SAISON 22/23

Formation

# Introduction Deep Learning

Séquence 2

## Convolutional neural network (CNN) - 1/2

FIDLE

https://fidle.cnrs.fr

Cette session va être enregistrée.
Retrouvez-nous sur notre chaine YouTube :-)

This session will be recorded.
Find us on our YouTube channel :-)

https://fidle.cnrs.fr/youtube

SAISON 22/23

Formation

# Introduction Deep Learning

Séquence 2

## Convolutional neural network (CNN) - 1/2

FIDLE

https://fidle.cnrs.fr

# Resources

https://fidle.cnrs.fr

Powered by CNRS CRIC, and UGA DGDSI of Grenoble, Thanks !

- Course materials (pdf)
- Practical work environment*
- Corrected notebooks
- Videos (YouTube)

(*) Procedure via Docket or pip
Remember to get the latest version !

# Resources

You can also subscribe to :

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

DEVLOG RÉSEAU DU DEVELOPPEMENT LOGICIEL Recherche et Enseignement Supérieur https://listes.services.cnrs.fr/wws/info/devlog[1]

GROUPE CALCUL https://listes.math.cnrs.fr/wws/info/calcul[2]

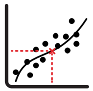
(1) List of ESR* developers,
(2) List of ESR* « calcul » group
Where ESR is Enseignement Supérieur et Recherche, french universities and public academic research organizations

# Previously on Fidle...

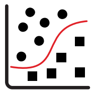
Few little things and concepts to keep in mind

1

**History, Fundamental Concepts**



**Basic Regression**
DNN

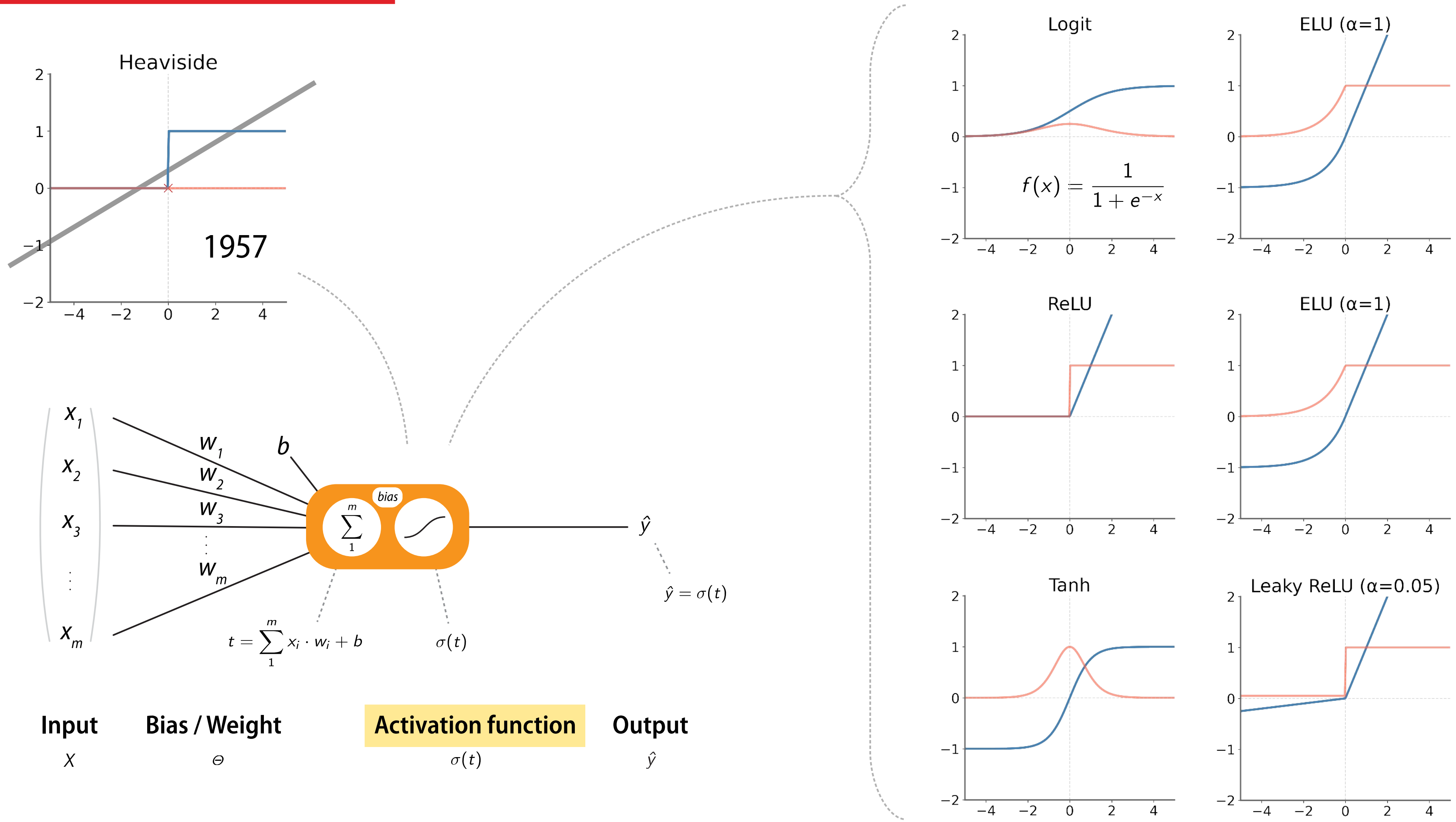


**Basic Classification**
DNN

Easy :-)

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

# Activation functions




Notebook [ACTF1]

# 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

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

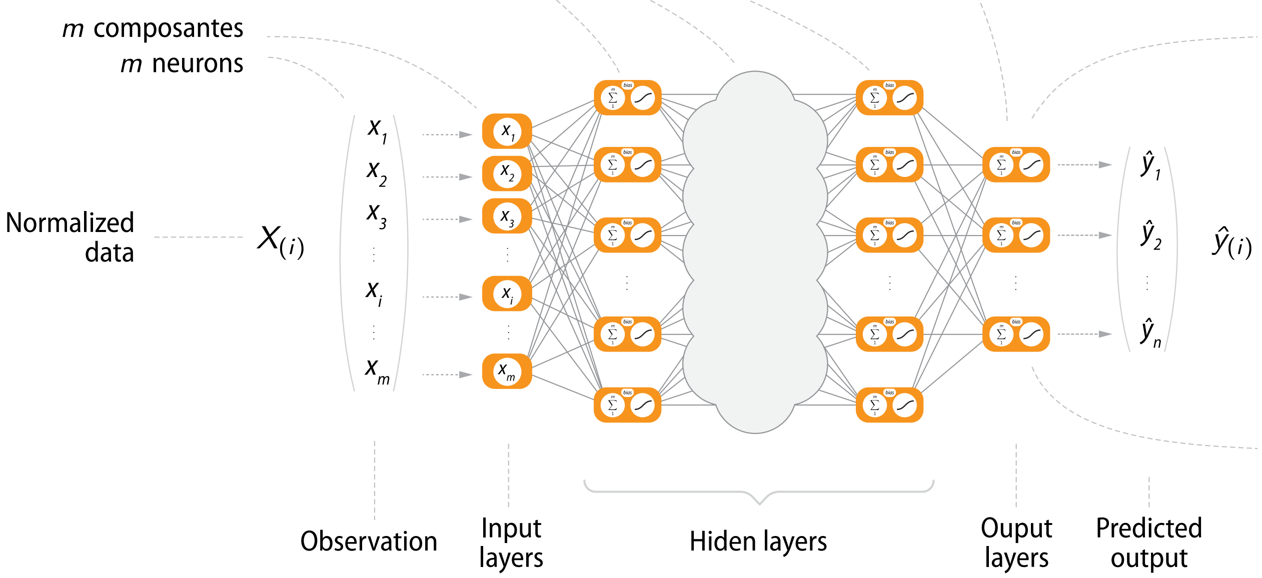
**Muticlass classification**

Activation : Softmax

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

Loss: Categorical cross entropy

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



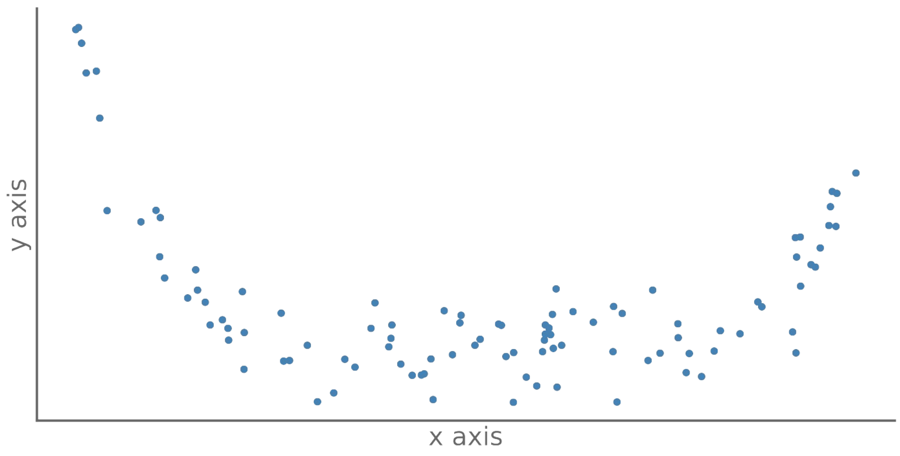

**Regression**

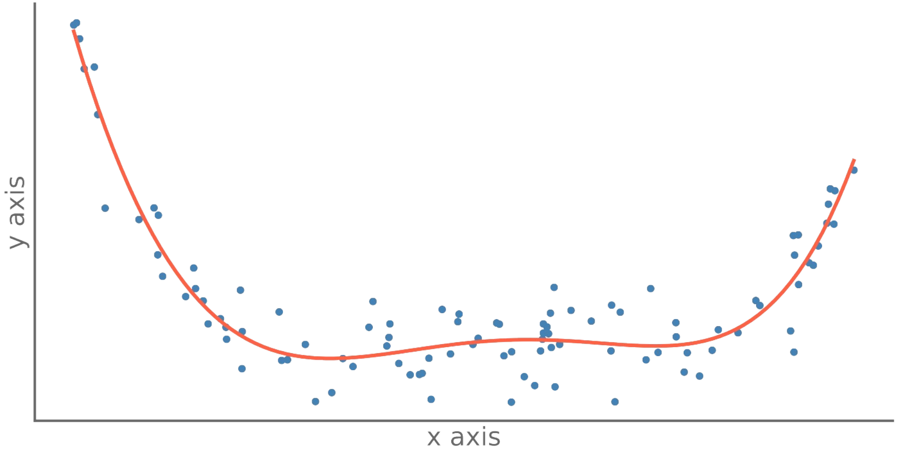
No activation

Loss : MSE, ...

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

# About overfiting

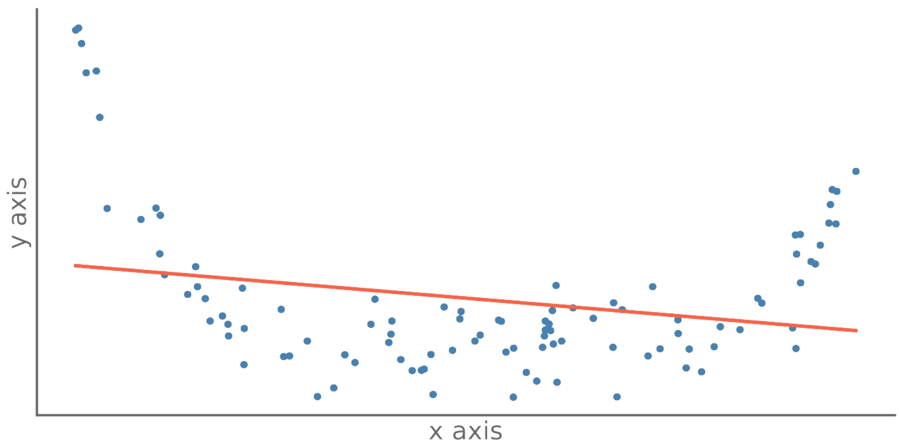


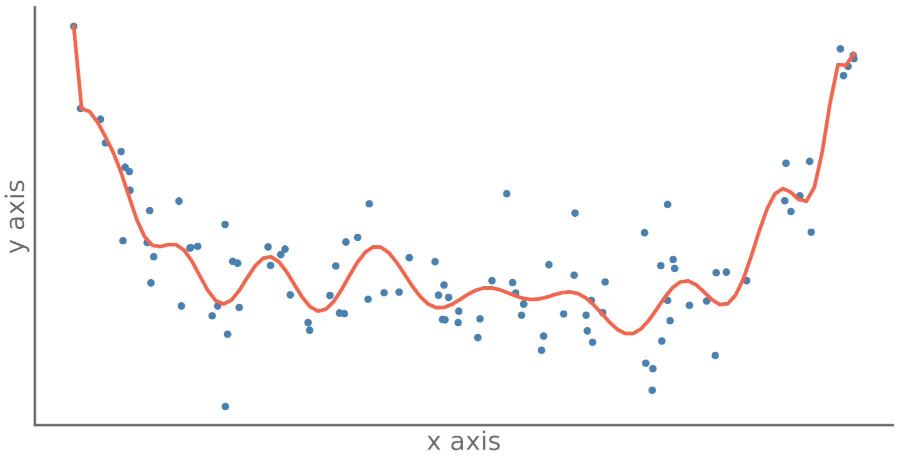

$$P_n(x) = a_0 + a_1 \cdot x + a_2 \cdot x^2 + \cdots + a_n \cdot x^n = \sum_{i=0}^{n} a_i \cdot x^n$$

Notebook [POLR1]

# About overfiting




Underfiting




Overfiting

Notebook [POLR1]

# Program

**20 Séquences**
du 17 novembre
au 14 mai 2023

1. **History, Fundamental Concepts**
   - **Basic Regression** DNN
   - **Basic Classification** DNN

2. 3. **Hight Dimensional Data** CNN

4. Demystify **mathematics** for neural networks.

5. **Training strategies** Evaluation — **Sparse data (text)** Embedding

6. **Sequences data** RNN

7. PyTorch — A small detour with **PyTorch**.

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

9. New ! **Graph Neural Network** GNN

10. **Autoencoder networks** AE

11. **Variational Antoencoder** VAE

12. **Project session** «My project in 180 s»

13. **Generative Adversarial Networks** GAN

14. New ! **Diffusion Model** Text to image

15. **AI, Law, Society** and **Ethics**

16. New ! **Model and training optimization** Resource efficiency

17. **Jean-Zay** GPU acceleration

18. New ! **Physics-Informed Neural Networks** PINNS

19. **Deep Reinforcement Learning** RL

20. **What will be tomorrow's AI** Review & perspectives !

FIDLE

SAISON 22/23

# Program

**20 Séquences**
du 17 novembre
au 14 mai 2023

1. **History, Fundamental Concepts**
   - **Basic Regression** DNN
   - **Basic Classification** DNN
2. 3. **Hight Dimensional Data** CNN
4. Demystify **mathematics** for neural networks.
5. **Training strategies** Evaluation / **Sparse data (text)** Embedding
6. **Sequences data** RNN
7. PyTorch — A small detour with **PyTorch**.
8. **«Attention is All You Need»** Transformers
9. New ! **Graph Neural Network** GNN
10. **Autoencoder networks** AE
11. **Variational Antoencoder** VAE
12. **Project session** «My project in 180 s»
13. **Generative Adversarial Networks** GAN
14. New ! **Diffusion Model** Text to image
15. **AI, Law, Society** and **Ethics**
16. New ! **Model and training optimization** Resource efficiency
17. **Jean-Zay** GPU acceleration
18. New ! **Physics-Informed Neural Networks** PINNS
19. **Deep Reinforcement Learning** RL
20. **What will be tomorrow's AI** Review & perspectives !

FIDLE

SAISON 22/23

# Roadmap

2 3

**Hight Dimensionnal Data**

CNN

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

- Understanding what a CNN is
- Identify use cases

2.2 **Example 1 : MNIST**

- Implementation of a simple case

3.1 **Example 2 : GTSRB**

- The devil is also hiding in the data
- How to work with « large » dataset
- Monitoring the training phase and managing our models
- Improve our results with data augmentation
- Datasets and models: how to automate testing
- How to go from notebook to HPC

# Roadmap

2 3

**Hight Dimensionnal Data**
CNN

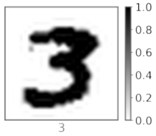
2.1 What is a **Convolutional Neuron Network** (CNN) ?
- Understanding what a CNN is
- Identify use cases

2.2 **Example 1 : MNIST**
- Implementation of a simple case

3.1 **Example 2 : GTSRB**
- The devil is also hiding in the data
- How to work with « large » dataset
- Monitoring the training phase and managing our models
- Improve our results with data augmentation
- Datasets and models: how to automate testing
- How to go from notebook to HPC

# Why convolutional Neural Networks (CNN) ?

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

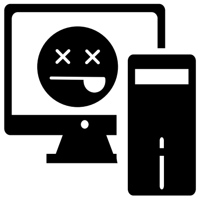



0.0008 M pixels
28x28, 8 bits

785.000 params

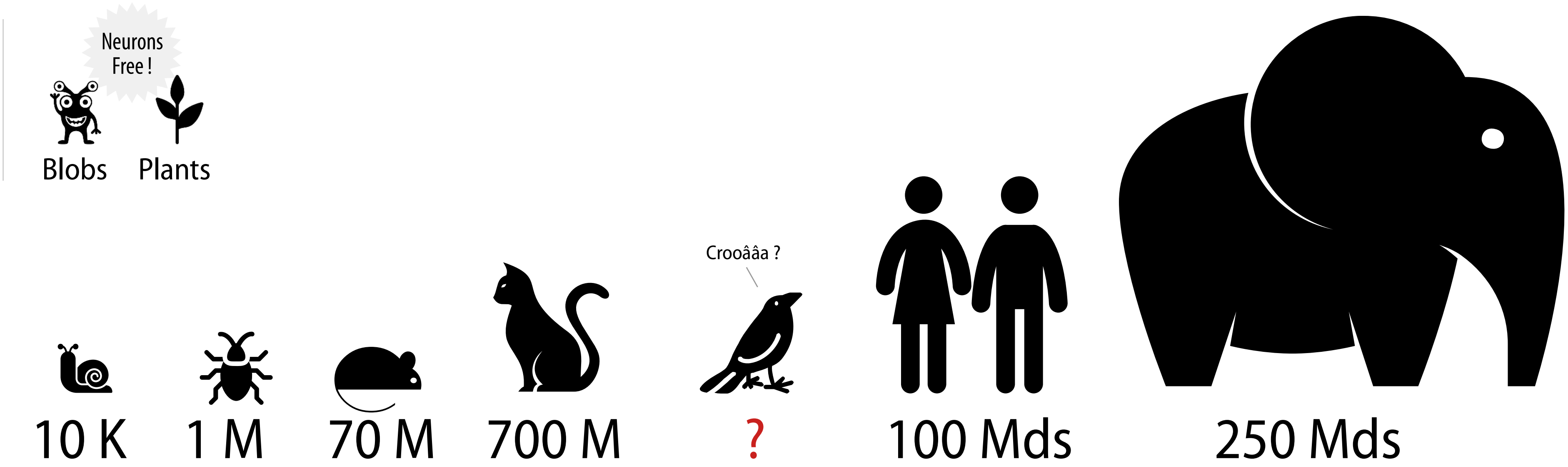
24 M pixels
(r,v,b) 3x8 bits

72. 10E9 params...



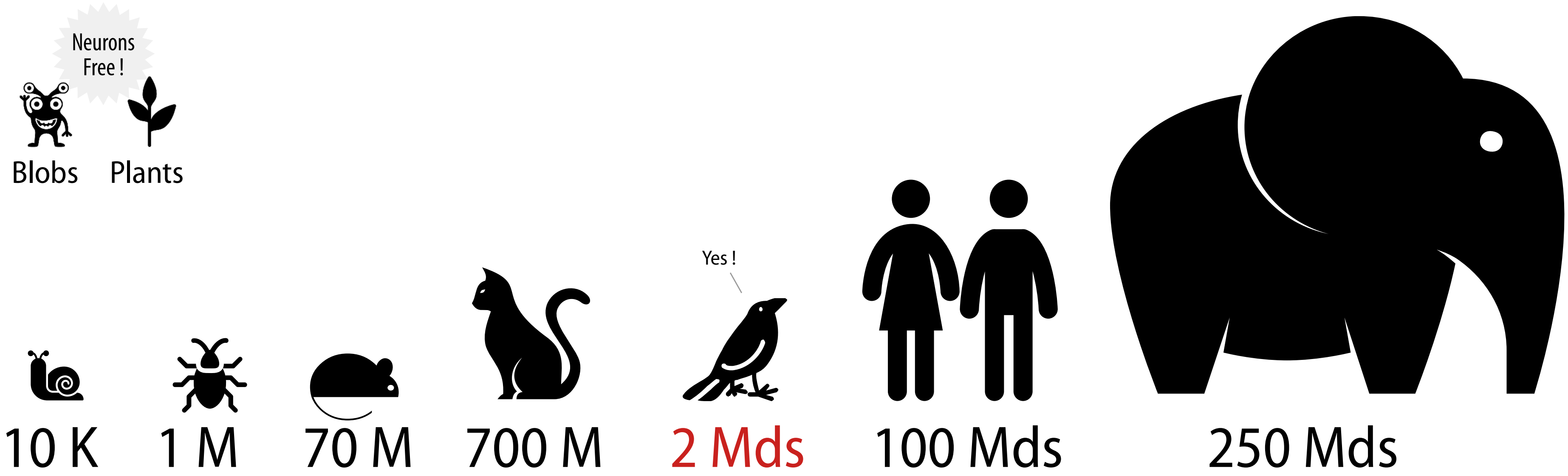
Short Disgression !

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



Short Disgression !

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

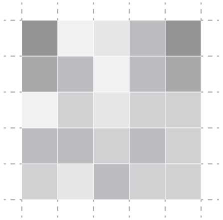


Short Digression !

Yes !

# Principle of convolutions

By Jan Kroon, from Pexels.com



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

Image piece — Kernel 3x3

$$\underset{X}{\begin{bmatrix} 5 & 2 & 1 \\ 4 & 3 & 2 \\ 0 & 2 & 1 \end{bmatrix}} \otimes \underset{\omega}{\begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}} = \underset{y}{\boxed{10}}$$

$$\begin{aligned} y &= 5\text{x}1 + 2\text{x}0 + 1\text{x}1 \\ &+ 4\text{x}0 + 3\text{x}1 + 2\text{x}0 \\ &+ 0\text{x}1 + 2\text{x}0 + 1\text{x}1 = 10 \end{aligned}$$

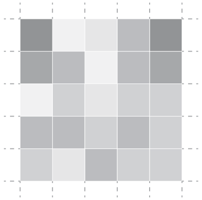
$$y = \sum_{i=1}^{n} \sum_{j=1}^{m} x_{i,j} \cdot \omega_{i,j} \quad \text{with} \begin{cases} n & \text{kernel width} \\ m & \text{kernel height} \end{cases}$$

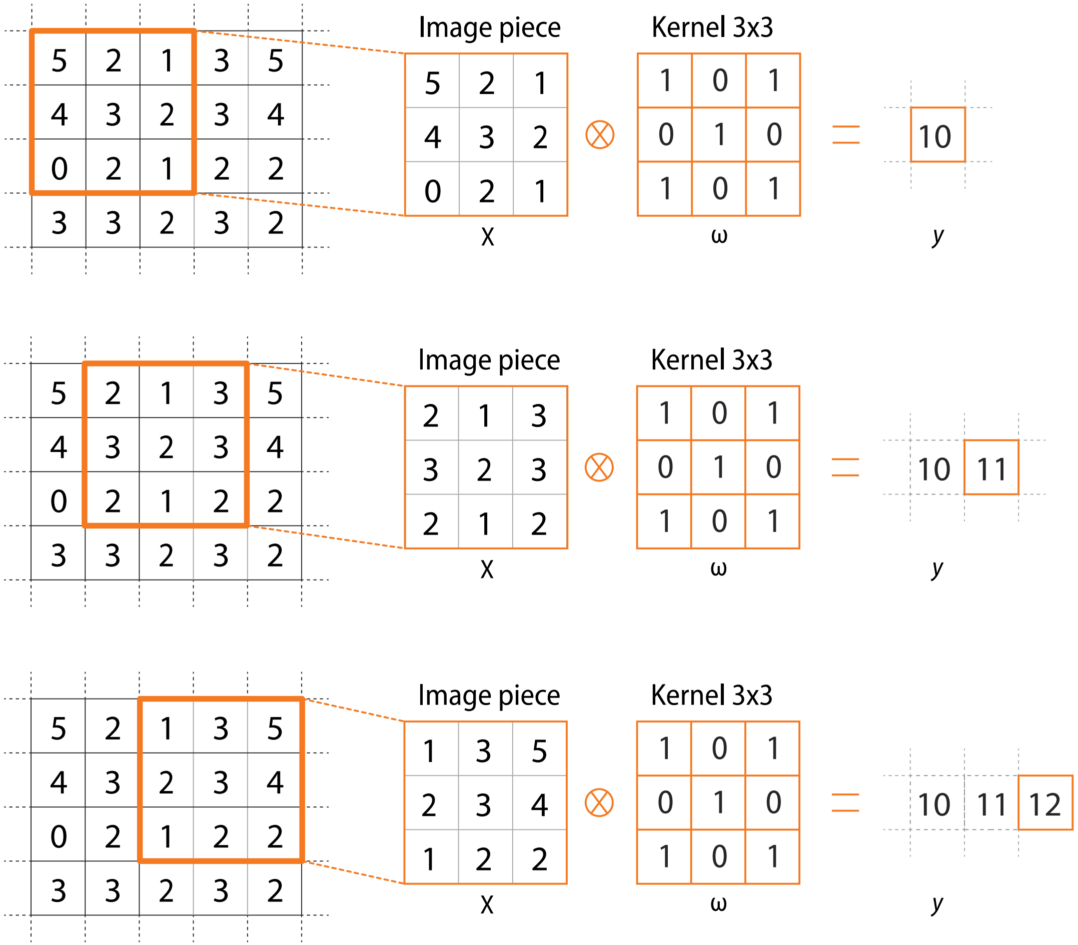
## 2D convolution

$\otimes$ Hadamard product

# Principle of convolutions

By Jan Kroon, from Pexels.com

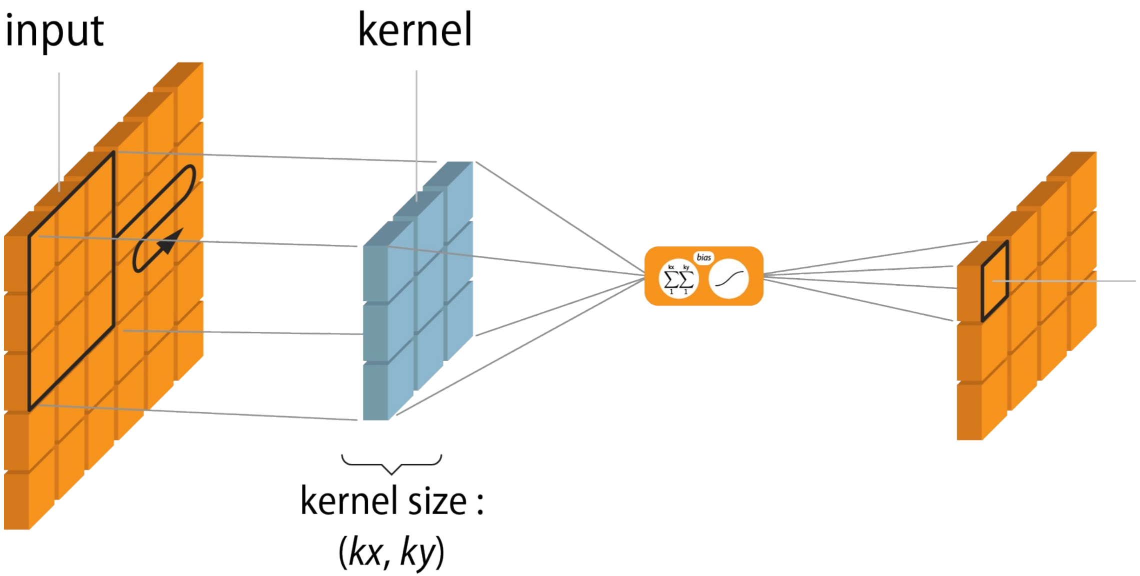





We can perform convolutions in 1, 2, 3 ...or n-dimensional spaces !

$\otimes$ is Hadamard product

# Convolutional layers

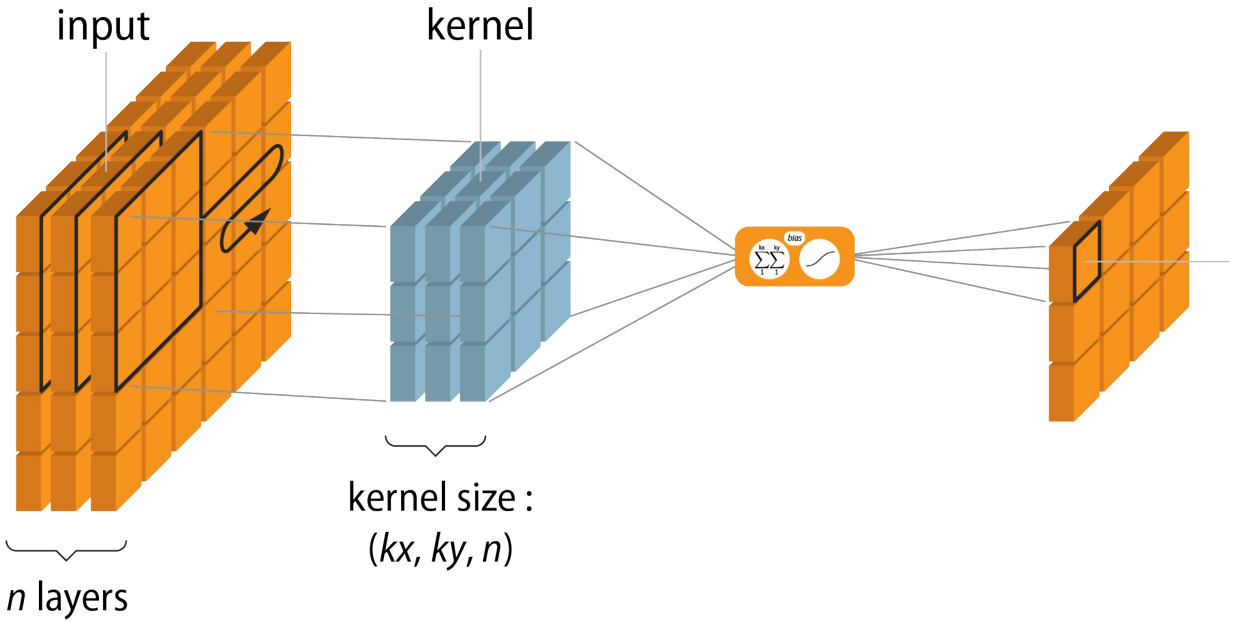



$$y = \sigma\left(\sum \text{input} \otimes \text{kernel} + b\right)$$

Where :
$\sigma$ : is an activation function
$b$ : is a bias

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

# Convolutional layers

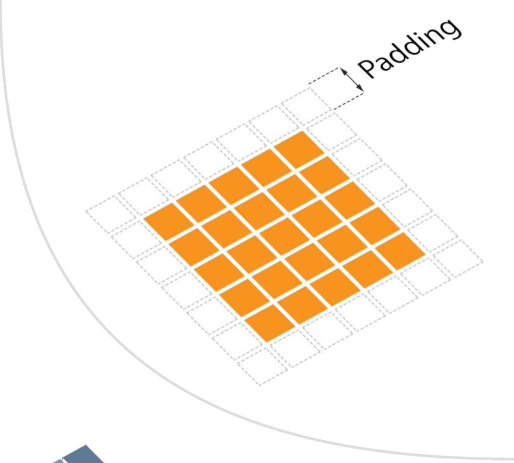



$$y = \sigma(\sum \text{input} \otimes \text{kernel} + b)$$

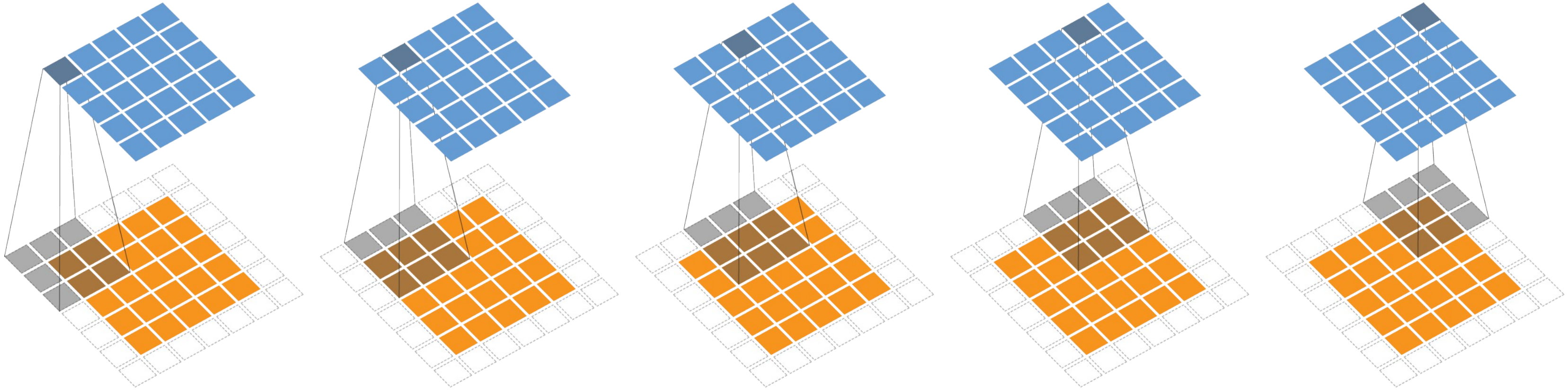
Where :
$\sigma$ : is an activation function
$b$ : is a bias

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

# Convolution parameters

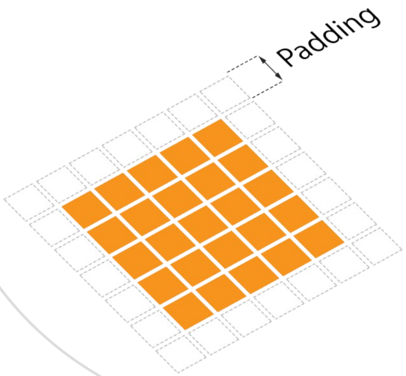



Parameters of a convolutional layer : padding

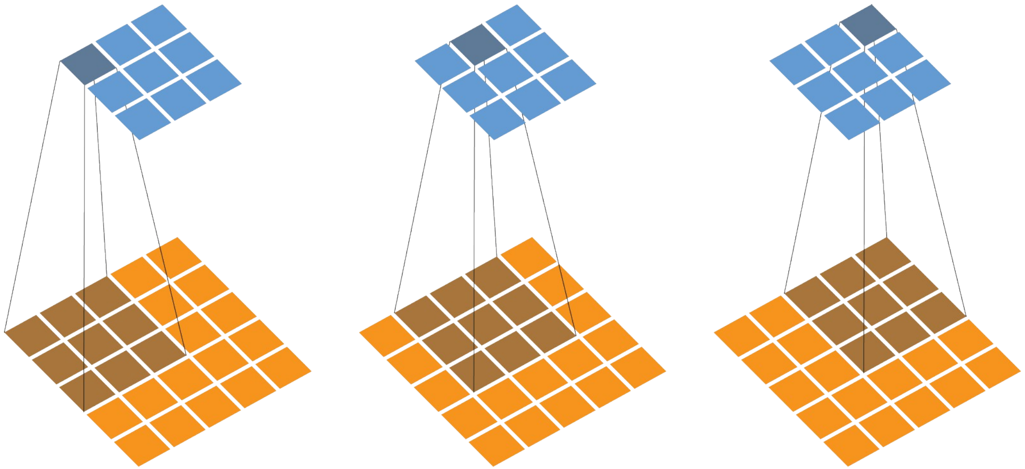


Keras : padding = 'same' $\Rightarrow$ size is preserved

# Convolution parameters

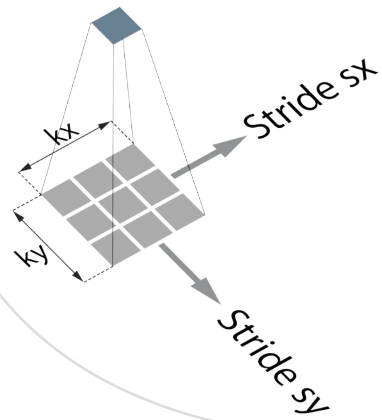



Parameters of a convolutional layer : padding

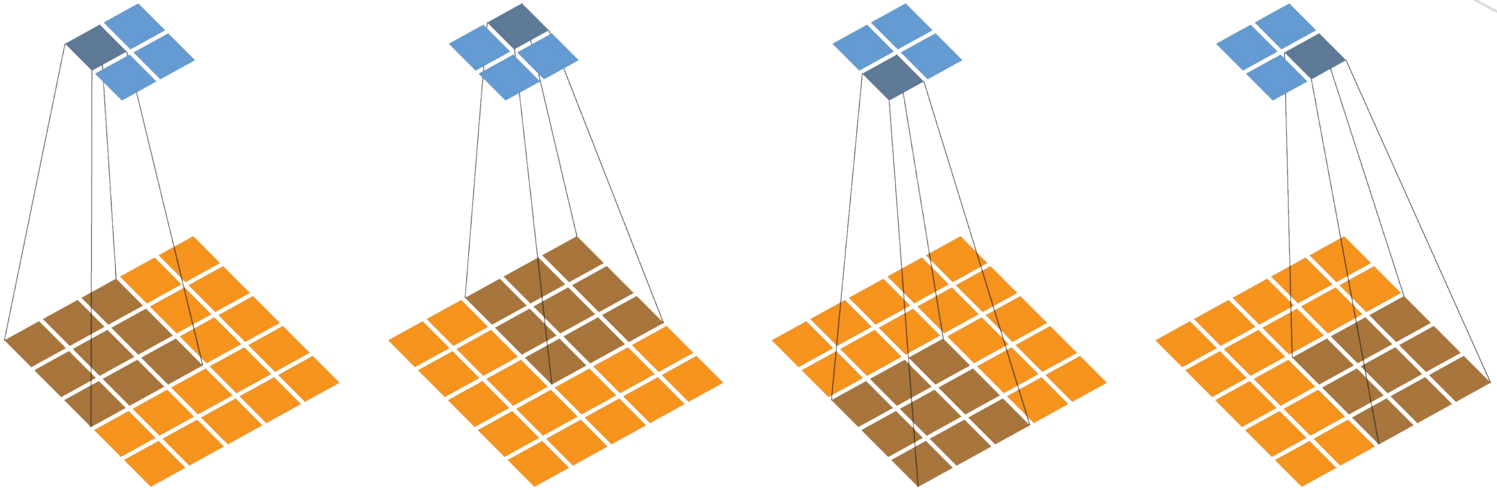


Keras : padding = 'valid' $\Rightarrow$ Size is not preserved (no padding)

# Convolution parameters

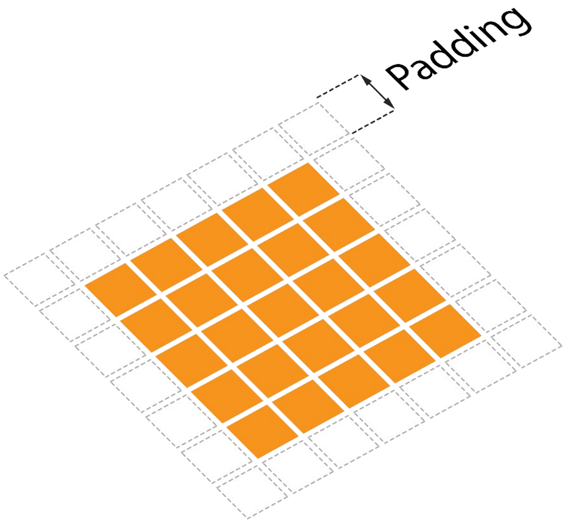


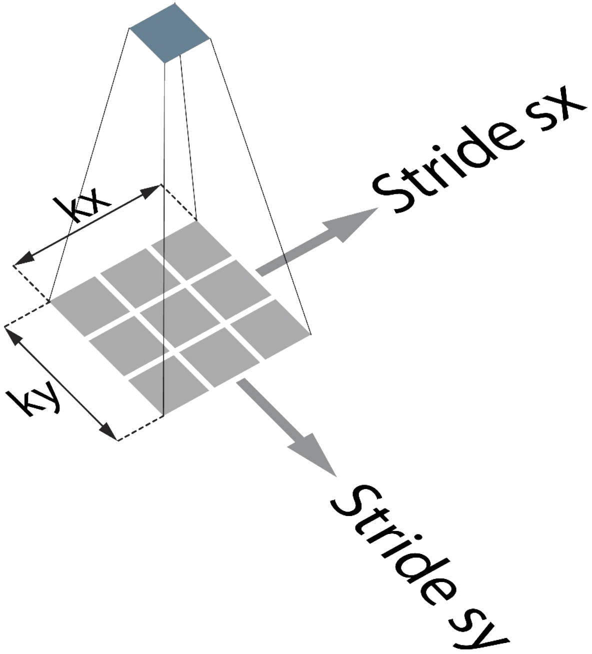

Parameters of a convolutional layer : Strides



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

# Convolution parameters




padding

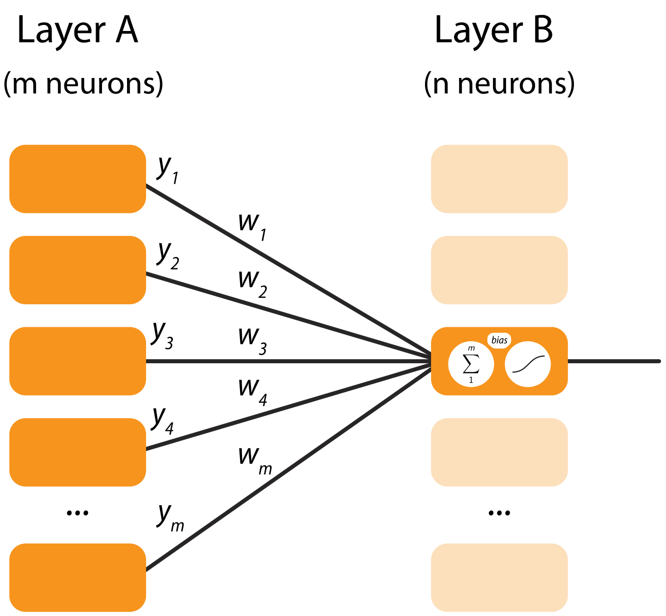
'same' 'valid'

kernel size

strides

# Number of parameters

For a fully connected layer :



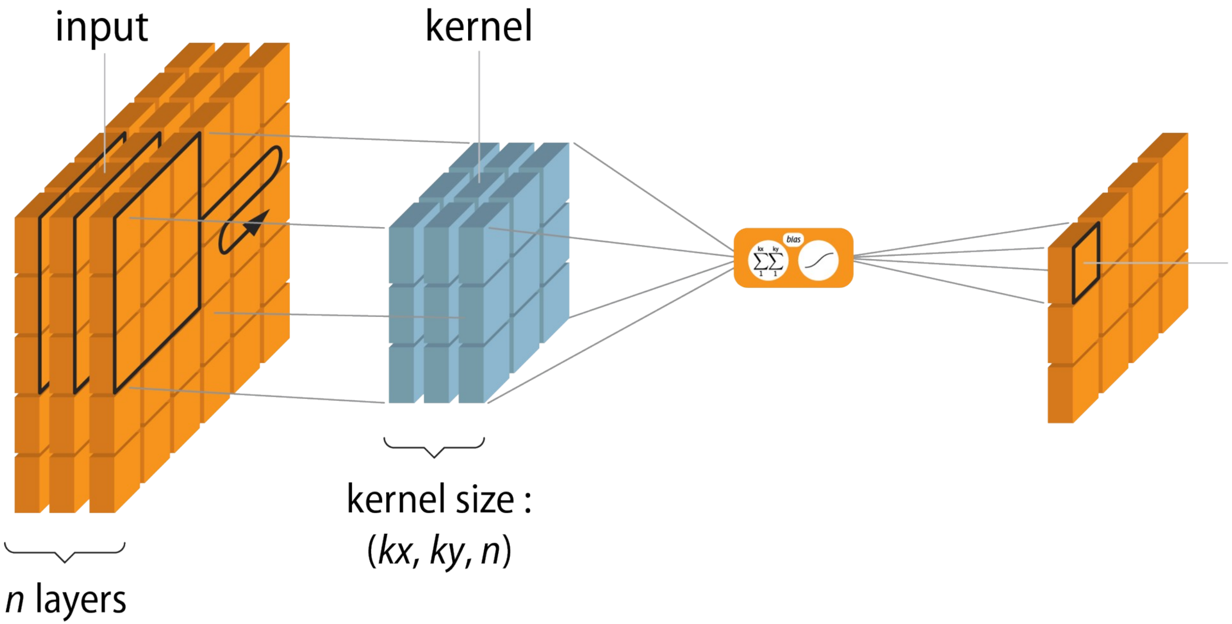

$$y = \sigma\left(\sum_{1}^{m} y_i \cdot w_i + b\right)$$

Where :
$\sigma$ : 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 :



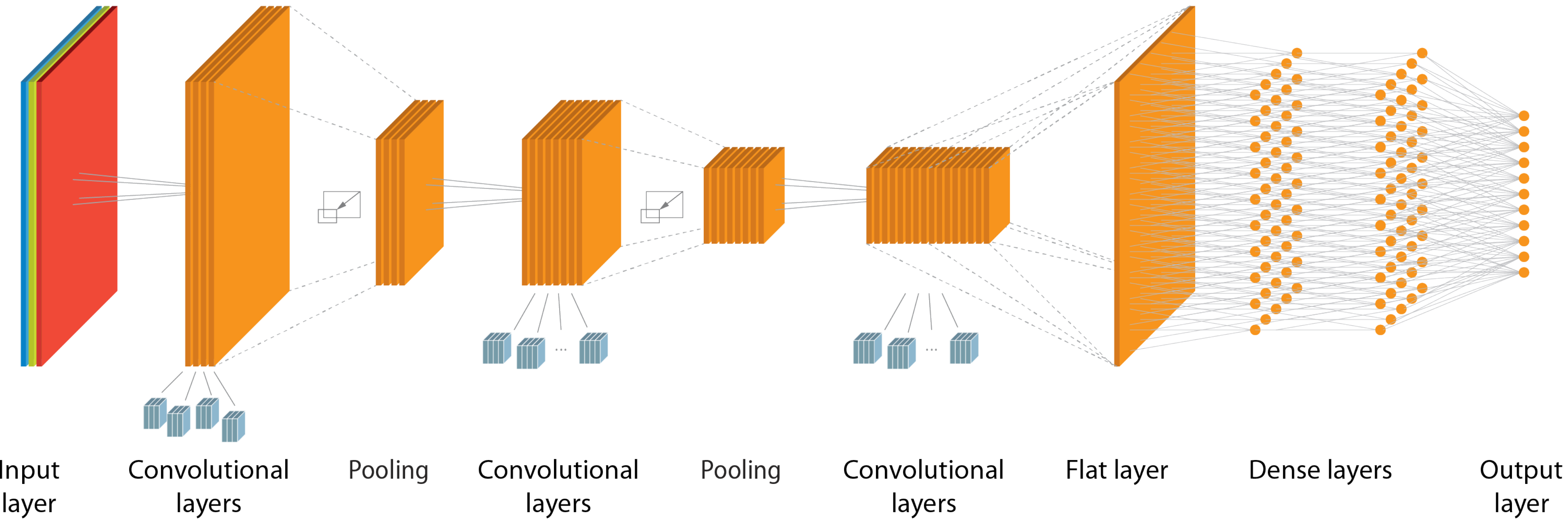

$$y = \sigma\left(\sum \text{input} \otimes \text{kernel} + b\right)$$

Where :
- $\sigma$ : is an activation function
- $b$ : is a bias

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

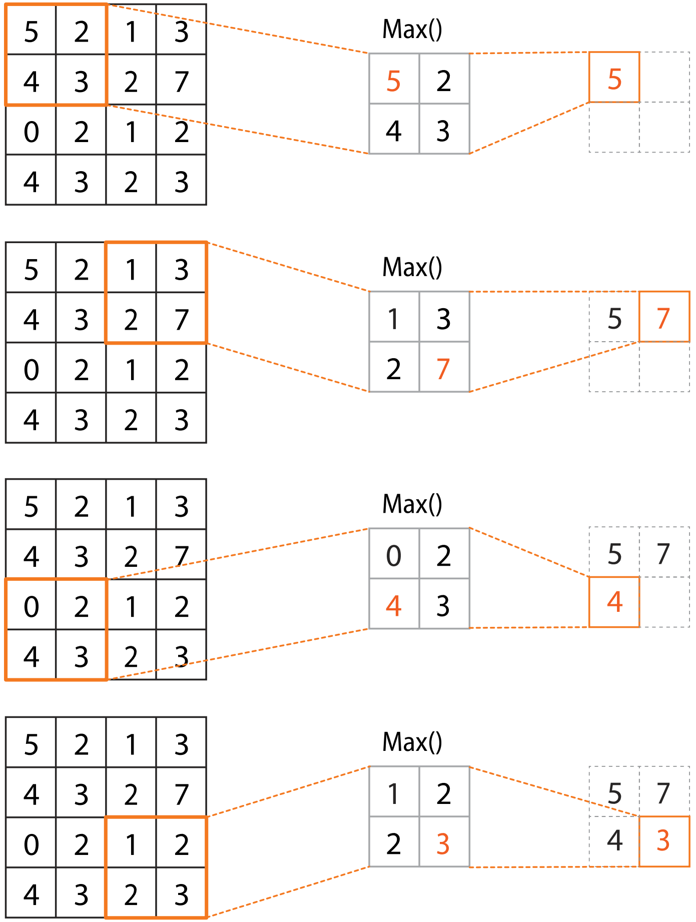
If we want to generate *m* convolutional layers, we will need *m* convolutional neurons, so, number of parameters is : $m.(n.kx.ky + 1)$

# Convolutional Neural Networks (CNN)



# Convolutional Neural Networks (CNN)

Principle of Max Pooling :

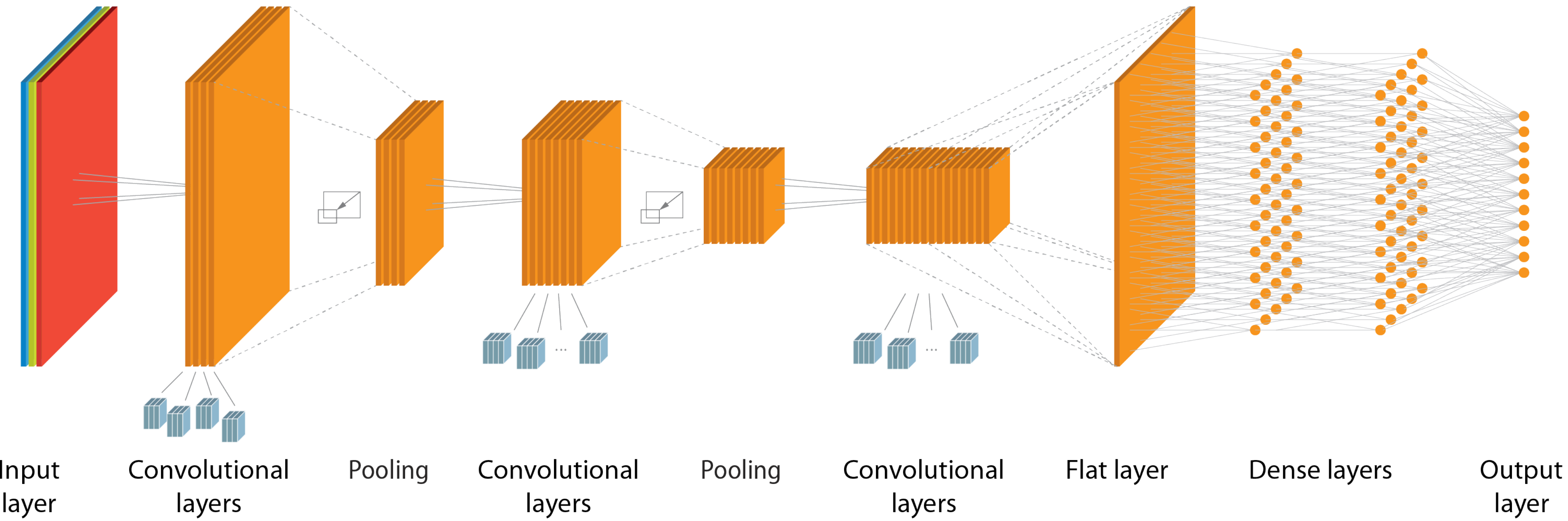



It is possible to set the **window size, padding** mode and **strides**.

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)



# Roadmap

2 3

**Hight Dimensionnal Data**
CNN

2.1

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

- ➔ Understanding what a CNN is
- ➔ Identify use cases

2.2

## Example 1 : MNIST

- ➔ Implementation of a simple case

3.1

## Example 2 : GTSRB

- ➔ The devil is also hiding in the data
- ➔ How to work with « large » dataset
- ➔ Monitoring the training phase and managing our models
- ➔ Improve our results with data augmentation
- ➔ Datasets and models: how to automate testing
- ➔ How to go from notebook to HPC

# Roadmap

2 3

**Hight Dimensionnal Data**
CNN

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

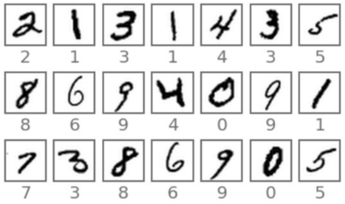
- → Understanding what a CNN is
- → Identify use cases

2.2 **Example 1 : MNIST**

- Implementation of a simple case

3.1 **Example 2 : GTSRB**

- → The devil is also hiding in the data
- → How to work with « large » dataset
- → Monitoring the training phase and managing our models
- → Improve our results with data augmentation
- → Datasets and models: how to automate testing
- → How to go from notebook to HPC




# Image classification with CNN

## Notebook : [MNIST2]

**Objective :**
Recognizing handwritten numbers

**Dataset :**
Modified National Institute of Standards and Technology (MNIST)

# MNIST2

**97.7 %**

Previously with a DNN

START

Step 1 – Import and init

Step 2 – Retrieve data

keras

Step 3 – Preparing the data

Train set

Test set

Step 4 – Build a model

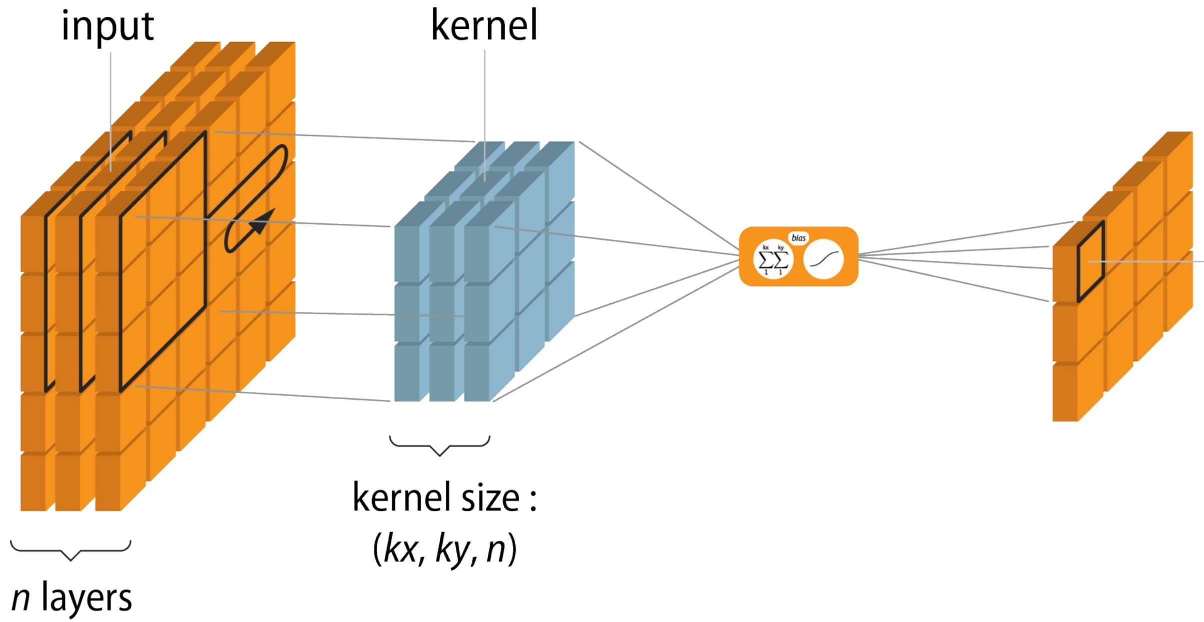
Step 5 – Train the model

Step 6 – Evaluate

END

## Objectives :

Classify handwritten numbers (MNIST dataset) via a CNN




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

Where :
$\sigma$ : is an activation function
$b$ : is a bias

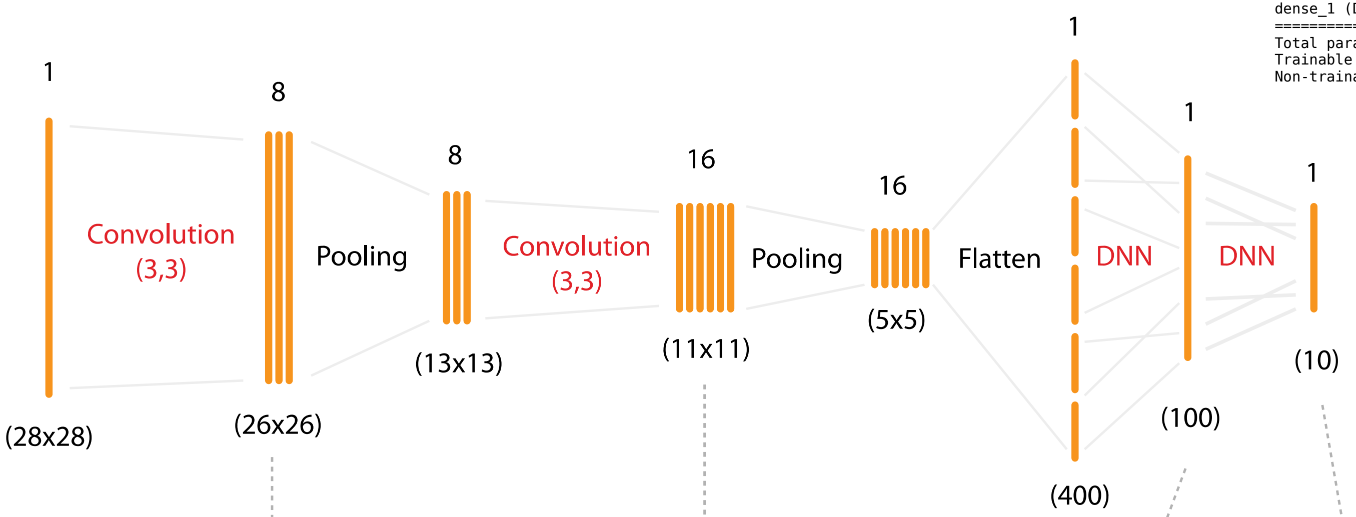
Number of parameters for a convolutional layer : $n \,.\, kx \,.\, Ky + 1$

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

# MNIST2

```
Model: "sequential"
_________________________________________________________________
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)        1168
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 5, 5, 16)          0
_________________________________________________________________
dropout_1 (Dropout)          (None, 5, 5, 16)          0
_________________________________________________________________
flatten (Flatten)            (None, 400)               0
_________________________________________________________________
dense (Dense)                (None, 100)               40100
_________________________________________________________________
dropout_2 (Dropout)          (None, 100)               0
_________________________________________________________________
dense_1 (Dense)              (None, 10)                1010
=================================================================
Total params: 42,358
Trainable params: 42,358
Non-trainable params: 0
```

## Understand how it works by understanding where the parameters are...




(3x3 + 1)*8

80 params

(8x3x3 + 1)*16

1168 params

100*(400 + 1)

40100 params

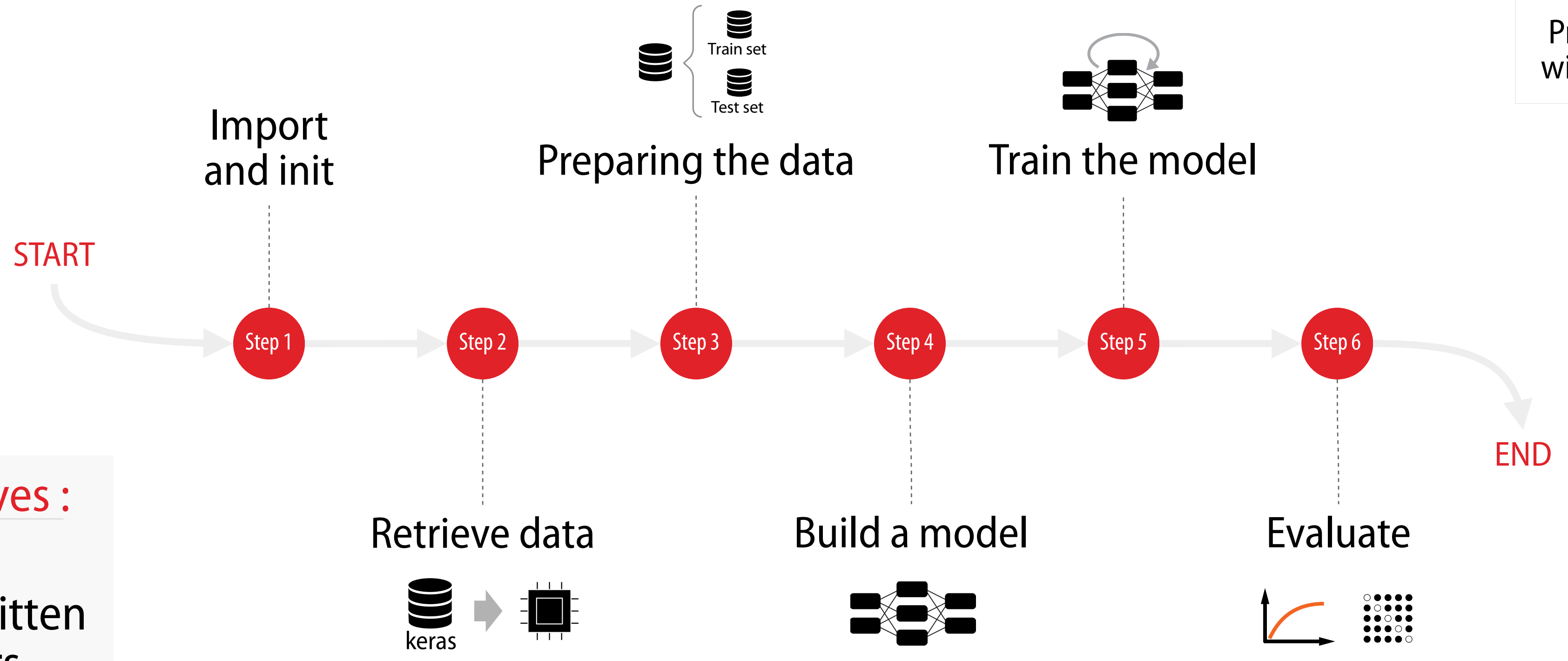
10*(100 + 1)

1010 params

# MNIST2

**97.7 %**

Previously with a DNN




## Objectives :

Classify handwritten numbers (MNIST dataset) via a CNN

# Next, on Fidle :

2 3

**Hight Dimensionnal Data**

CNN

## Jeudi 1 décembre, 14h00

Séquence 3 :

## Réseaux convolutifs, partie 2

Quand les datasets et les calculs grossissent, problématiques liées à la gestion des données

- Rappel sur les convolutions
- Monitoring et Tensorboard
- Augmentation de données
- Passage à l'échelle (du notebook au batch)
- Points de reprise (checkpoint)

Exemple proposé :

Classification de panneaux routiers

Durée : 2h00

# Next on Fidle :

2 3

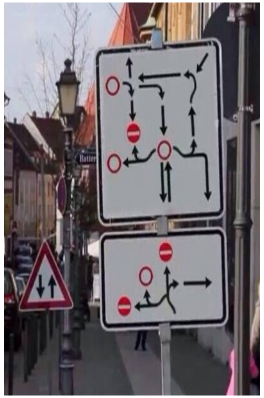
**Hight Dimensionnal Data**

CNN

**Jeudi 1er décembre, 14h00**

Séquence 3 :

**Réseaux convolutifs, partie 2**
**Quand les datasets et les calculs grossissent, problématiques liées à la gestion des données**

To be continued...


Contact@fidle.cnrs.fr
FIDLE https://fidle.cnrs.fr
YouTube https://fidle.cnrs.fr/youtube


FIDLE