{BnF

Atelier IA

Classification supervisée d’images

Plan

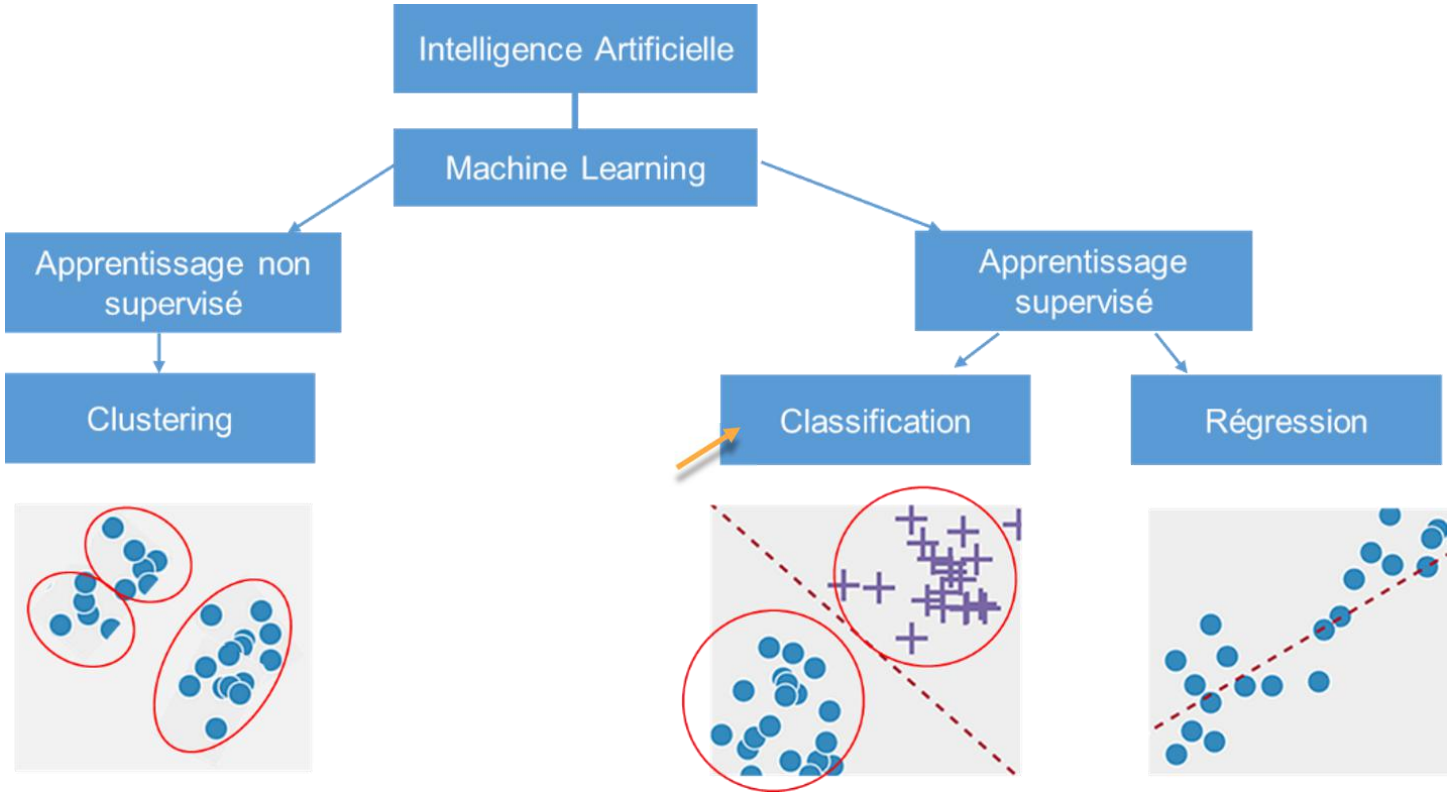
Introduction :

- Classification supervisée
- Le neurone formel
- Perceptron
- Perceptron multicouche
- Réseaux convolutifs (CNN)

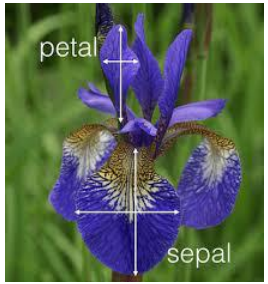
Classification avec Watson Studio/Google AutoML

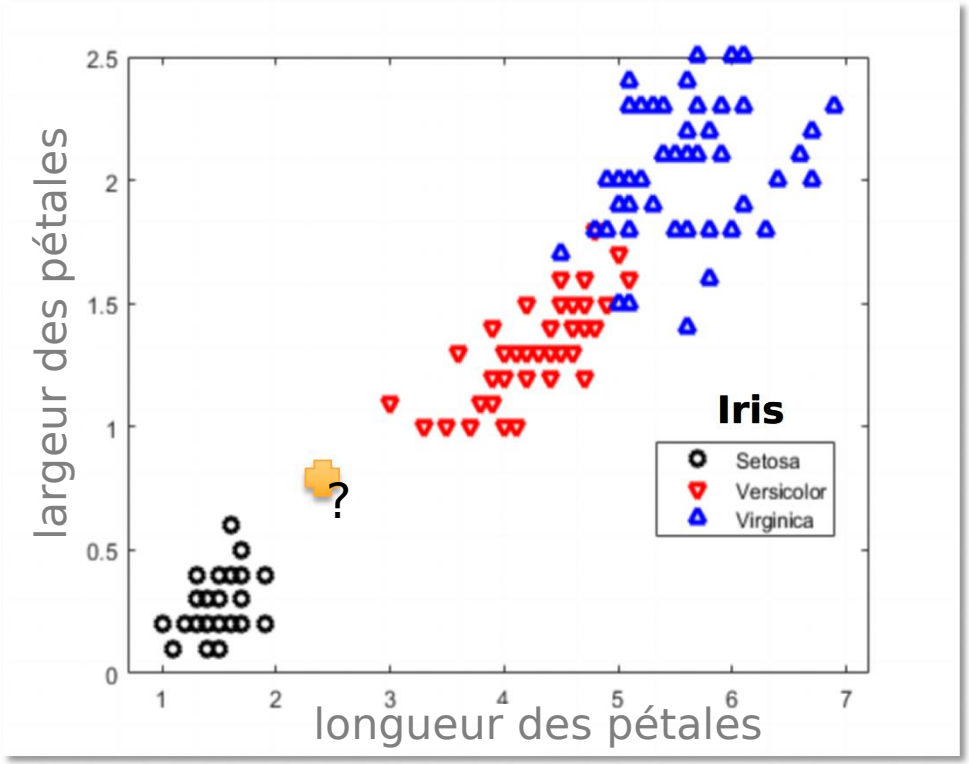
Avec une plateforme IA (TensorFlow)

Classification supervisée



Classification supervisée : exemple

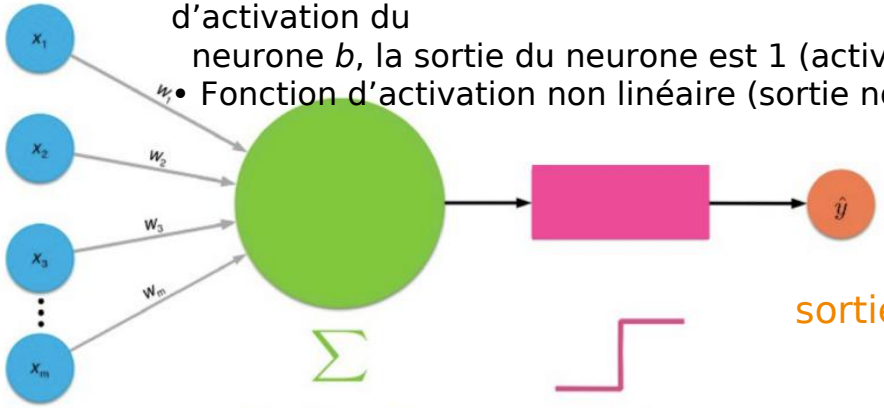




Le neurone formel

The Formal Neuron: 1943 [McCulloch and Pitts, 1943]

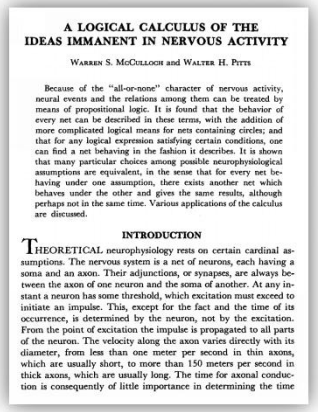
- Règle de calcul qui permet d'associer aux *m* entrées une sortie *y*
- Les entrées sont excitatrices ou inhibitrices (*y* = 0 dès qu'une entrée inhibitrice = 1)
- Si la somme pondérée *s* des entrées excitatrices dépasse le seuil d'activation du neurone *b*, la sortie du neurone est 1 (activée)
- Fonction d'activation non linéaire (sortie non proportionnelle à l'entrée)



sortie

$$s = \mathbf{w}^\top \mathbf{x} + b = \sum_{i=1}^{m} w_i x_i + b$$

fonction « échelle »



A LOGICAL CALCULUS OF THE IDEAS IMMANENT IN NERVOUS ACTIVITY

WARREN S. MCCULLOCH and WALTER H. PITTS

Because of the "all-or-none" character of nervous activity, neural events and the relations among them can be treated by means of propositional logic. It is found that the behavior of every net can be described in these terms, with the addition of more complicated logical means for nets containing circles; and that for any logical expression satisfying certain conditions, one can find a net behaving in the fashion it describes. It is shown that many particular choices among possible neurophysiological assumptions are equivalent, in the sense that for every net behaving under one assumption, there exists another net which behaves under the other and gives the same results, although perhaps not in the same time. Various applications of the calculus are discussed.

INTRODUCTION

THEORETICAL neurophysiology rests on certain cardinal assumptions. The nervous system is a net of neurons, each having a soma and an axon. Their adjunctions, or synapses, are always between the axon of one neuron and the soma of another. At any instant a neuron has some threshold, which excitation must exceed to initiate an impulse. This, except for the fact and the time of its occurrence, is determined by the neuron, not by the excitation. From the point of excitation the impulse is propagated to all parts of the neuron. The velocity along the axon varies directly with its diameter, from less than one meter per second in thin axons, which are usually short, to more than 150 meters per second in thick axons, which are usually long. The time for axonal conduction is consequently of little importance in determining the time

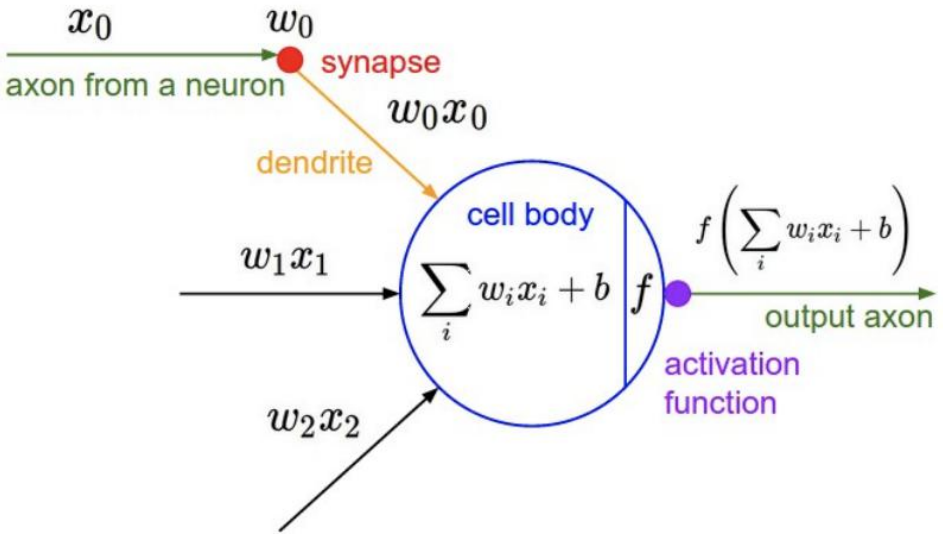
Voir Fun MOOC, « Deep Learning »

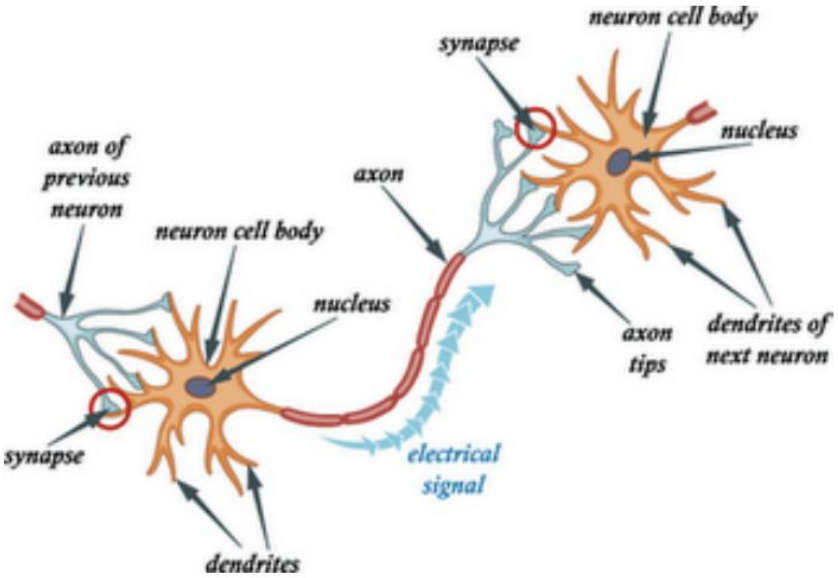
Deep Learning - session 1

CNAM - 01031

Terminé - avril 22, 2018

Neurone biologique ?

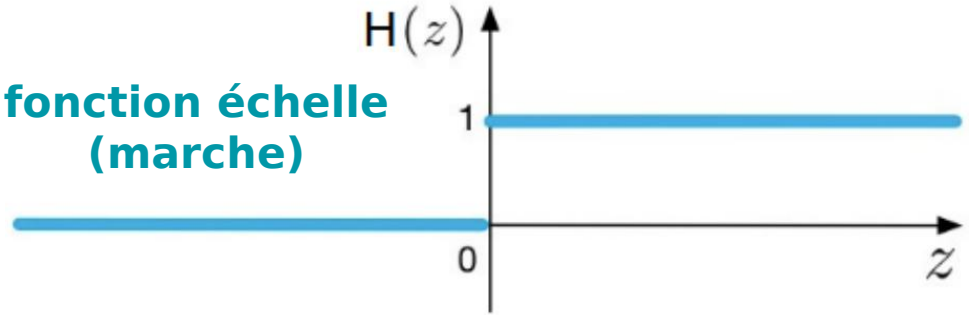




Analogie entre ce modèle computationnel de neurones formels et les neurones biologiques : « si le neurone est suffisamment excité, il atteint un potentiel d'activation suffisant »

Fonctions d’activation

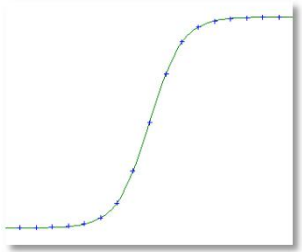
fonction échelle (marche)



La fonction « échelle » affecte une valeur de sortie égale à 1 si la sortie *s* est positive et une valeur 0 sinon

On peut voir le neurone formel comme effectuant une étape de **binarisation**, où le neurone est activé si la valeur de la combinaison des entrées et des poids est supérieure à un
seuil – *b* (seuil d’activation)

Autres fonctions possibles, permettant une prise de décision moins raide et une sortie à valeur réelle : fonction sigmoïde



courbe en « S »

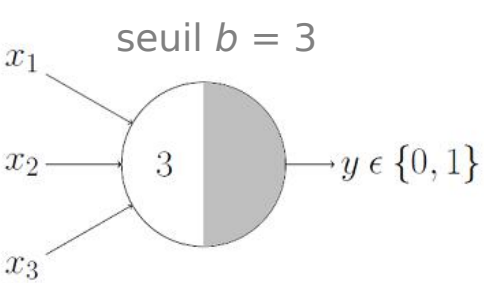
Application à la classification (binaire)

Fonctions booléennes

- Tous les poids sont égaux (1)
- Entrées binaires
- Sortie binaire :

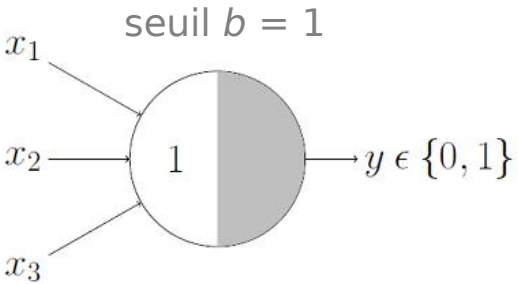
$\hat{y} = f(s)$ = 1 si somme(x_i) ⩾ b

= 0 si somme(x_i) < b



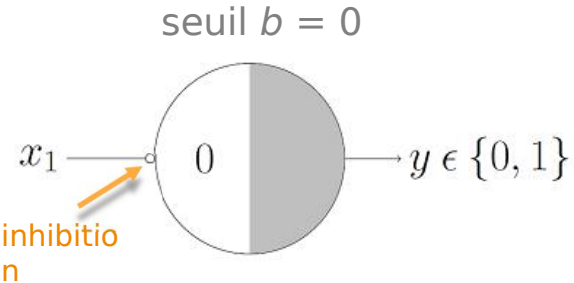
fonction ET

(grandes_pétales ET couleur_foncée)



fonction OU

(grandes_pétales OU grandes_étamines OU …)



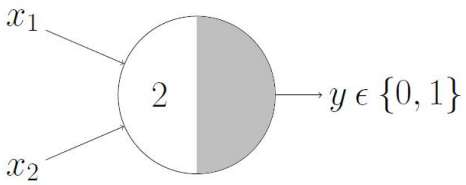
fonction NON (pas foncé)

- Si $x_1 = 0$, le neurone n'est pas inhibée et l'excitation est 0, qui est égale au seuil de 0. La sortie est 1.
- Si $x_1 = 1$, le neurone est inhibé, la sortie est de 0.

https://mcneela.github.io/machine_learning/2017/07/07/McCulloch-Pitts.html

Application à la classification (binaire)

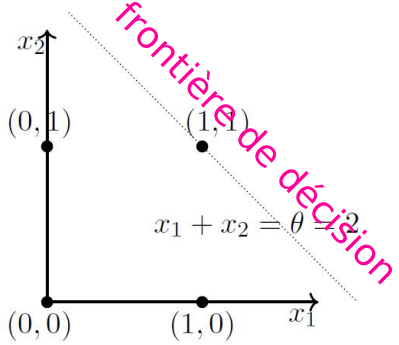
Frontière de décision : ET, deux entrées



AND function

$$x_1 + x_2 = \sum_{i=1}^{2} x_i \geq 2$$

seuil $b = 2$

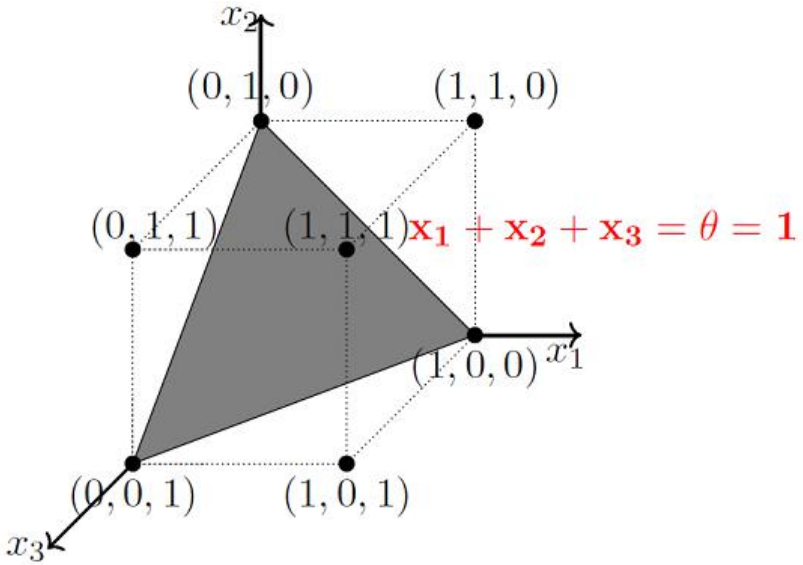


fonction ET
(grandes_pétales ET couleur_foncée)

- frontière de décision (linéaire)
  = **ligne**

Application à la classification (binaire)

Frontière de décision : OU, 3 entrées



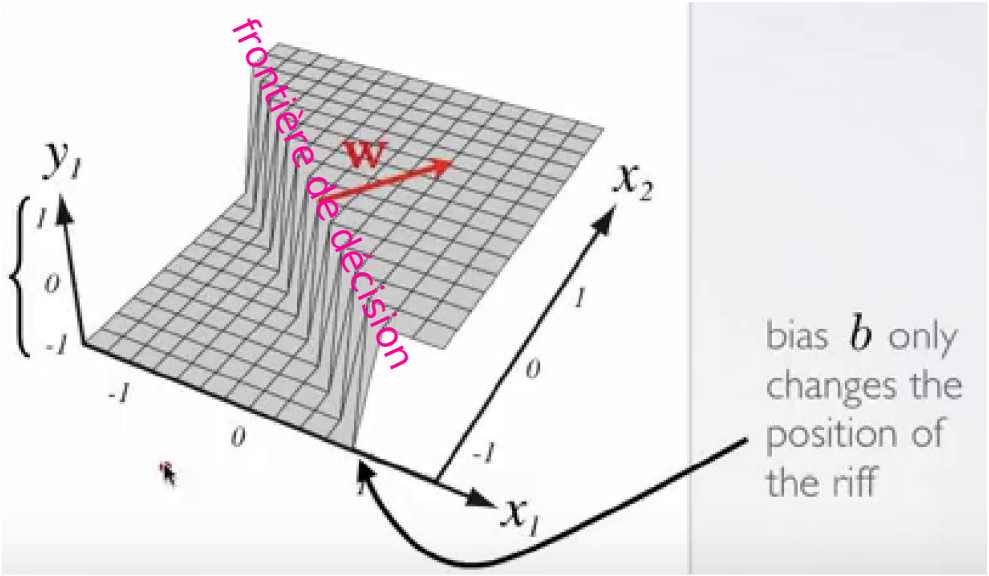
- frontière de décision (linéaire)
  = **plan**

seuil $b = 1$

fonction OU
(grandes_pétales OU grandes_étamines OU ...)

Application à la classification (binaire)

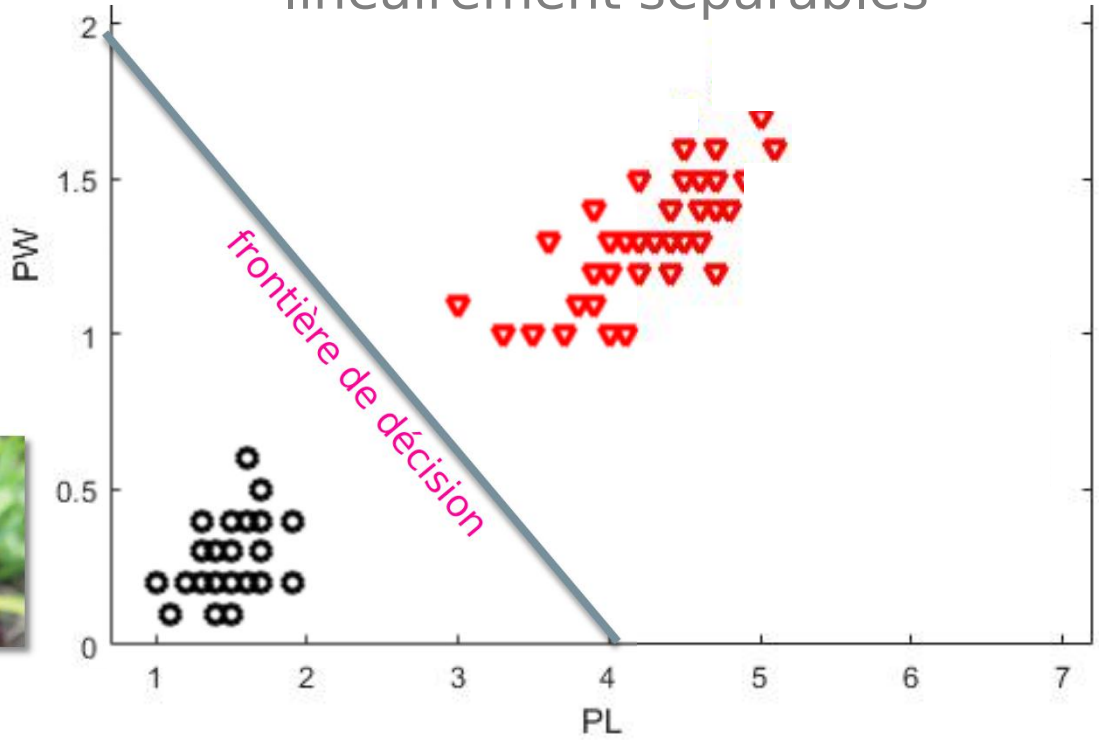
Fonctions scalaires



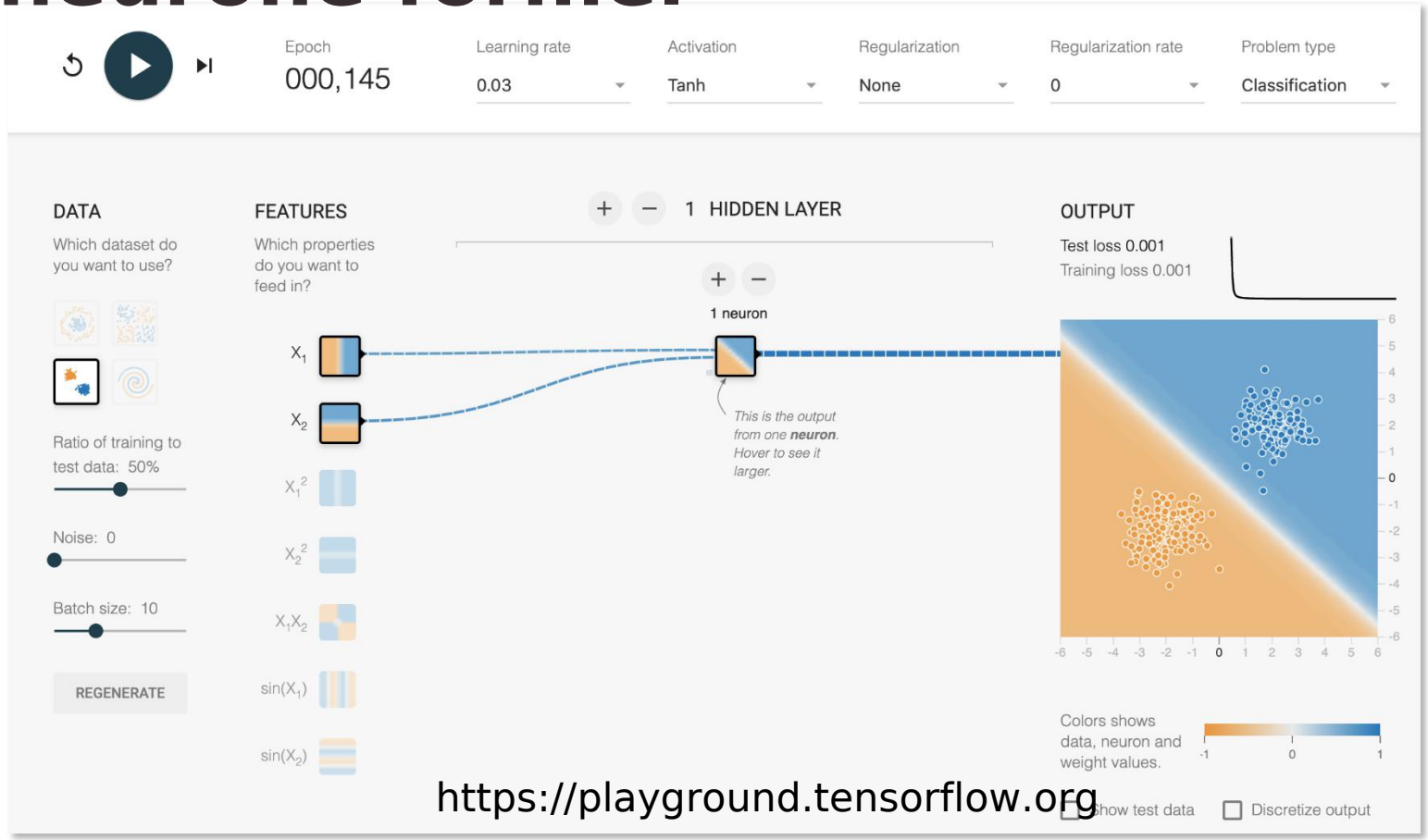
- L'entrée x a deux dimensions
- La sortie $\hat{y}$ est prédite par le neurone formel
- $\hat{y}$ = probabilité d'être dans la classe 1, la classe positive, sachant l'entrée x

Application à la classification (binaire)

trois espèces d'iris à classer, linéairement séparables



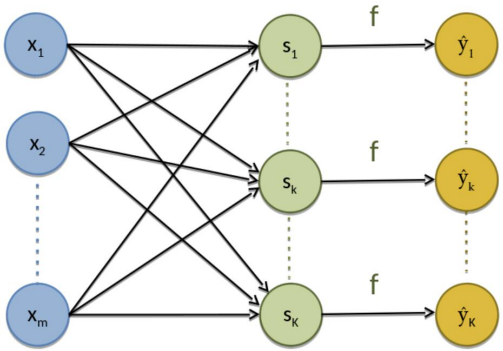
Simulation de l'apprentissage : neurone formel



https://playground.tensorflow.org

Classification multiclasse : Perceptron

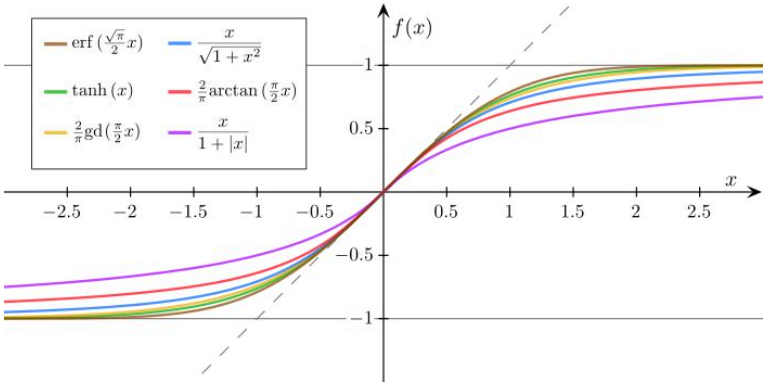
Frank Rosenblatt (1958), Minsky and Papert (1969)




Book Reviews

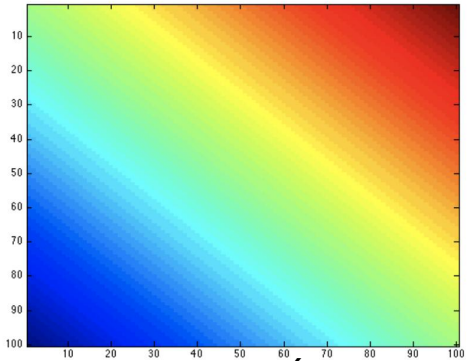
A Step toward the Understanding of Information Processes

- Poids et entrées à valeurs réelles
- Assemblage de neurones formels
- Pour chaque neurone de sortie :
  $\hat{y}_k$ = probabilité d'être dans la classe *k* connaissant l'entrée *x*
- Fonction d'activation *f* softmax

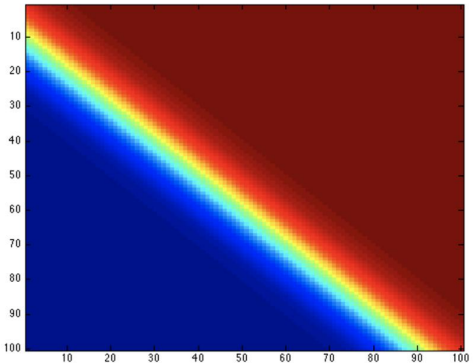


Application à la classification (binaire)

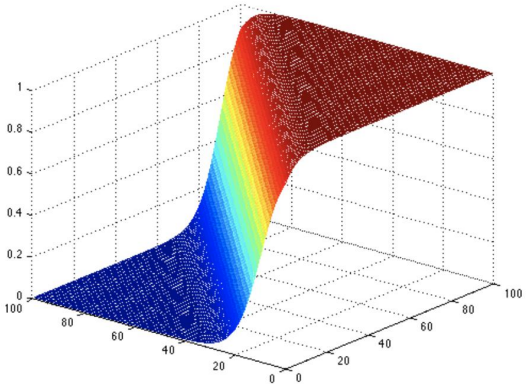
- 2d example: $m = 2$,
  $\mathbf{x} = \{x_1, x_2\} \in [-5; 5] \times [-5; 5]$
- Linear mapping: $\mathbf{w} = [1; 1]$ and $b = -2$
- Result of linear mapping : $s = \mathbf{w}^\top \mathbf{x} + b$
- Sigmoid activation function:
  $\hat{y} = \left(1 + e^{-a(\mathbf{w}^\top \mathbf{x} + b)}\right)^{-1}$, $a = 1$



entrée

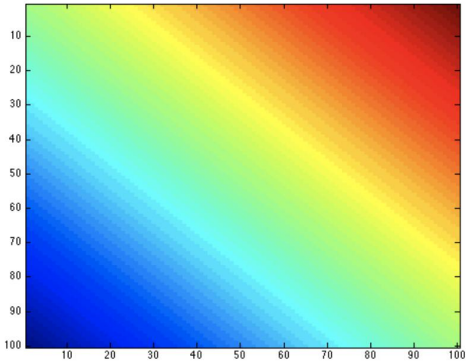


sortie

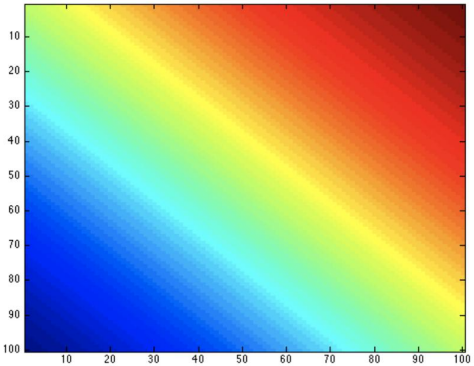


Application à la classification (binaire)

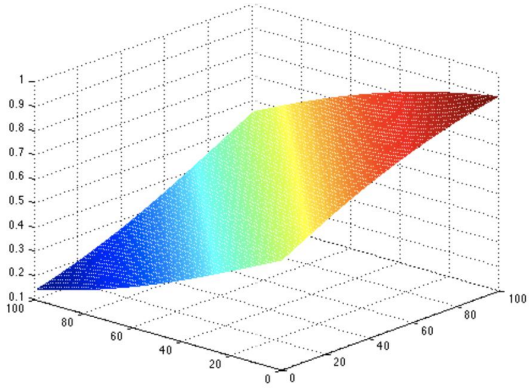
- 2d example: $m = 2$, $\mathbf{x} = \{x_1, x_2\} \in [-5; 5] \times [-5; 5]$
- Linear mapping: $\mathbf{w} = [1; 1]$ and $b = -2$
- Result of linear mapping : $s = \mathbf{w}^\top \mathbf{x} + b$
- Sigmoid activation function: $\hat{y} = \left(1 + e^{-a(\mathbf{w}^\top \mathbf{x} + b)}\right)^{-1}$, $a = 0.1$



entrée



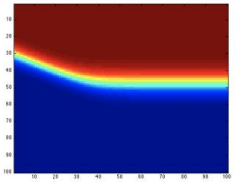
sortie

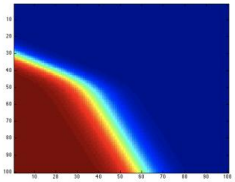


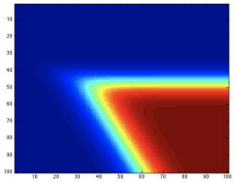
Classification multiclasse : Perceptron

- $\mathbf{x} = \{x_1, x_2\} \in [-5;5] \times [-5;5]$, $\hat{y}$: 3 outputs (classes)

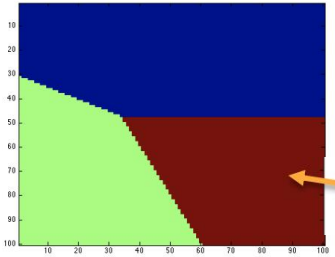
Soft-max output:
$P(k/\mathbf{x}, \mathbf{W})$

$\mathbf{w_1} = [1;1]$, $b_1 = -2$ 

$\mathbf{w_2} = [0;-1]$, $b_2 = 1$ 

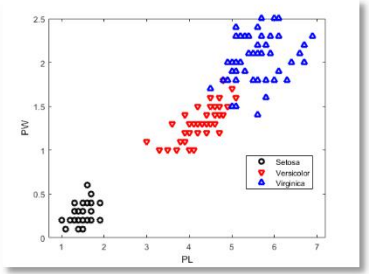
$\mathbf{w_3} = [1;-0.5]$, $b_3 = 10$ 

Class Prediction:
$k^* = \arg\max_k P(k/\mathbf{x}, \mathbf{W})$

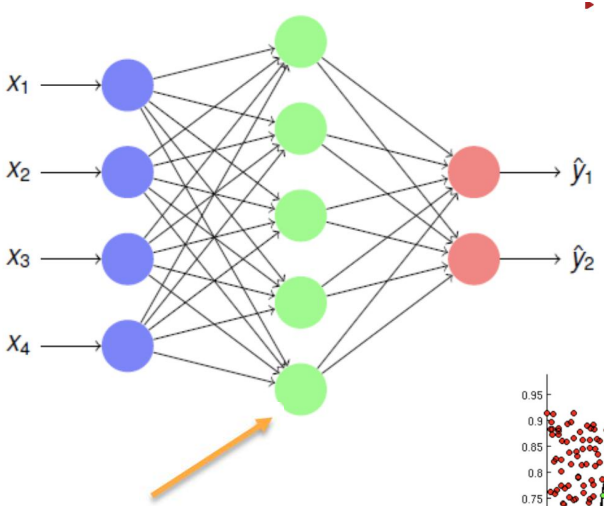


frontières de décision linéaires

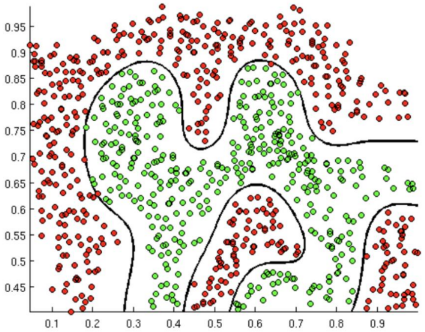
trois espèces d'iris

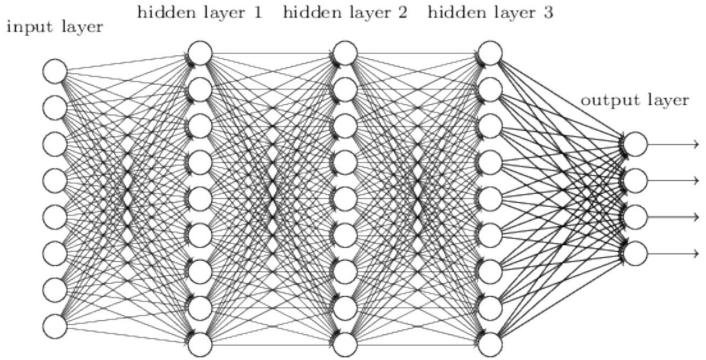


Classification multiclasse : Perceptron multicouche



- Nouvelles représentations intermédiaires produites à partir de la couche précédente vers la couche suivante
- Frontières de décision non linéaires
- Plus de couches : réseaux profonds (*deep neural network*)

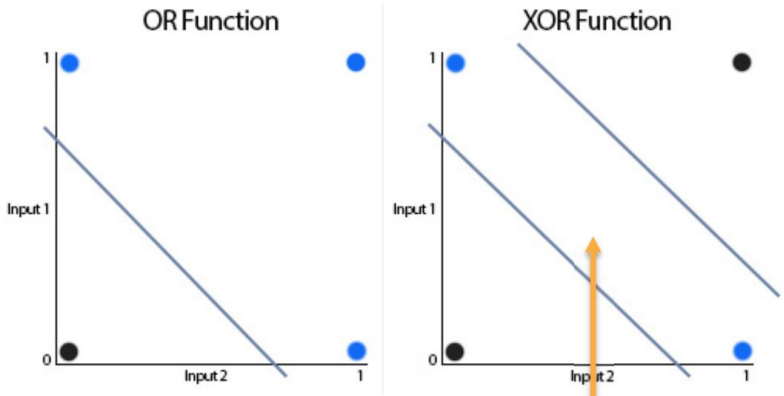


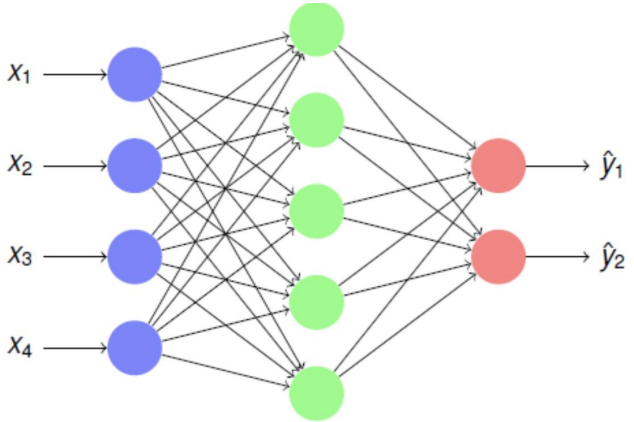


Classification multiclasse : fonction XOR

OU exclusif : sortie=1 que si exclusivement l'une ou l'autre des entrées vaut 1

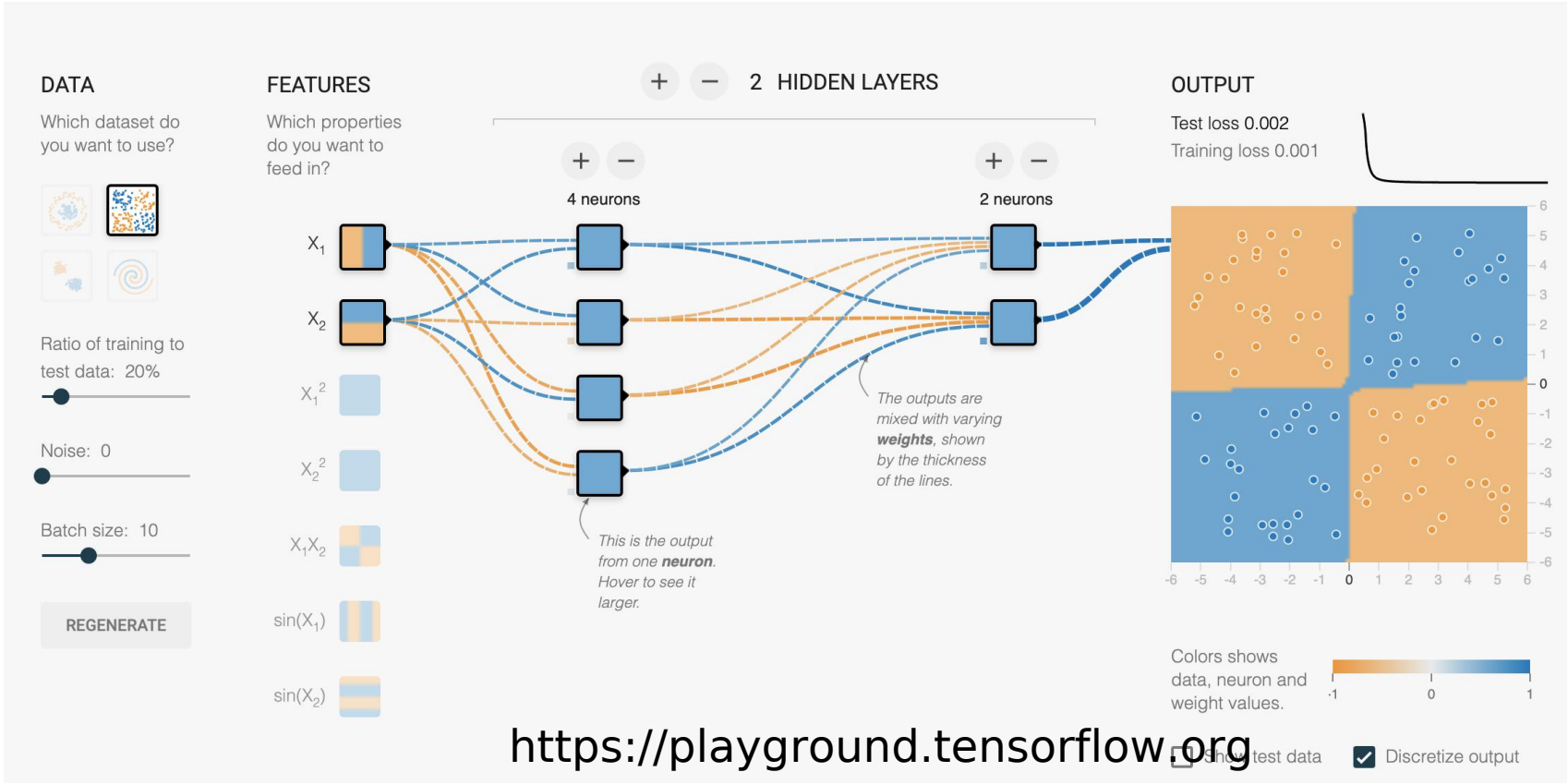
Minksy and Papert (1969) : ils pensaient que ce problème n'était pas modélisable par un réseau de neurones





- 4 neurones
- une couche cachée
- 2 classes en sortie

Simulation : XOR



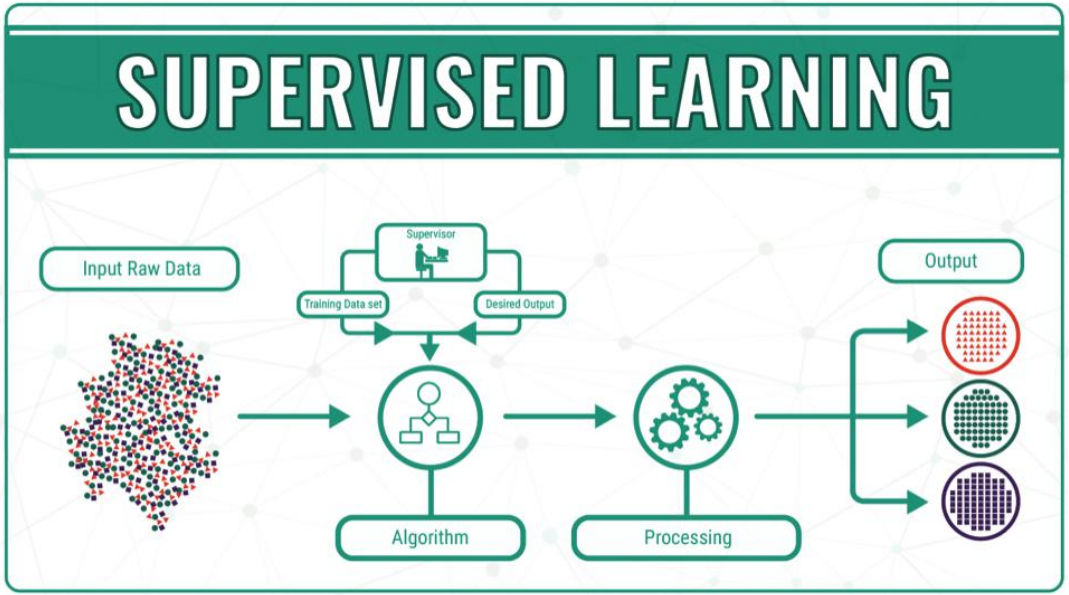
https://playground.tensorflow.org

Apprentissage supervisé

Problème : déterminer la force optimale des connexions entre les neurones du réseau suivant le problème à résoudre

But : apprendre les paramètres du réseau à partir des données

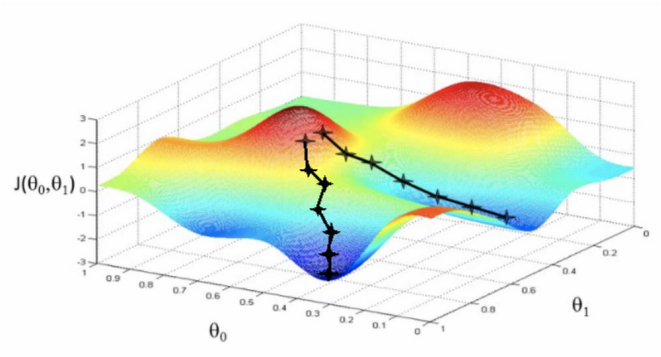
Méthode : algorithme de *rétropropagation du gradient* (1975,1985)



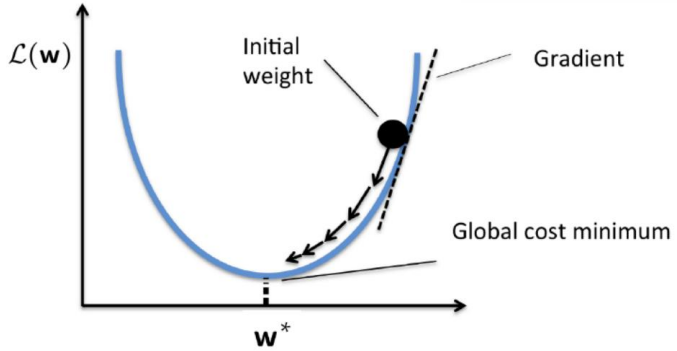
Algorithme de rétropropagation du gradient

Méthode

(a) on définit une fonction de coût, ($l\,\ell(\hat{\mathbf{y}}_i, \mathbf{y}_i^*)$
qui modélise la pénalisation entre la sortie prédite par le réseau et la sortie donnée par la supervision
(b) on minimise la fonction de coût sur l'ensemble d'apprentissag($\mathbf{x}_i, \mathbf{y}_i^*$) , soit la moyenne des fonctions de coût sur les différents exemples d'apprentissage (*gradient* de la fonction de coût)

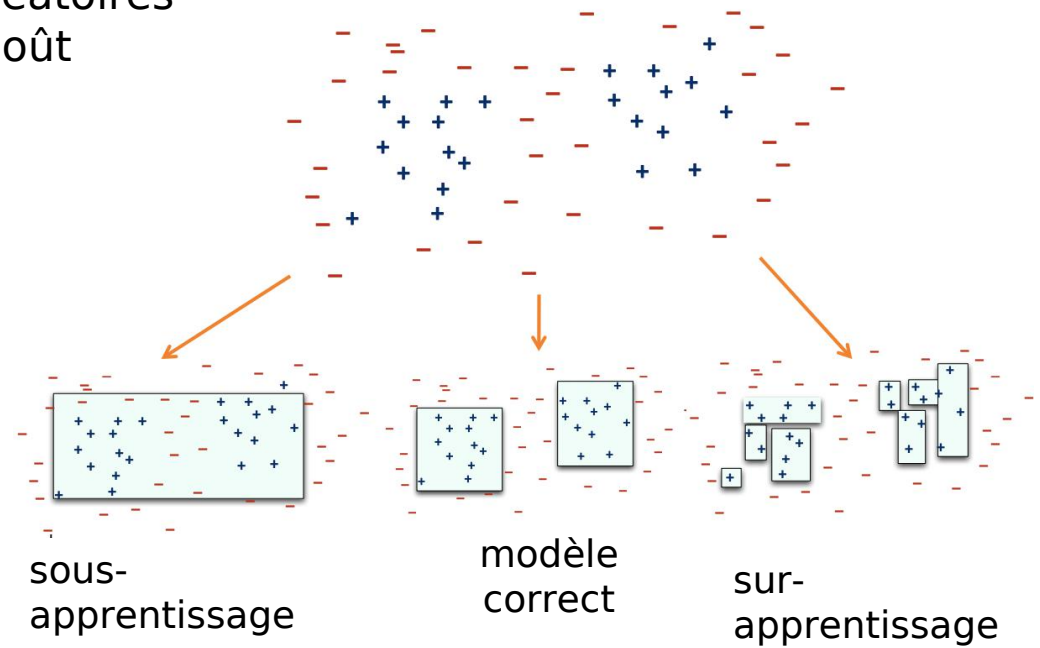


$$\mathcal{L}(\mathbf{w}) = \frac{1}{N}\sum_{i=1}^{N} \ell(\hat{\mathbf{y}}_i, \mathbf{y}_i^*)$$



gradient

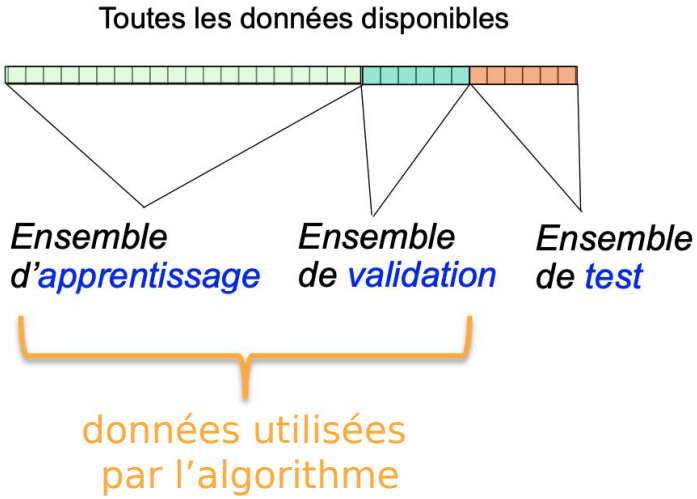
- Algorithme itératif
- Initialisation des poids aléatoires
- Choix de la fonction de coût
- Arrêt de la convergence
- Optimisation
- Sur-apprentissage

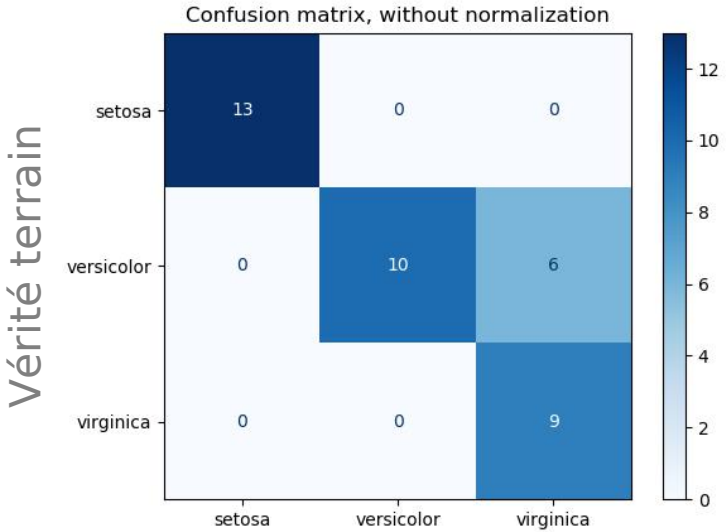


http://www2.agroparistech.fr/ufr-info/membres/cornuejols/Teaching/Centrale-Executive/cours-app-superv
1x4.pdf

Evaluation des performances

- Ensemble de données labelisées (vérité terrain)
- Evaluation empirique des performances (sur l'ensemble de test)
- Matrice de confusion

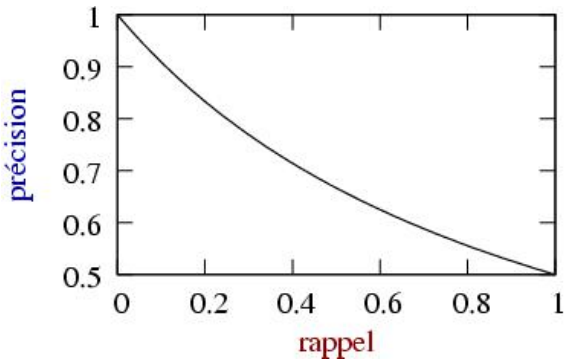


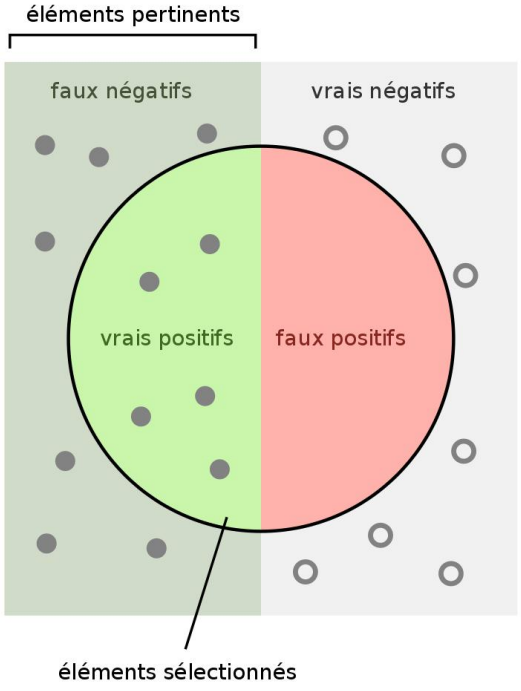


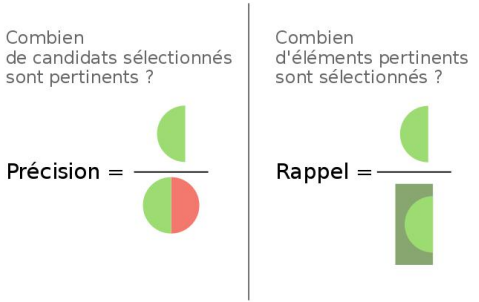
Matrice de confusion pour trois classe d'iris

Evaluation des perforr

- FP, FN, TP, TN
- Métriques : taux de rappel et taux de précision
- Score F (moyenne harmonique de la précision et du rappel)
- Compromis rappel/précision

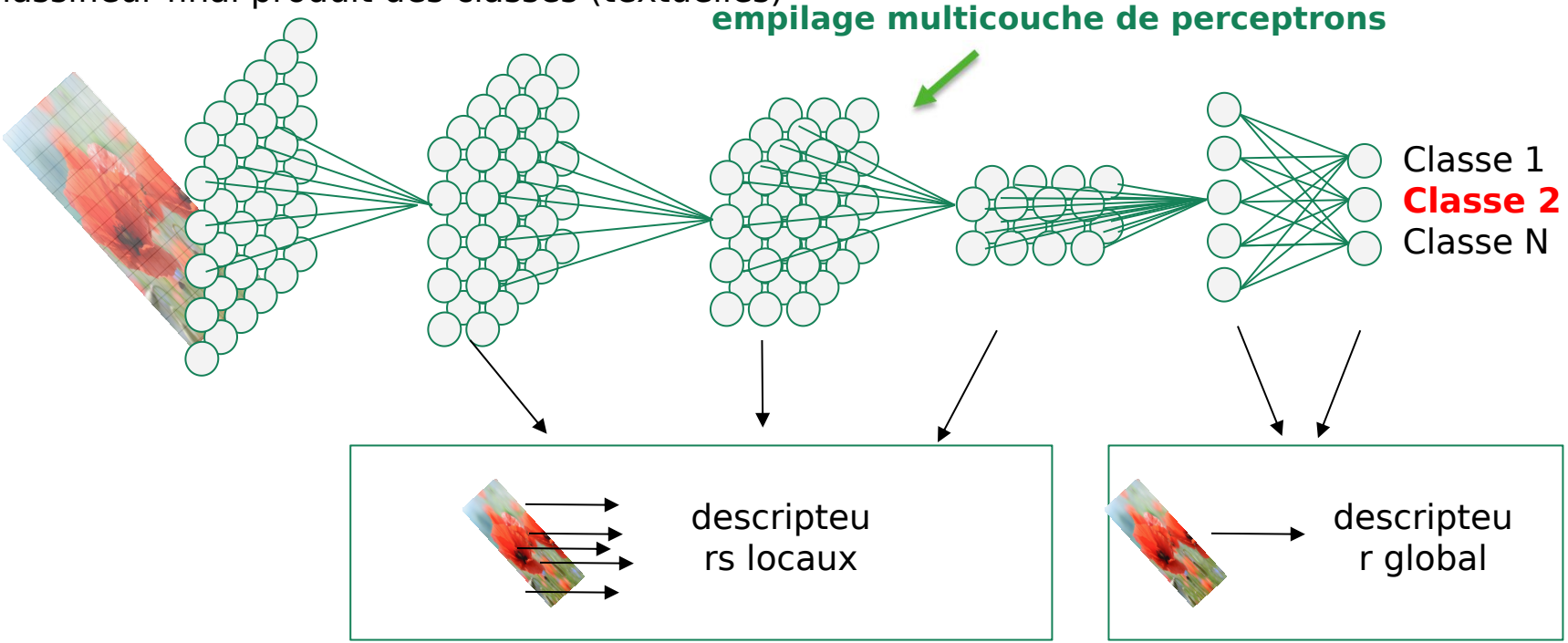




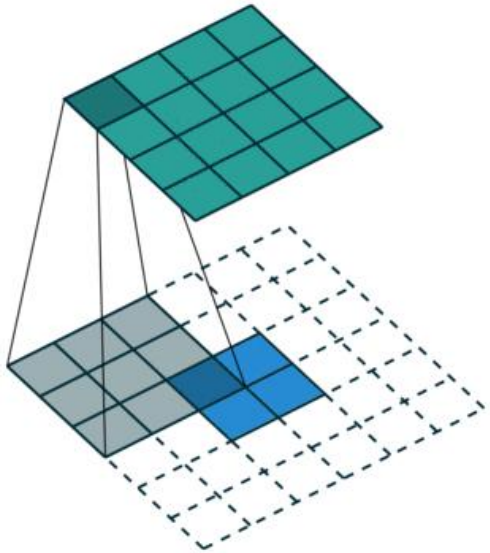


CNN (*convolutional neural network*)

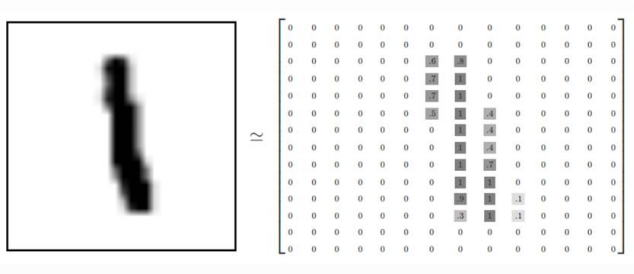
Les réseaux neuronaux convolutifs « simulent » le cortex visuel animal (chevauchement des neurones lors du pavage du champ visuel). Le réseau produit des descripteurs de plus en plus « abstraits ».
Le classifieur final produit des classes (textuelles)



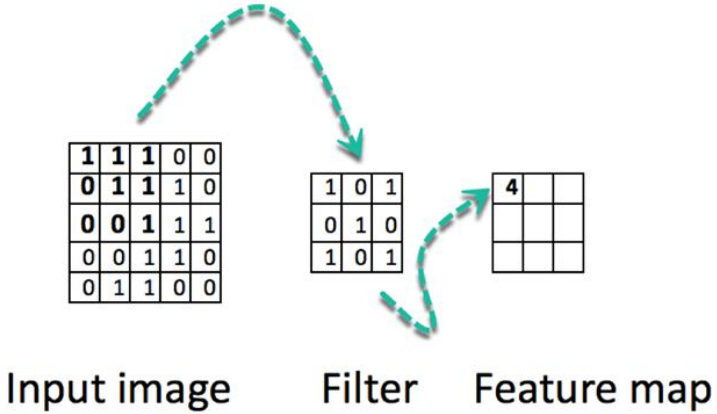
Convolution et filtres



Convolution, filtre 3x3

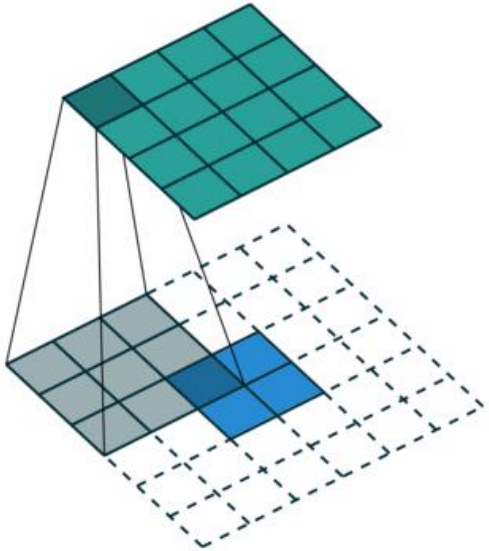


une image = matrice de pixels



Application d’un filtre

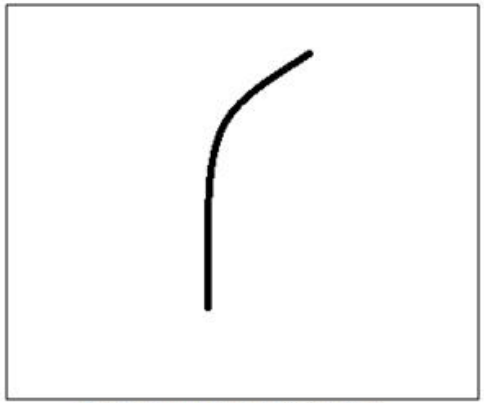
Exemple d'application d'un filtre



Convolution, filtre 3x3

0	0	0	0	0	30	0
0	0	0	0	30	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	0	0	0	0

Pixel representation of filter



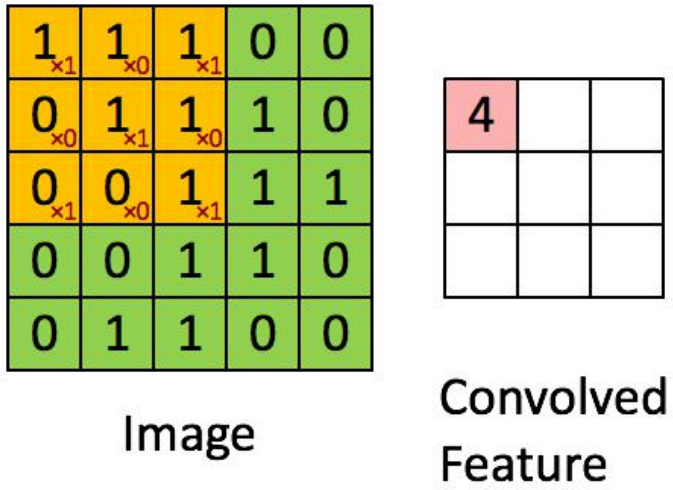
Visualization of a curve detector filter

Filtre de détection de courbes penchées vers la droite (tige de fleur)

Exemple de filtres

Operation	Kernel	Image result
Identity	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
Edge detection	$\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$	
	$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	
	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	
Sharpen	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	

Convolution et filtres



Application de la convolution sur l’image et génération d’une carte de caractéristique
- réduction de la taille de l’image
- extraction des caractéristiques visuelles
- suppression du bruit

Exemple d'application d'un filtre

Visualization of the receptive field

0	0	0	0	0	0	30
0	0	0	0	50	50	50
0	0	0	20	50	0	0
0	0	0	50	50	0	0
0	0	0	50	50	0	0
0	0	0	50	50	0	0
0	0	0	50	50	0	0

Pixel representation of the receptive field

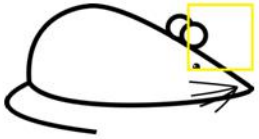
*

0	0	0	0	0	30	0
0	0	0	0	30	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	0	0	0	0

Pixel representation of filter

Multiplication and Summation = (50*30)+(50*30)+(50*30)+(20*30)+(50*30) = 6600 (A large number!)

Application à une portion d'image : détection de la courbe



Visualization of the filter on the image

0	0	0	0	0	0	0
0	40	0	0	0	0	0
40	0	40	0	0	0	0
40	20	0	0	0	0	0
0	50	0	0	0	0	0
0	0	50	0	0	0	0
25	25	0	50	0	0	0

Pixel representation of receptive field

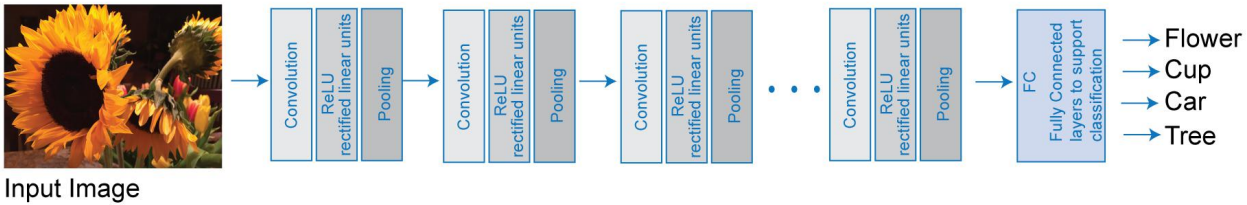
*

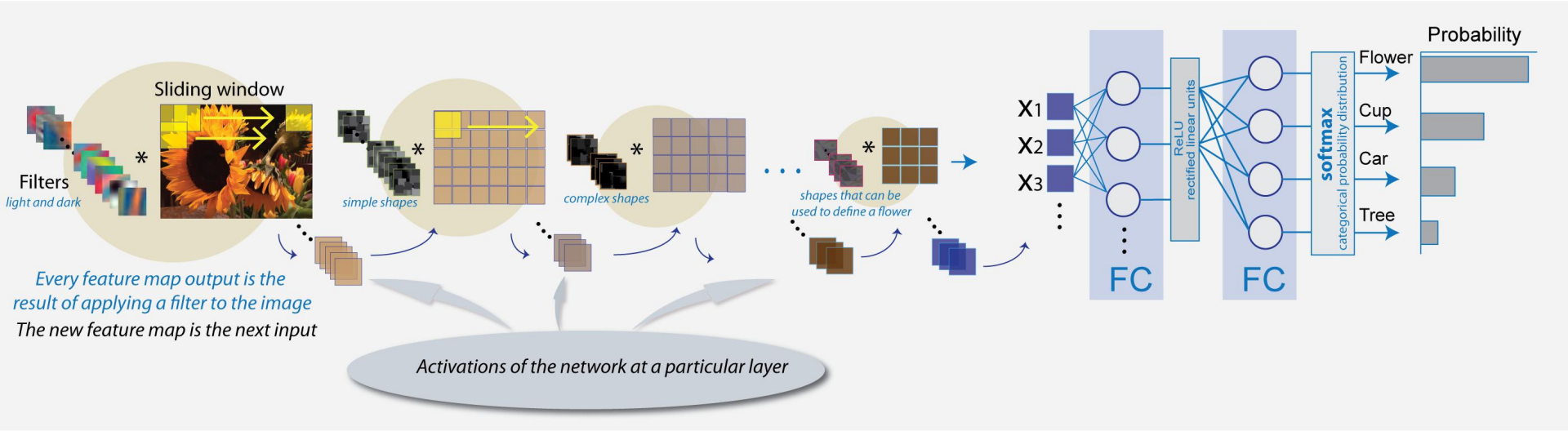
0	0	0	0	0	30	0
0	0	0	0	30	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	0	0	0	0

Pixel representation of filter

Multiplication and Summation = 0

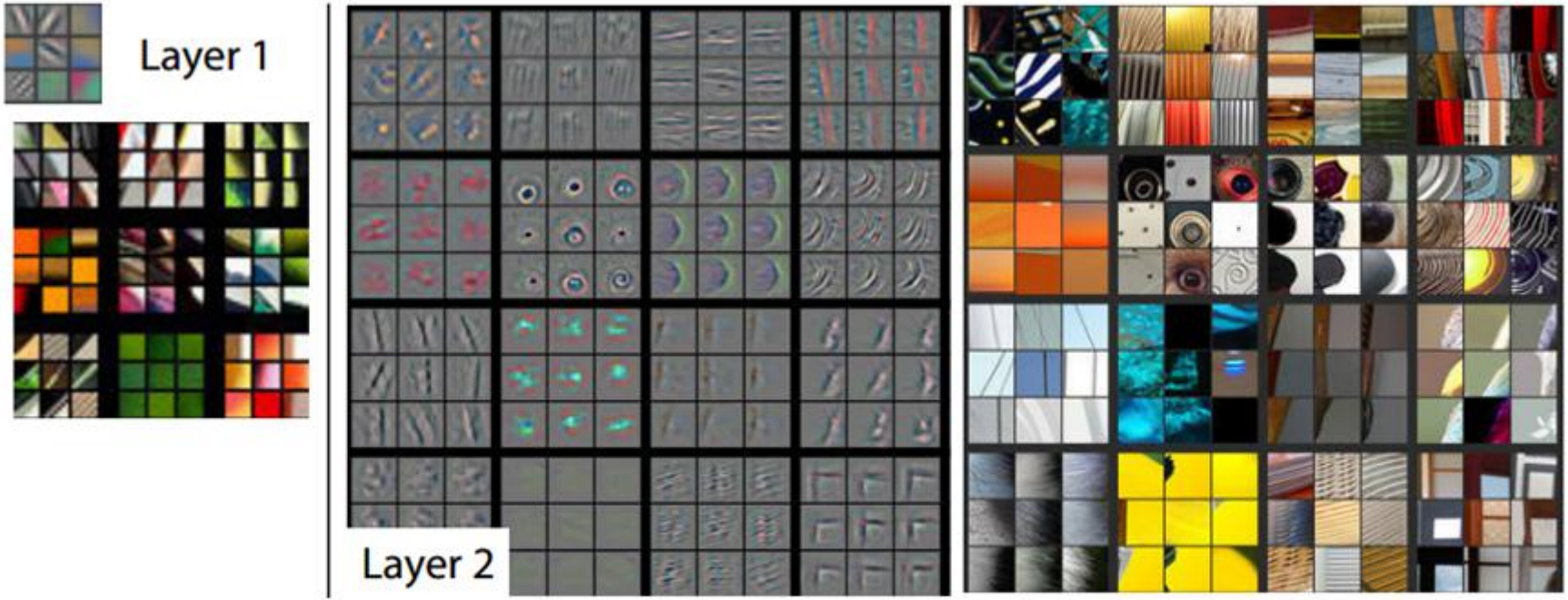
CNN : architecture

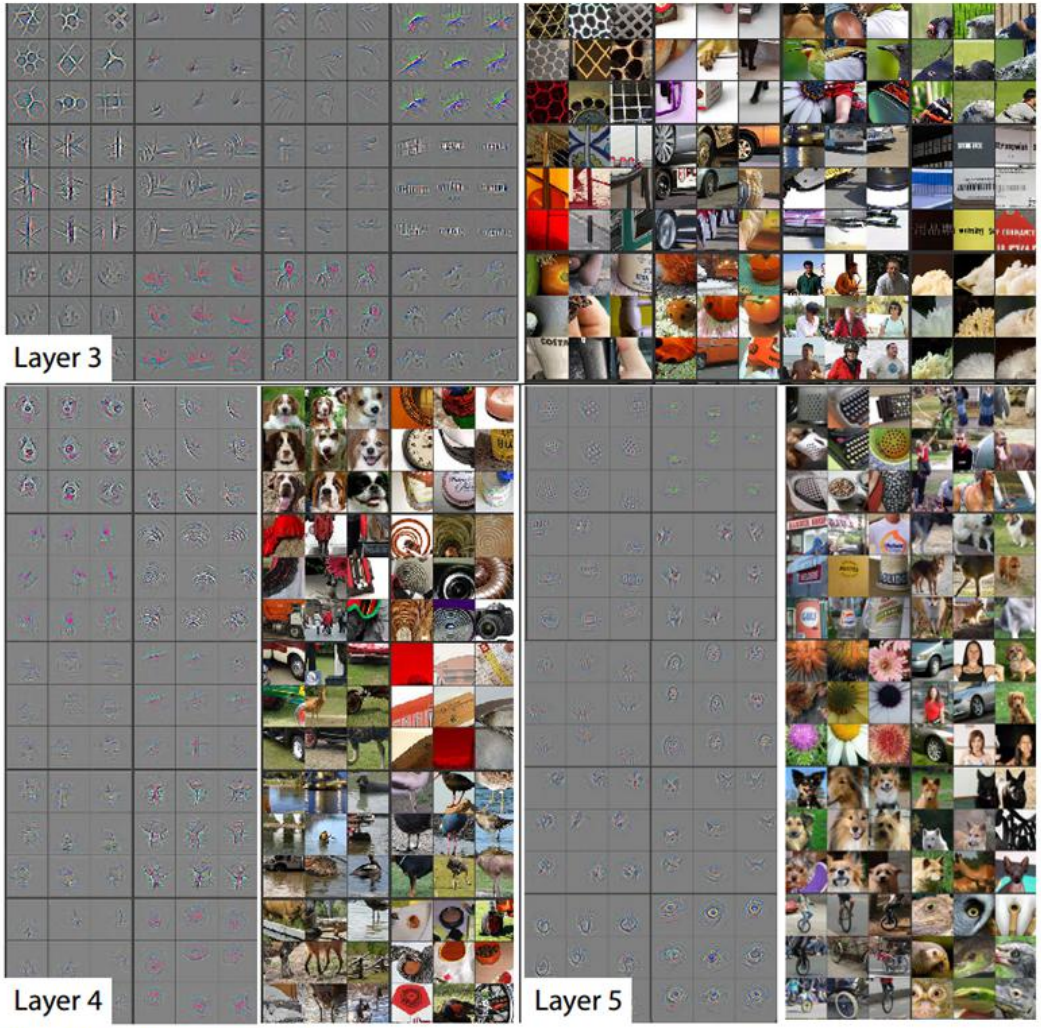




CNN : extraction de caractéristiques

- Première couche : extraction de caractéristiques élémentaires (contours, aplats)
- Les couches supérieures agrègent ces caractéristiques pour détecter des formes plus complexes

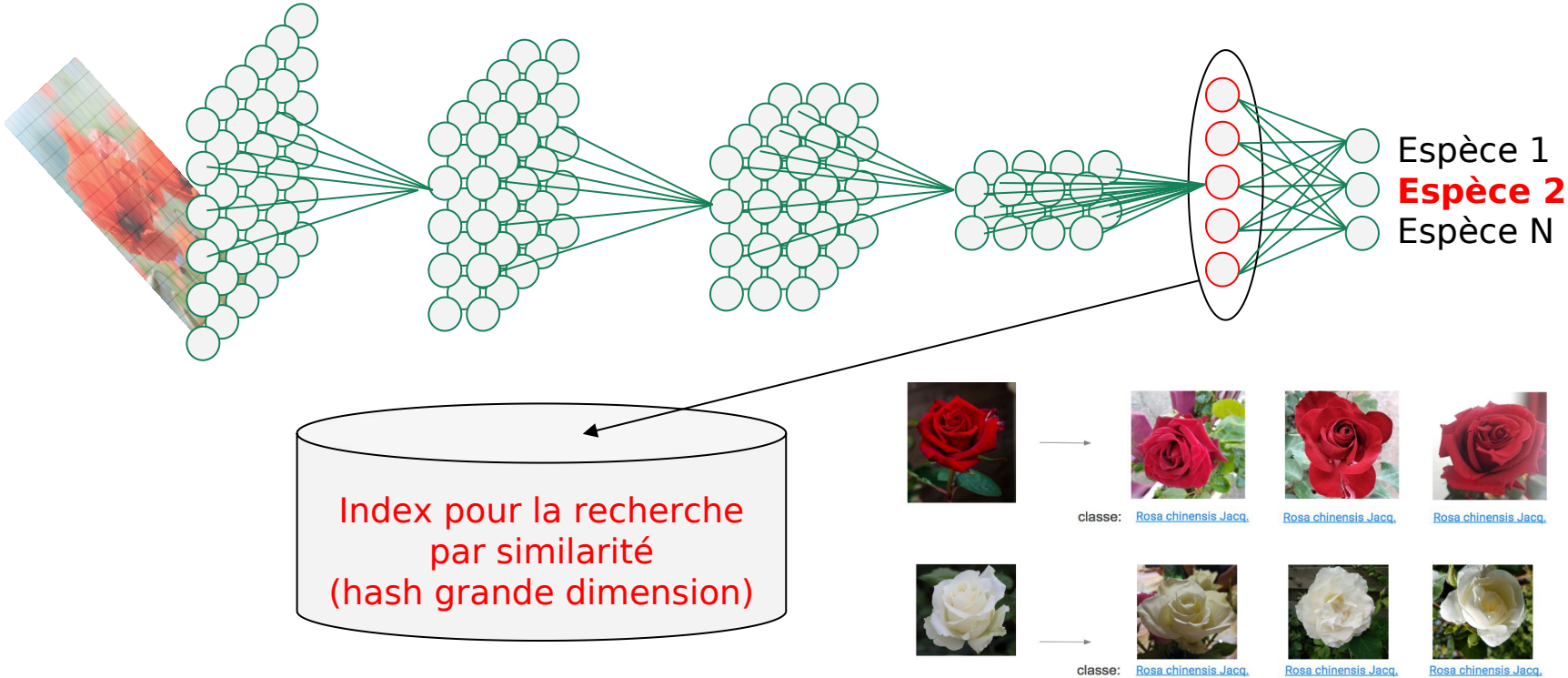




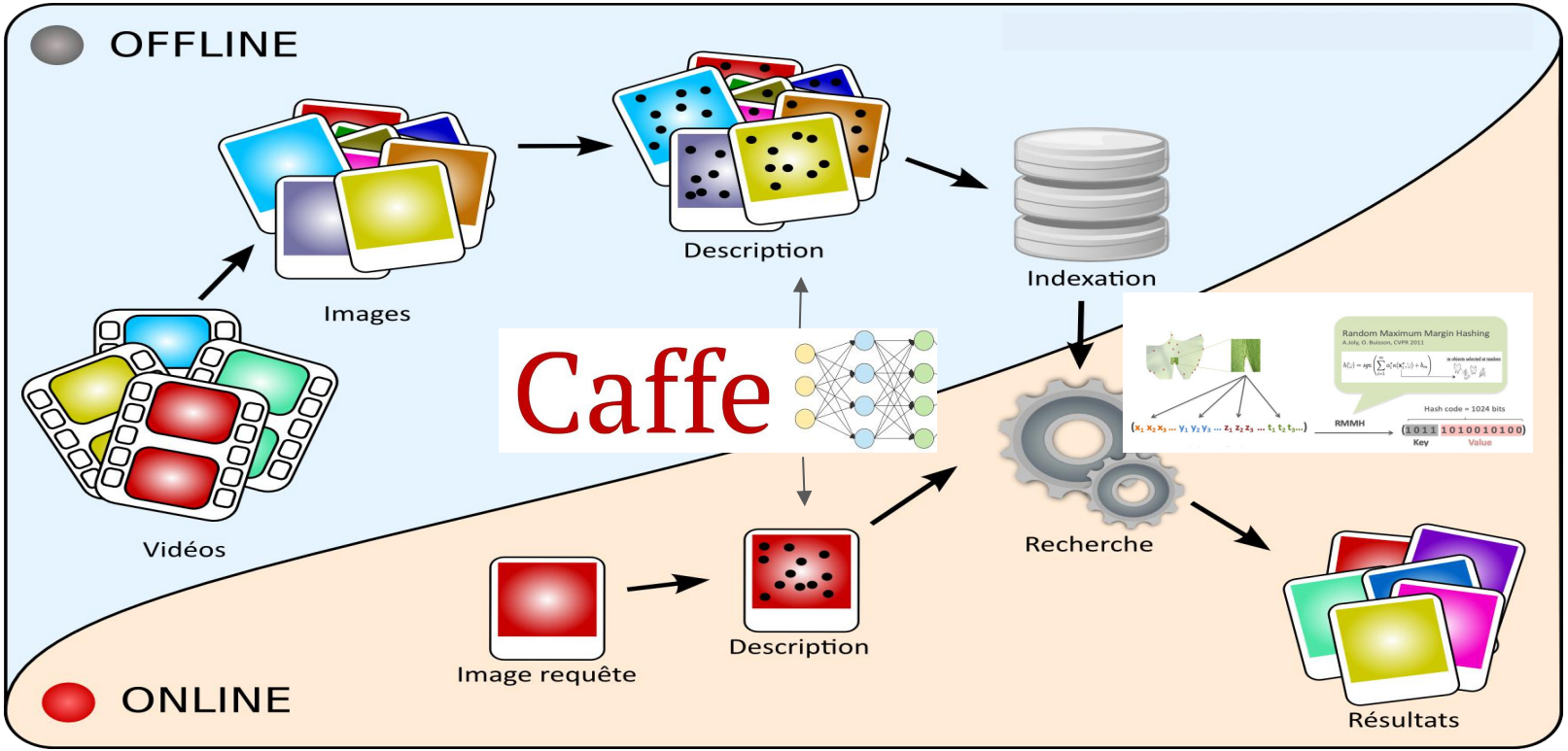
Visualizations of Layers 3, 4, and 5

Exemple Snoop

Pl@ntNet



Snoop : moteur de recherche visuel

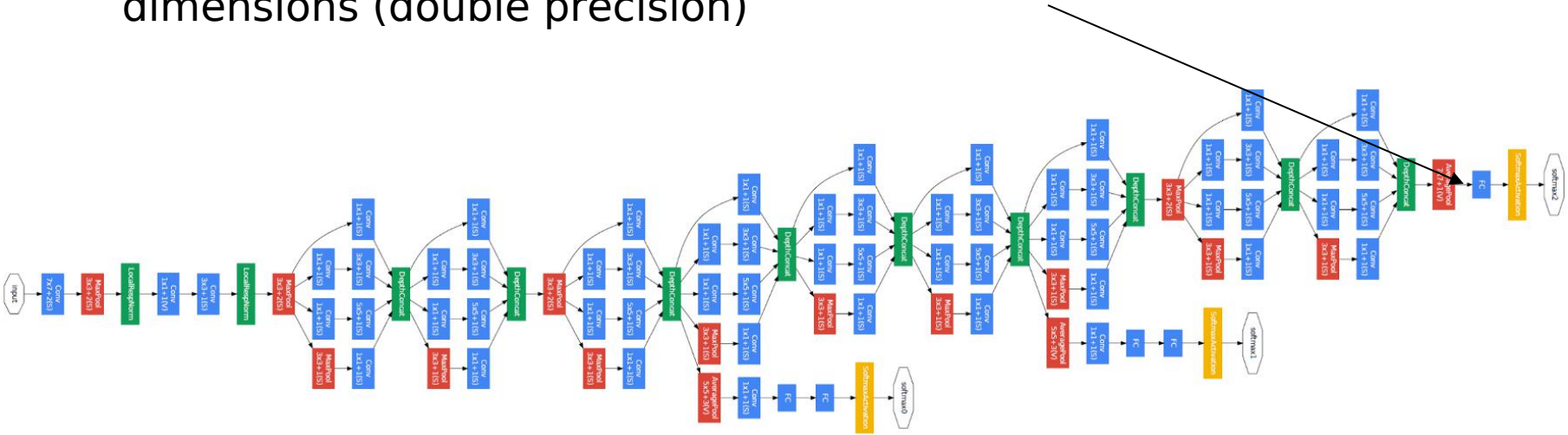


Ajout du deep learning en 2016

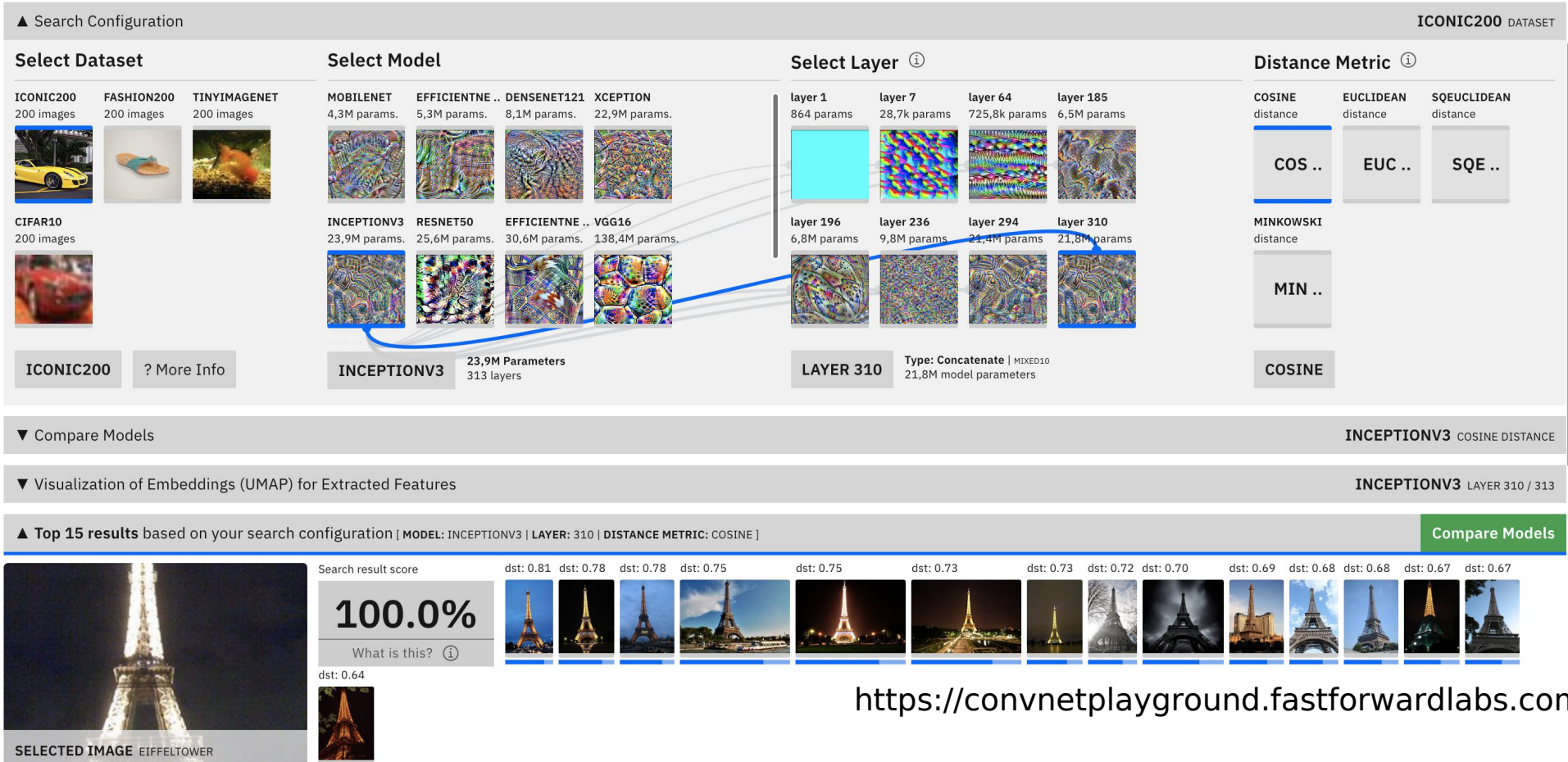
Snoop : configuratic

Pl@ntNet

- **Extraction de caractéristiques :** Inception v2, dernière couche (pré-classifieur), vecteur de caractéristiques de 1 024 dimensions (double précision)



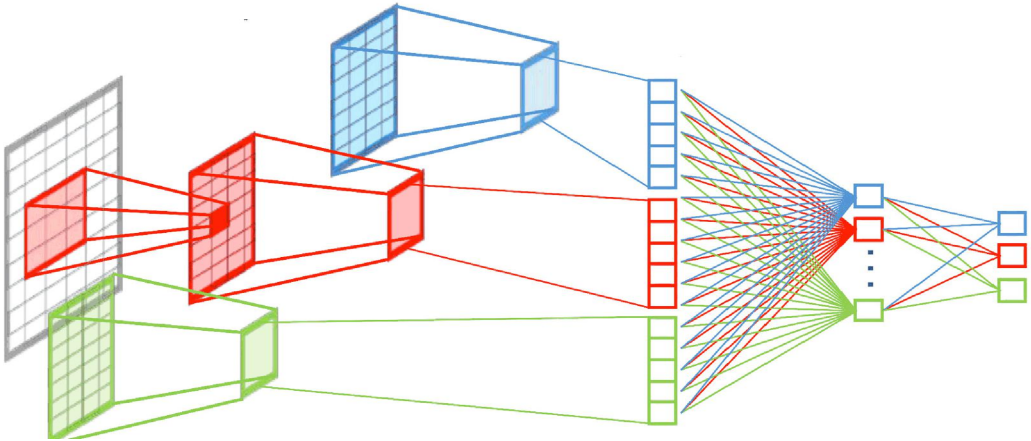
Simulation CNN

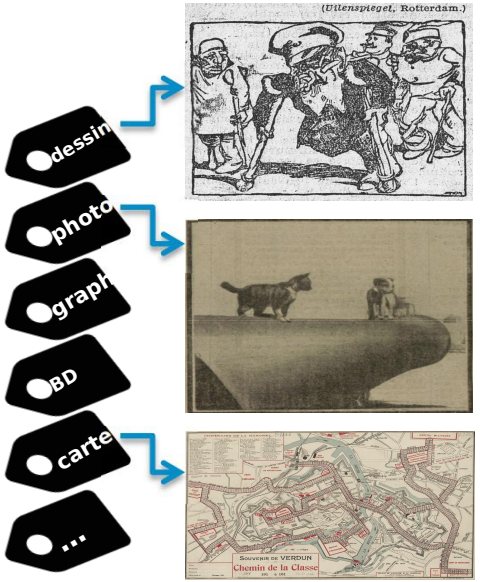


https://convnetplayground.fastforwardlabs.com

BnF : classification de types d'image

- **Classification** avec un réseau de neurones artificiels convolutionnel et une approche par « transfer learning »
- Identification du « **type** » des illustrations (photos, dessins, cartes, BD, graphes et schémas…)





Classification de genres

- **Transfer learning** : seule la dernière couche du réseau est réentrainée sur un jeu de données Gallica (**12 classes)**
- **4 classes de « bruit »** : couverture, page blanche, ornement, texte

« Bruit »

Classification avec Watson Studio ou Google AutoML

1. Préparer des images (*x* par classe, 10 mini pour Watson, un zip par classe)
2. Entraîner le modèle de classification
3. Evaluer, tester
4. Utiliser le modèle :

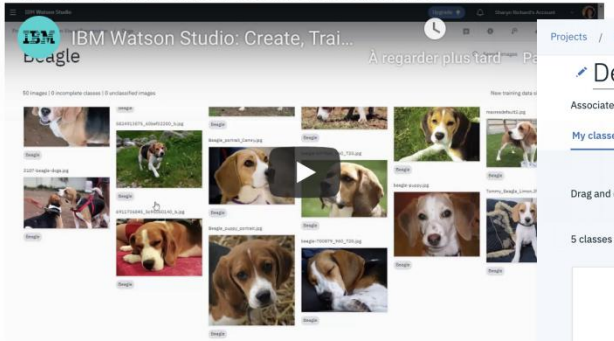
- dans l'écosystème IBM/Google
- Apple CoreML (IBM)
- avec du code (via les API)

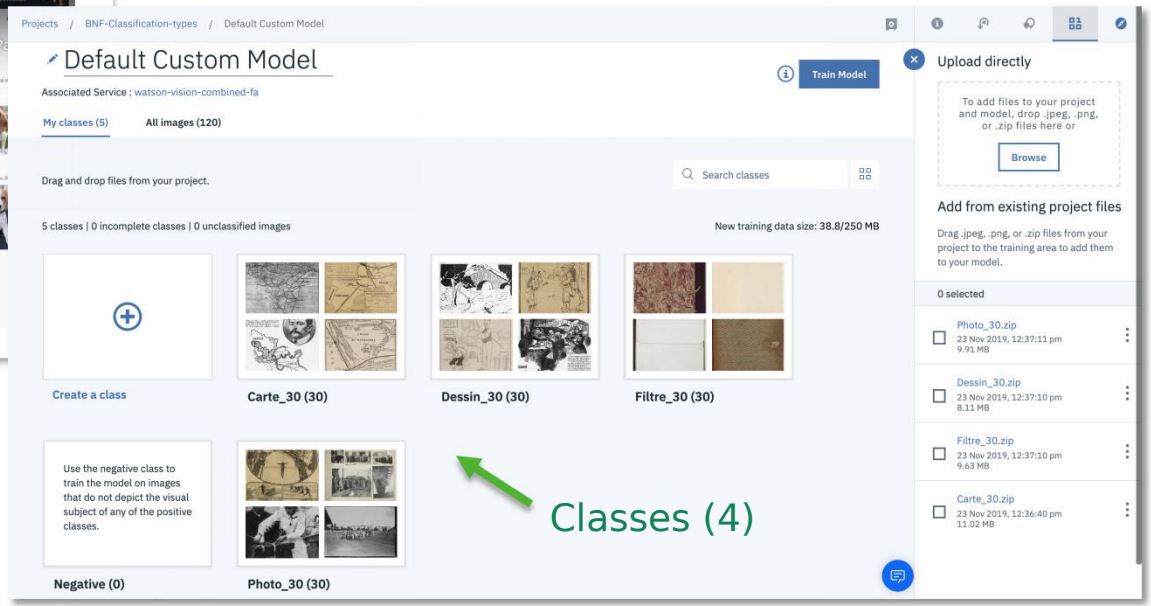
Avec IBM Watson Studio : entraînement

Demonstration

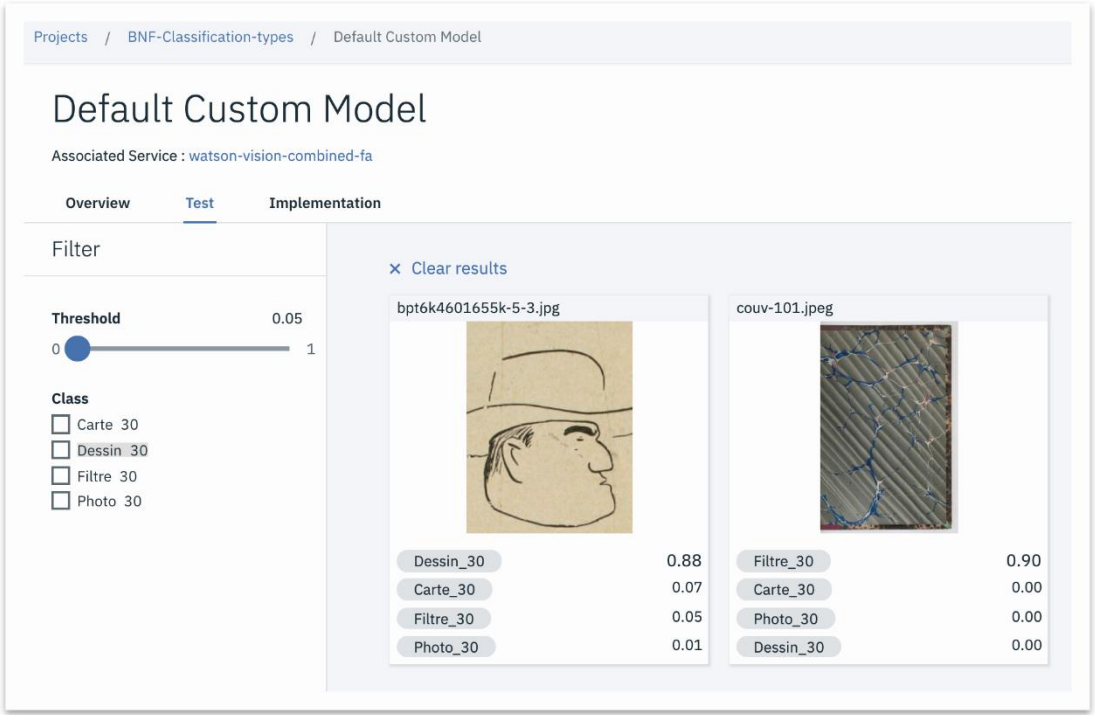
Watch this video to see how to create, train, and test a visual recognition custom model.

Figure 1. Create, train, and test a visual recognition custom model

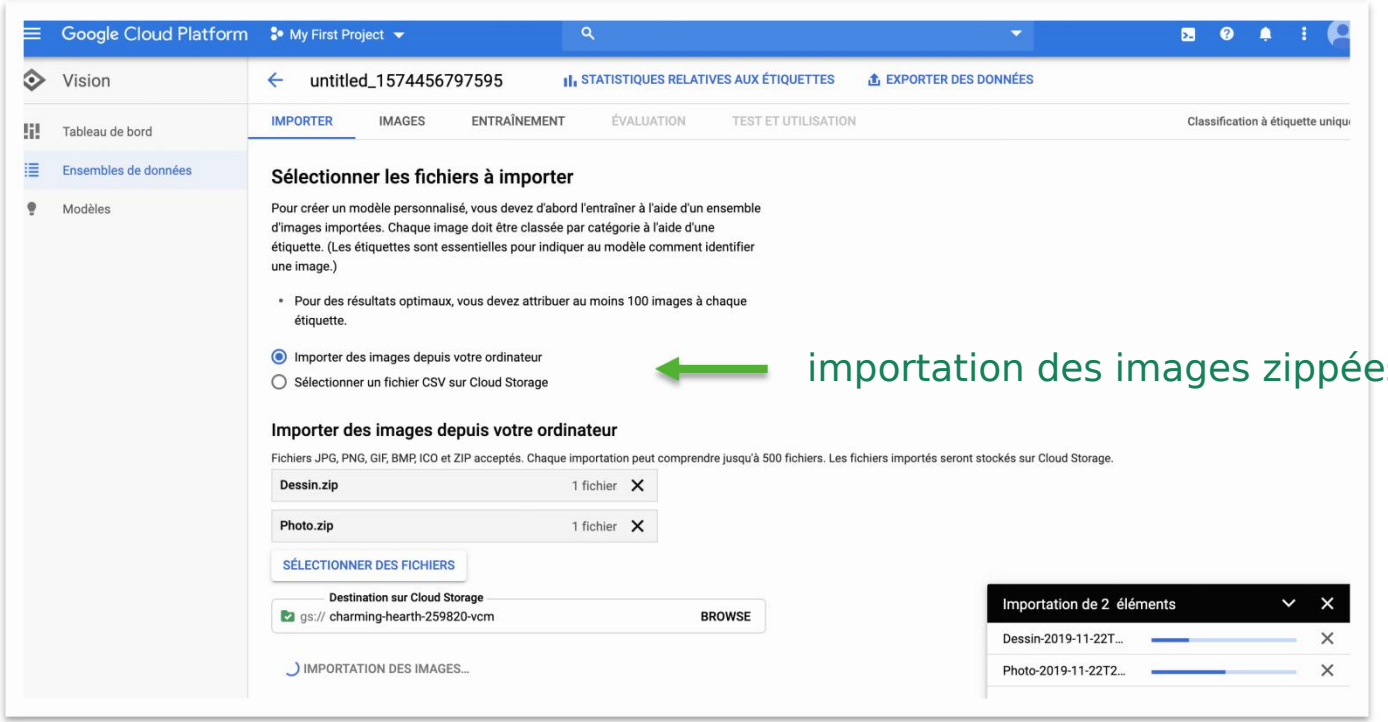




Avec IBM Watson Studio : test

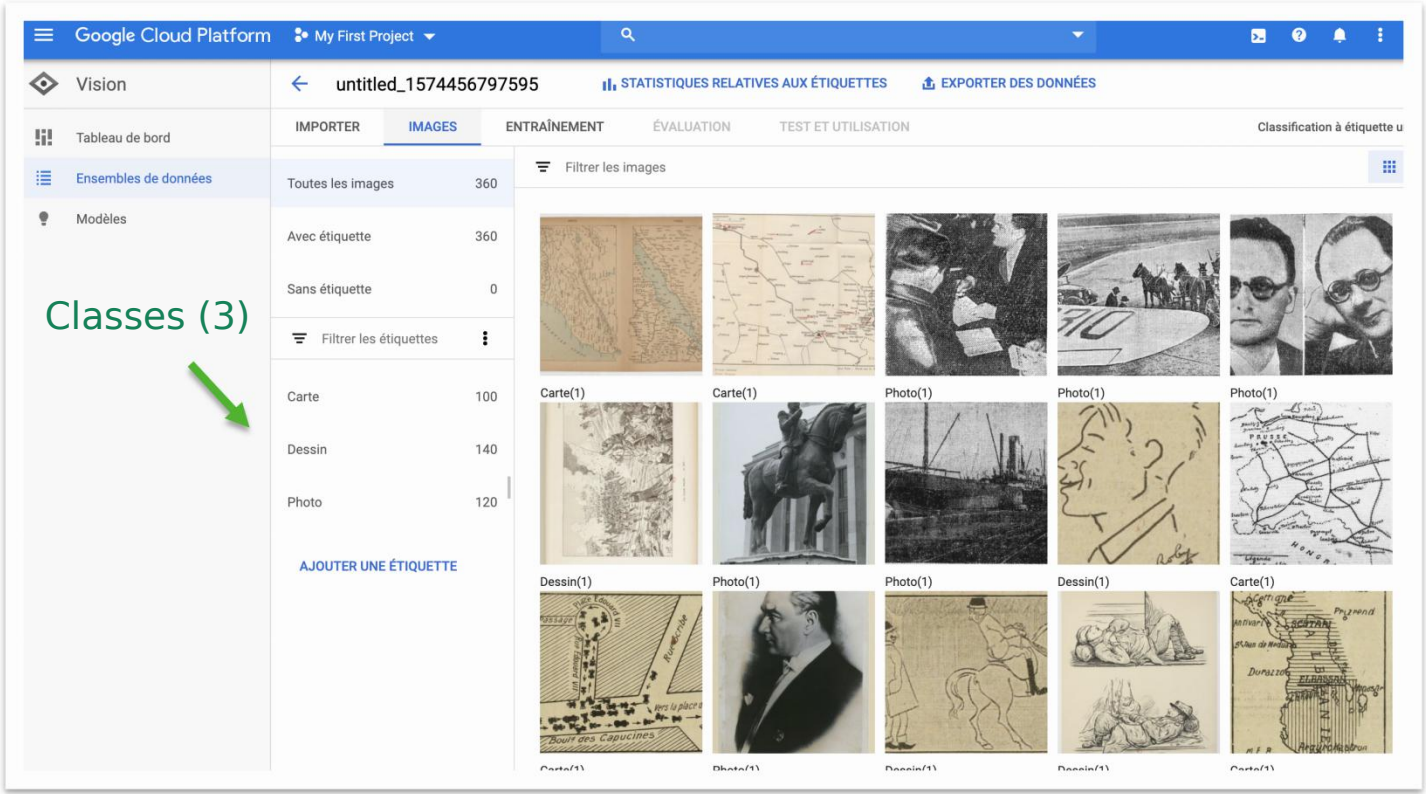


Avec Google Cloud Platform/AutoML

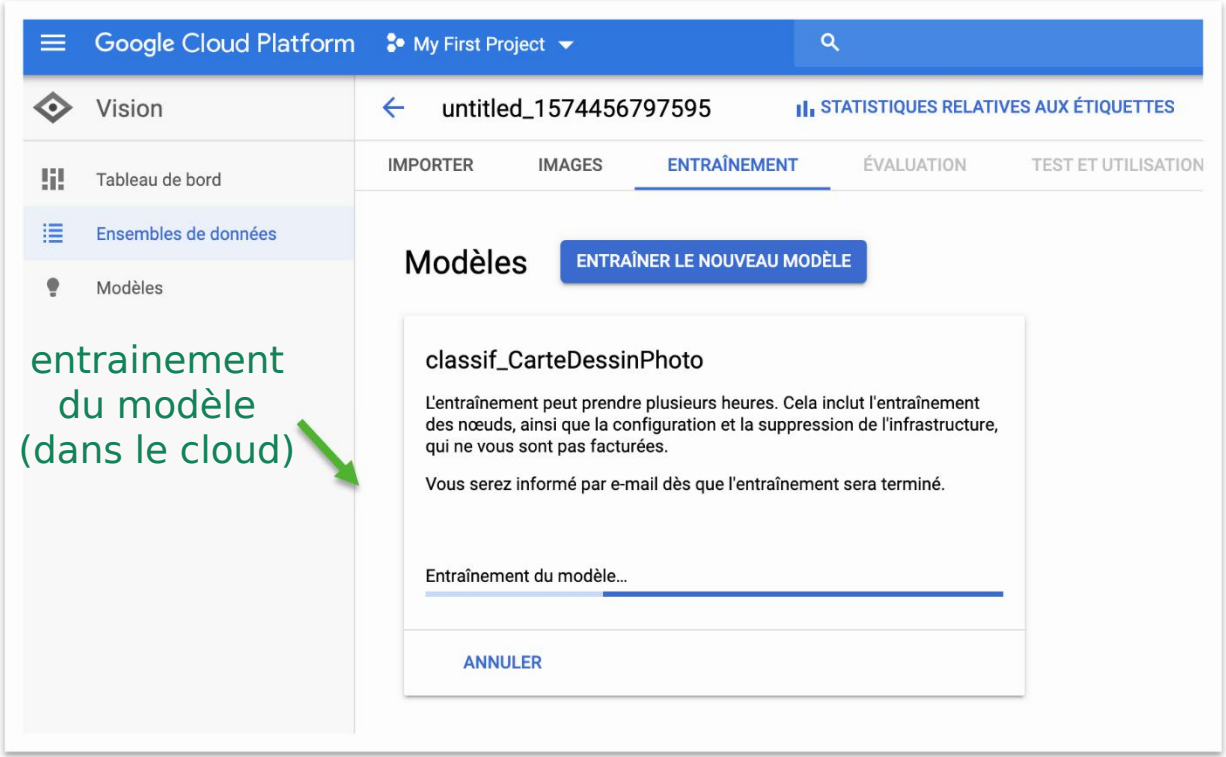


Voir aussi : https://cloud.google.com/vision/automl/docs/beginners-guide?hl=fr

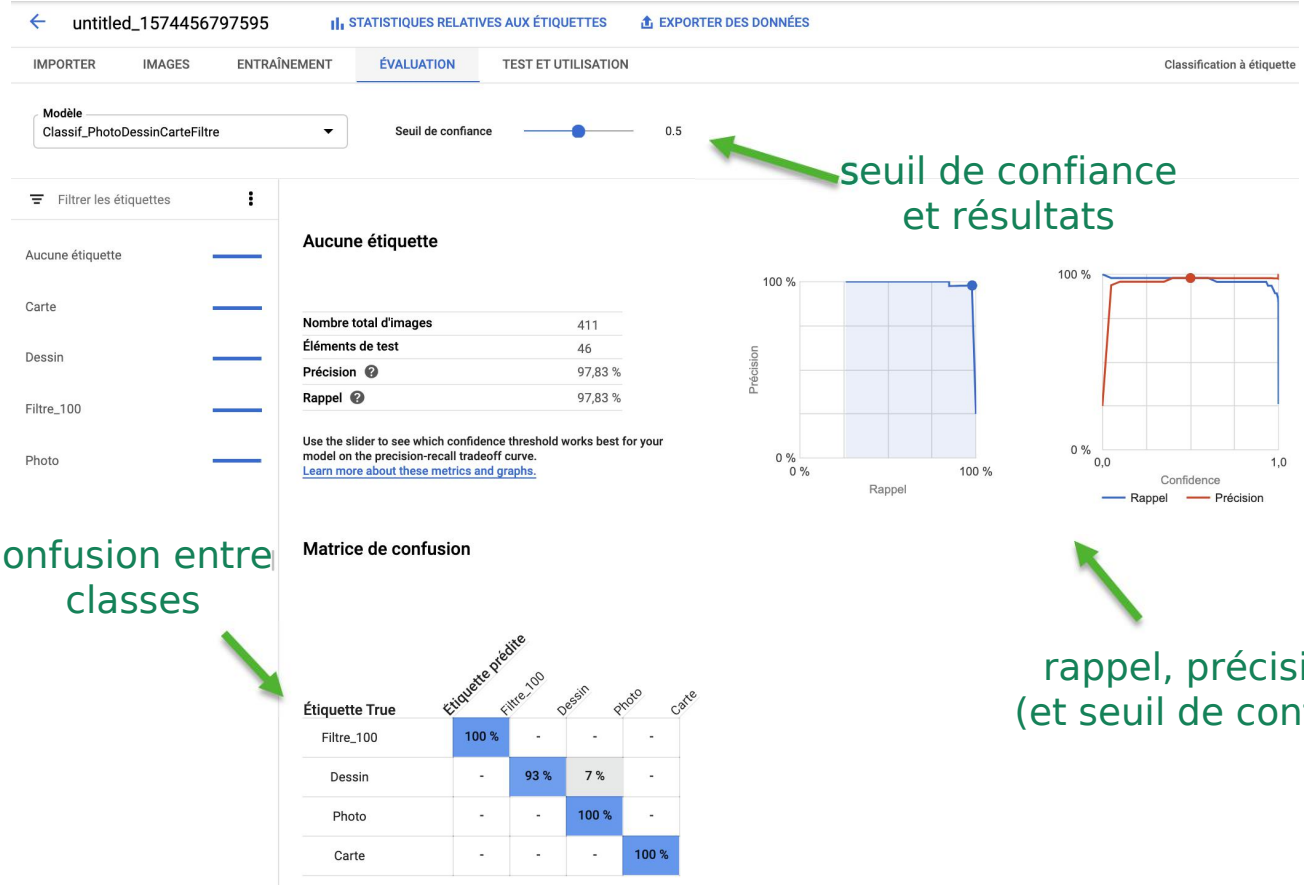
Avec Google Cloud Platform/AutoML



Avec Google Cloud Platform/AutoML

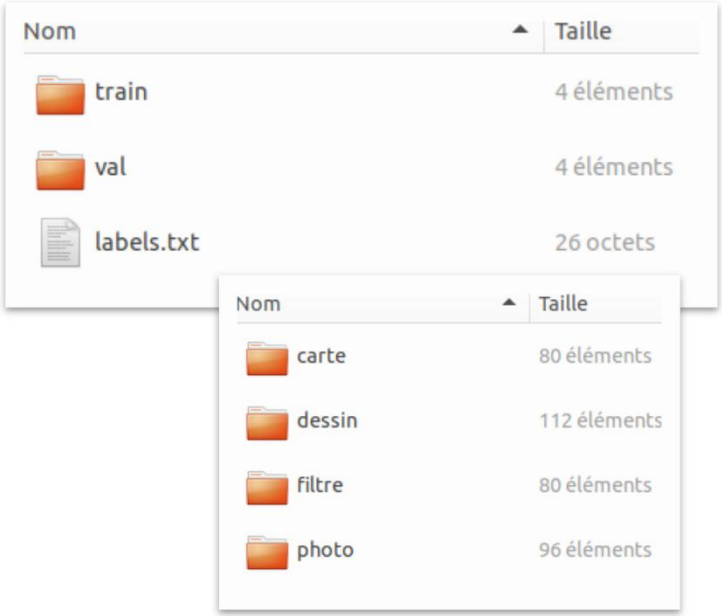


Avec Google Cloud Platform/AutoML



Classification avec un framework IA

- Offre : TensorFlow (Google), PyTorch (Facebook), CNTK (Microsoft), Caffe2, Keras…




```
TRT]   binding -- index   1
               -- name    'output_0'
               -- type    FP32
               -- in/out  OUTPUT
               -- # dims  1
TRT]   warning -- unknown nvinfer1::DimensionType (127)
               -- dim #0  4 (UNKNOWN)
TRT]   binding to input 0 input_0  binding index:  0
TRT]   binding to input 0 input_0  dims (b=1 c=3 h=224 w=224) size=602112
TRT]   binding to output 0 output_0  binding index:  1
TRT]   binding to output 0 output_0  dims (b=1 c=4 h=1 w=1) size=16
evice GPU, img-entrainement/resnet18.onnx initialized.
TRT]   img-entrainement/resnet18.onnx loaded
mageNet -- loaded 4 class info entries
mg-entrainement/resnet18.onnx initialized.
lass 0000 - 0.413331  (carte)
lass 0001 - 0.353872  (dessin)
lass 0002 - 0.199865  (filtre)
lass 0003 - 0.032932  (photo)
mage is recognized as 'carte' (class #0) with 41.333064% confidence


TRT]   ------------------------------------------------
TRT]   Timing Report img-entrainement/resnet18.onnx
TRT]   ------------------------------------------------
TRT]   Pre-Process   CPU   0.08125ms  CUDA   0.39380ms
TRT]   Network       CPU  33.51479ms  CUDA  32.94156ms
TRT]   Post-Process  CPU   0.09761ms  CUDA   0.09849ms
TRT]   Total         CPU  33.69365ms  CUDA  33.43386ms
TRT]   ------------------------------------------------

TRT]   note -- when processing a single image, run 'sudo jetson_clocks' before
               to disable DVFS for more accurate profiling/timing measurements

etson.utils -- freeing CUDA mapped memory
```

- Jeux de données : https://www.dropbox.com/sh/7puok3g2cc8hcda/AACw6h6fPfx93_TKn5ZipWqSa?dl=0
- Comptes :
  IBM : jp.moreux@yahoo.fr
  Google : : jpmoreuxbnf@gmail.com
  Jetson Nano : jpm / nanojp