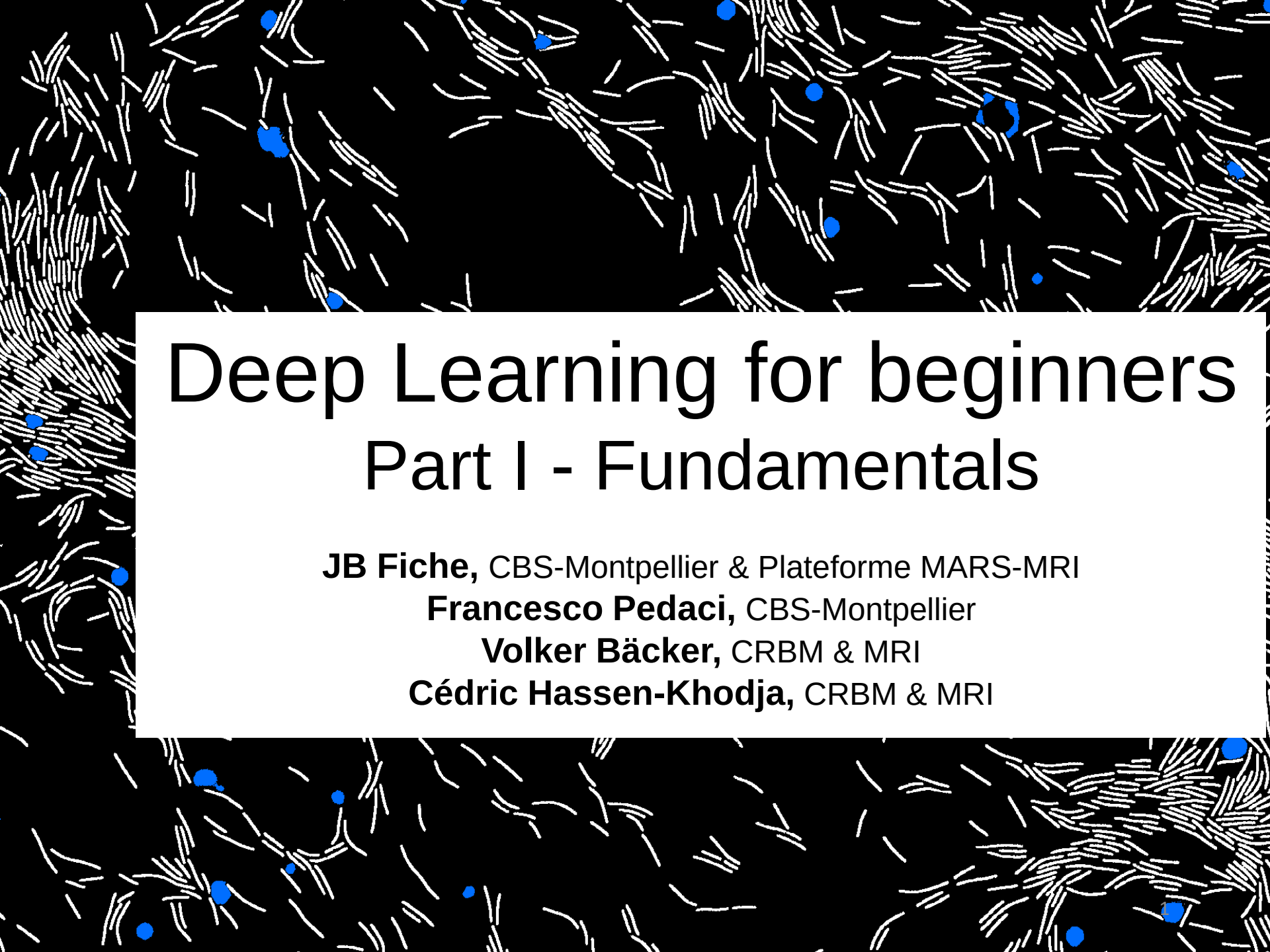


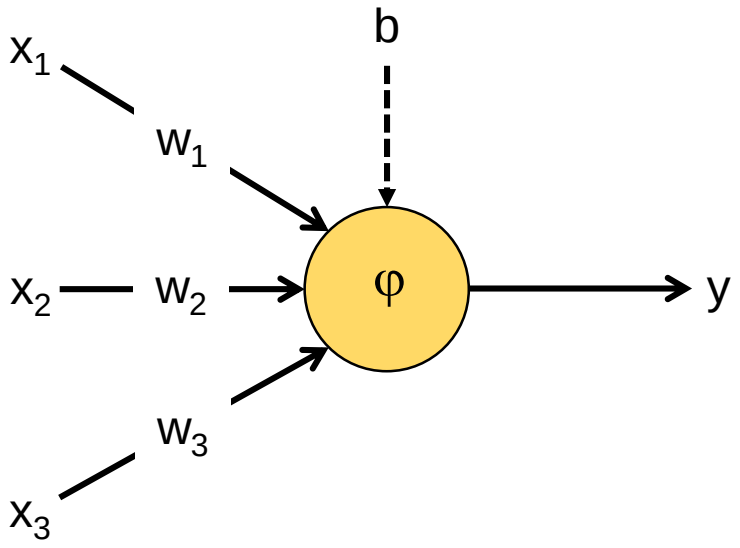
Deep Learning for beginners

Part I - Fundamentals

JB Fiche, CBS-Montpellier & Plateforme MARS-MRI
Francesco Pedaci, CBS-Montpellier
Volker Bäcker, CRBM & MRI
Cédric Hassen-Khodja, CRBM & MRI

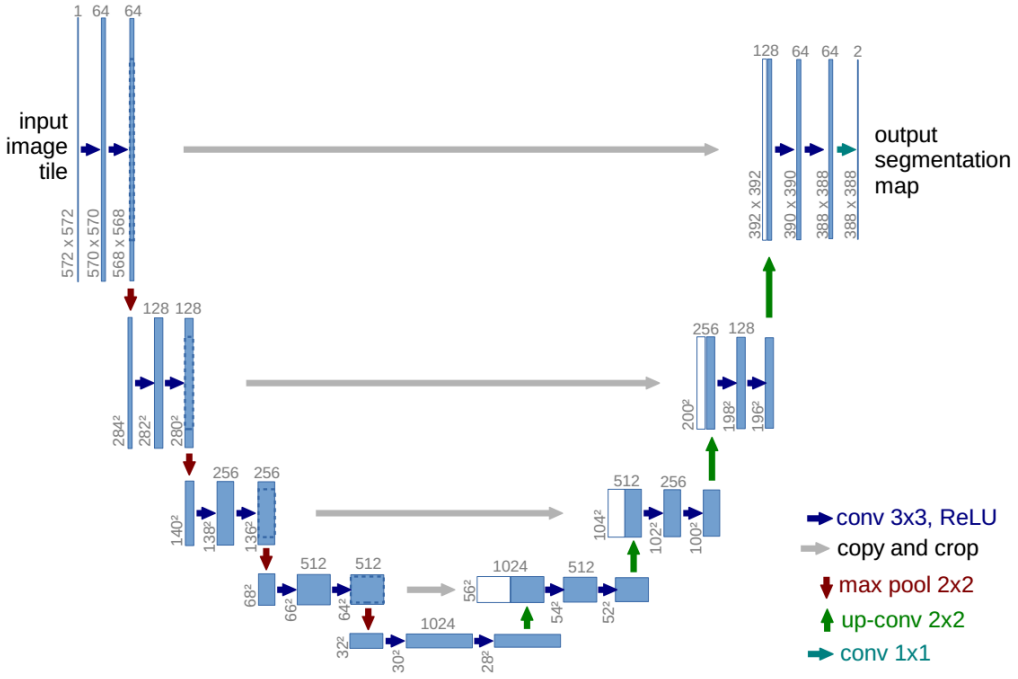
End-of-the-day goal

- Understand the differences between **machine-learning** & **deep learning**
- Understand what an **Artificial Neuron Network (ANN)** is and what are the main parameters to characterize them
- What are the **fundamentals for building and training an ANN**



End-of-the-day goal

- What is a **Convolutional Neural Network (CNN)** and why they are useful for image analysis
- Have seen a **few applications of Deep Learning** for image analysis and several of the most common architectures.



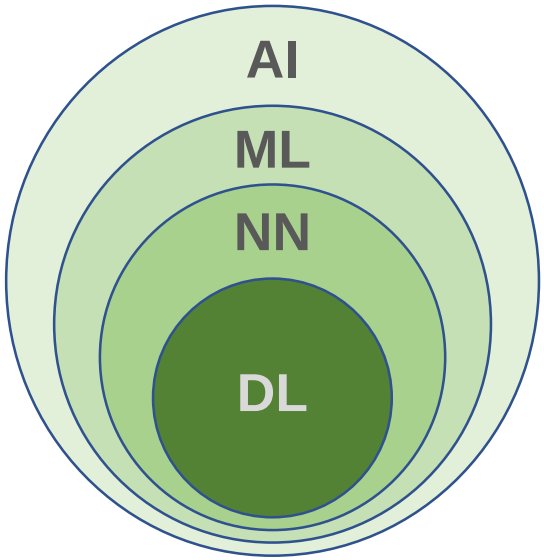
Pic Credit: Ronneberger et al. 2015. International Conference on Medical image computing.

Outline

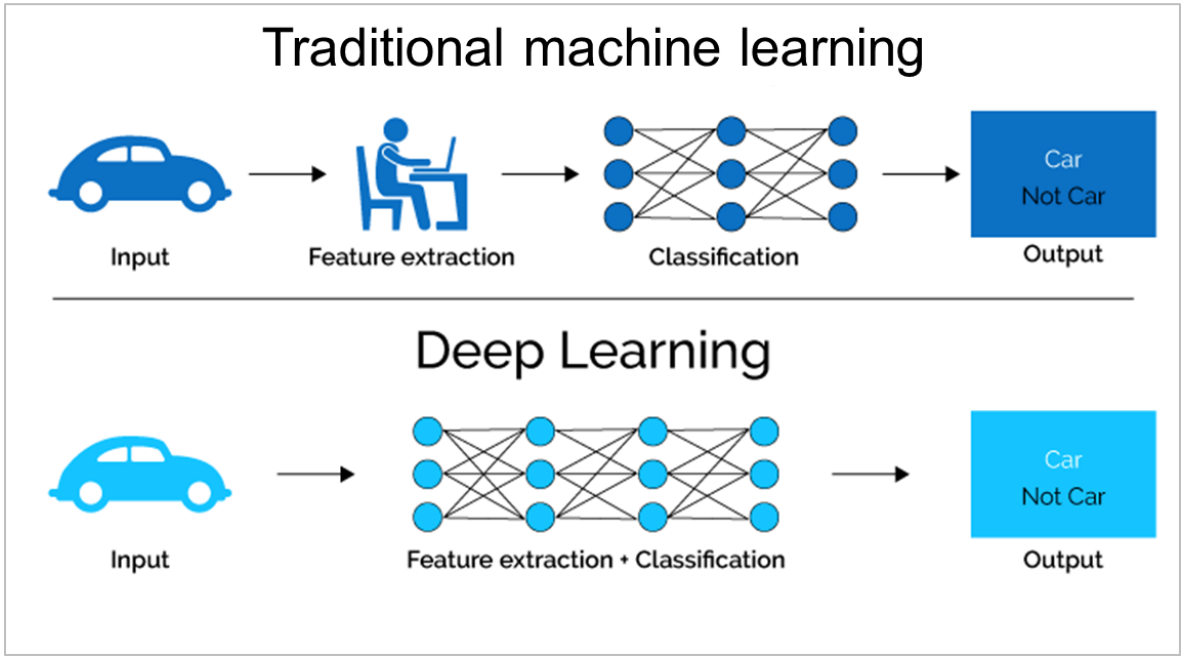
I. Introduction

i. **Difference between Machine Learning (ML) & Deep Learning (DL)**
ii. Brief historical overview
iii. Applications for image analysis
iv. When using DL and which tools are available?

Machine Learning vs. Deep Learning



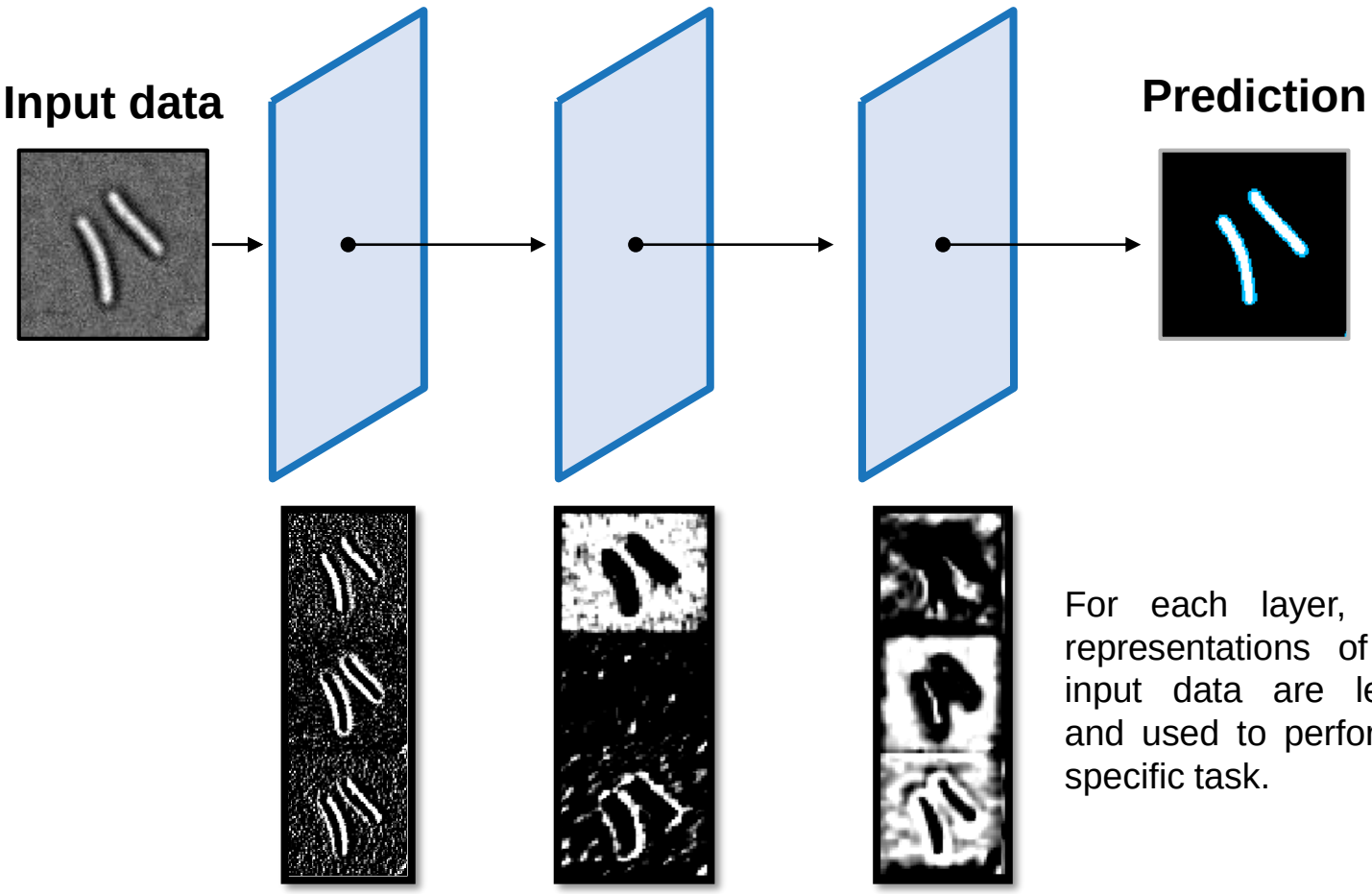
AI = artificial intelligence
ML = machine learning
NN = neural network
DL = deep learning



Pic Credit: Xenonstack | Machine Learning vs Deep Learning

Machine Learning algorithms are searching for a useful representations of the input data.

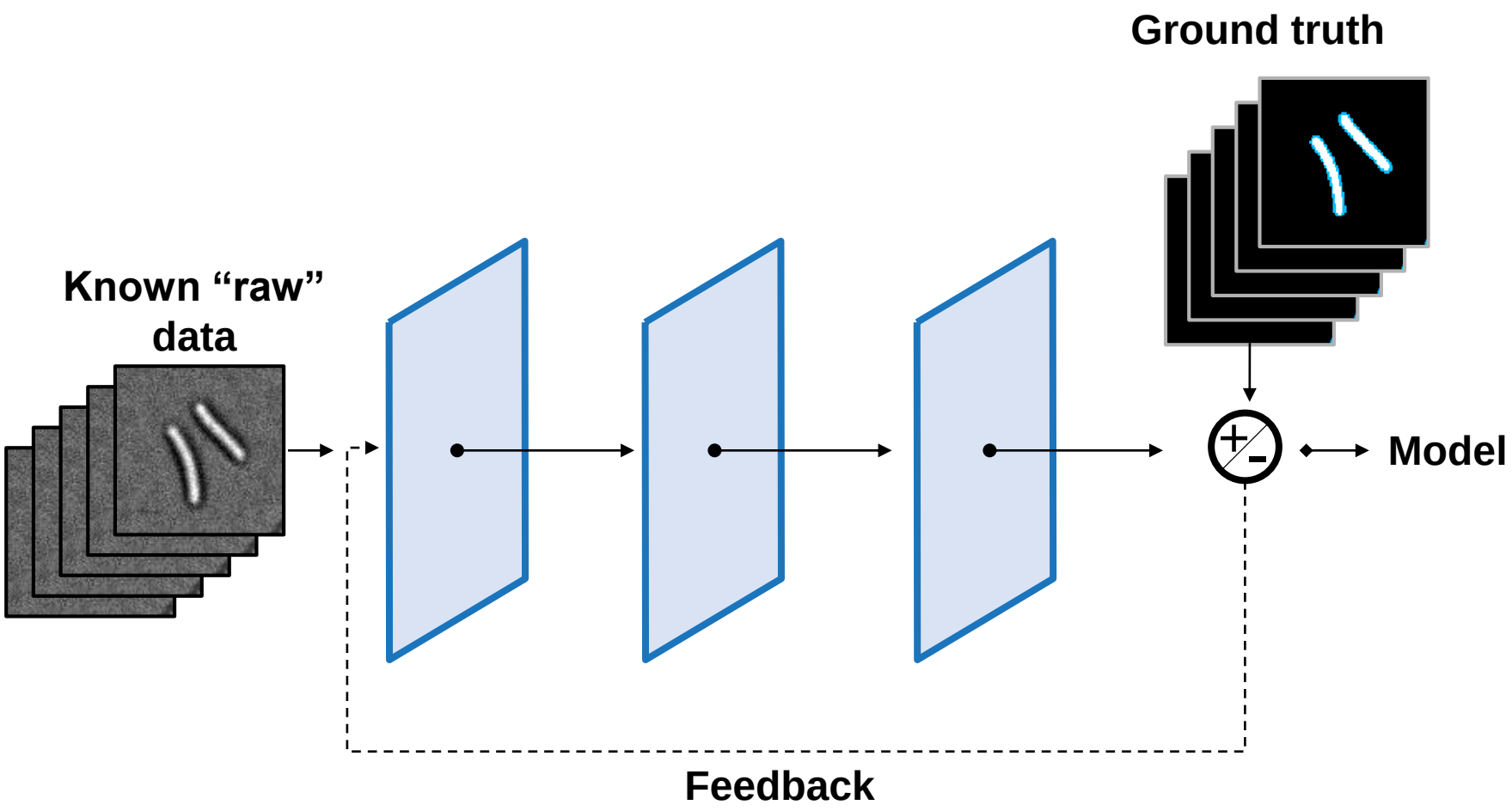
Deep Learning : why “Deep” ?



For each layer, new representations of the input data are learnt and used to perform a specific task.

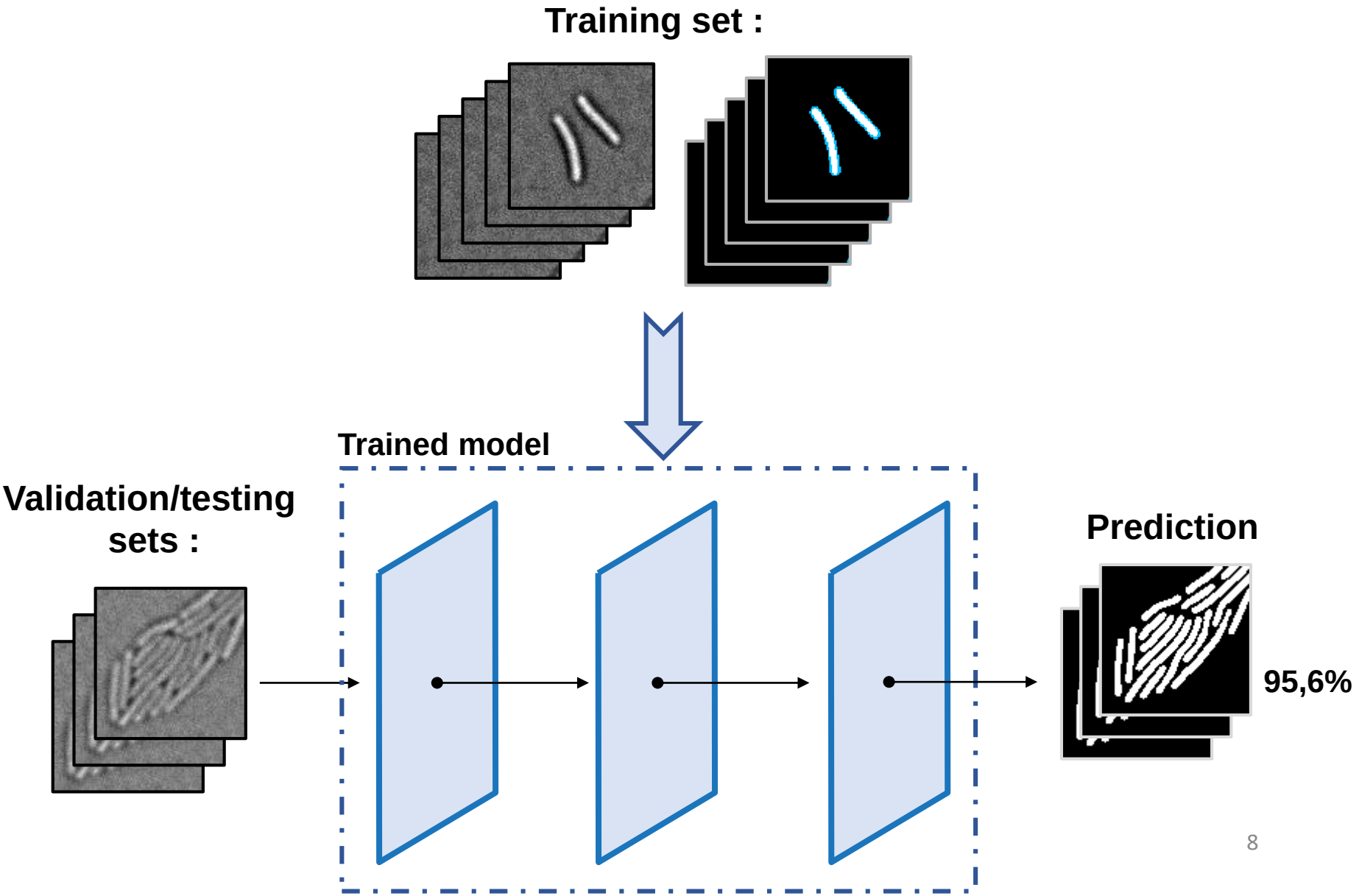
The « Deep » in Deep Learning is a reference to the **successive layers of representations** the network is using to perform its task.

Supervised Deep Learning : what “training” means?



During the training, the network will learn how to represent the data in a useful way in order to perform a specific task (segmentation, denoising, etc.)

Supervised Deep Learning : what “training” means?

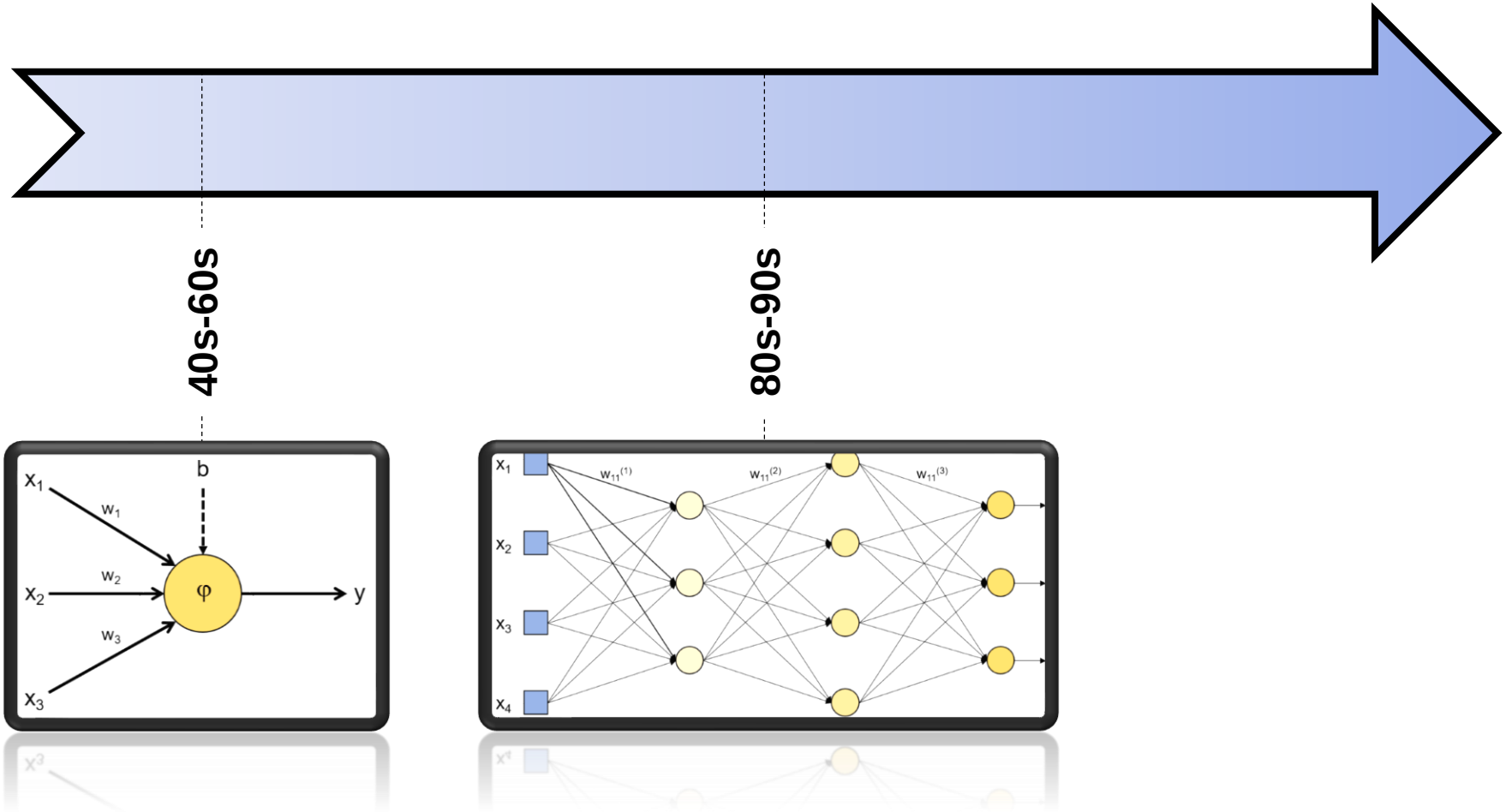


Outline

I. Introduction

i. Difference between Machine Learning (ML) & Deep Learning (DL)
ii. Brief historical overview
iii. Applications for image analysis
iv. When using DL and which tools are available?

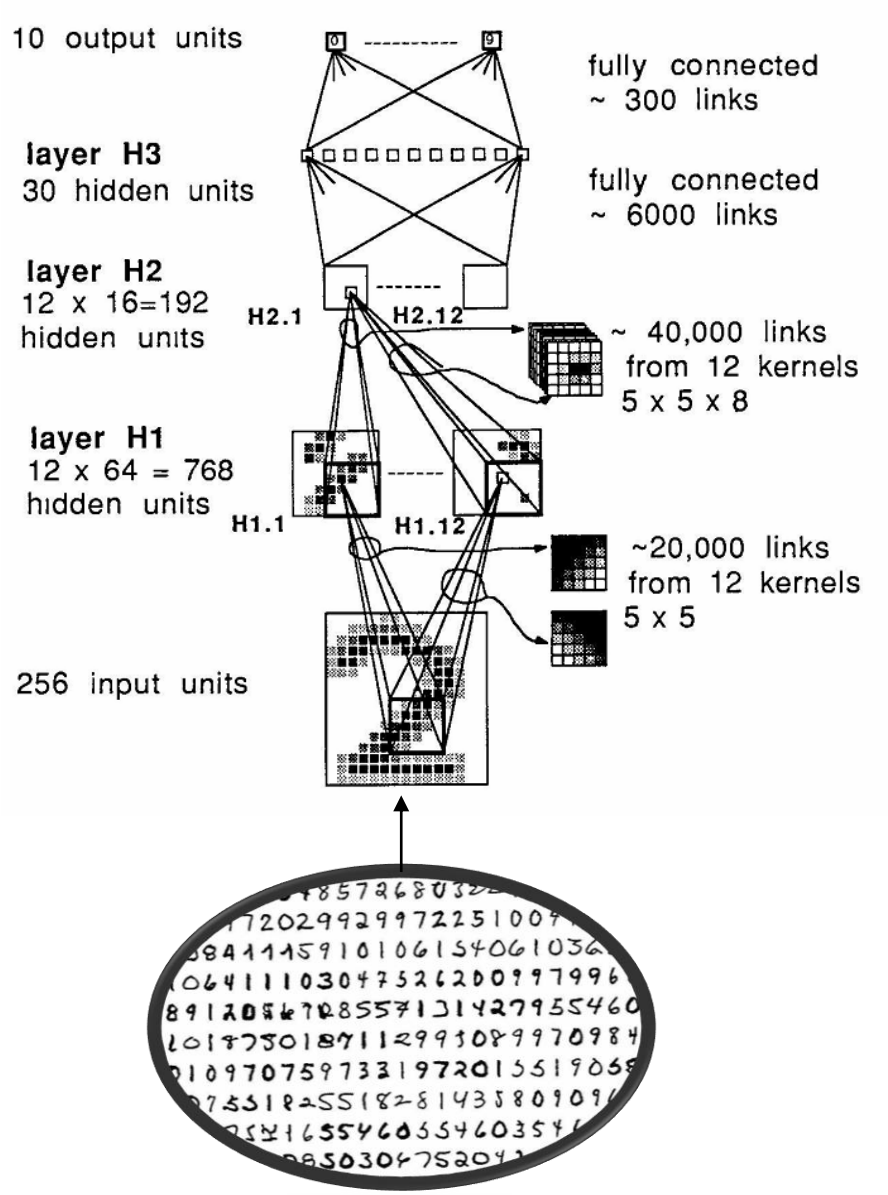
Brief historical overview



Cybernetics : linear **neural network** models able to handle simple classification tasks

Connectivism : Development of the **back-propagation** algorithm for the training of deep neural network. First **convolutional network**.

Hand-written zipcode classification - 1989



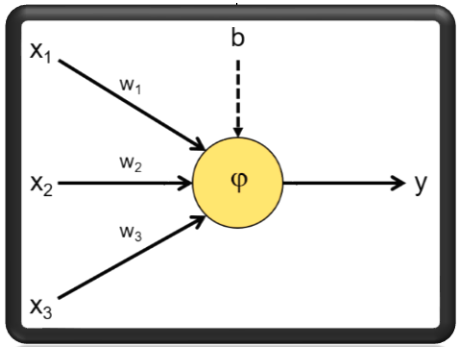
Deep learning models were already successfully used in the 90's for complicated tasks such as hand-written digit classification.

The same algorithms are still used nowadays for image analysis.

Pic Credit: LeCun et al. 1989. Neural Computation

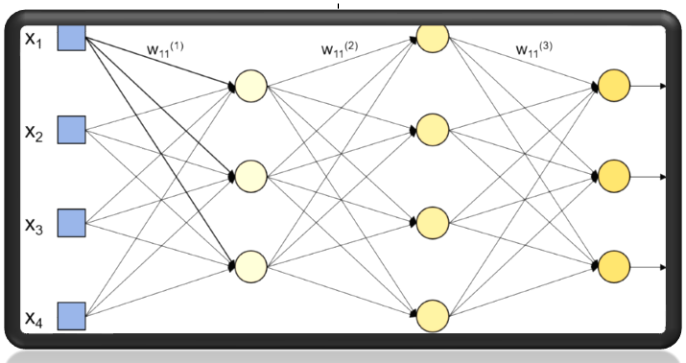
Brief historical overview

40s-60s



<u>Cybernetics</u> : linear **neural network** models able to handle simple classification tasks

80s-90s

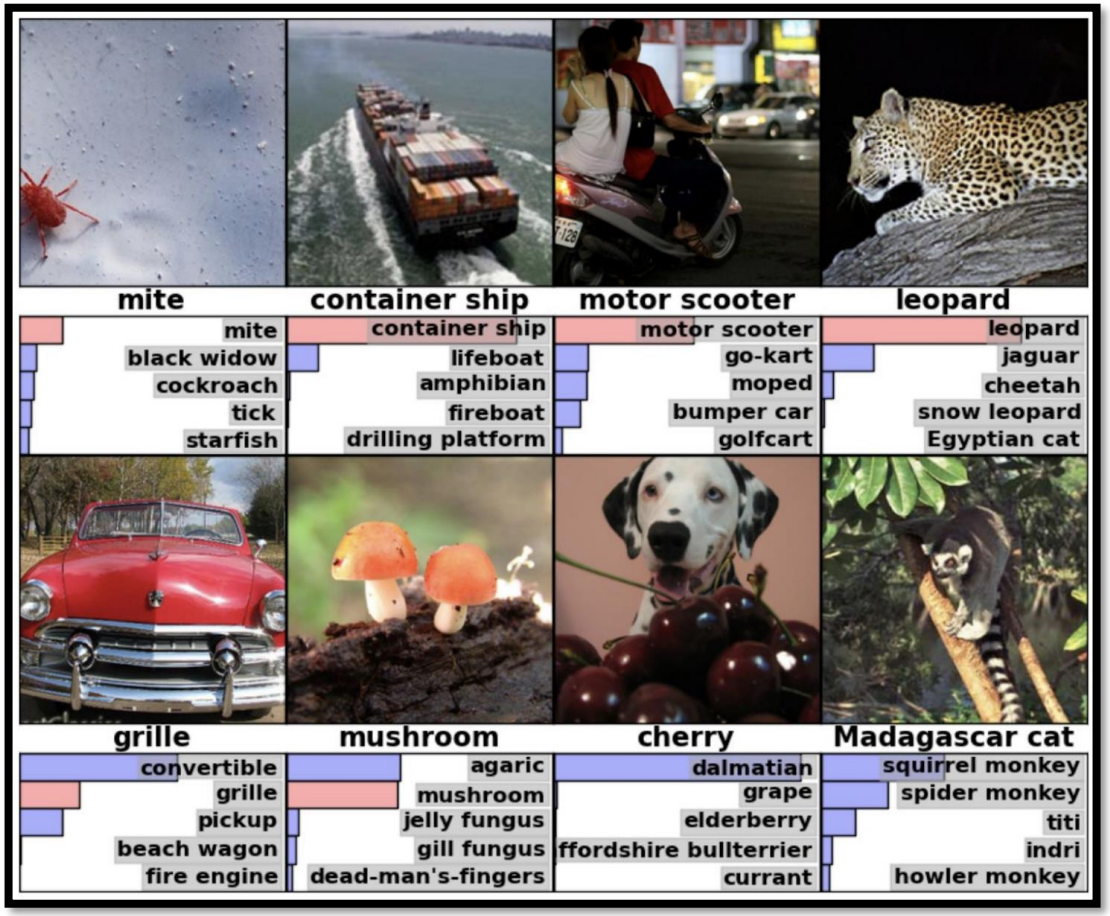


<u>Connectivism</u> : Development of the **back-propagation** algorithm for the training of deep neural network. First **convolutional network**.

2012

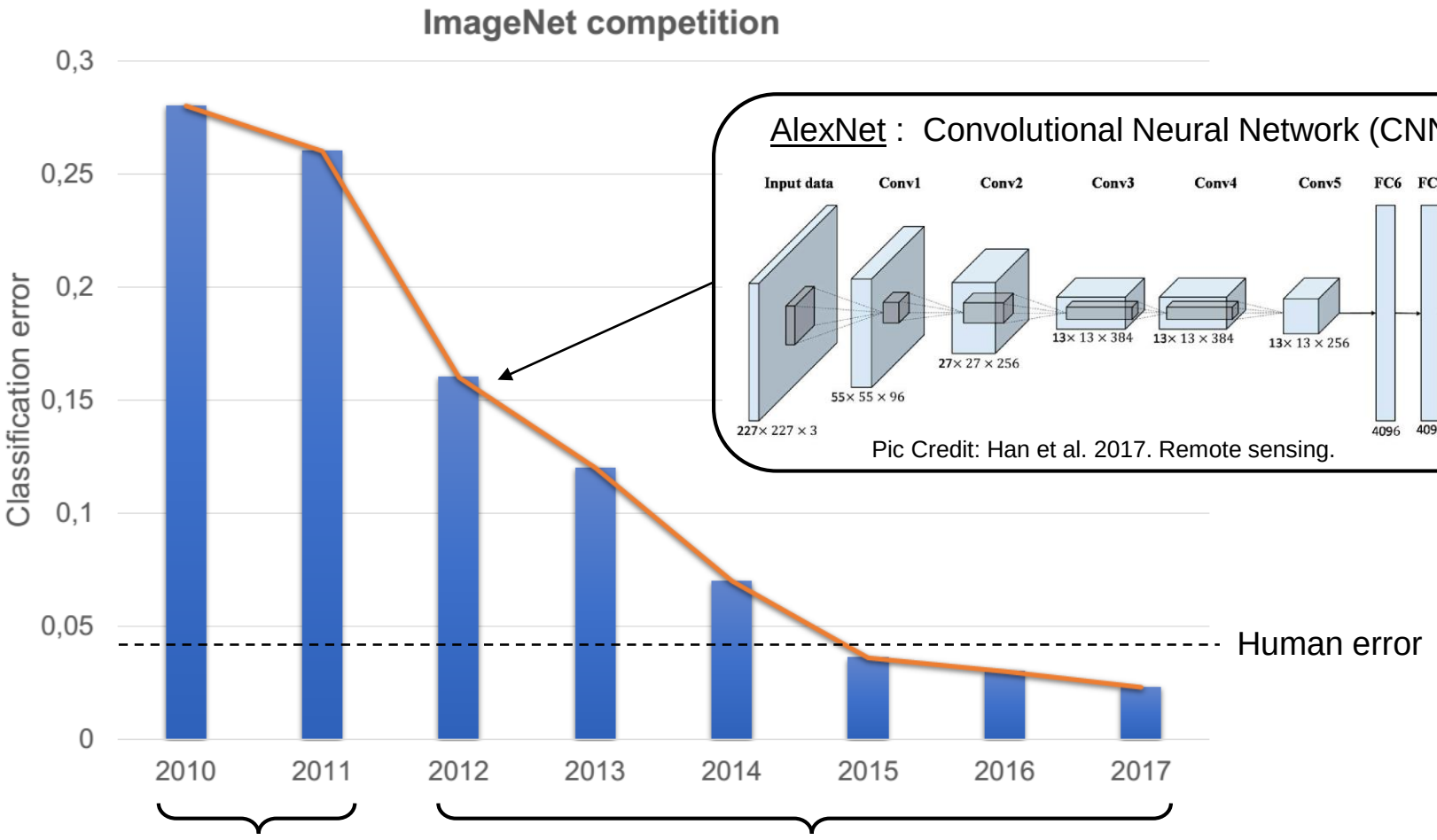
First time a **Deep Learning convolutional algorithm** wins the ImageNet competition

2012 : Breakthrough for image classification



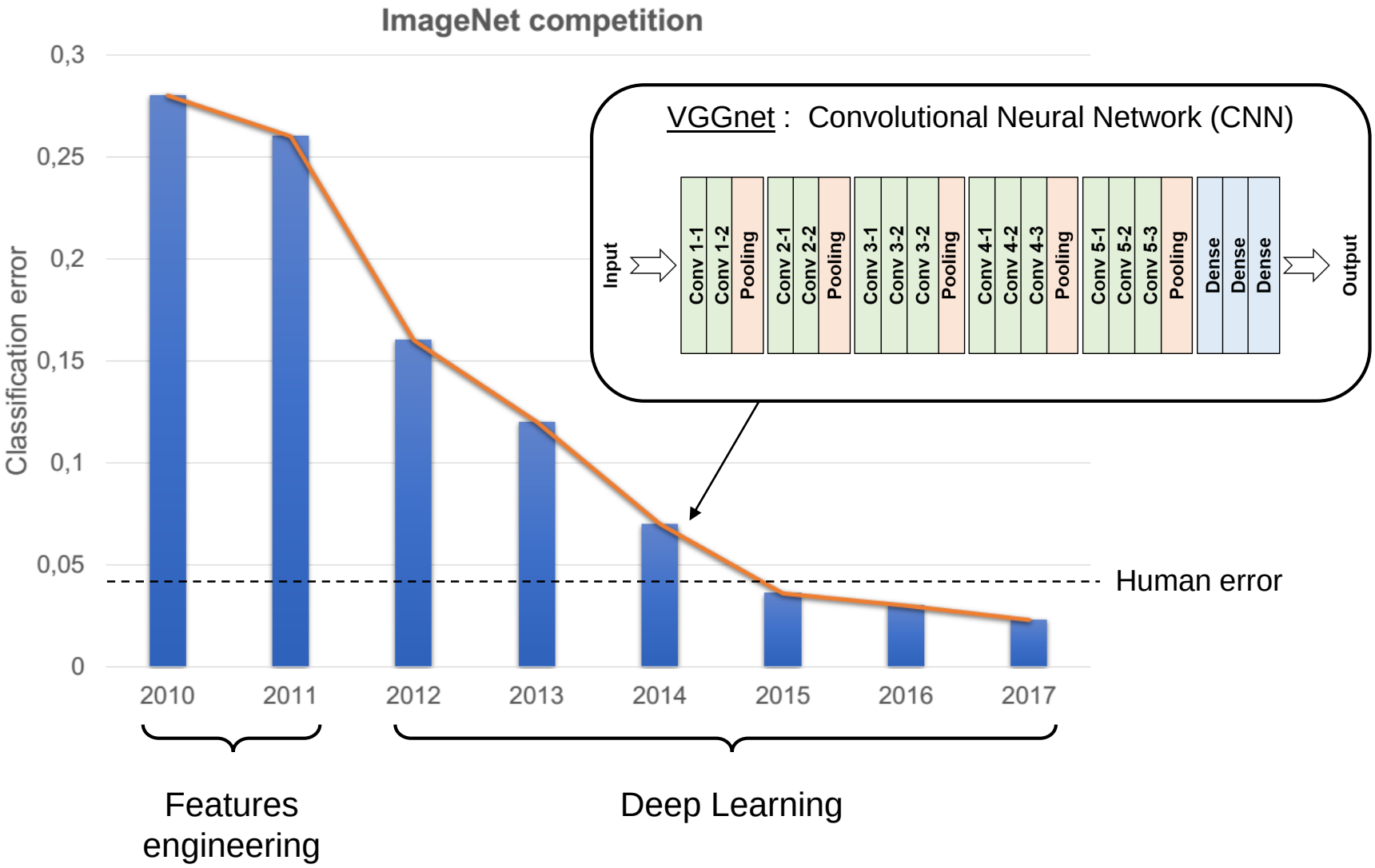
The **Image Large Scale Visual Recognition Challenge** started in 2010 using the ImageNet database (15M labelled images and 22,000 different object categories).

2012 : Breakthrough for image classification



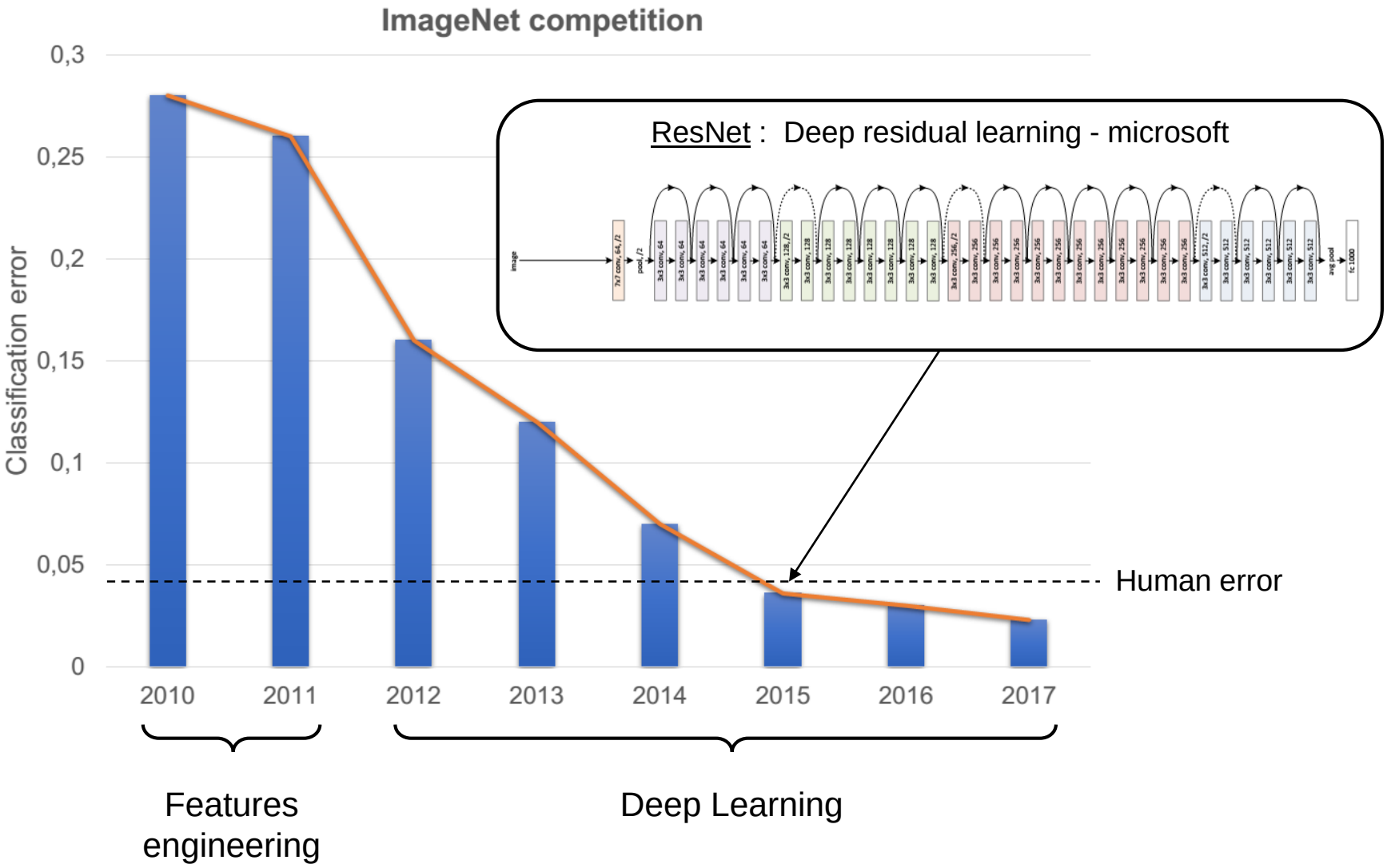
Krizhevsky et al. 2012. Advances in neural information processing systems

2012 : Breakthrough for image classification



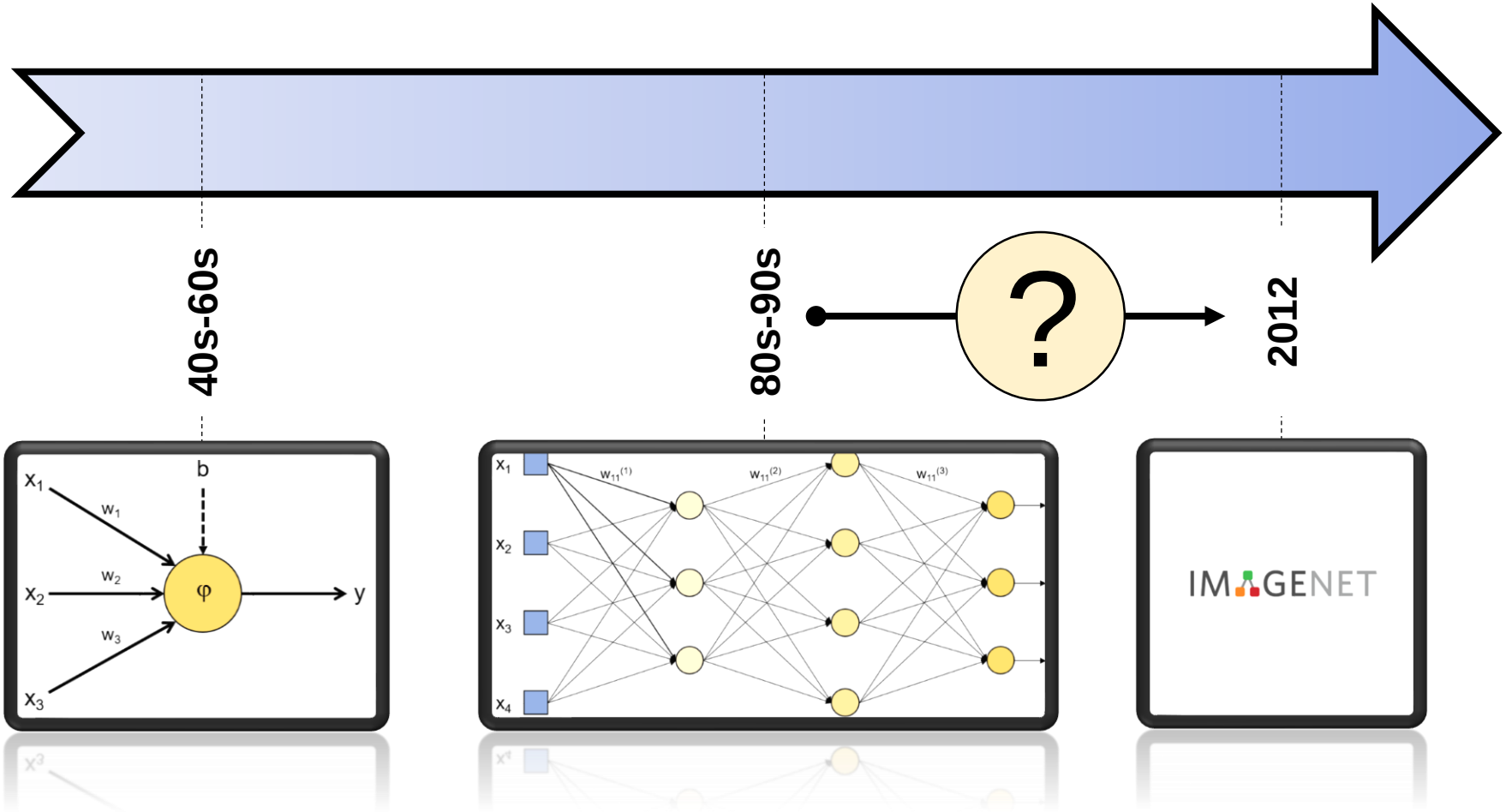
Simonyan & Zisserman. 2015. Internantional conference on learning representations

2012 : Breakthrough for image classification



Pic Credit: He et al. 2016. Conference on Computer Vision and Pattern Recognition

Why only now?

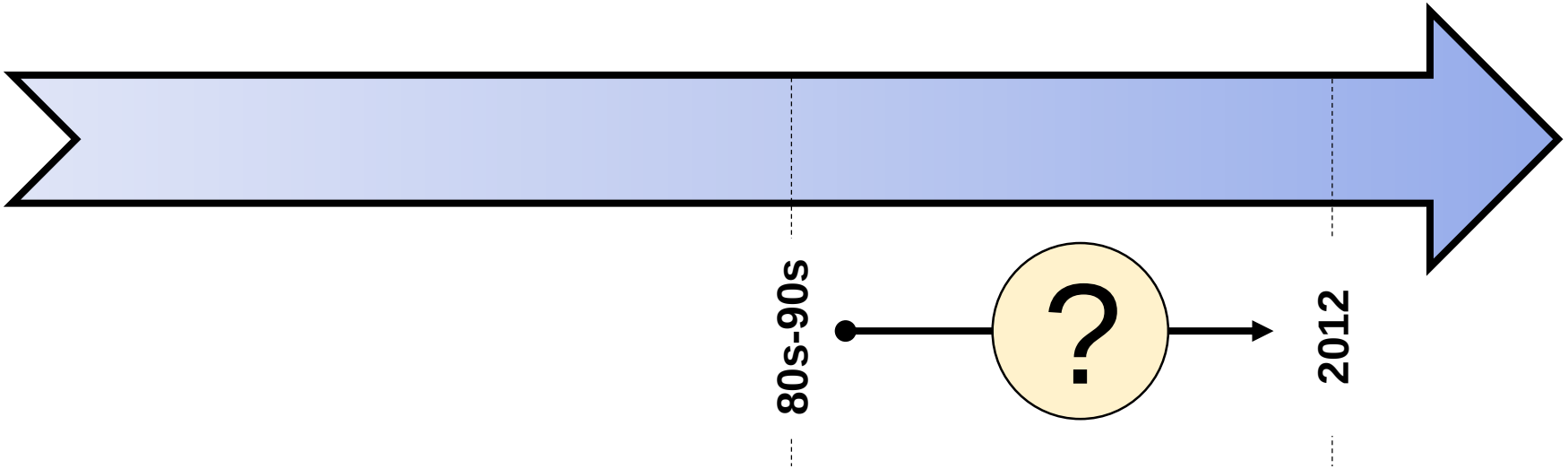


Cybernetics : linear **neural network** models able to handle simple classification tasks

Connectivism : Development of the **back-propagation** algorithm for the training of deep neural network. First **convolutional network**.

First time a **Deep Learning convolutional algorithm** wins the ImageNet competition

Why only now?



1- Internet : offers a huge source of **high-resolution annotated images**, essential for an efficient **training**.

2- Computer resources : faster CPU and GPU, able to perform **>10^{12} operations/s**

NVIDIA TESLA
>3000€

3- New algorithms : activation function (ReLu), faster optimizers, methods to avoid overfitting (Dropout), etc.

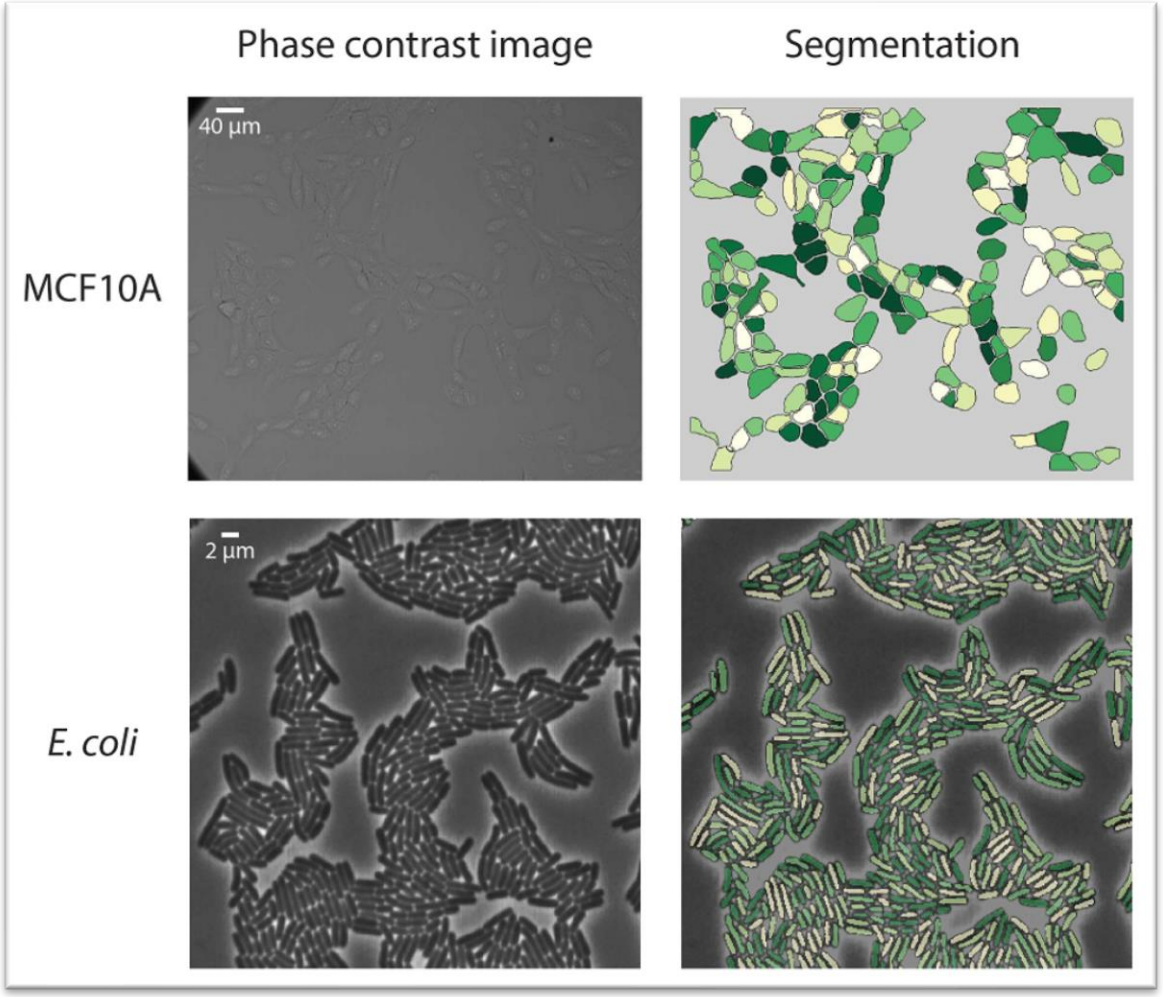
Outline

I. Introduction

i. Difference between Machine Learning (ML) & Deep Learning (DL)
ii. Brief historical overview
iii. Applications for image analysis
iv. When using DL and which tools are available?

Deep Learning applications : a few examples

I. Segmentation

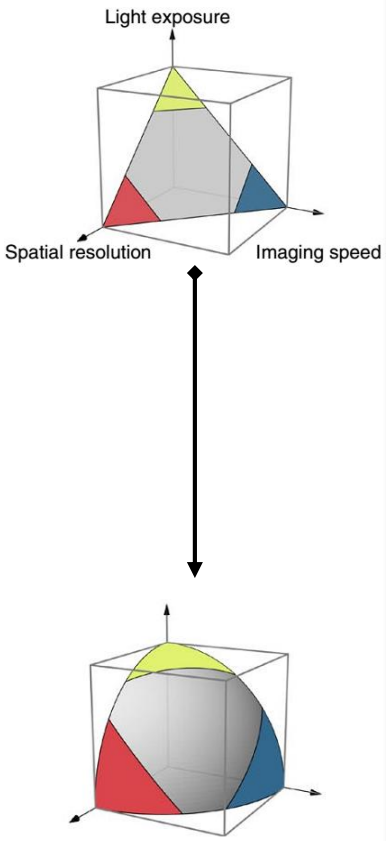


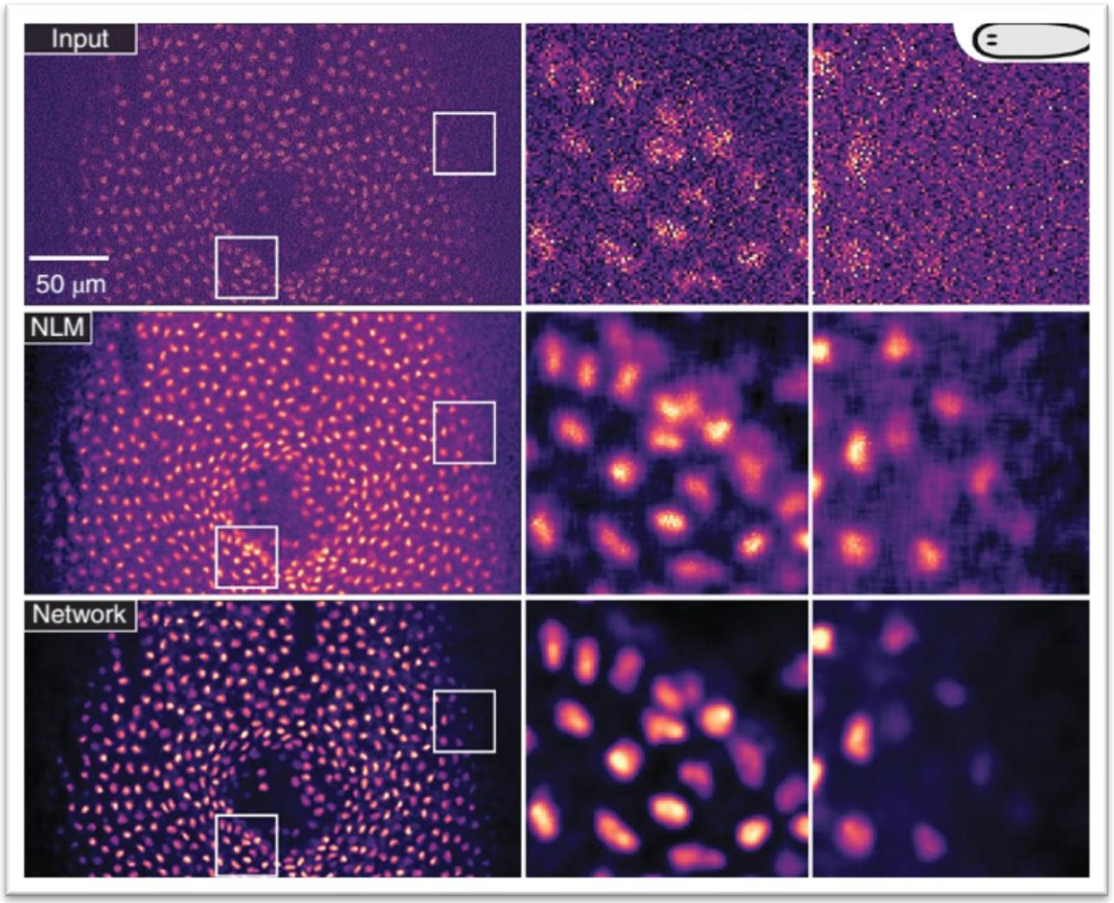
Pic credit : Van Valen et al. 2016. Plos Computational Biology

Deep Learning applications : a few examples

I. Segmentation

II. Image restoration



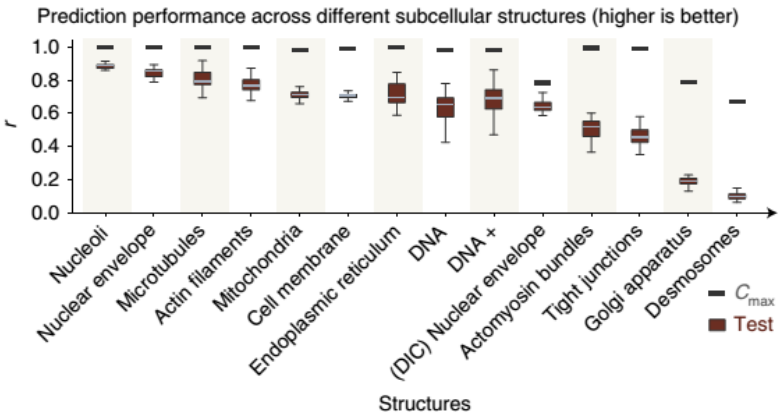


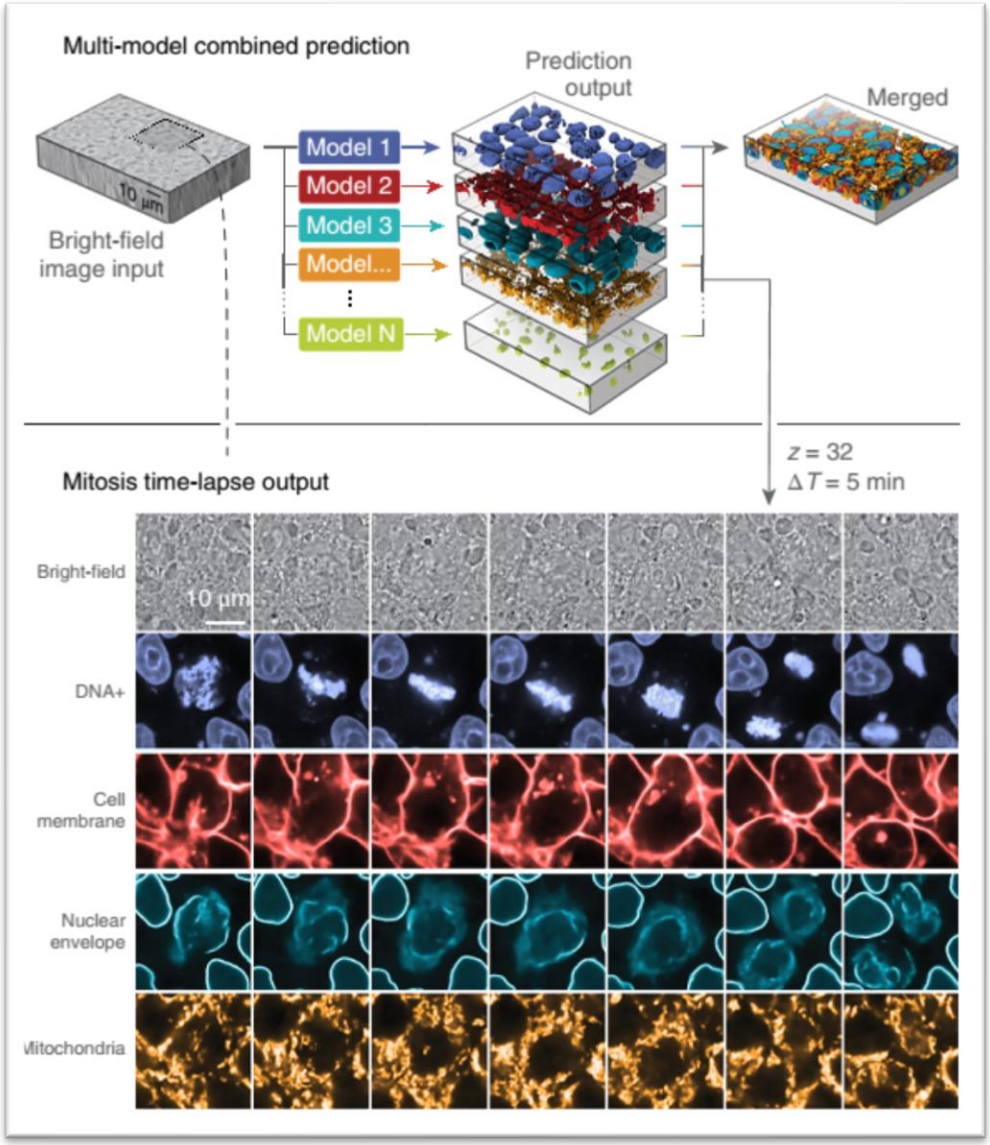
Nucleus stained of S. meditarranea (RedDot1)

Pic credit : Weigert et al. 2018. Nature Methods – Buades et al. 2005. Conference on Computer Vision and Pattern Recognition

Deep Learning applications : a few examples

I. Segmentation
II. Image restoration
III. Image prediction

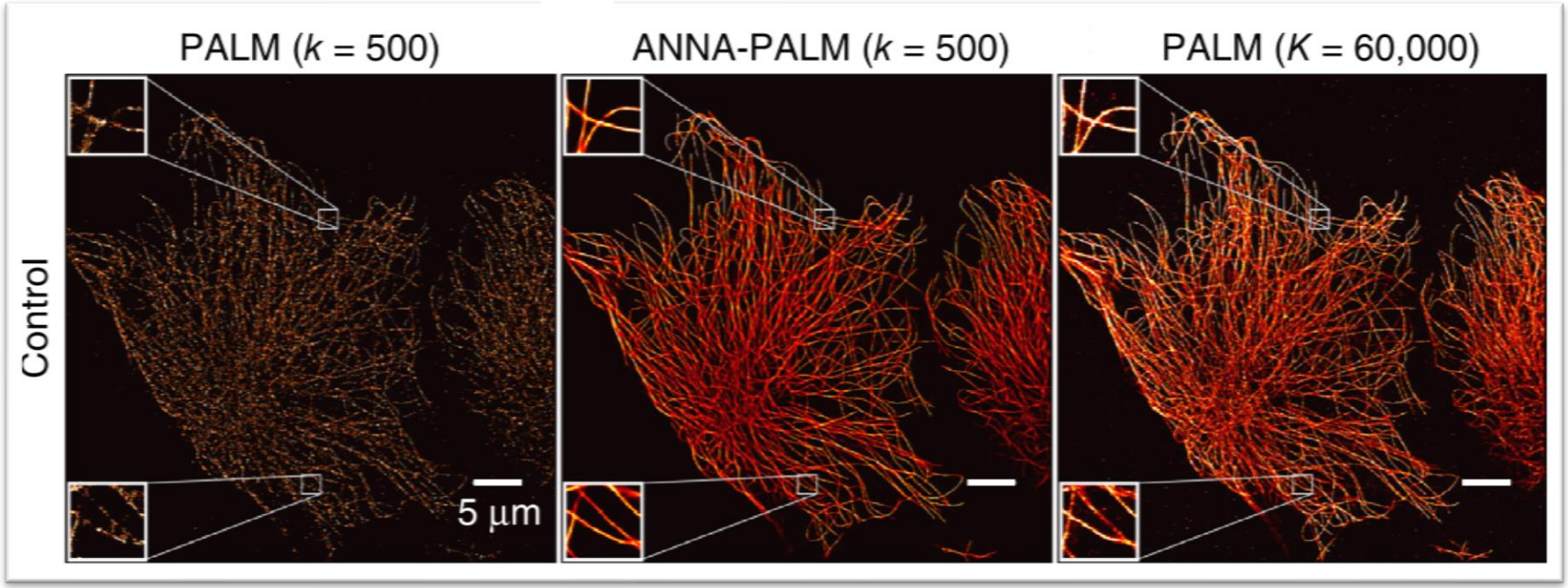




Pic credit : Ounkomol – 2018. Nature Methods

Deep Learning applications : a few examples

I. Segmentation
II. Image restoration
III. Image prediction
IV. Super-resolution prediction



Pic credit : Ouyang – 2018. Nature Biotechnology

Outline

I. Introduction

i. Difference between Machine Learning (ML) & Deep Learning (DL)
ii. Brief historical overview
iii. Applications for image analysis
iv. When using DL and which tools are available?

Why using Deep Learning?

When **classic image processing/analysis tools** are not efficient or do not exist for the task we want to perform (eg. PALM image prediction)

Need to have enough analyzed data to train the network

Need to label the data in order to get database large enough for the training

Time consuming

Network are trained for a specific set of data. New type of data means new training.

Not (always) flexible

Deep Learning needs large computational resources for image analysis

Expensive

Which tools are available for Deep Learning?

Matlab 2018 version and later

Toolkits dedicated for ANN and deep learning.

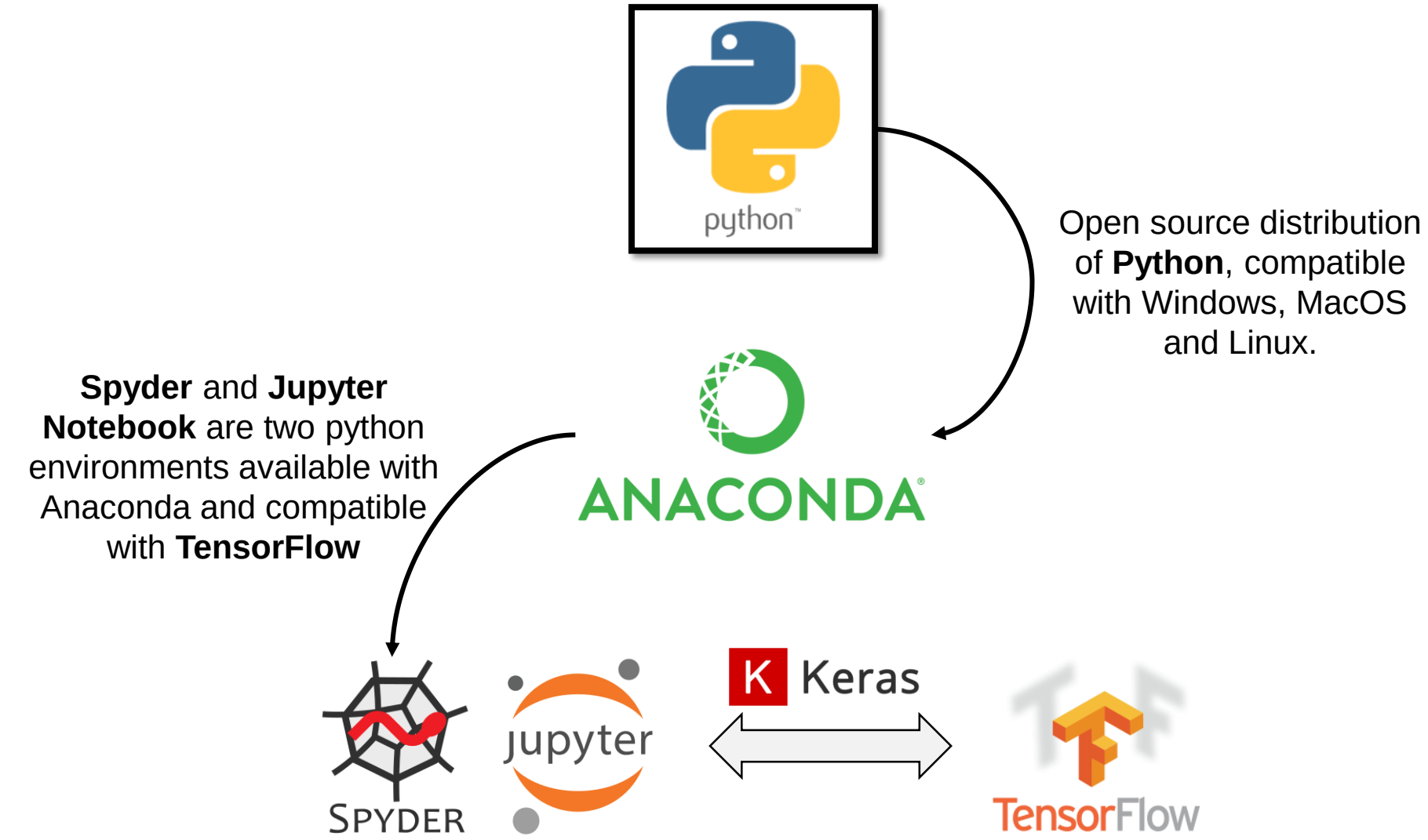
Python 3 – open source

For DL, the open-source **TensorFlow** library from Google is used.

imJoy – open source

Plugin developed by the MIT and freely available on GitHub.

Deep Learning with Python



Outline

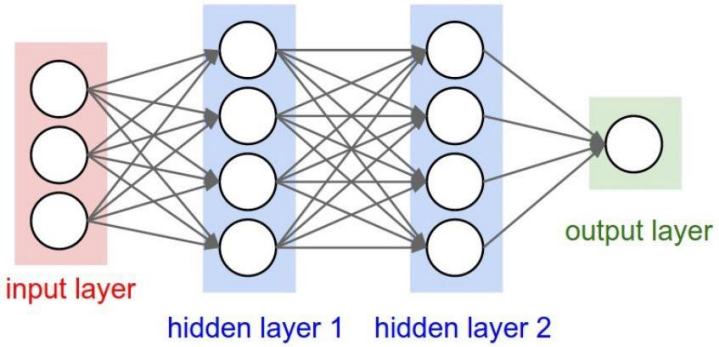
I. Introduction

II. Starting with Deep Learning and Artificial Neuron Network

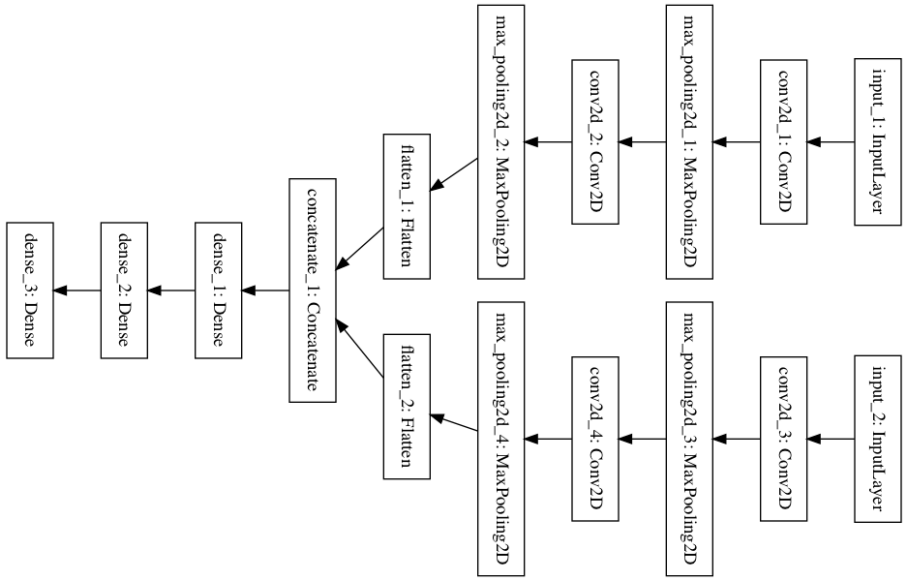
i. **A bit of vocabulary before starting**
ii. Case 1 : single neuron network
iii. Case 2 : how to construct a densely connected network
iv. Case 3 : multi-class clusterization
v. Summary

Sequential vs. non-sequential models

Sequential model

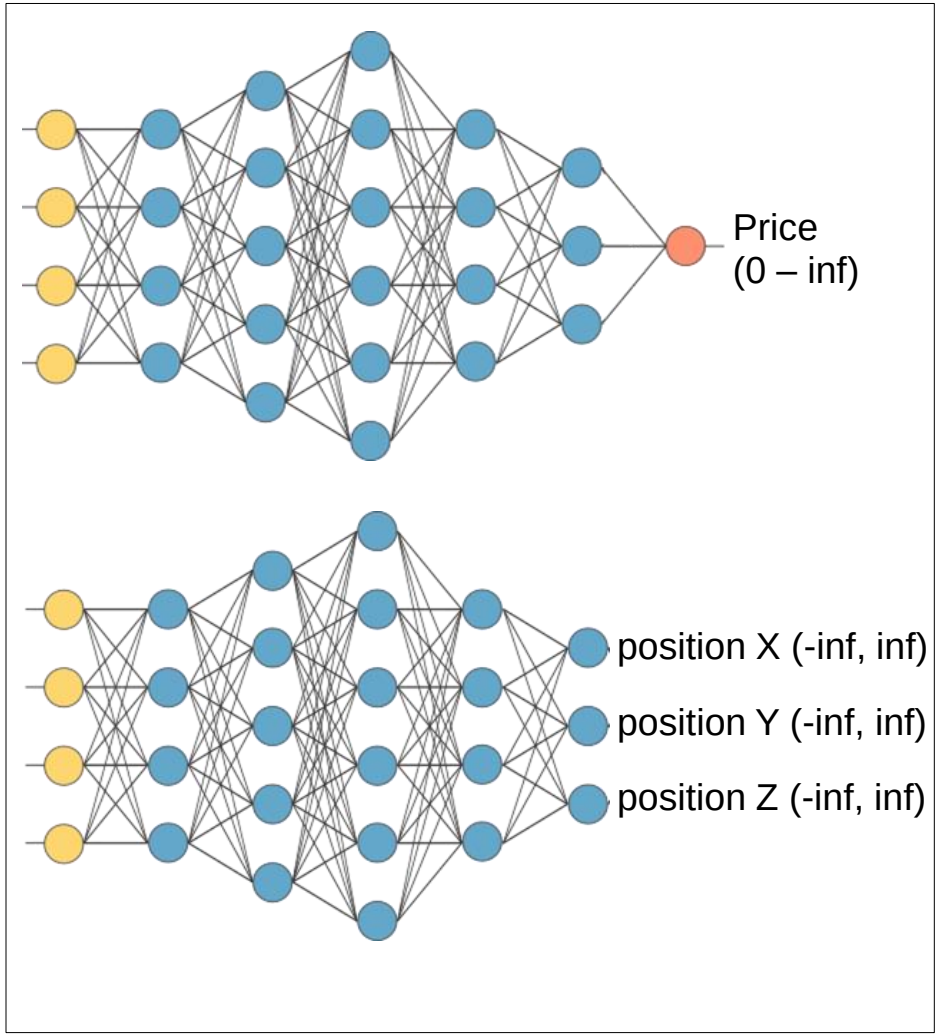


Non-sequential model

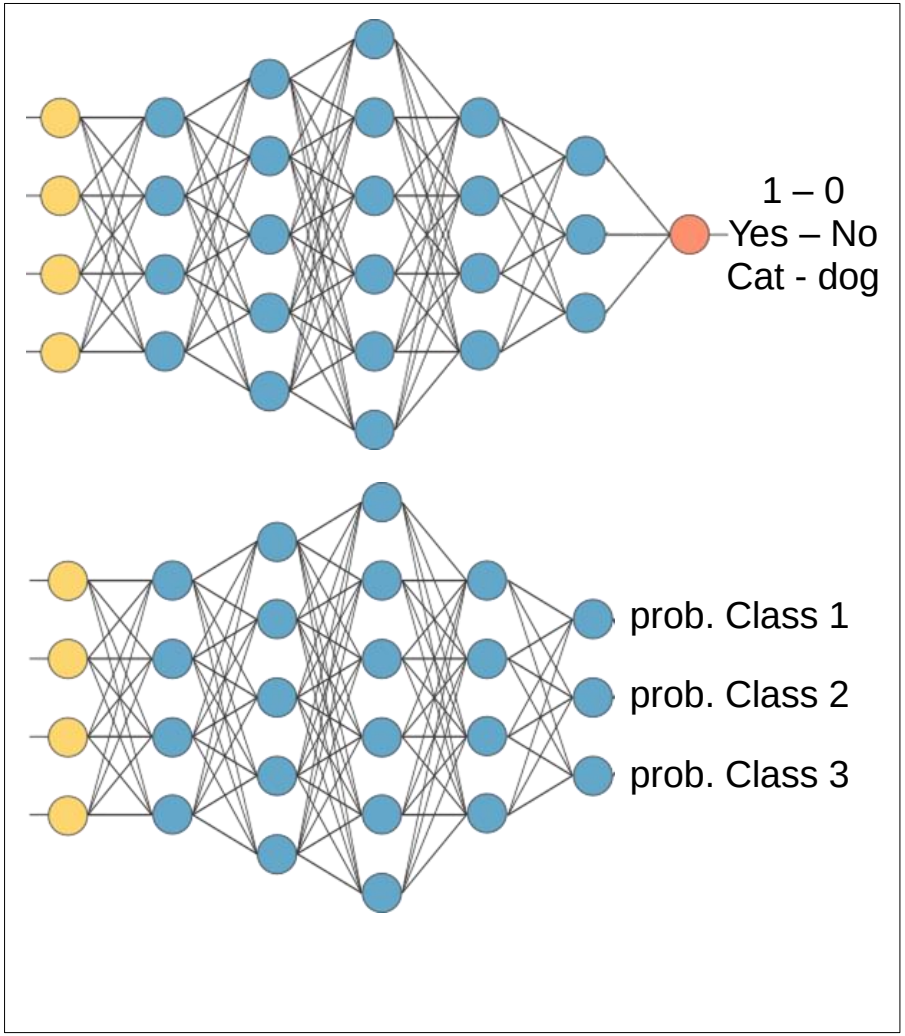


Regression vs. Classification

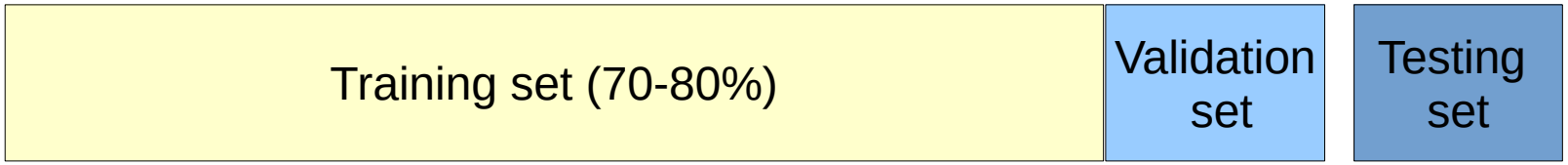
Regression : output is one or more real numbers

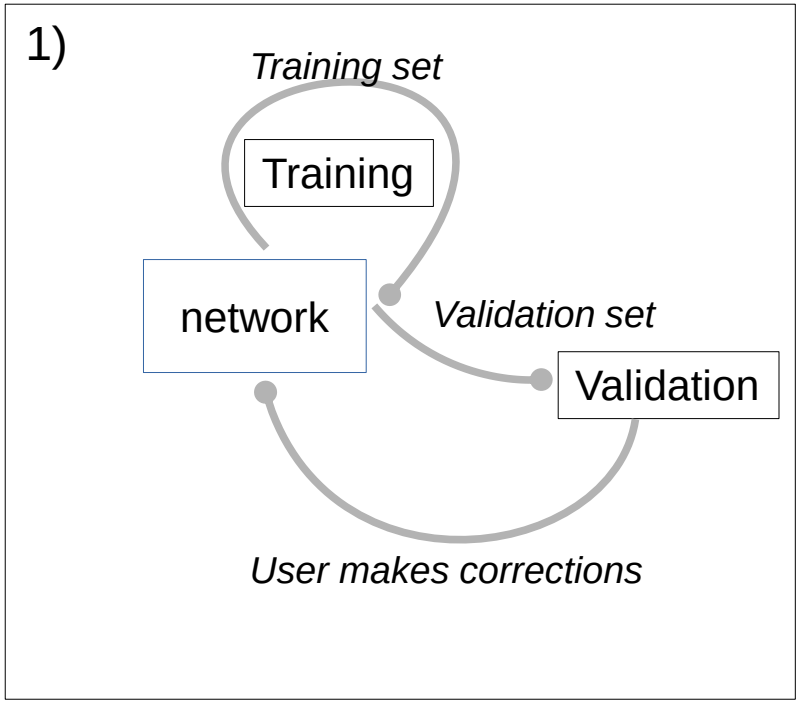


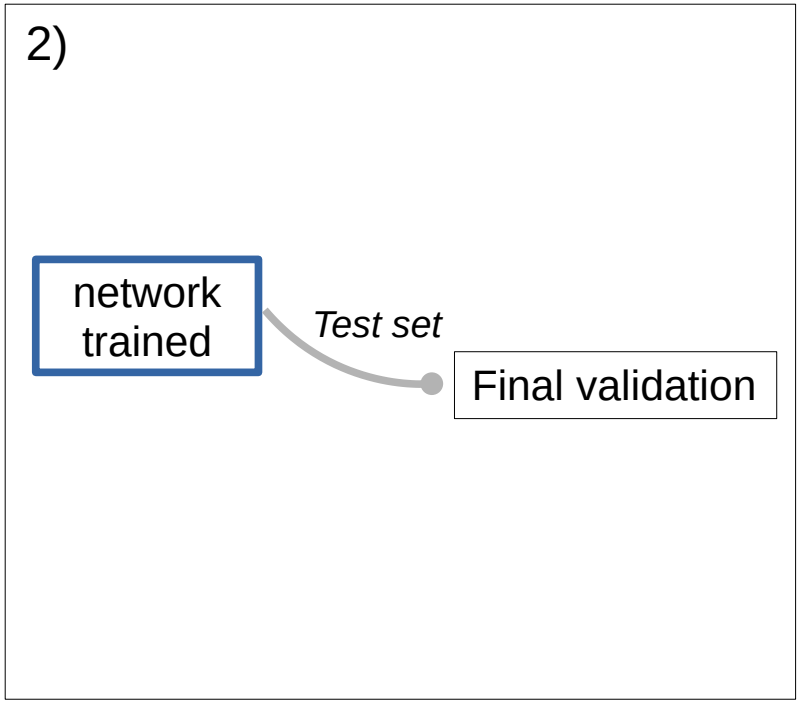
Classification : output is the probability that input belong to one or more classes



Training, Testing and validation sets







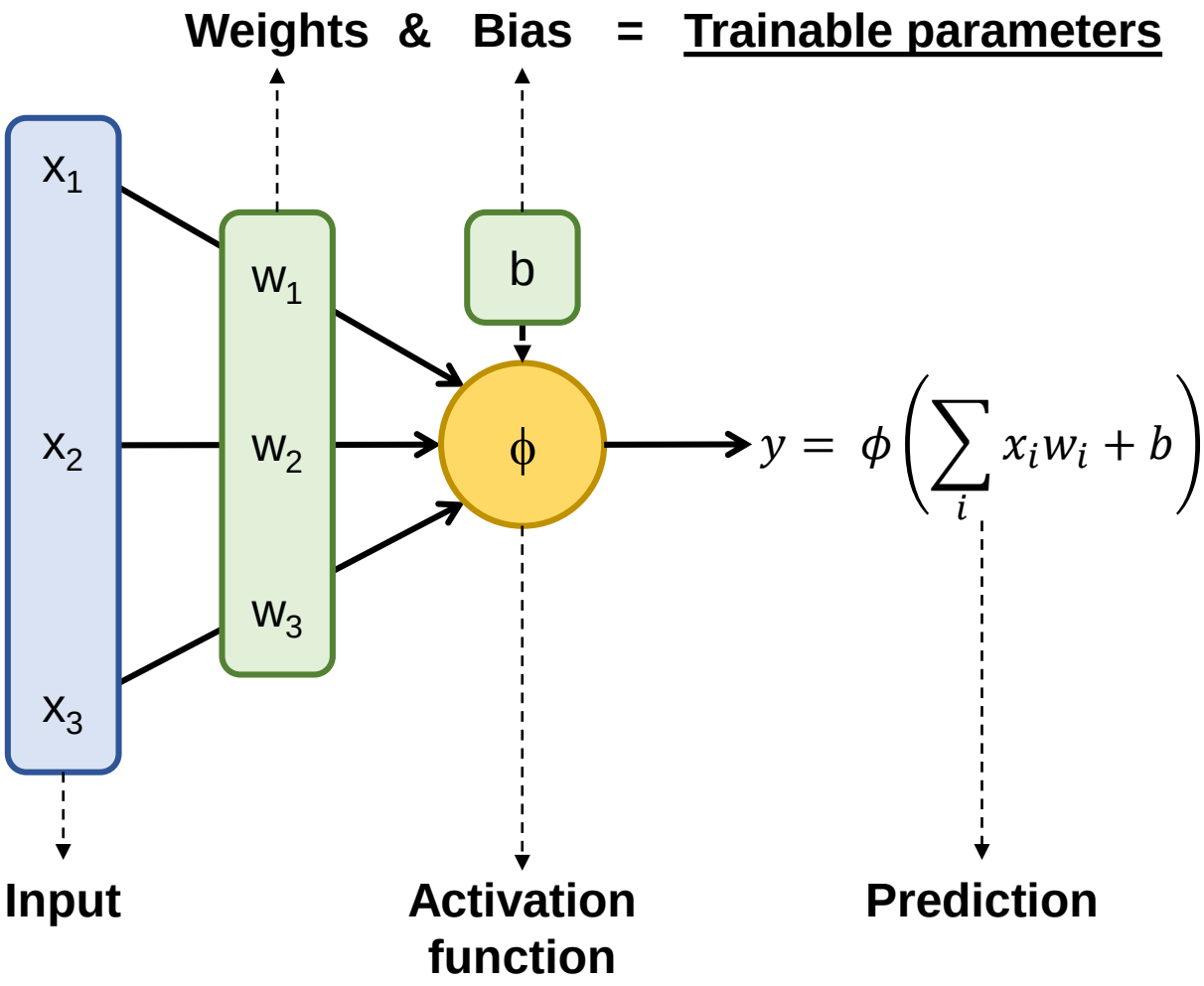
Outline

I. Introduction

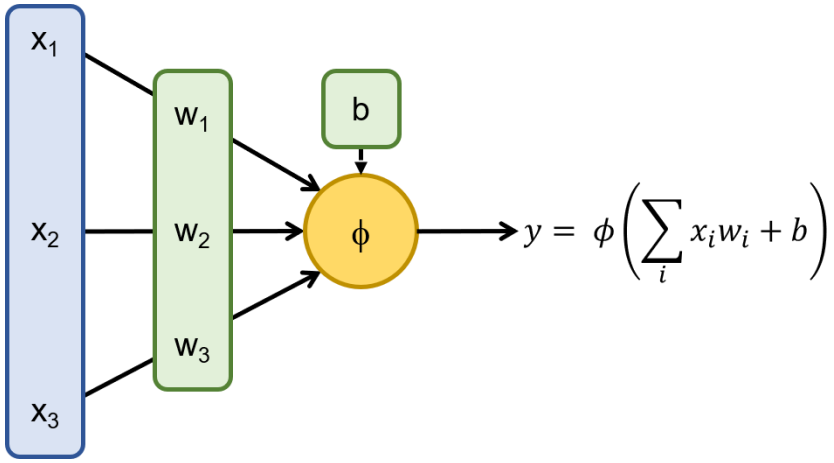
II. Starting with Deep Learning and Artificial Neuron Network

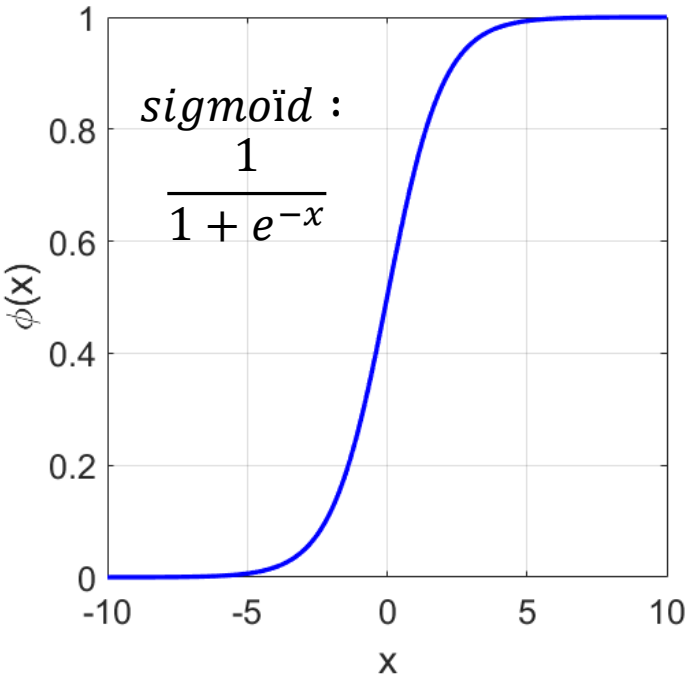
i. A bit of vocabulary before starting
ii. Case 1 : single neuron network
iii. Case 2 : how to construct a densely connected network
iv. Case 3 : multi-class clusterization
v. Summary

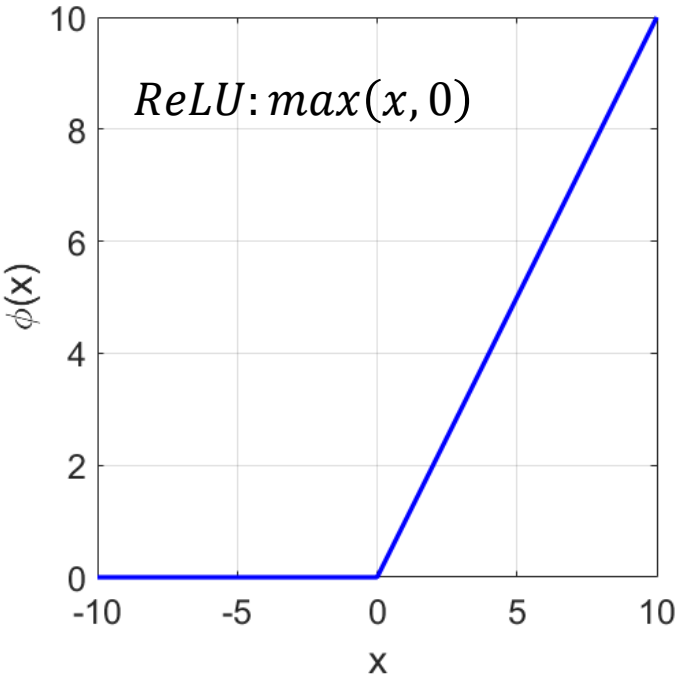
Definition of an artificial neuron



Most common activation functions



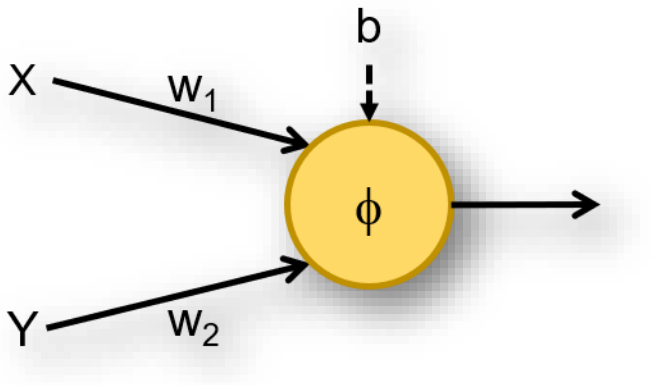




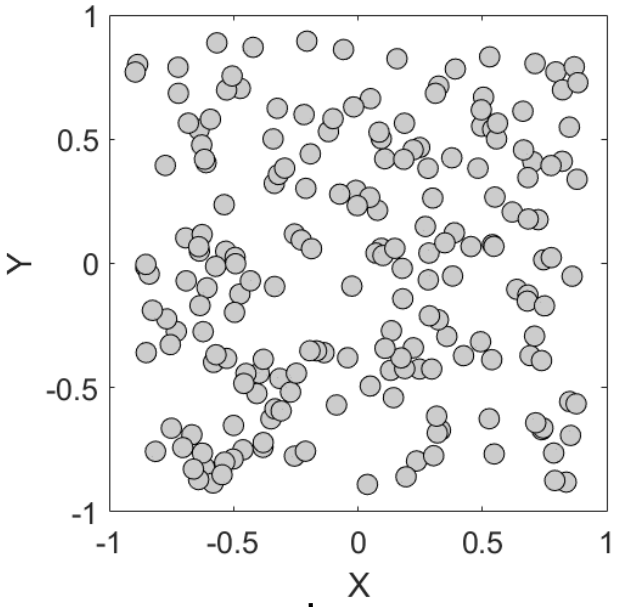
Example 1 : train a simple classifier with one neuron

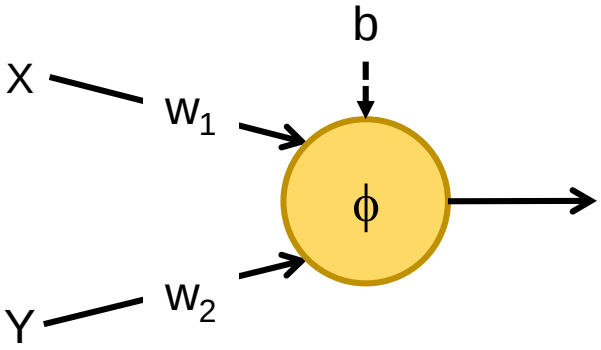
Example n°1 : Clusterization_linearly_separated.ipynb

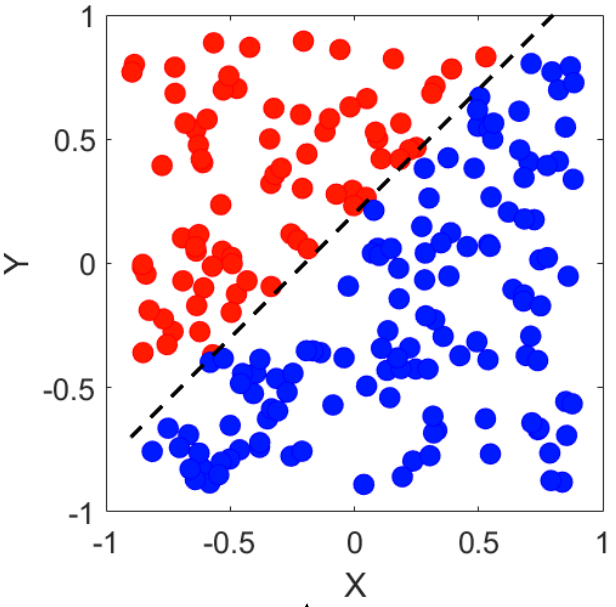
1. Understand the principle of the training
2. Train the classifier and test its accuracy
3. First step with Keras/TensorFlow



Example 1 : A one neuron classifier







Input data: X,Y coordinates

Goal of the training = find the best set of parameters {w_1, w_2, b} allowing for an accurate clusterization of the input data

Prediction : class red or blue

= 0

= 1

Definition of the classifier with Keras

1- Definition of the network architecture

```
from keras import models
from keras import layers

model = models.Sequential()
model.add(layers.Dense(1,activation='sigmoid', input_shape=(2,)))
```

2- Definition of the training options

```
model.compile(optimizer='sgd',
              loss='binary_crossentropy',
              metrics=['accuracy'])
```

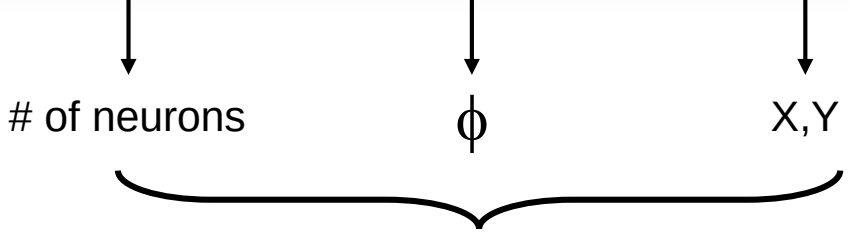
3- Training

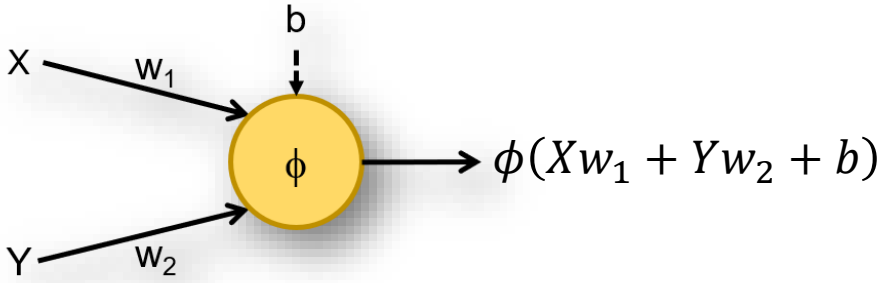
```
history = model.fit(Training_data,
                    Training_label,
                    epochs = 50,
                    validation_data = (Validation_data, Validation_label))
```

Definition of the classifier with Keras

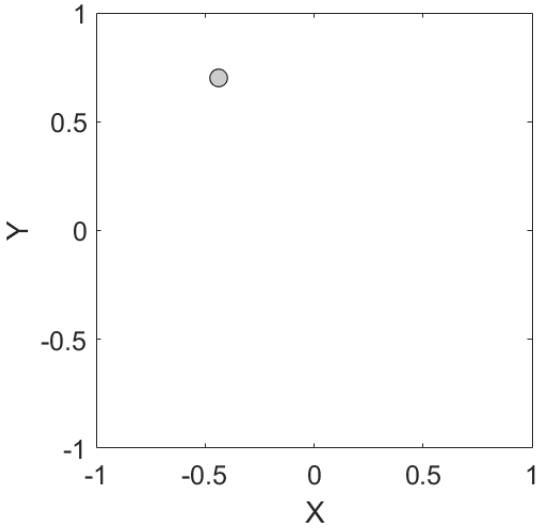
1- Definition of the network architecture

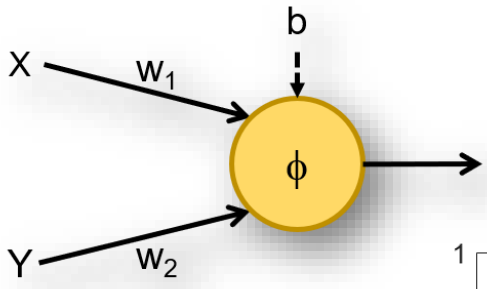
```
from keras import models
from keras import layers

model = models.Sequential()
model.add(layers.Dense(1,activation='sigmoid', input_shape=(2,)))
```





Training : step by step explanation

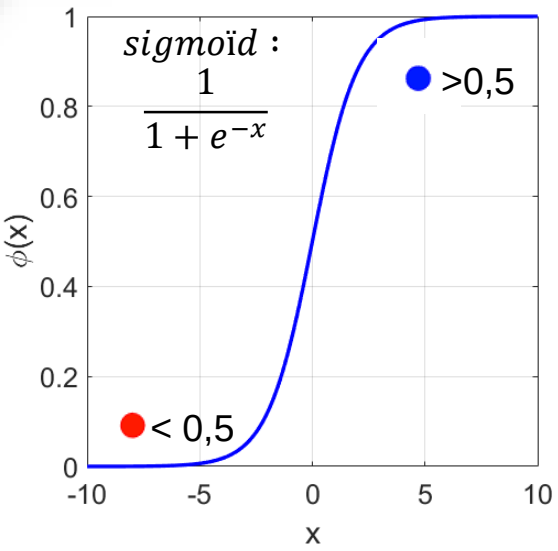




$\phi(Xw_1 + Yw_2 + b)$

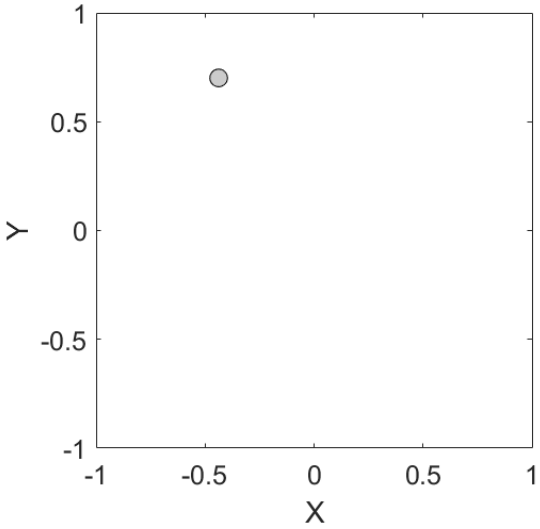
The sigmoid is chosen as activation function and a “prediction” is calculated

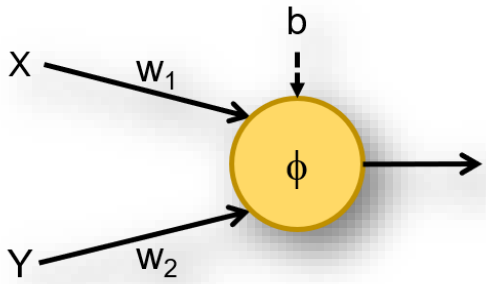
The weights are usually randomly initialized and the bias set to zero.



Training : step by step explanation

The **loss function** is used to **measure how far the prediction is from the expected result.**

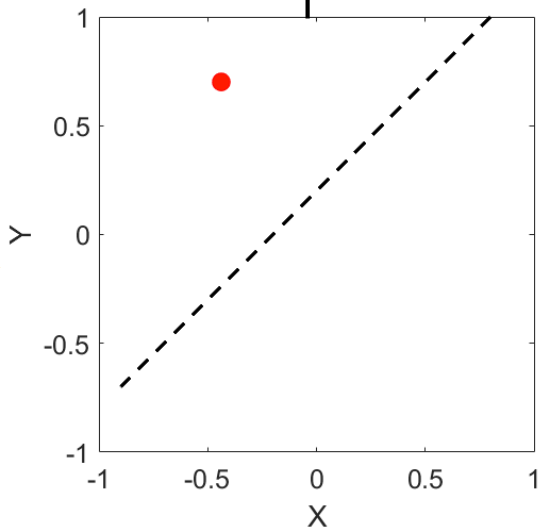




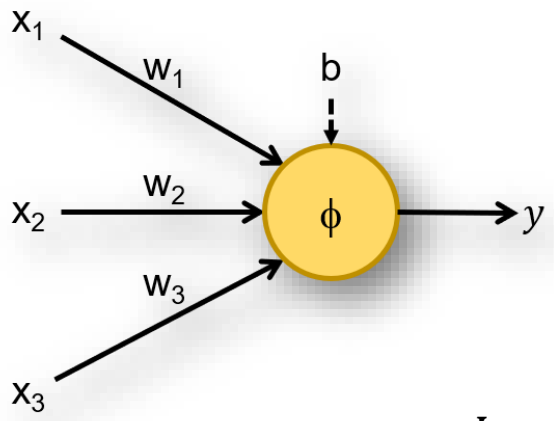
$$\phi(Xw_1 + Yw_2 + b)$$

+/-
Loss
function

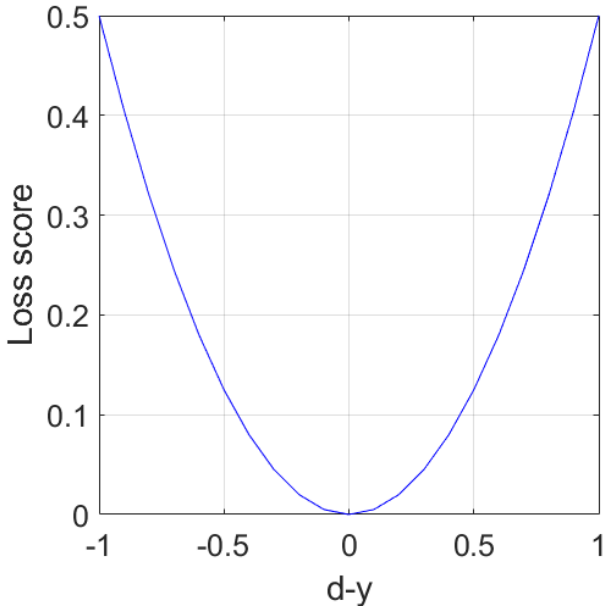
The **calculated output** from the neuron is **compared to the expected result.**



What is the loss function?

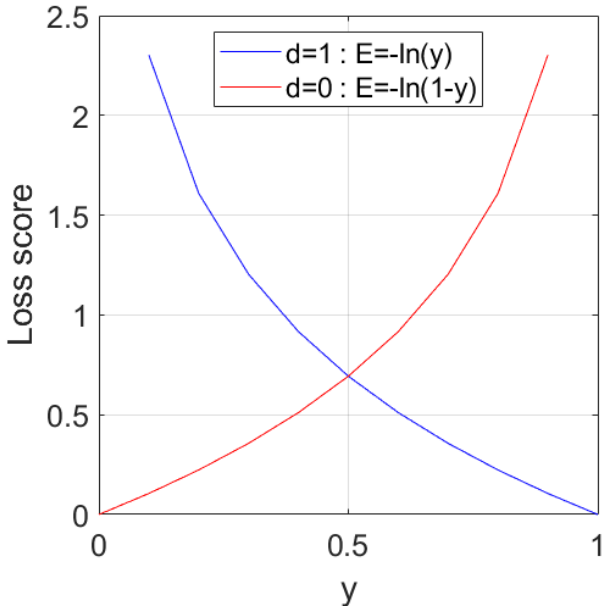


$$J = \frac{1}{2}(d_i - y_i)^2$$



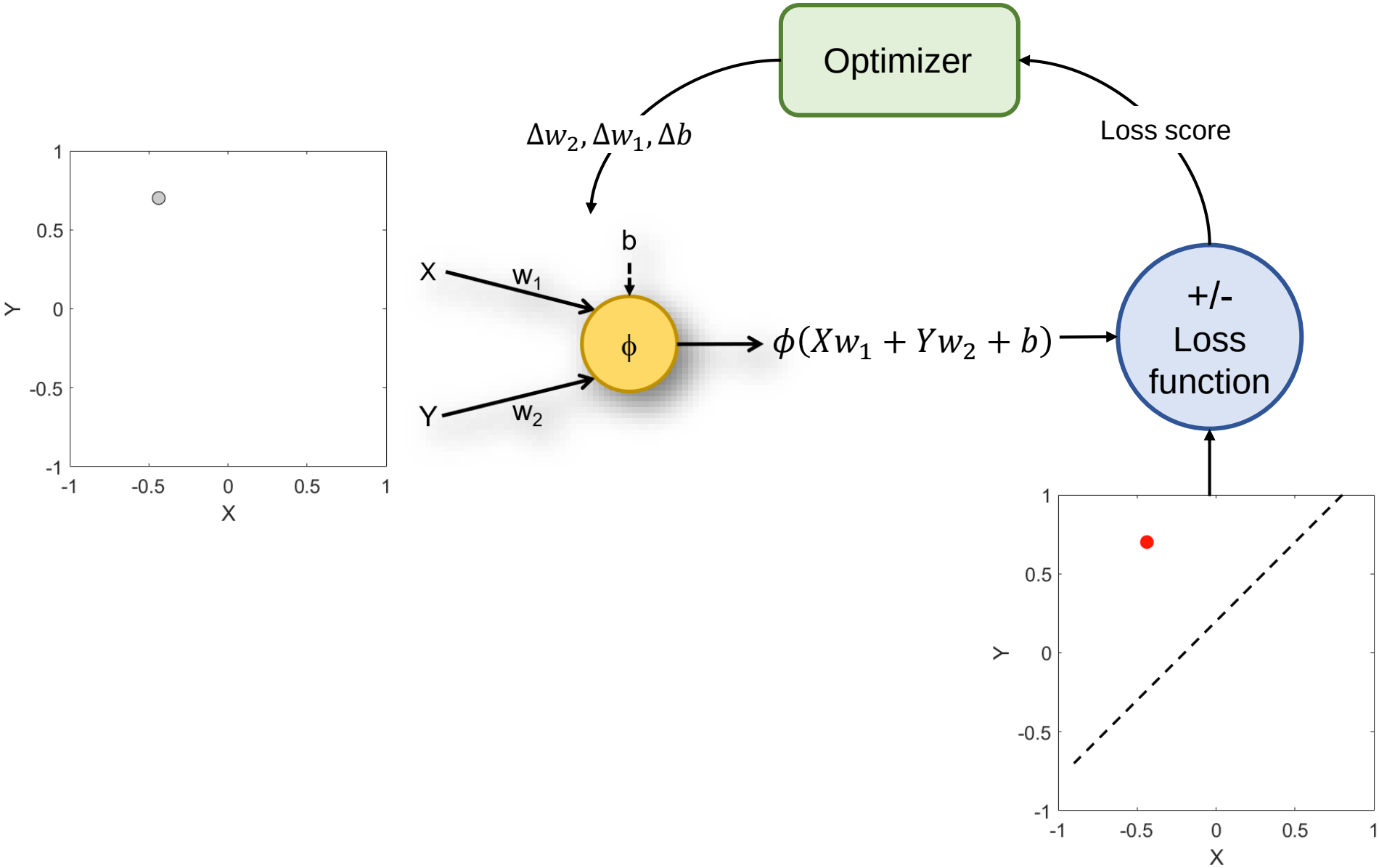
Squared error function, mainly used for regression problems.

$$J = \{-d_i ln(y_i) - (1 - d_i) ln(1 - y_i)\}$$



Binary crossed entropy, used in classification problems.

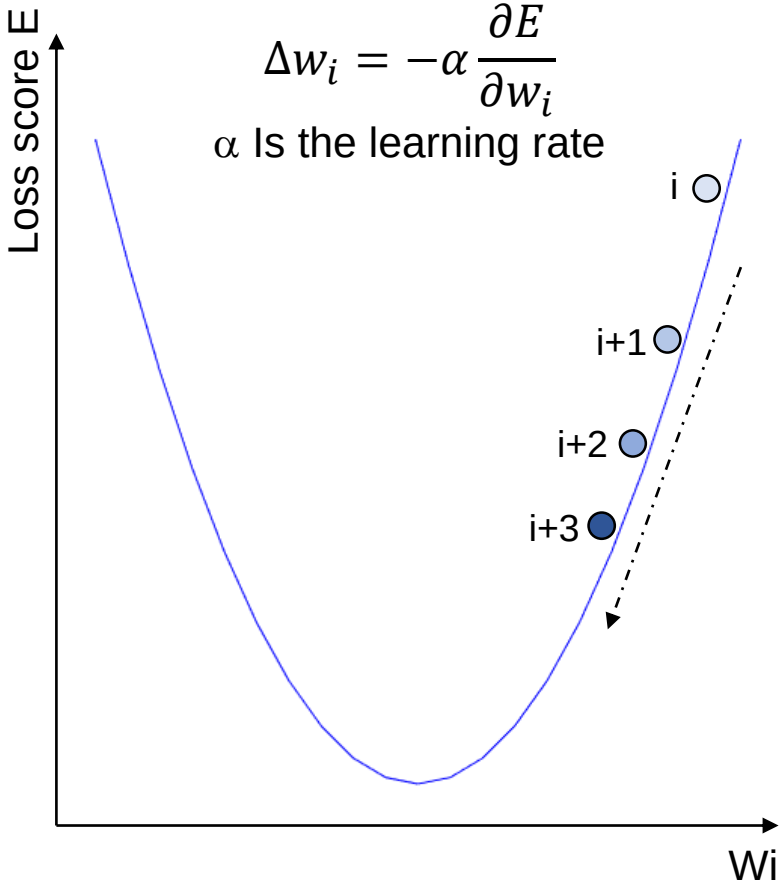
Training : step by step explanation

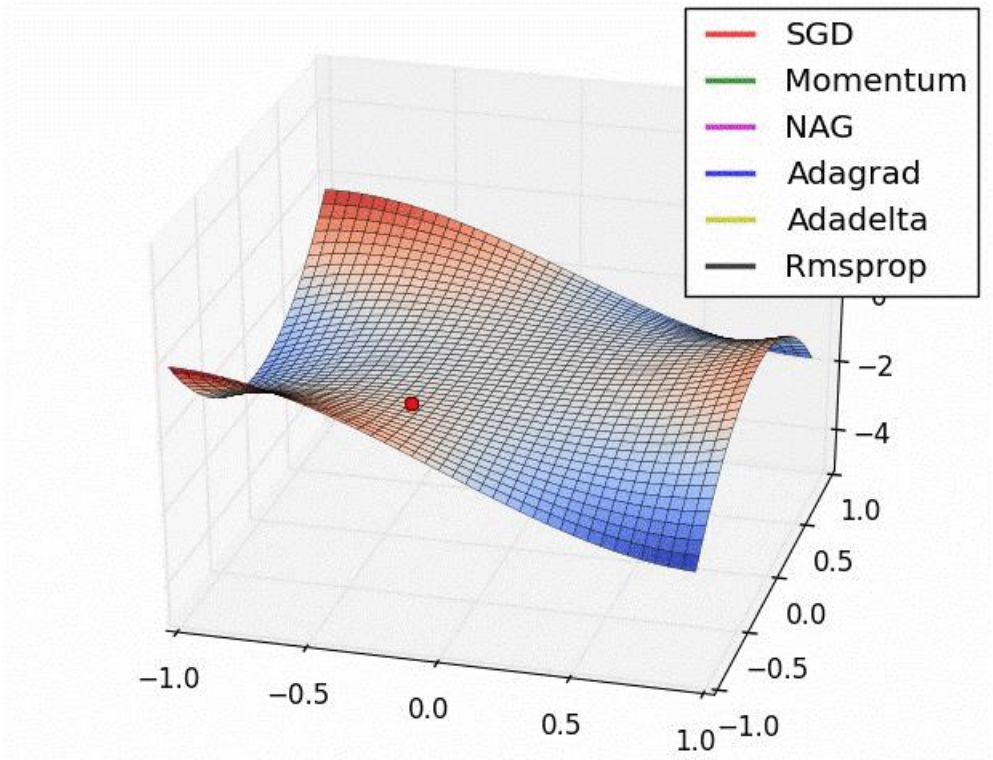


Optimizer : single gradient descent SGD

$$\Delta w_i = -\alpha \frac{\partial E}{\partial w_i}$$

α Is the learning rate



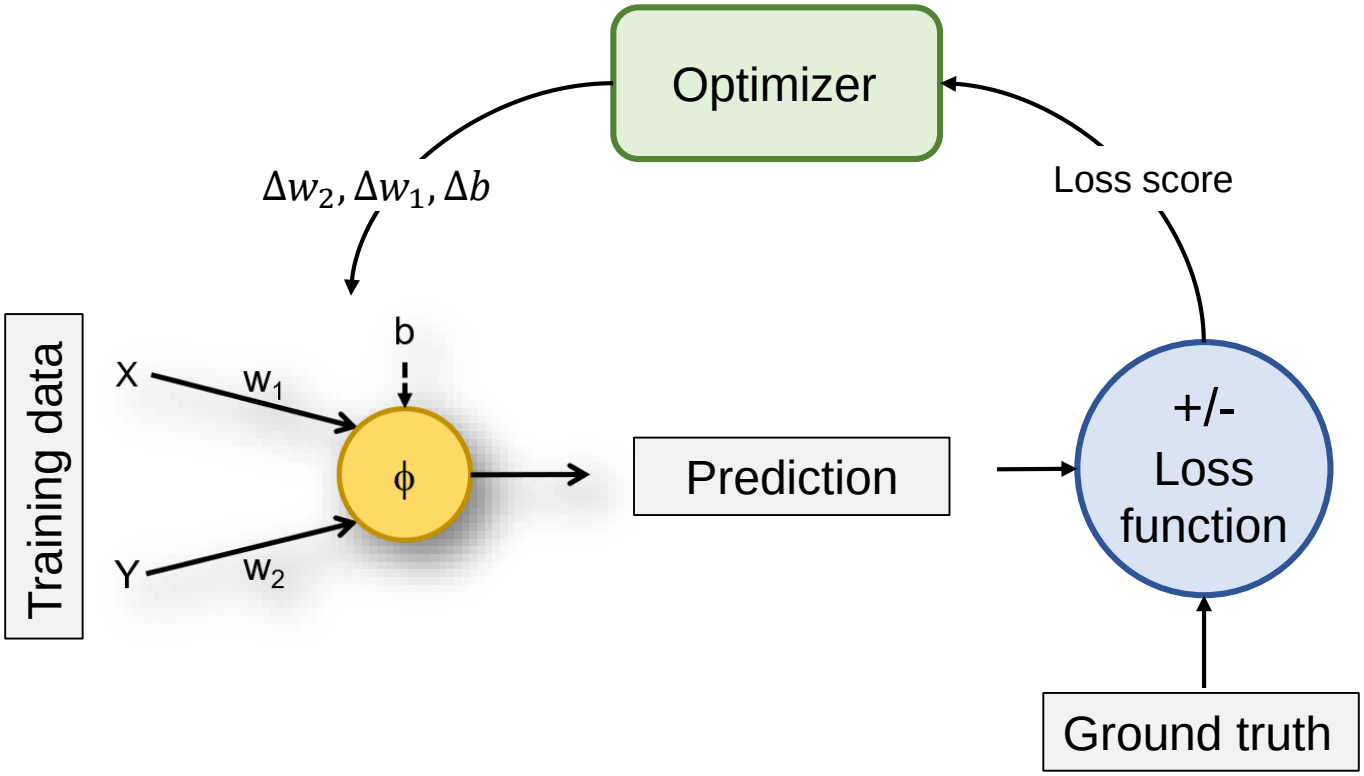


Pic credit : https://medium.com/@ramrajchandradevan/the-evolution-of-gradient-descend-optimization-algorithm-4106a6702d39

Training : step by step explanation

2- Definition of the training options

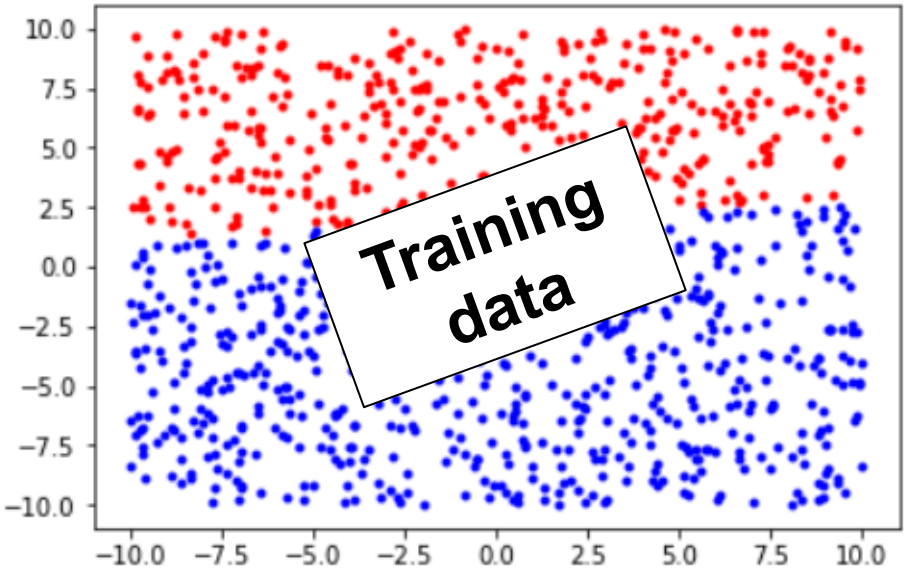
```
model.compile(optimizer='sgd',
              loss='binary_crossentropy',
              metrics=['accuracy'])
```

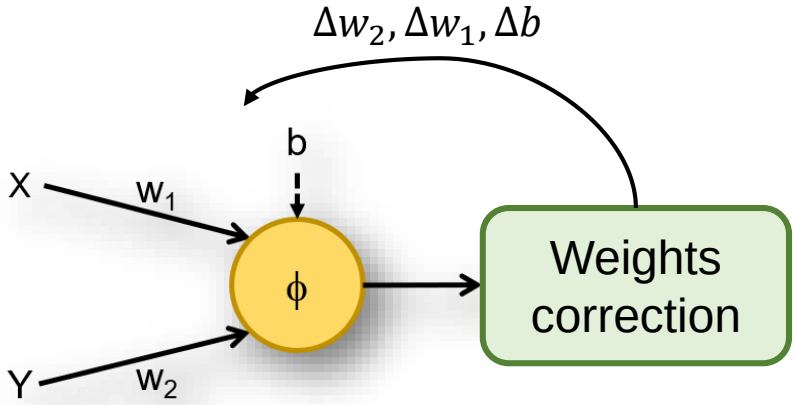


Training results

3- Training

```
history = model.fit(Training_data,
                    Training_label,
                    epochs = 20,
                    validation_data = (Validation_data, Validation_label))
```

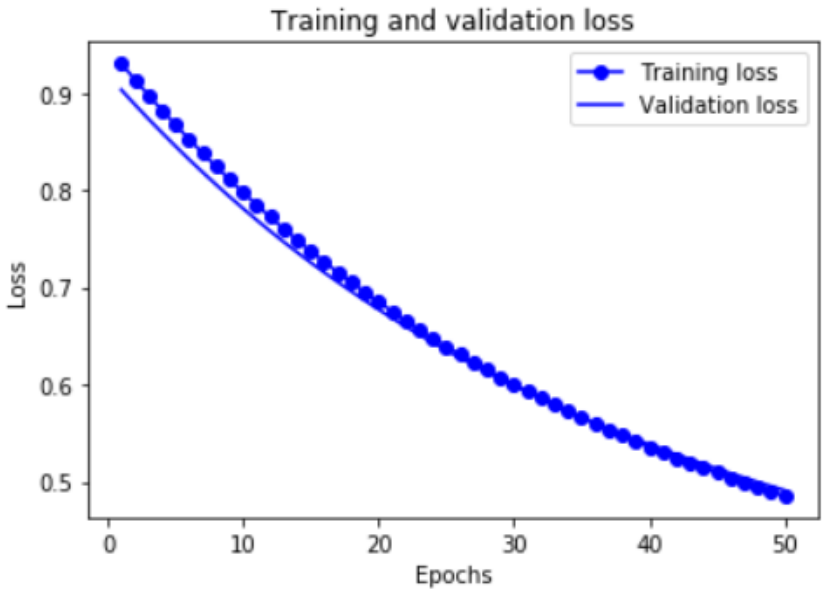


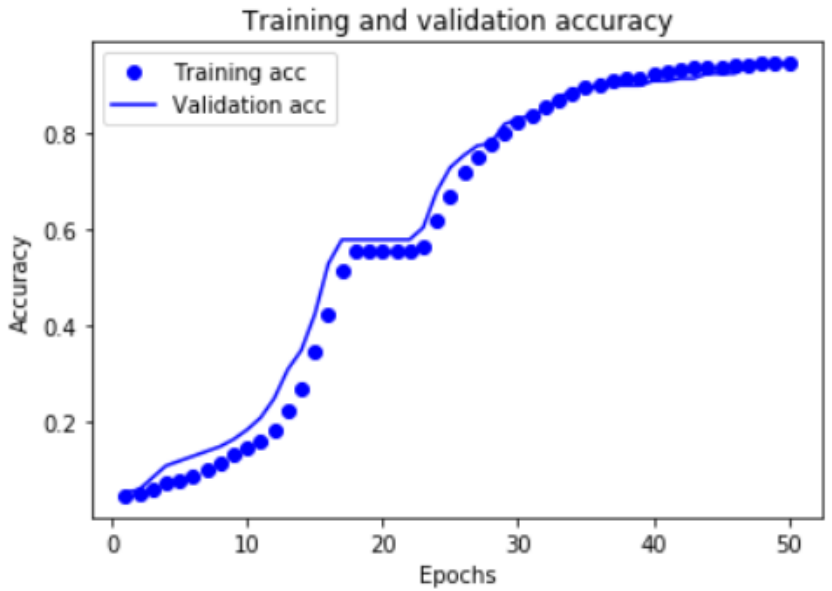


Training results

3- Training

```
history = model.fit(Training_data,
                    Training_label,
                    epochs = 50,
                    validation_data = (Validation_data, Validation_label))
```

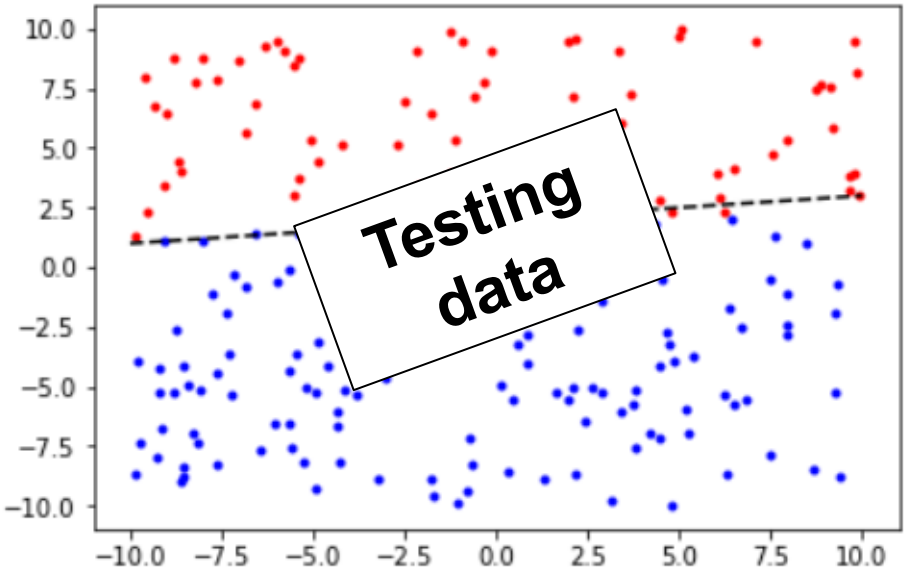




Validation of the training using the testing set

Estimation of the model accuracy

```
Predication_accuracy = model.evaluate(Testing_data, Testing_label)
print(Predication_accuracy)
```

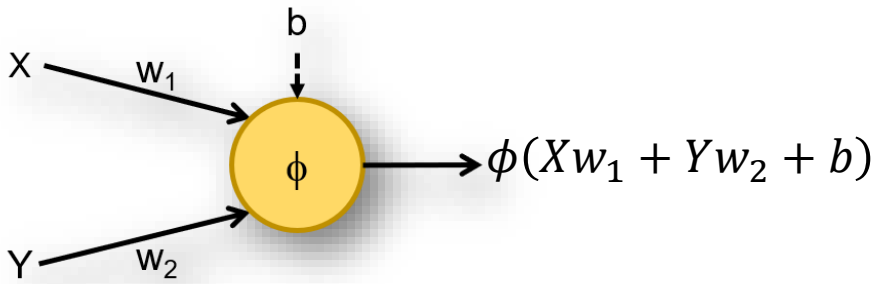


Global accuracy 98%

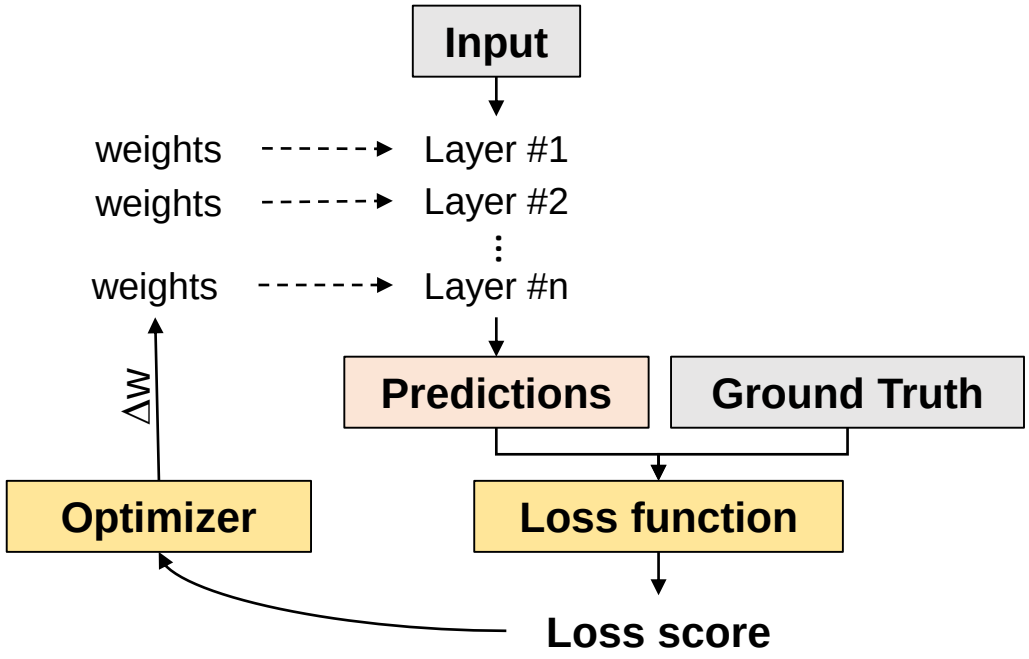
What did we learn so far?

- How a single neuron works :
  - **Activation function**
  - **Input / output**
  - Weights and bias

  } To be defined by the user



- How the training works :
  - **Loss function**
  - **Optimizer**
  - **Learning rate**



Outline

I. Introduction

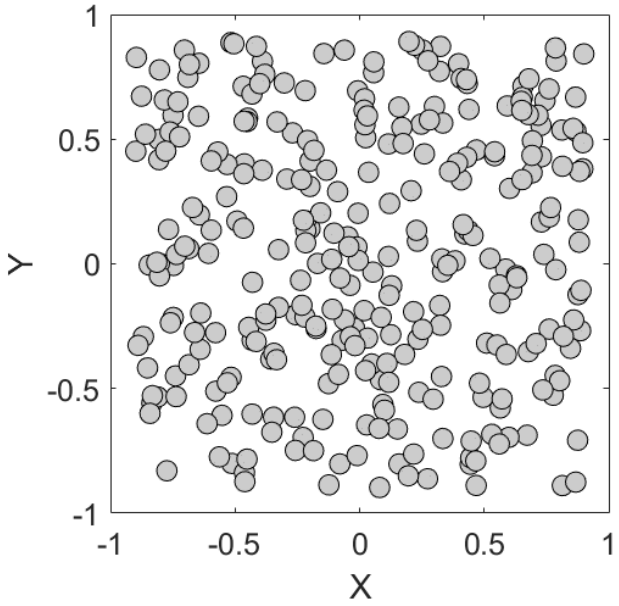
II. Starting with Deep Learning and Artificial Neuron Network

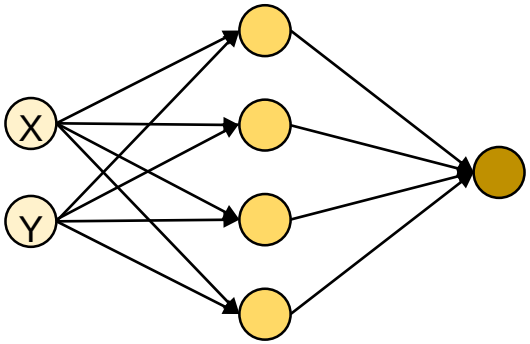
i. A bit of vocabulary before starting
ii. Case 1 : single neuron network
iii. Case 2 : how to construct a densely connected network
iv. Case 3 : multi-class clusterization
v. Summary

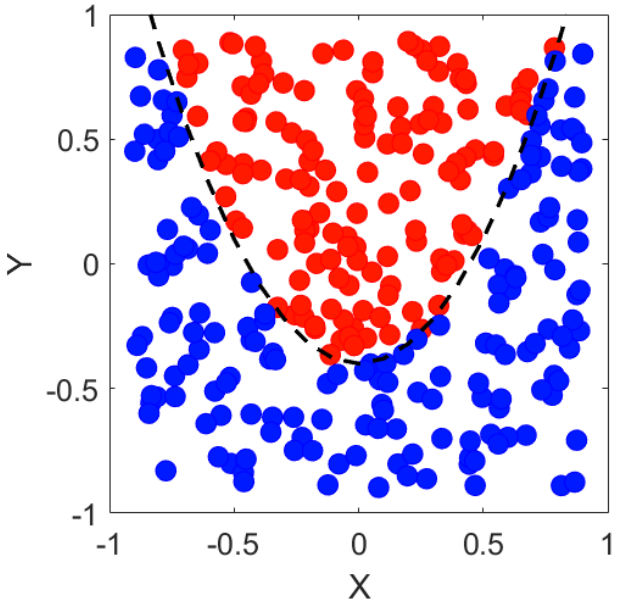
Example 2 : classify non-linearly separable data

Example n°2 : Clusterization_not_linearly_separated_parabole

1. Observe the limitations of single-layer model
2. Find a simple architecture able to solve this classification problem



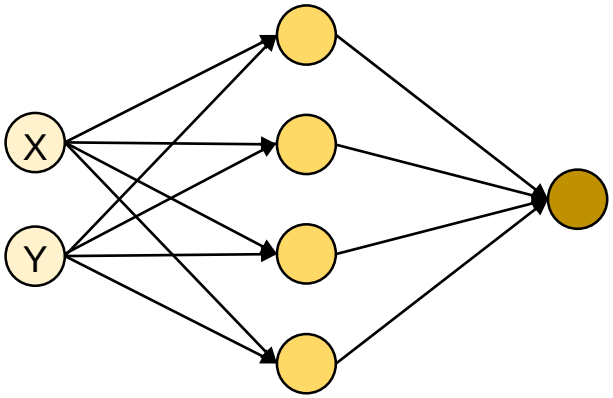




Example 2 : classify non-linearly separable data

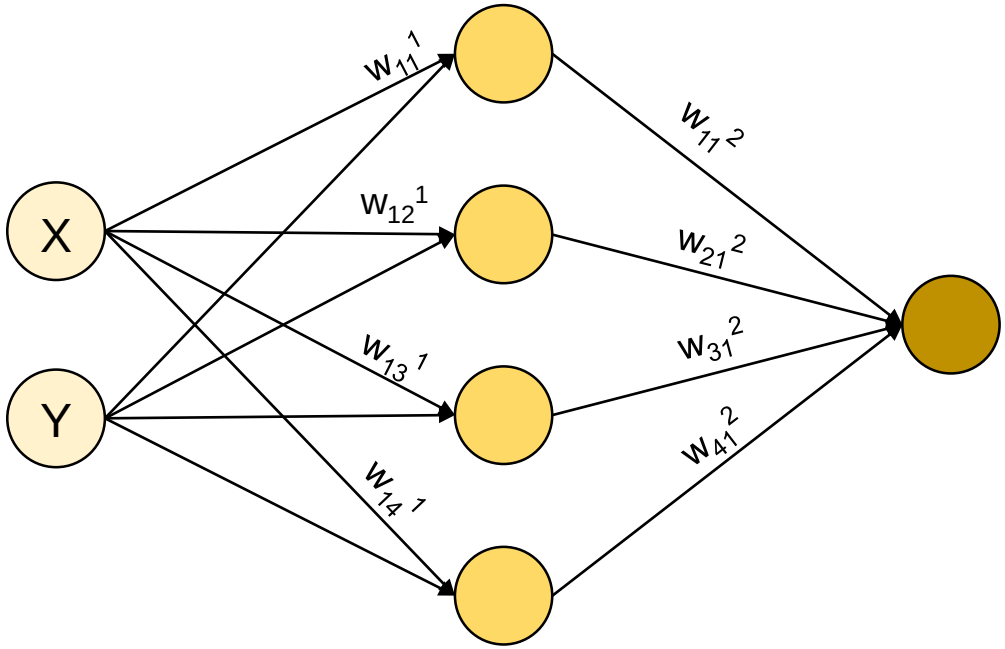
I. Try to train the classifier with one single neuron :

II. Repeat the same procedure using more neurons to “fit” the model :




```
model = models.Sequential()
model.add(layers.Dense(4,activation='sigmoid', input_shape=(2,)))
model.add(layers.Dense(1,activation='sigmoid'))

model.compile(optimizer = optimizers.sgd(lr=5e-2),
              loss='binary_crossentropy',
              metrics=['accuracy'])
```

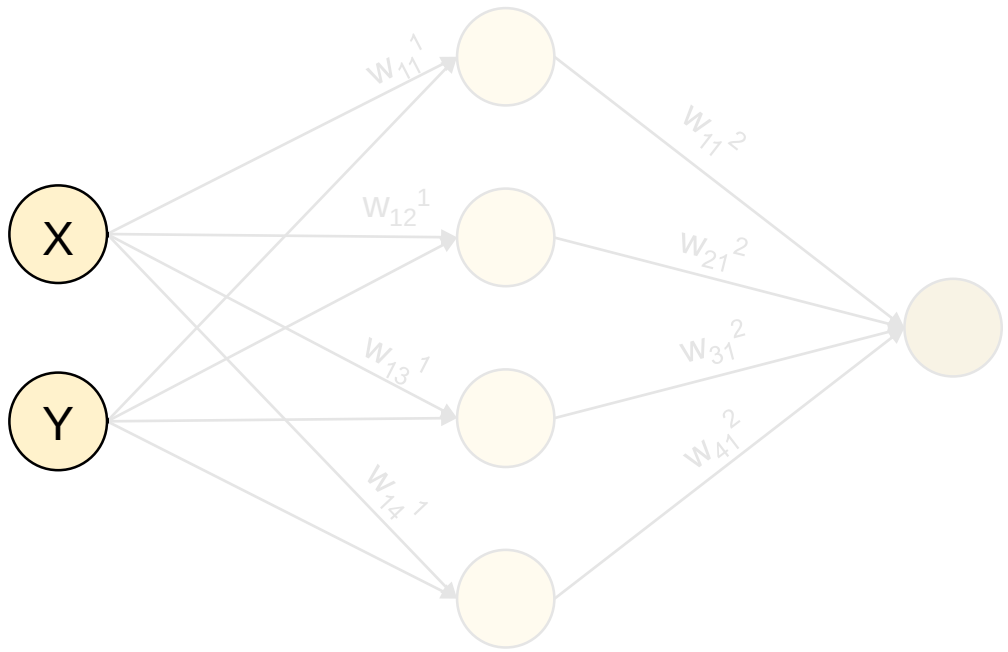
Back-propagation algorithm : a quick glimpse



Input | Hidden layer 4 neurons | Output

Total number of **trainable parameters** : 4*2 + 4 + 4 + 1 = 17

Back-propagation algorithm : a quick glimpse



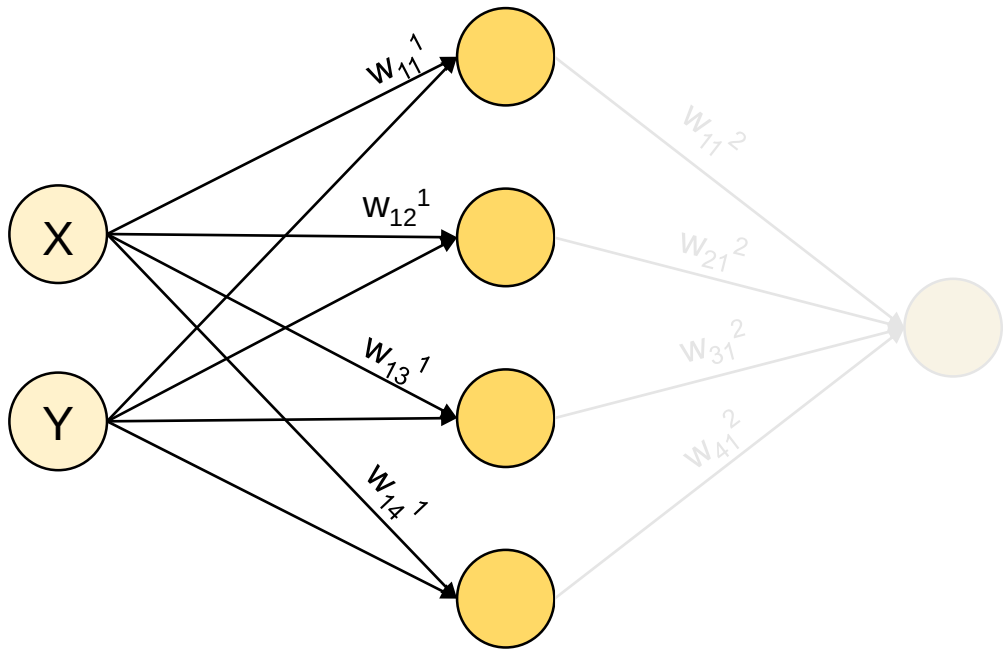
Input

Hidden layer
4 neurons

Output

Forward pass

Back-propagation algorithm : a quick glimpse



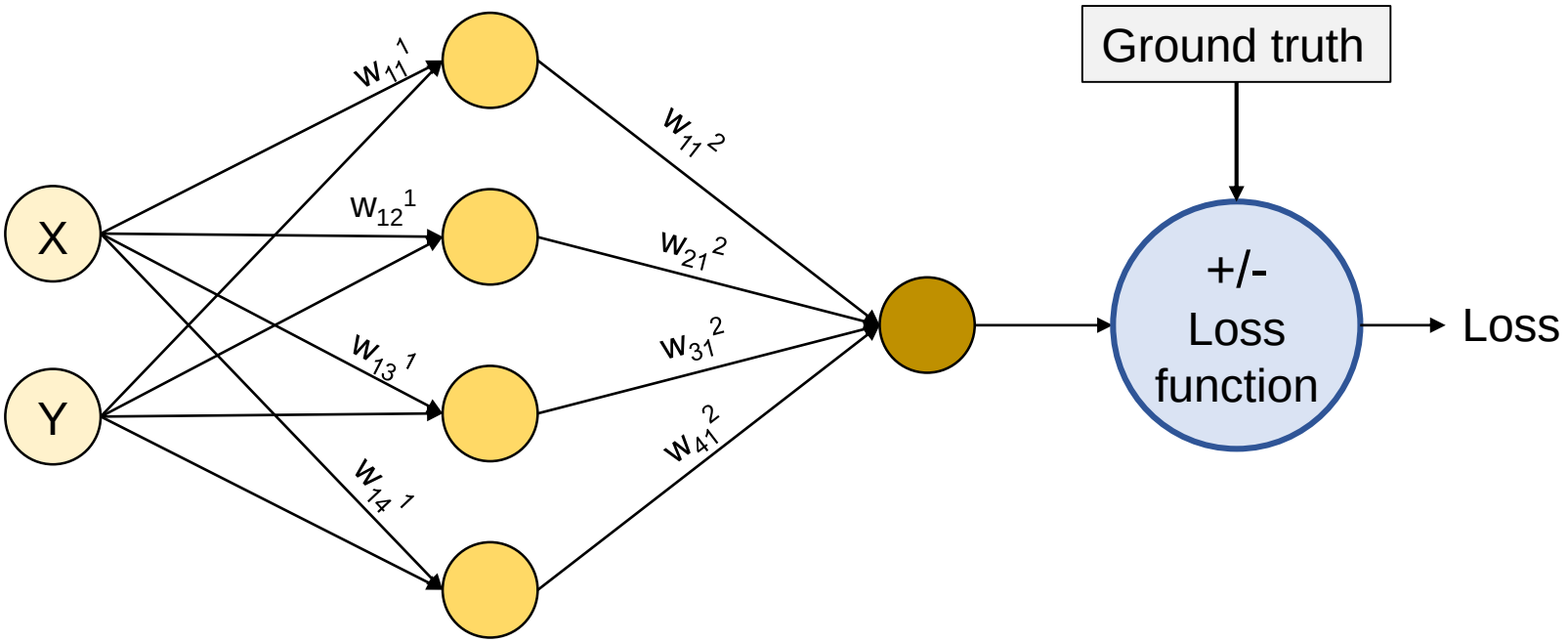
Input

Hidden layer
4 neurons

Output

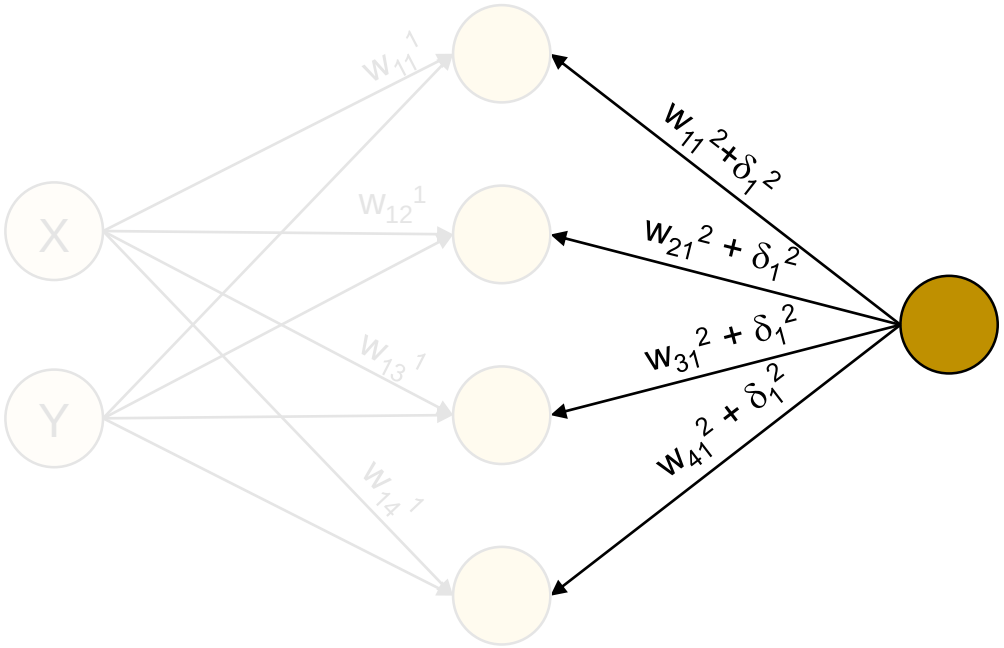
Forward pass

Back-propagation algorithm : a quick glimpse



Forward pass

Back-propagation algorithm : a quick glimpse



$$Minimize\ \frac{\partial Loss}{\partial W}$$

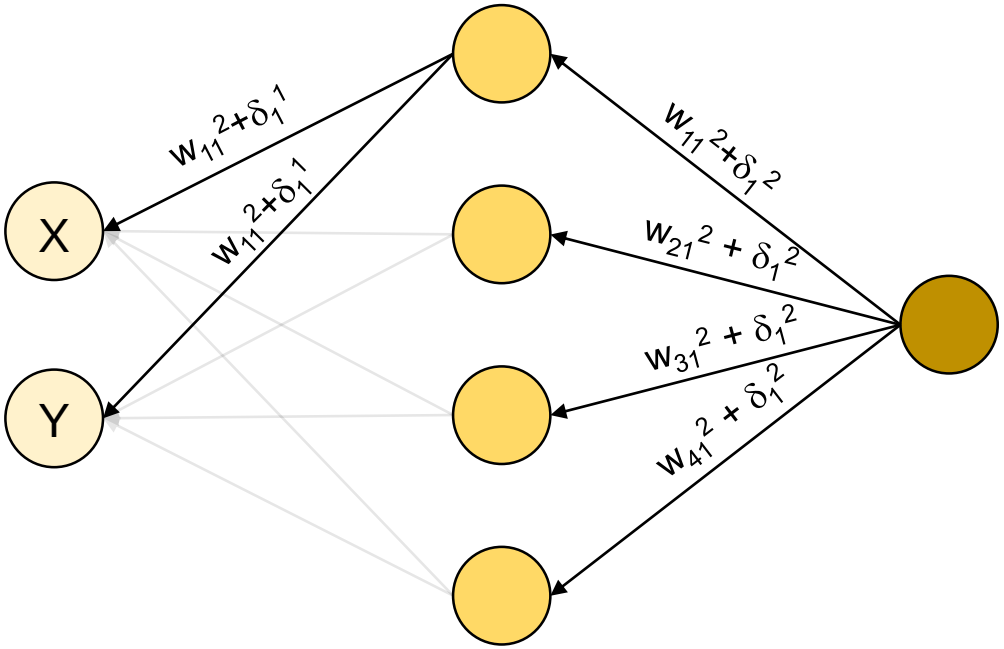
Input

Hidden layer
4 neurons

Output

Backward pass

Back-propagation algorithm : a quick glimpse



$$Minimize \frac{\partial Loss}{\partial W}$$

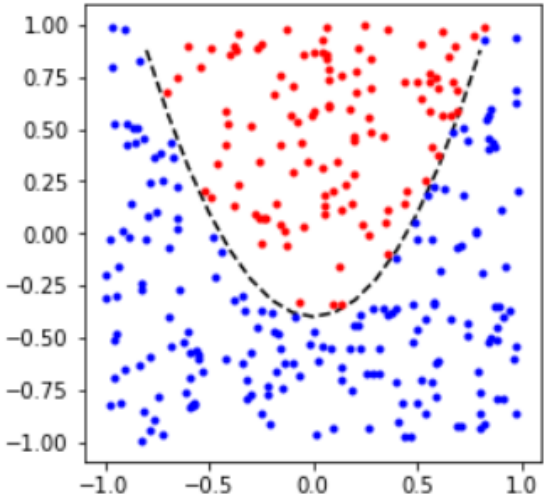
Input

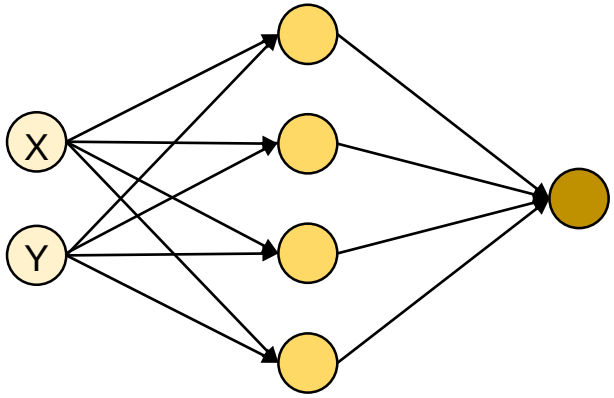
Hidden layer
4 neurons

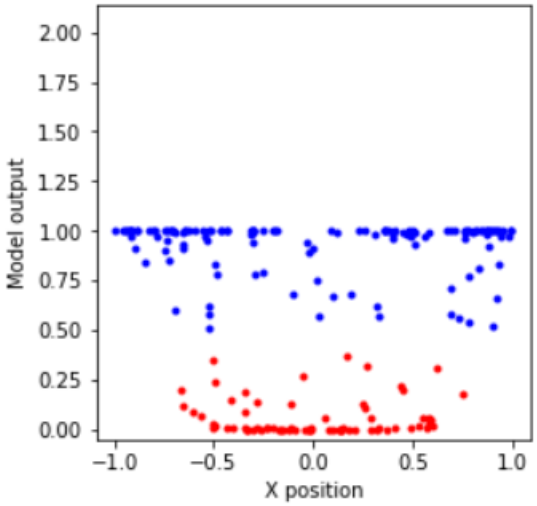
Output

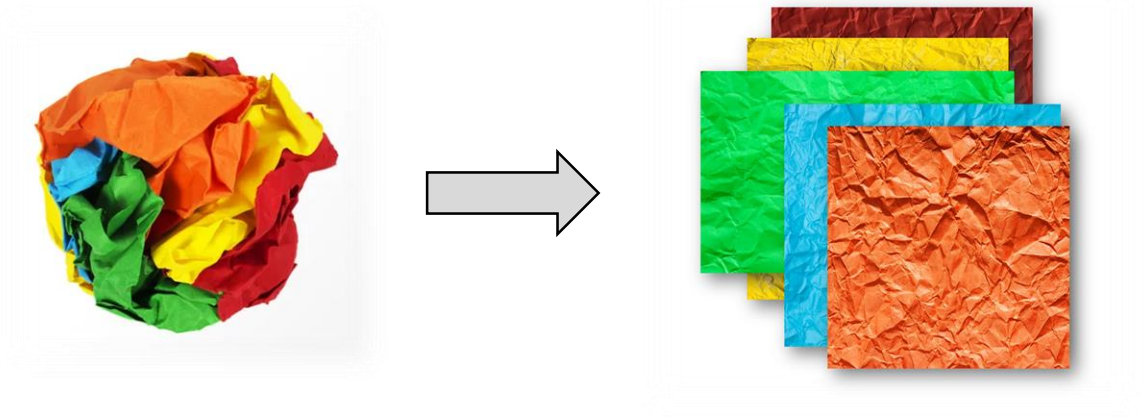
Backward pass

The network is learning a “better” representation

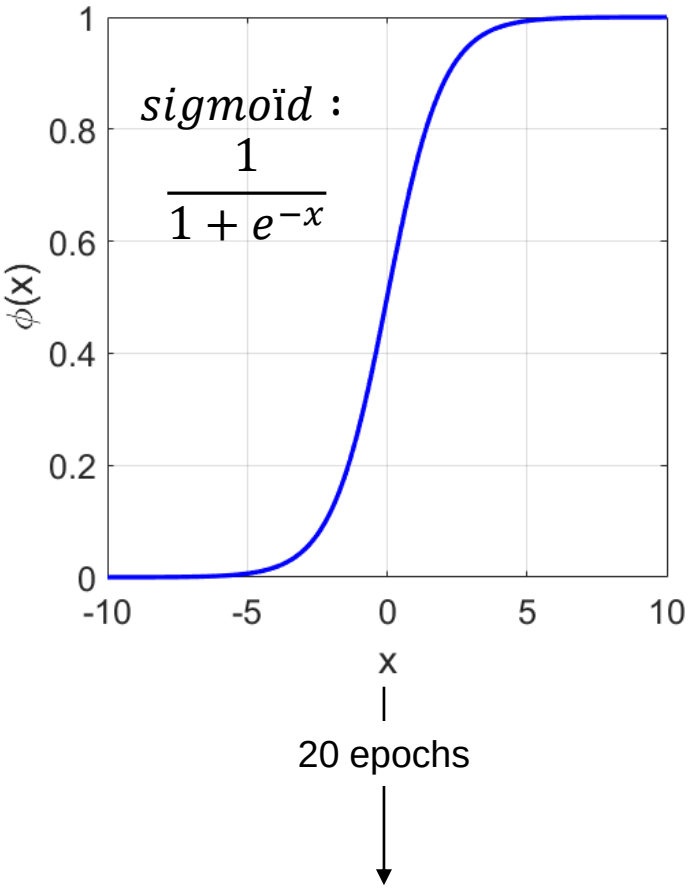








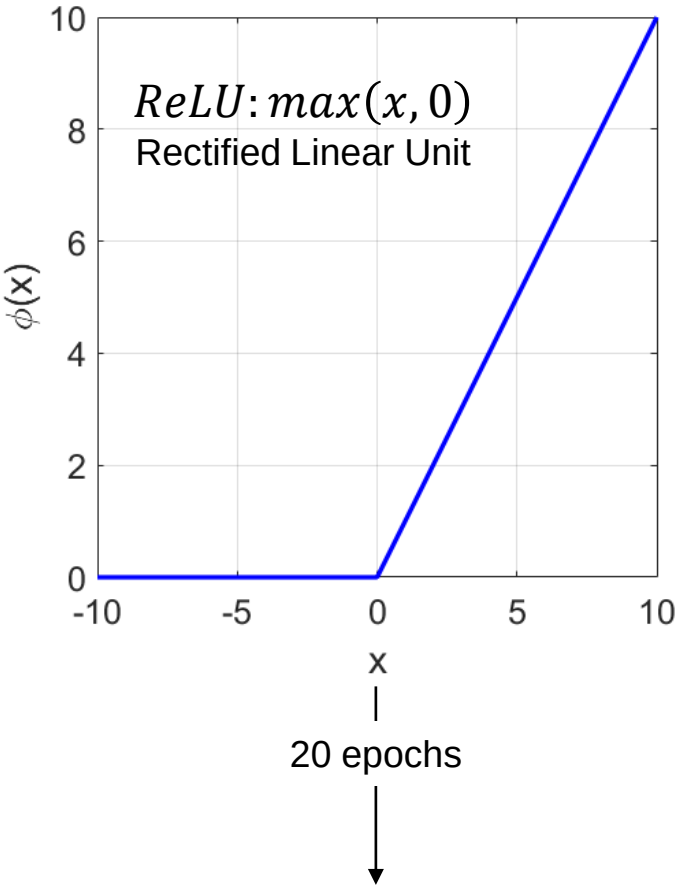
ReLu vs. Sigmoid



20 epochs

~90% accuracy

Gradient is saturating for large output values and exp() is a computer-expensive operation



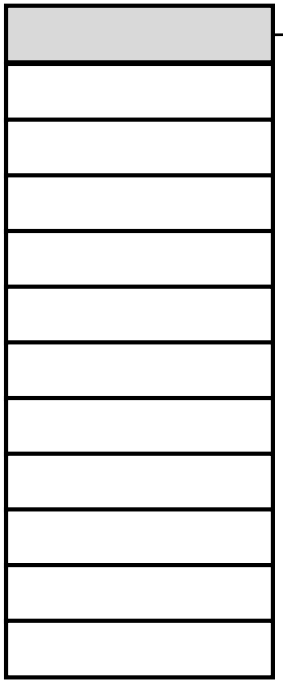
20 epochs

~96% accuracy

Non-saturating gradient and very fast operation. However, negative values are discarded.

Batch processing

```
history = model.fit(Training_data,
                    Training_label,
                    epochs = 20,
                    batch_size = 1,
                    validation_data = (Validation_data, Validation_label))
```

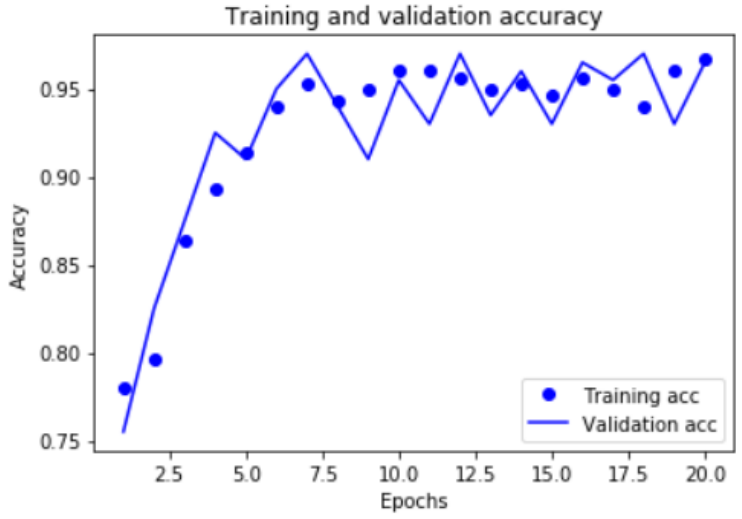


Training set = N

For each training data the weights are adjusted

Δw_i

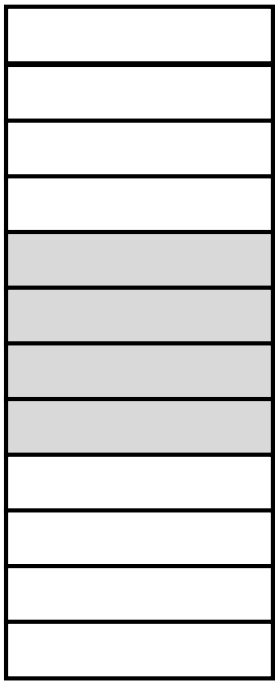
The weights are adjusted N times per epoch



Batch processing

```
history = model.fit(Training_data,
                    Training_label,
                    epochs = 50,
                    batch_size = 4,
                    validation_data = (Validation_data, Validation_label))
```

Better to use a power of 2



For each batch, the weights **are adjusted once** using the average

$$\frac{1}{batch\_size}\sum \Delta w_i$$

The weights are adjusted **N/batch_size**

Training set = N

What did we learn so far?

- How to build a multi-layer network :

```
model = models.Sequential()
model.add(layers.Dense(5,activation='relu', input_shape=(2,)))
model.add(layers.Dense(5,activation='relu'))
model.add(layers.Dense(1,activation='sigmoid'))
```

  - The network learning capability increases with the **number of layers & neurons** → **complex data will require deeper network**
  - But be careful to avoid **overfitting**

- For the training :

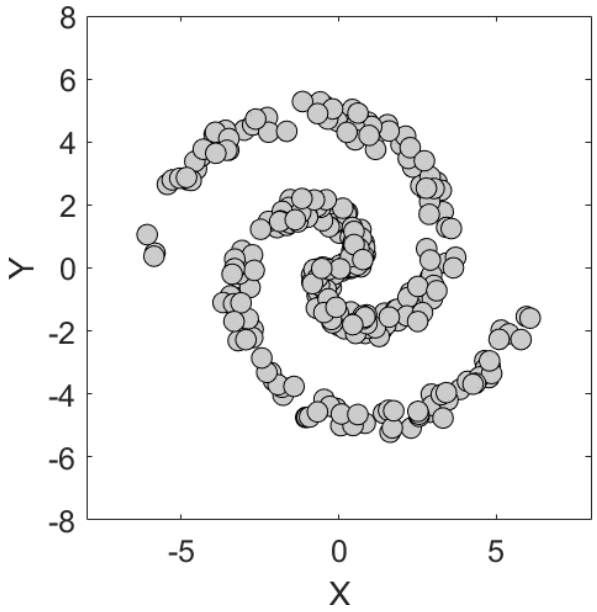
  - The **relu activation function** is more effective and allow the network to learn faster
  - Using **mini-batch** allows for a faster and more stable training

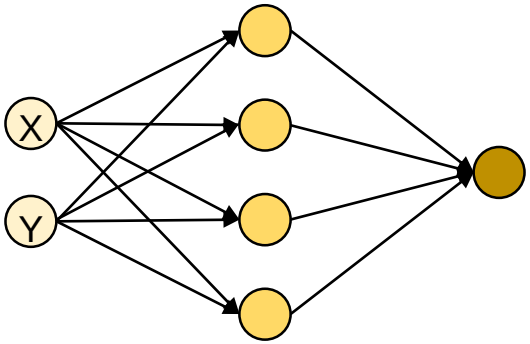
https://playground.tensorflow.org/

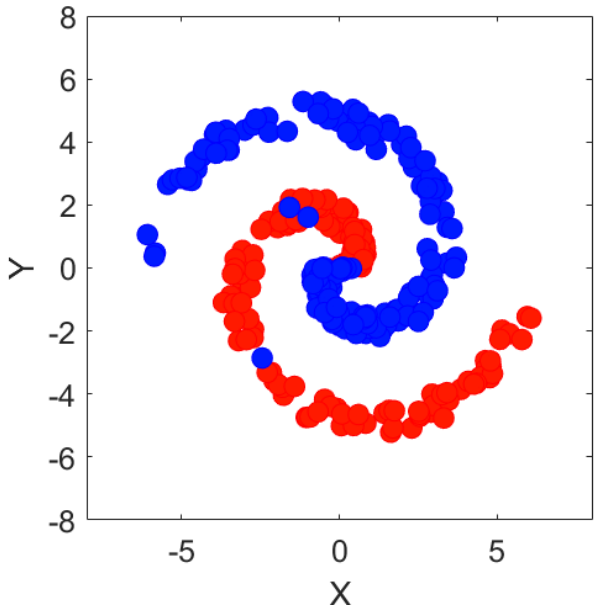
https://cs230.stanford.edu/lecture/

Challenge : build your own ANN to solve a problem

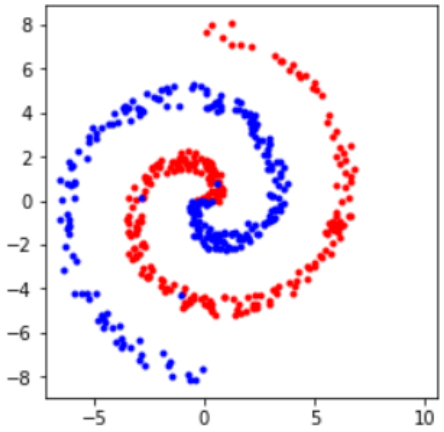
Example n°3 : Clusterization_not_linearly_separated_spirale

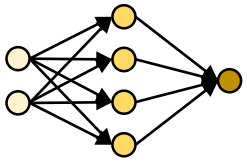




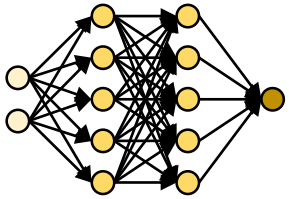


Number of neurons vs. fitting capacity

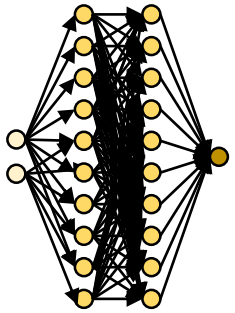




(17)

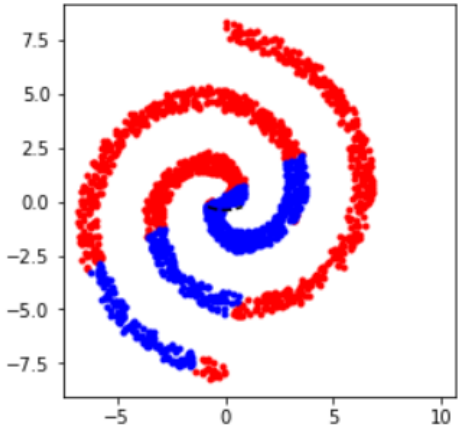


(51)

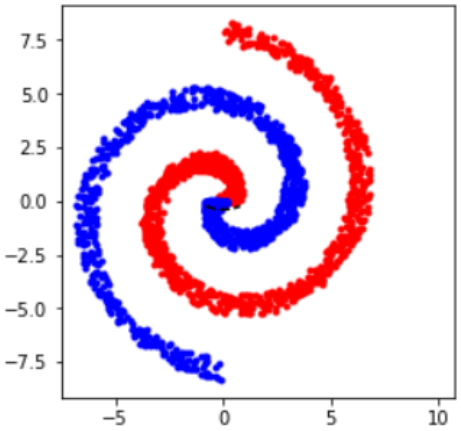


(151)

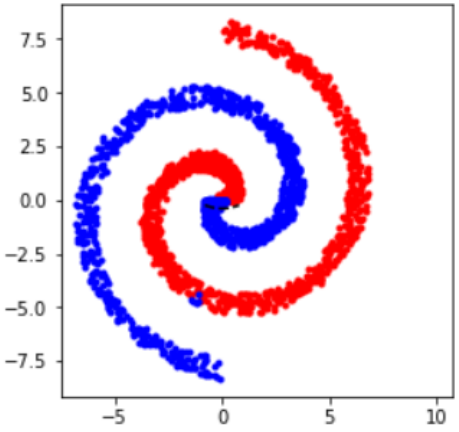
Underfitting



Optimized



Overfitting



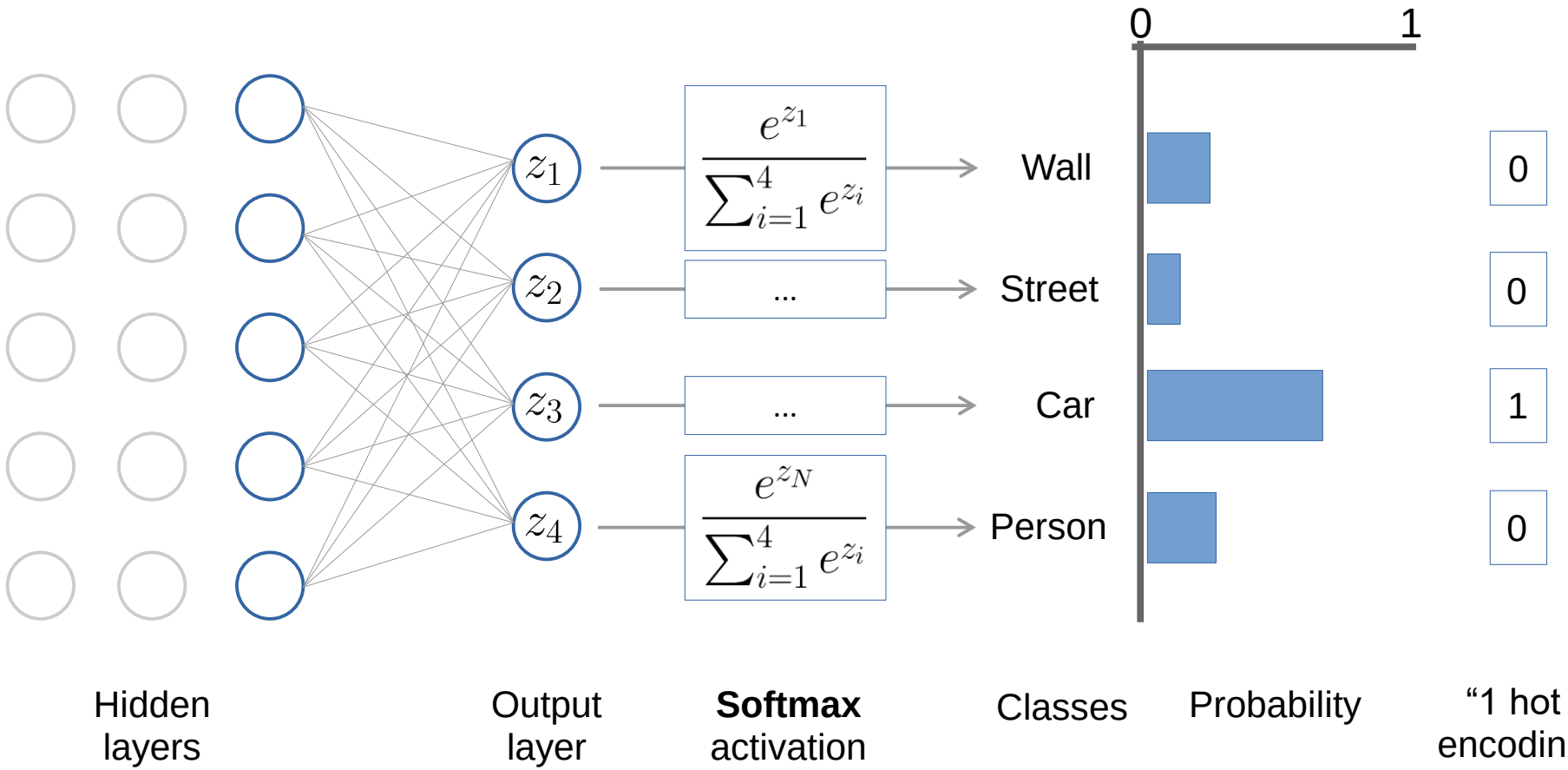
Outline

I. Introduction

II. Starting with Deep Learning and Artificial Neuron Network

i. A bit of vocabulary before starting
ii. Case 1 : single neuron network
iii. Case 2 : how to construct a densely connected network
iv. Case 3 : multi-class clusterization
v. Summary

Classifier with multiple classes : **softmax** activation



Outline

I. Introduction

II. Starting with Deep Learning and Artificial Neuron Network

i. A bit of vocabulary before starting
ii. Case 1 : single neuron network
iii. Case 2 : how to construct a densely connected network
iv. Case 3 : multi-class clusterization
v. Summary

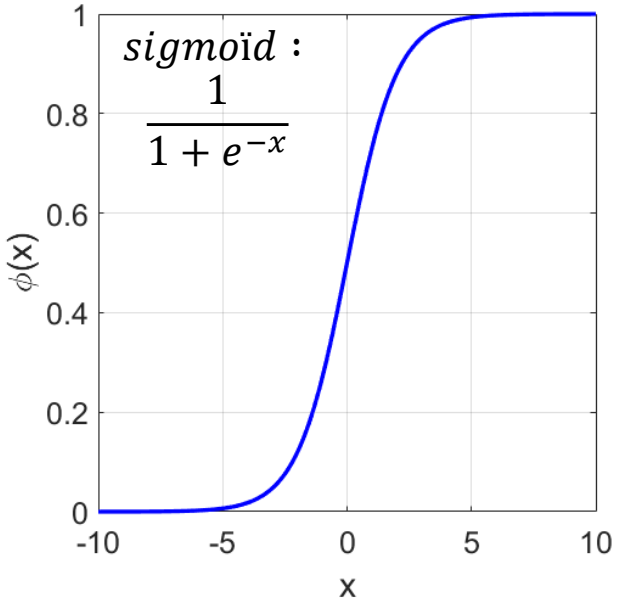
General summary

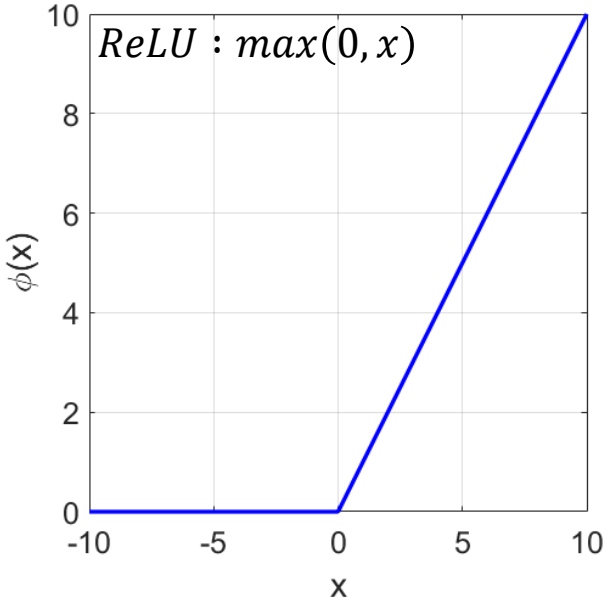
How to choose the activation & loss function?

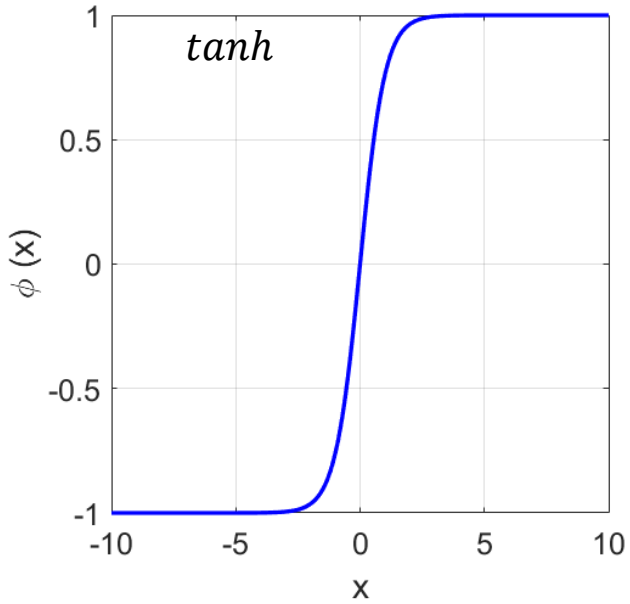
Problem type	Last-layer activation	Loss function	Number of neurons in the last layer
Binary classification	‘sigmoid’	‘binary-crossentropy’	1
Multiclass, single-label classification	‘softmax’	‘categorical_crossentropy’	As many as the number of classes
Regression to values between 0 and 1	‘sigmoid’ or ‘none’	‘mse’	1

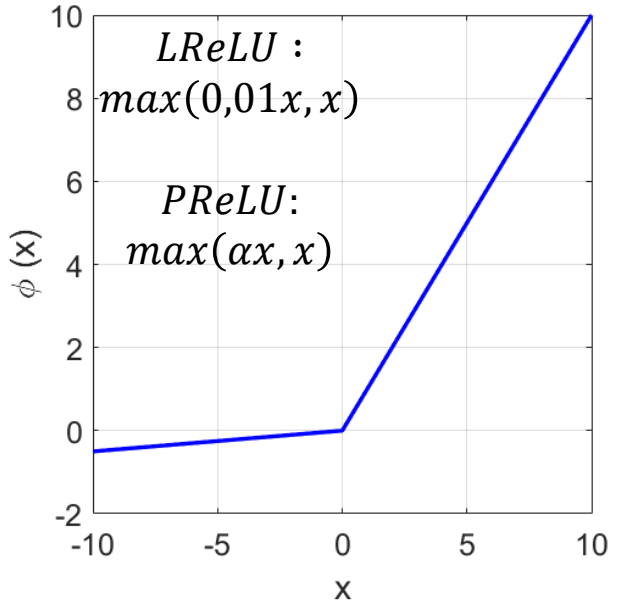
As a rule-of-thumbs, use ‘relu’ everywhere else as activation function

General summary









General summary

Most commonly used optimizers?

Optimize full name	Python call name	Learning-rate
Stochastic Gradient Descent	‘sgd’	Fixed by the user
Root Mean Square Propagation	‘RMSprop’	Adjusted automatically
Adaptive Moment Optimization	‘adam’	Adjusted automatically