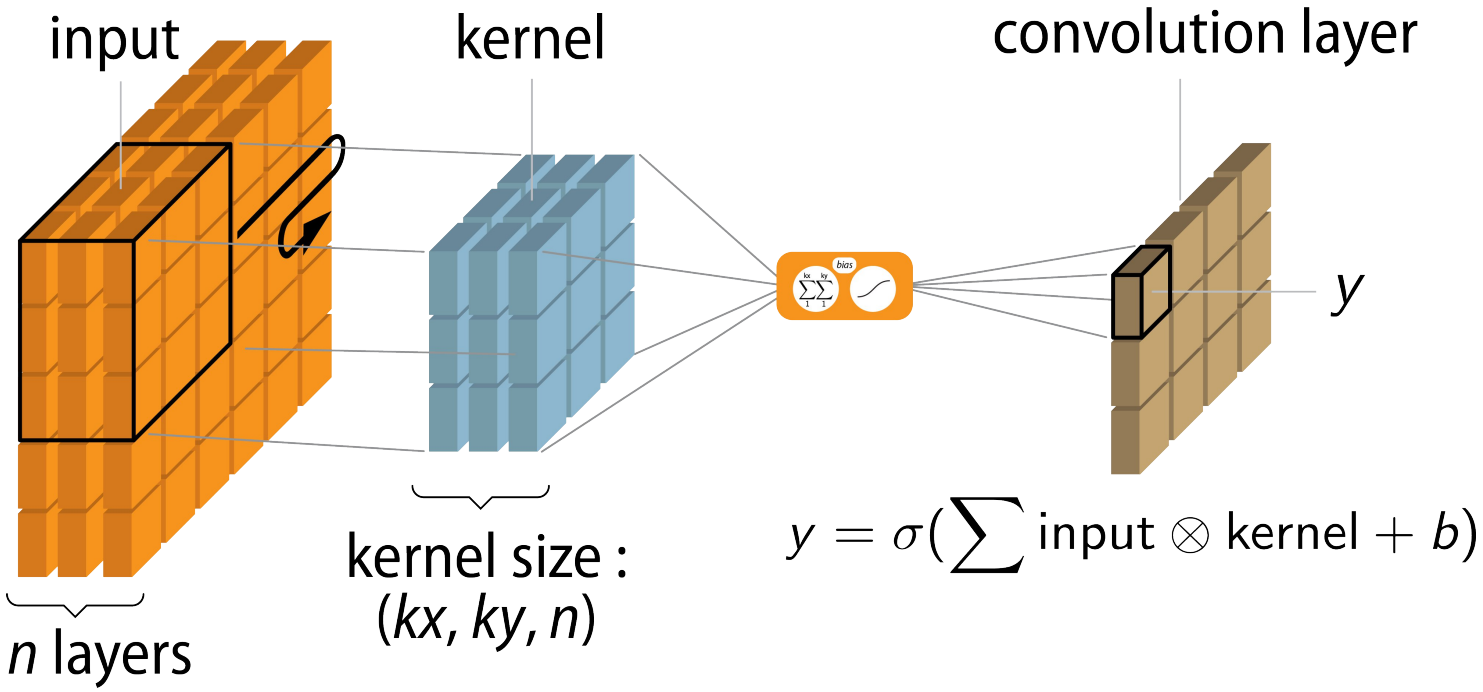
For a fully connected layer :




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

# Number of parameters

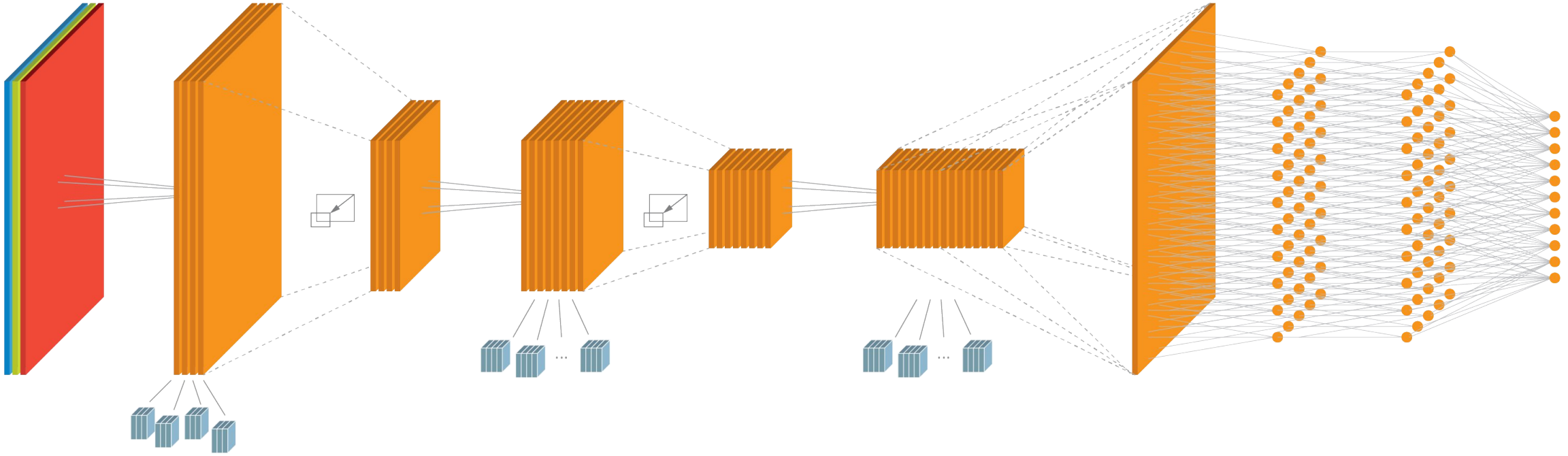
For a convolutional layer :



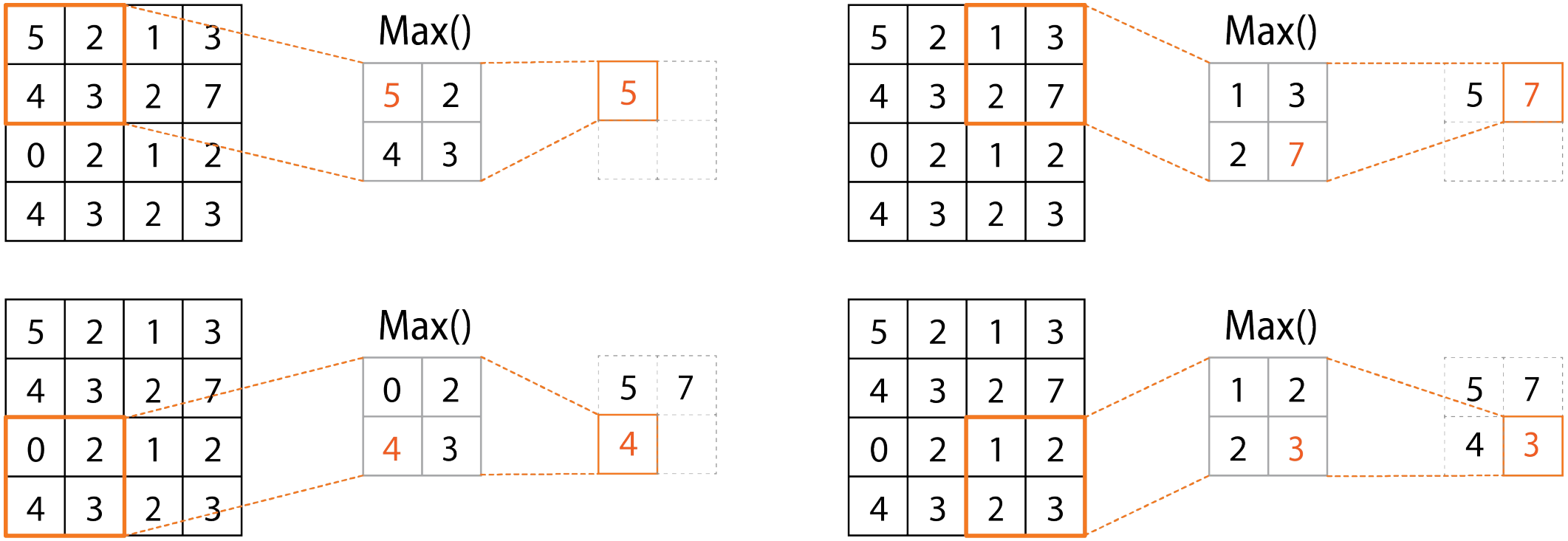

Number of parameters for a single convolutional layer : *n.kx.Ky + 1*

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)



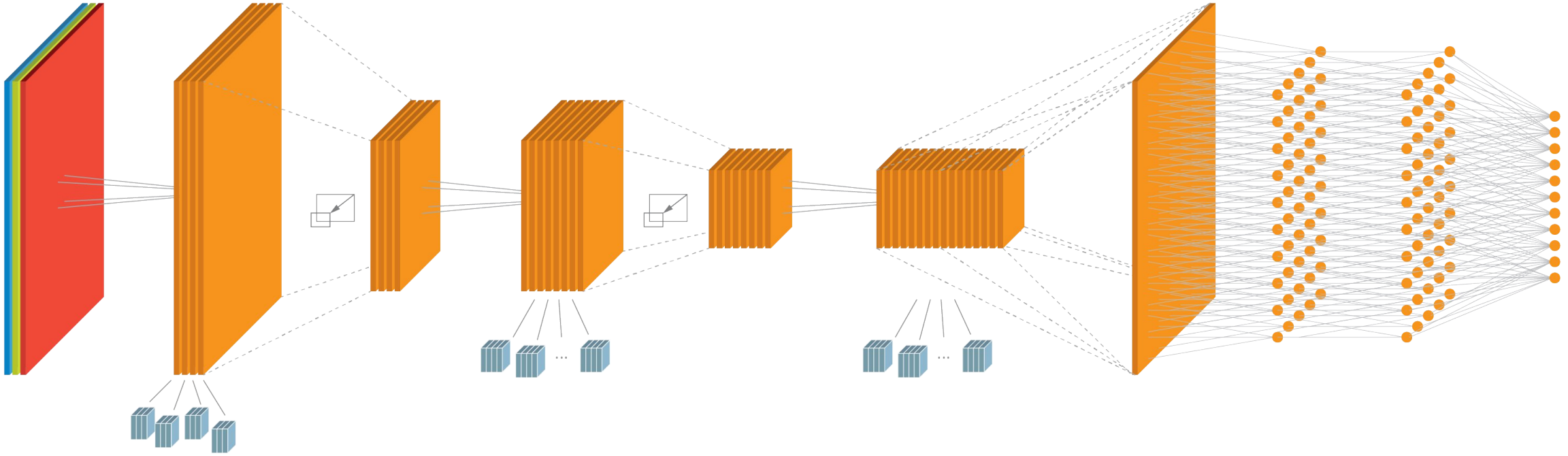
# Max Pooling




It is possible to set the window size, padding and strides.
By default, the strides correspond to the size of the window.

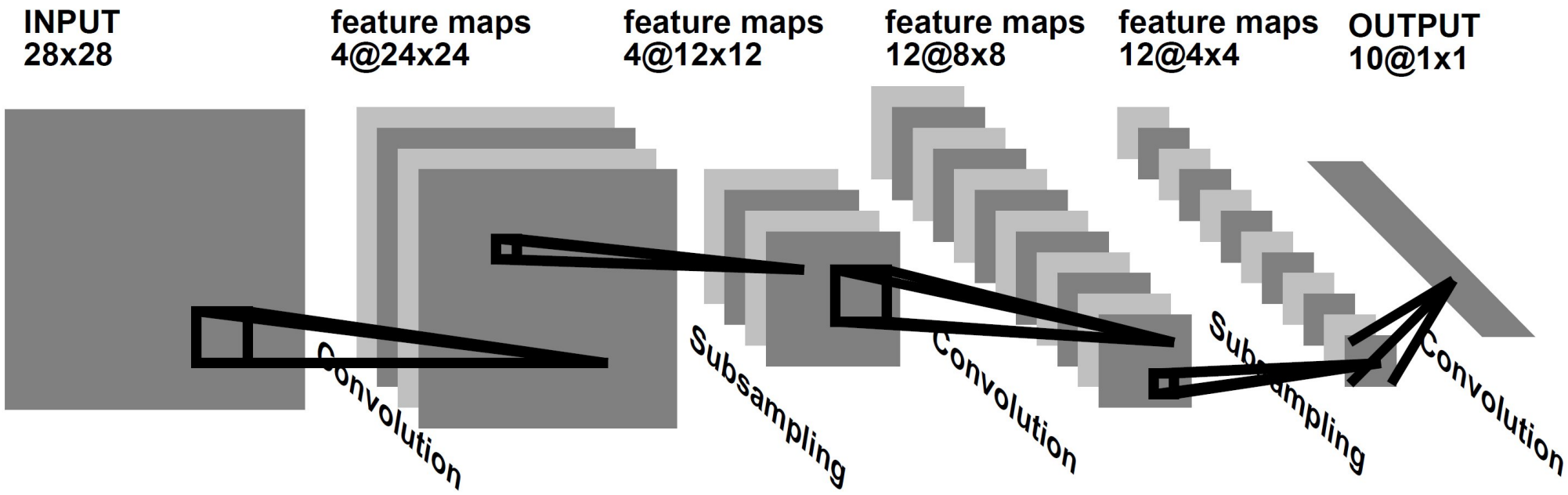
A window (2,2) generates an image twice as small.

# Convolutional Neural Networks (CNN)



# Convolutional Neural Networks (CNN)

## LeNet (1995)

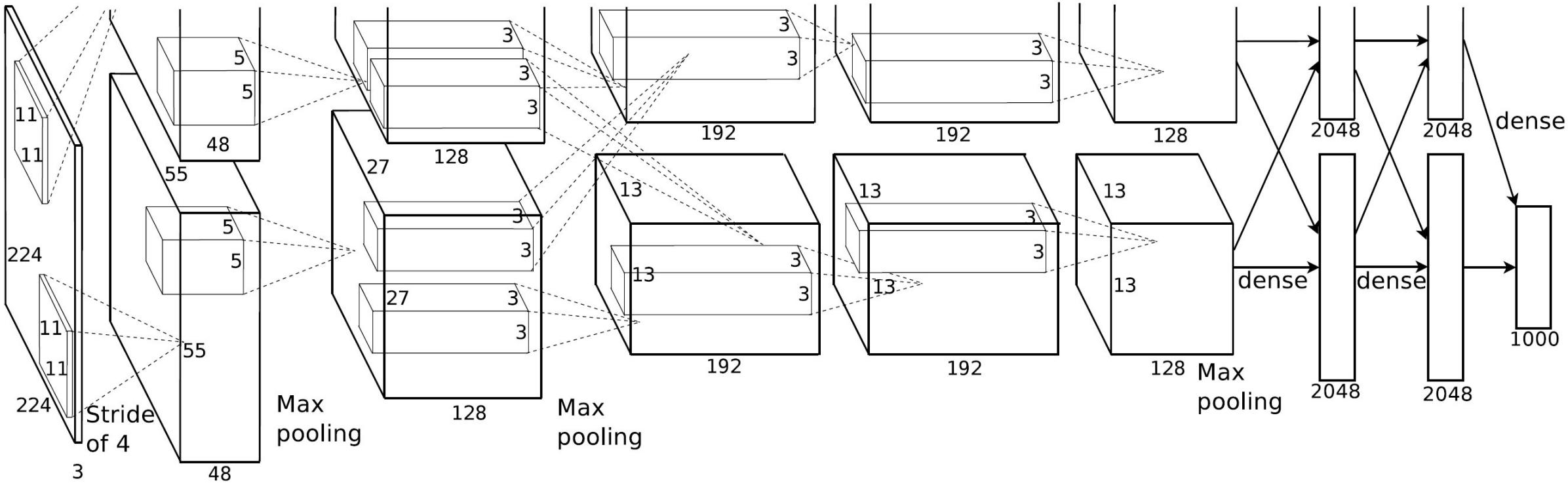



**« Learning algorithms for classification: A comparison on handwritten digit recognition »**
Yann LeCun, Lawrence D Jackel, Léon Bottou, Corinna Cortes, John S Denker, Harris Drucker, Isabelle Guyon, Urs A Muller, Eduard Sackinger, Patrice Simard, Vladimir Vapnik. (1995)
http://yann.lecun.com/exdb/publis/pdf/lecun-95a.pdf

**« Gradient-Based Learning Applied to Document Recognition »**
Yann Lecun, Leon Bottou, Yoshua Bengio, Patrick Haffner. (1998)
http://vision.stanford.edu/cs598_spring07/papers/Lecun98.pdf

# Convolutional Neural Networks (CNN)

## AlexNet (2012)




« Imagenet classification with deep convolutional neural networks »
Alex Krizhevsky, Ilya Sutskever, Geoffrey E Hinton, 2012
https://dl.acm.org/doi/10.1145/3065386

# Classic use of CNNs

## Classification

cat, 0.93

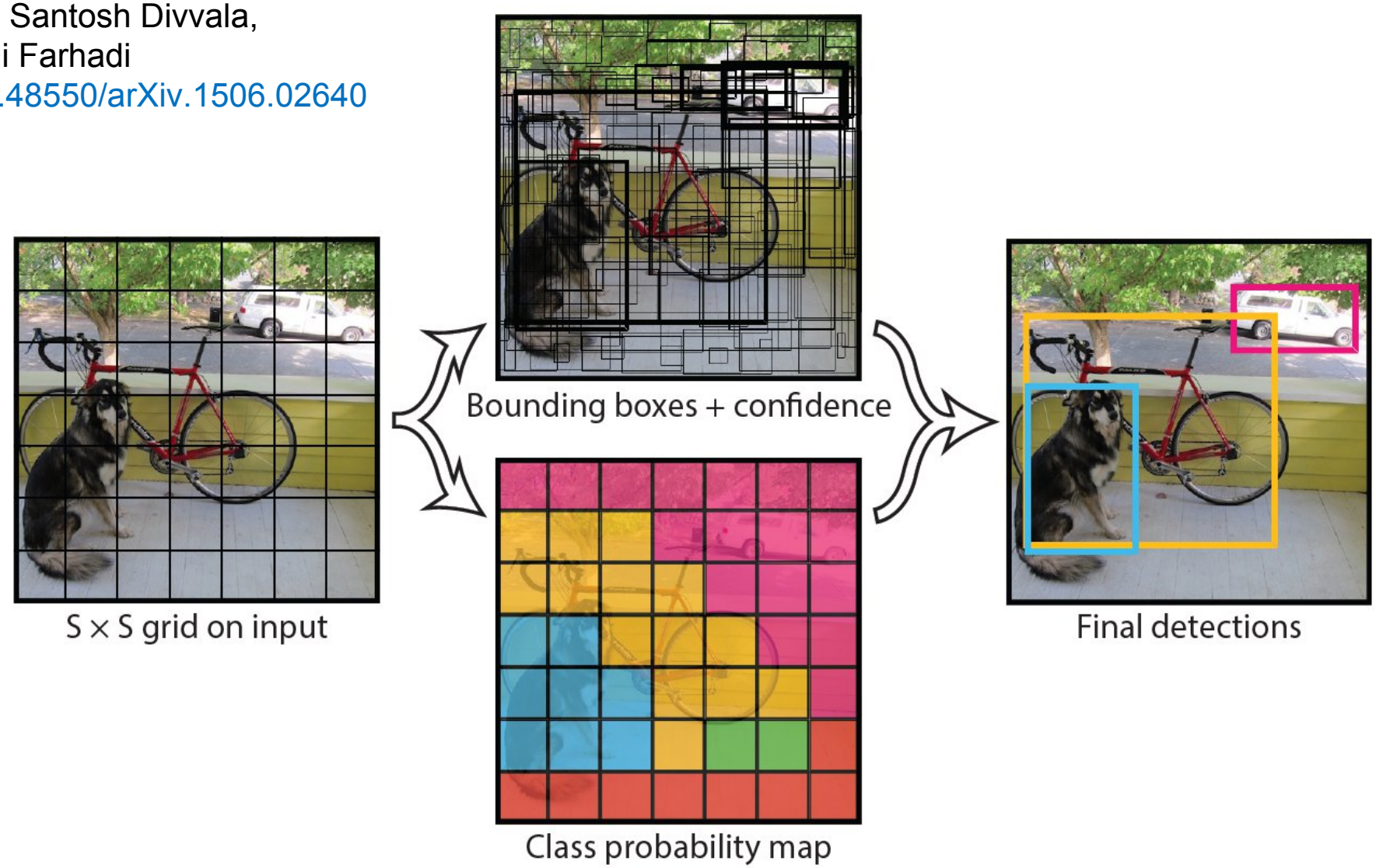
## Detection


## Segmentation


Photo de Loan sur Unsplash

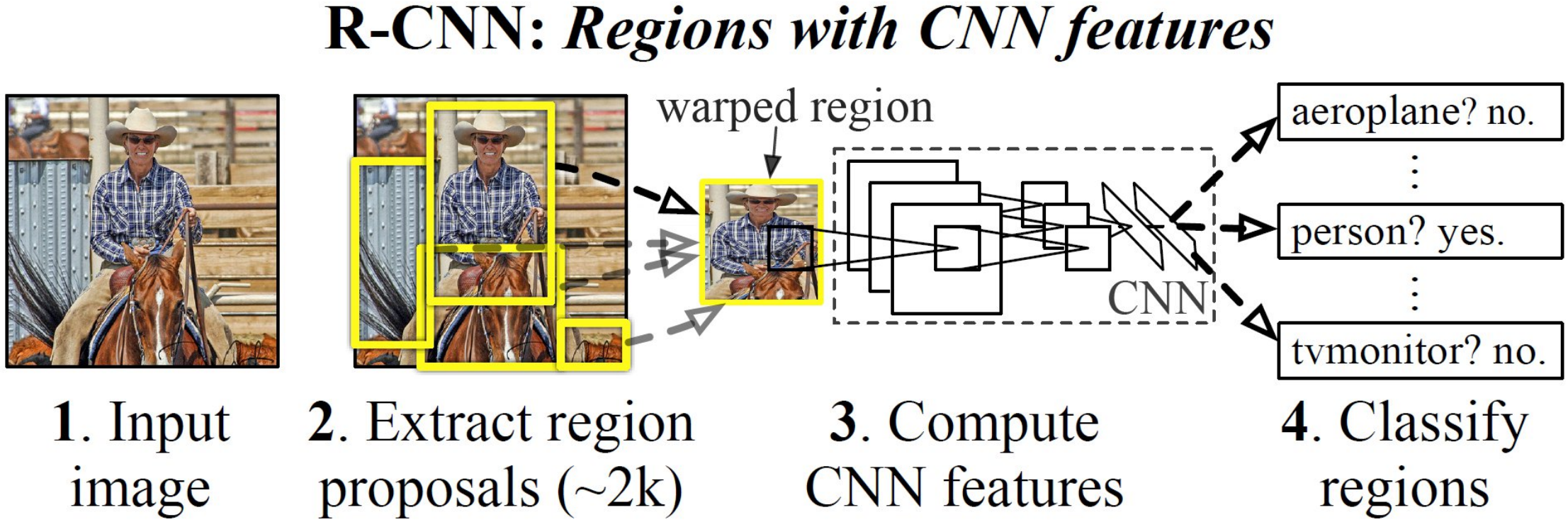
# An example of objecty detection model : YOLO

**You Only Look Once:**
**Unified, Real-Time Object Detection**
Joseph Redmon, Santosh Divvala,
Ross Girshick, Ali Farhadi
https://doi.org/10.48550/arXiv.1506.02640

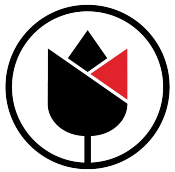


# Another example : R-CNN

**Rich feature hierarchies for accurate object detection and semantic segmentation**
Ross Girshick, Jeff Donahue, Trevor Darrell, Jitendra Malik
https://doi.org/10.48550/arXiv.1311.2524



7



**Convolutional models**

CNN

1 What is a **Convolutional Neuron Network** (CNN) ?

- Understanding what a CNN is
- Identify use cases

2 Example 1 : MNIST

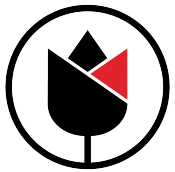
- A verys simple case

3 Example 2 : GTSRB

- A more elaborate case

And now, let's use our neurons !

7



**Convolutional models**

CNN

1 What is a **Convolutional Neuron Network** (CNN) ?

- Understanding what a CNN is
- Identify use cases

2 Example 1 : MNIST

- A very simple case

3 Example 2 : GTSRB

- A more elaborate case

# Image classification with CNN

Notebook : [K3MNIST2]

**Objective :**

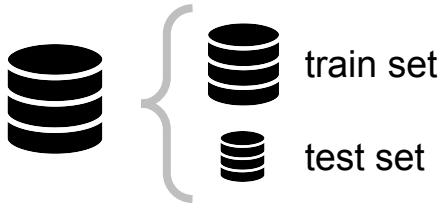
Recognizing handwritten numbers

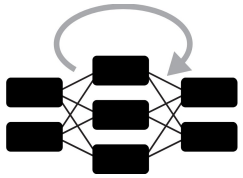
**Dataset :**

Modified National Institute of Standards and Technology (MNIST)

# K3MNIST2

180 seconds
on a CPU :-)



Import and init

Preparing the data

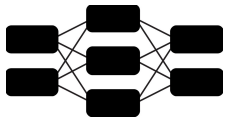
Train the model

START

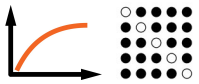
1 → 2 → 3 → 4 → 5 → 6 → END

Retrieve data

Build a model



Evaluate

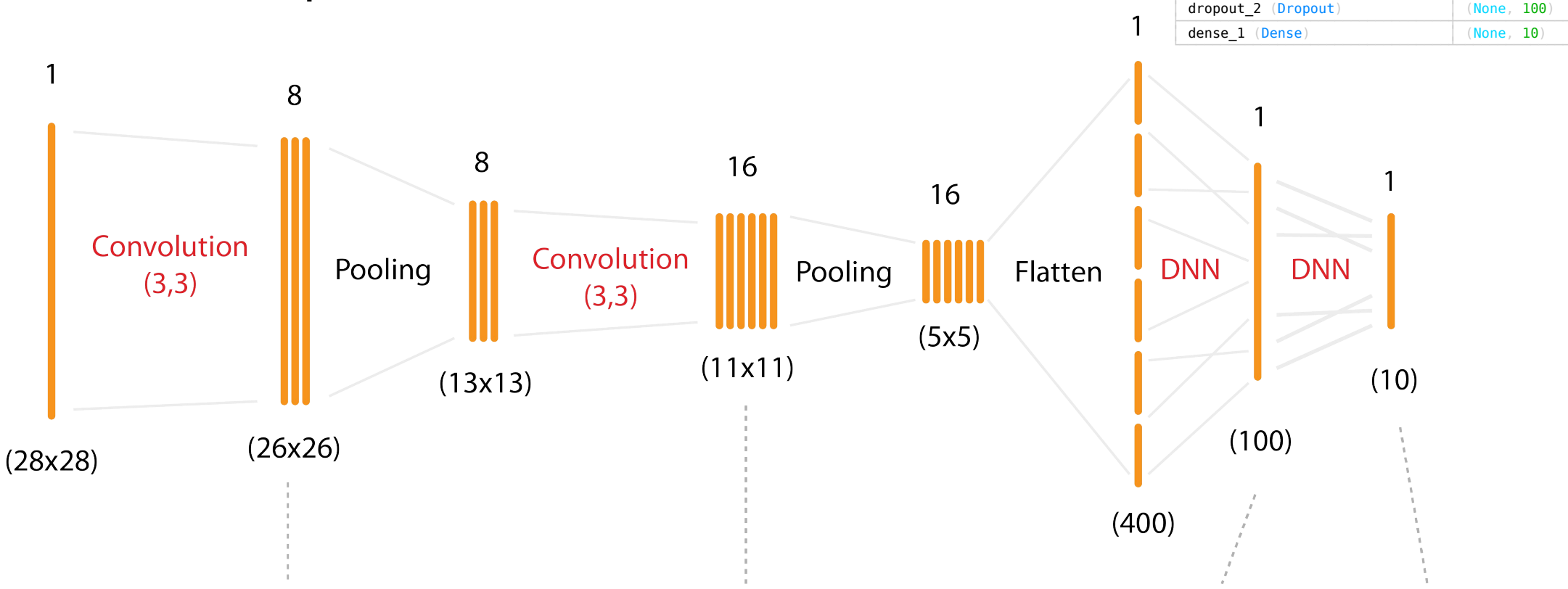


## Objectives :

Classify handwritten numbers (MNIST dataset) via a CNN

| Layer (type) | Output Shape | Param # |
|---|---|---|
| conv2d (Conv2D) | (None, 26, 26, 8) | 80 |
| max_pooling2d (MaxPooling2D) | (None, 13, 13, 8) | 0 |
| dropout (Dropout) | (None, 13, 13, 8) | 0 |
| conv2d_1 (Conv2D) | (None, 11, 11, 16) | 1,168 |
| max_pooling2d_1 (MaxPooling2D) | (None, 5, 5, 16) | 0 |
| dropout_1 (Dropout) | (None, 5, 5, 16) | 0 |
| flatten (Flatten) | (None, 400) | 0 |
| dense (Dense) | (None, 100) | 40,100 |
| dropout_2 (Dropout) | (None, 100) | 0 |
| dense_1 (Dense) | (None, 10) | 1,010 |

# Nombre de paramètres :




(3x3 + 1)*8

80 params

(8x3x3 + 1)*16

1168 params

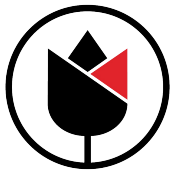
100*(400 + 1)

40100 params

10*(100 + 1)

1010 params

7



**Convolutional models**

CNN

1 What is a **Convolutional Neuron Network** (CNN) ?

- Understanding what a CNN is
- Identify use cases

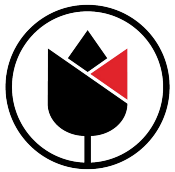
2 Example 1 : MNIST

- A very simple case

3 Example 2 : GTSRB

- A more elaborate case

7



**Convolutional models**

CNN

1 What is a **Convolutional Neuron Network** (CNN) ?

- Understanding what a CNN is
- Identify use cases

2 Example 1 : MNIST

- A very simple case

3 Example 2 : GTSRB

- A more elaborate case


# Image classification with CNN

Notebook : [K3GTSRB]

**Objective :**

Recognizing trafic signs

**Dataset :**

German Traffic Sign Recognition Benchmark (GTSRB) is a dataset with more than **50,000 photos** of **road signs** from about **40 classes**

http://benchmark.ini.rub.de/?section=gtsrb&subsection=dataset


# Image classification with CNN

Notebook : [K3GTSRB]

**Objective :**

Recognizing trafic signs

**Notebooks :**

| | |
|---|---|
| K3GTSRB1 : | Data analysis and creation of a usable dataset |
| K3GTSRB2 : | First convolutions and first results |
| K3GTSRB3 : | Monitoring training and managing checkpoints |

Calculation scale :
`scale`

- `scale = 1` — Use 100% of the dataset, Need a GPU !
- `scale = 0.1` — Use 10% of the dataset, For dev without GPU

Enhanced dataset location :
output
enhanced_dir

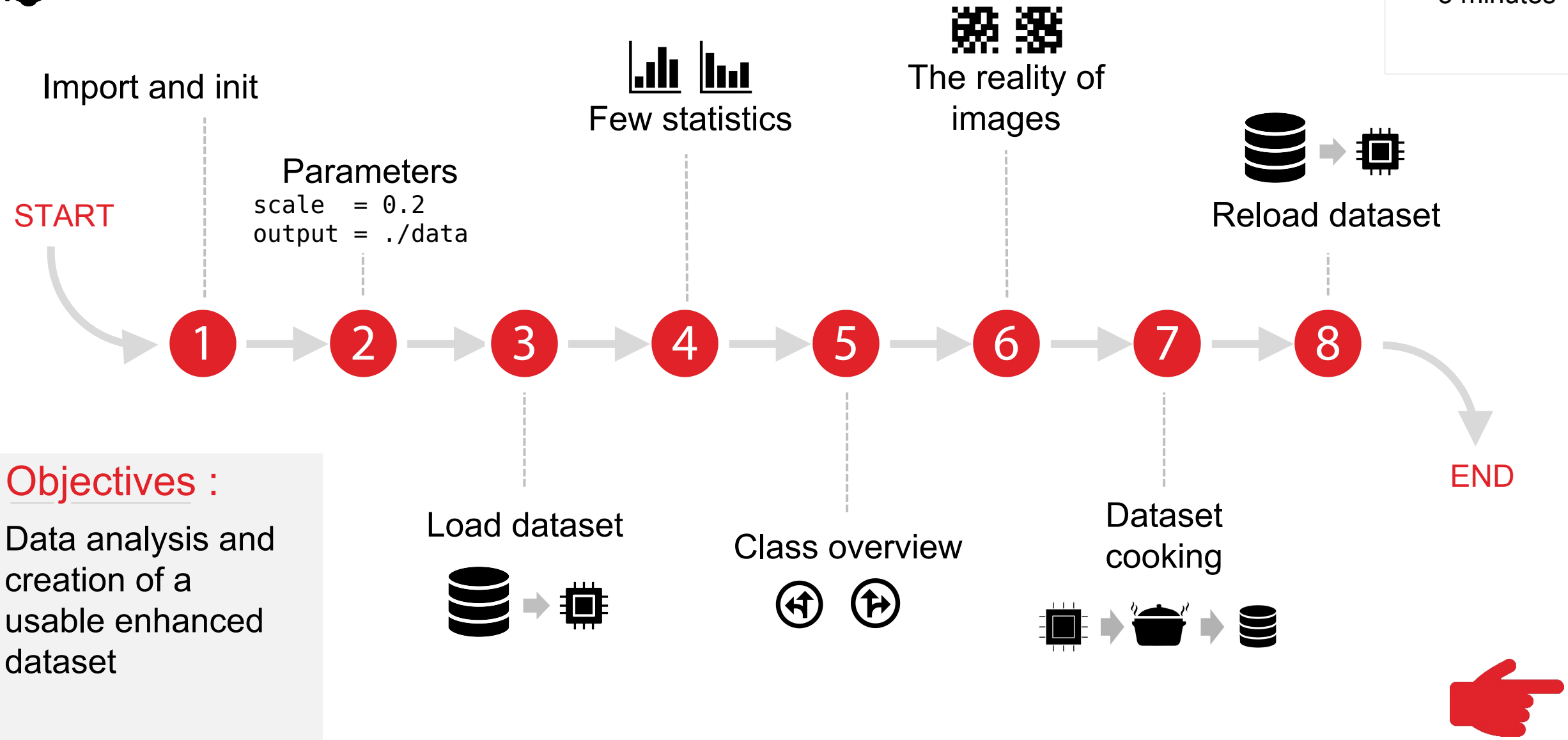
- `./data` — For tests and small scale
- `f'{datasets_dir}/GTSRB/enhanced'` — For production and scale 1

Notebooks outputs :
run_dir

- `./run/*` — By default

# K3GTSRB1
## Dataset construction

3 minutes




## Objectives :

Data analysis and creation of a usable enhanced dataset

# K3GTSRB2
## First convolutions

120 seconds
on a CPU :-)
(5 epochs)

train set

test set

Preparing the data

Import and init

```
scale  = 1
output = ./data
```

Train the model

START

1 → 2 → 3 → 4 → 5 → 6 → END

Load a dataset

Build a model

Evaluate

## Objectives :

Make a first classification via a convolutional network

# K3GTSRB1
## Second and better convolutions

3 minutes

Import and init

```
scale  = 1
output = ./data
```

START

1 → Load a dataset

2 → Create a model

3 → Train the model

TensorBoard

4 → Reload model

5 → Preparing the data

train set

test set

6 → Prepare callbacks

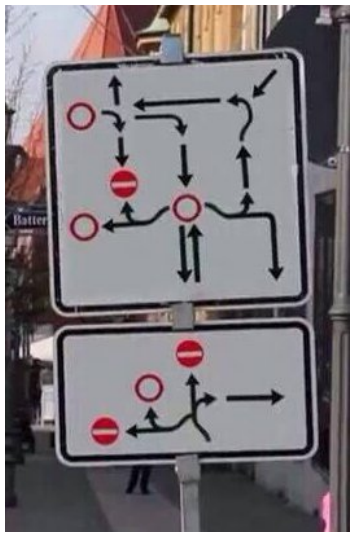
7 → Evaluation

8 → END

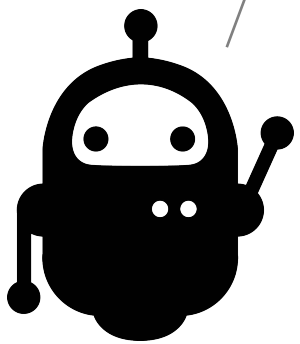
(90.7 %)

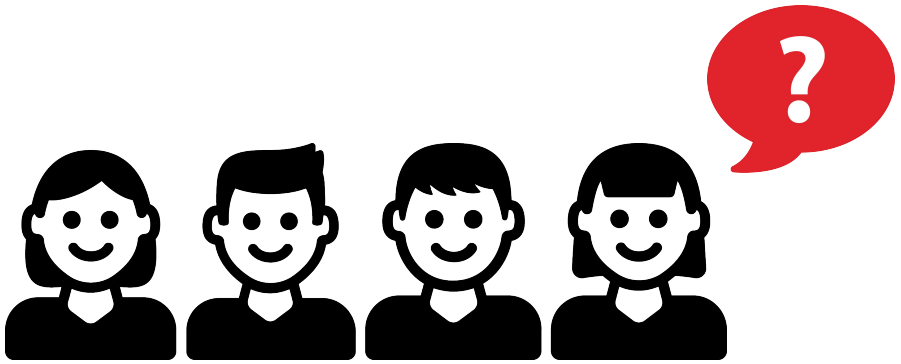
## Objectives :

The same example, but a little more elaborate :-)



Merci beaucoup !!
Quelques questions ?





FIDLE

https://youtube.com/@CNRS-FIDLE
https://fidle.cnrs.fr

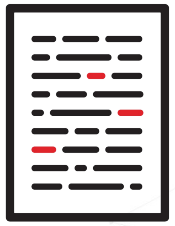


# Jeudi 1er février 2024, 14h

**Next, on Fidle :**

**L'IA comme un outil**

7



**Sparse (text) and sequences data**

Embedding, RNN