Réseau de neurones Deep Learning

d'après les transparents de :

- Antoine Cornuéjol
- Laurent Orseau
- Cours du CEA
- ...

Et les sites :

https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

https://cs231n.github.io/convolutional-networks/

Introduction

Pourquoi les réseaux de neurones ?

- Inspiration biologique
  - Le cerveau naturel : un modèle très séduisant
    - Robuste et tolérant aux fautes
    - Flexible. Facilement adaptable
    - S'accommode d'informations incomplètes, incertaines, vagues, bruitées ...
    - Massivement parallèle
    - Capable d'apprentissage
  - Neurones
    - ≈ 10^{11} neurones dans le cerveau humain
    - ≈ 10^{4} connexions (synapses + axones) / neurone
    - Signaux excitateurs / inhibiteurs

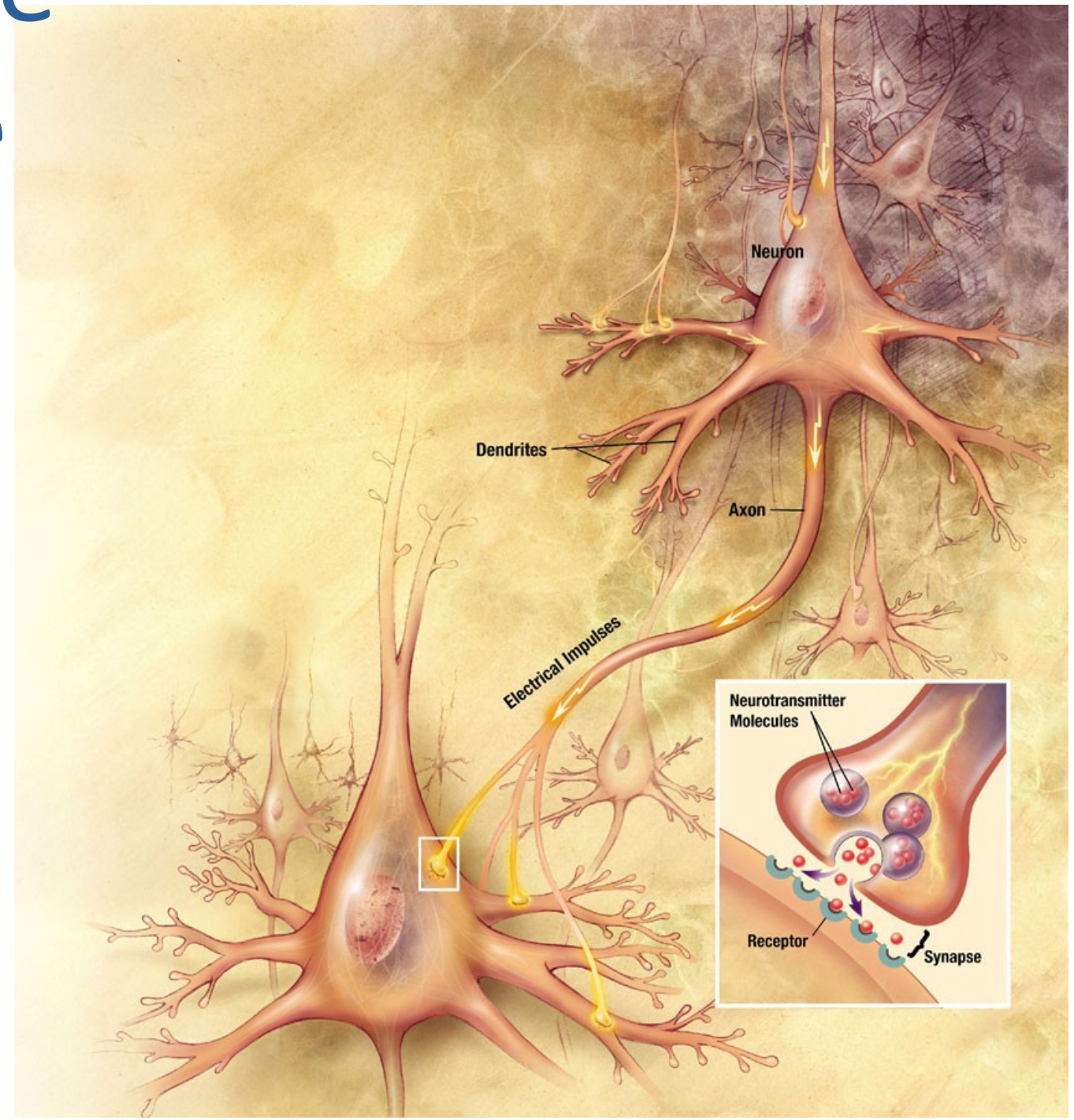
- Les attraits pratiques
  - Calculs parallélisables
  - Implantables directement sur circuits dédiés
  - Robustes et tolérants aux fautes (calculs et représentations distribués)
  - Algorithmes simples
  - D'emploi très général

- Les défauts
  - Opacité des "raisonnements"
  - et des résultats d'apprentissage

Historique

- Prémisses
  - Mc Culloch & Pitts (1943) : 1er modèle de neurone formel.
    Lien entre neurone et calcul logique : base de l'intelligence artificielle.
  - Règle de Hebb (1949) : apprentissage par renforcement du couplage synaptique
- Premières réalisations
  - ADALINE (Widrow-Hoff, 1960)
  - **PERCEPTRON** (Rosenblatt, 1958-1962)
  - Analyse de Minsky & Papert (1969)
- Nouveaux modèles
  - Kohonen (apprentissage compétitif), ...
  - Hopfield (1982) (réseau bouclé)
  - **Perceptron Multi-Couches** (1985)
- Analyse et développements
  - Théorie du contrôle, de la généralisation (Vapnik), ...

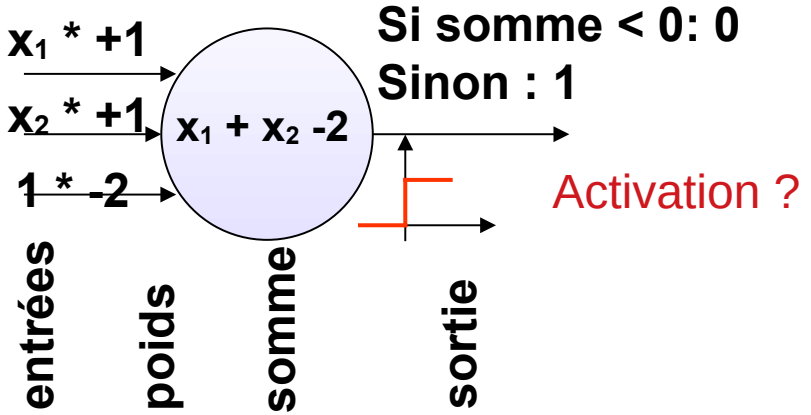
Le neurone biologique



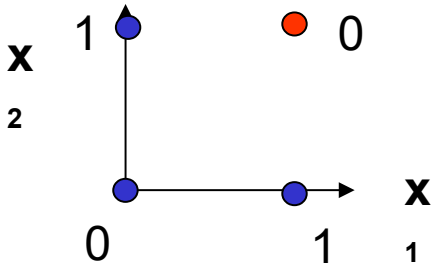
Qu'est-ce qu'un neurone artificiel ?

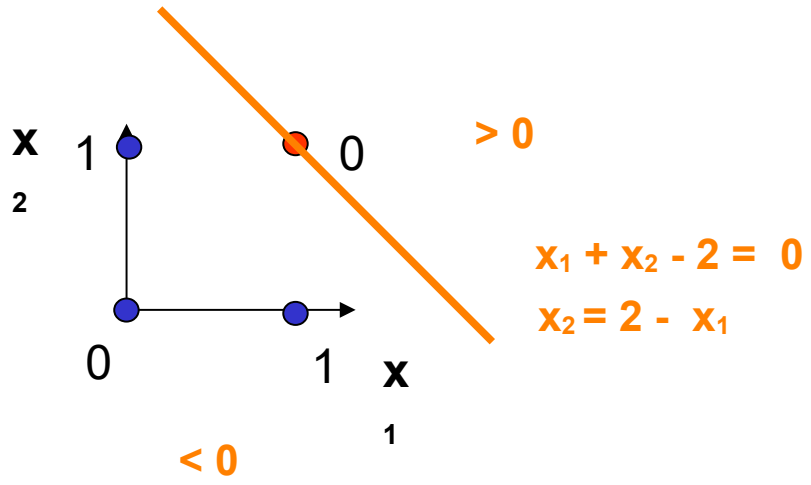
- Ensemble d'unités de calcul simples (les neurones) reliées de façon complexe
- Imaginés dans les années 40 (Mac Culloch et Pitts, Turing, …)
- Comprendre comment le cerveau peut réaliser certaines tâches qu'un ordinateur à du mal à réaliser
- [redacted] ut base (ET, OU, NON)

W.S. McCulloch & W. Pitts (1943). "A logical calculus of the ideas immanent in nervous activity", *Bulletin of Mathematical Biophysics*, 5, 115-137.

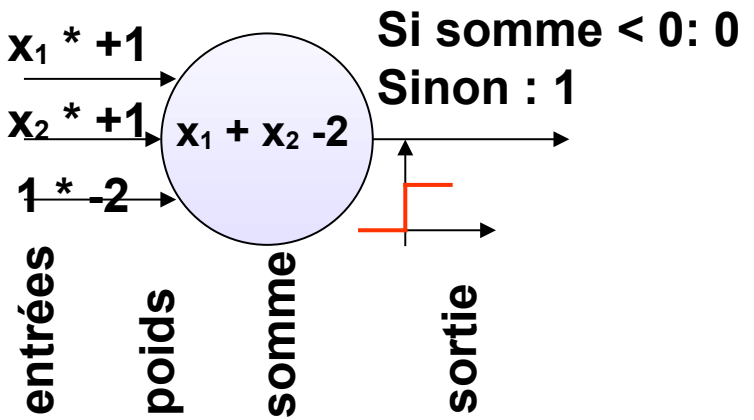


x_1	x_2	x_1 AND x_2
0	0	0
0	1	0
1	0	0
1	1	1





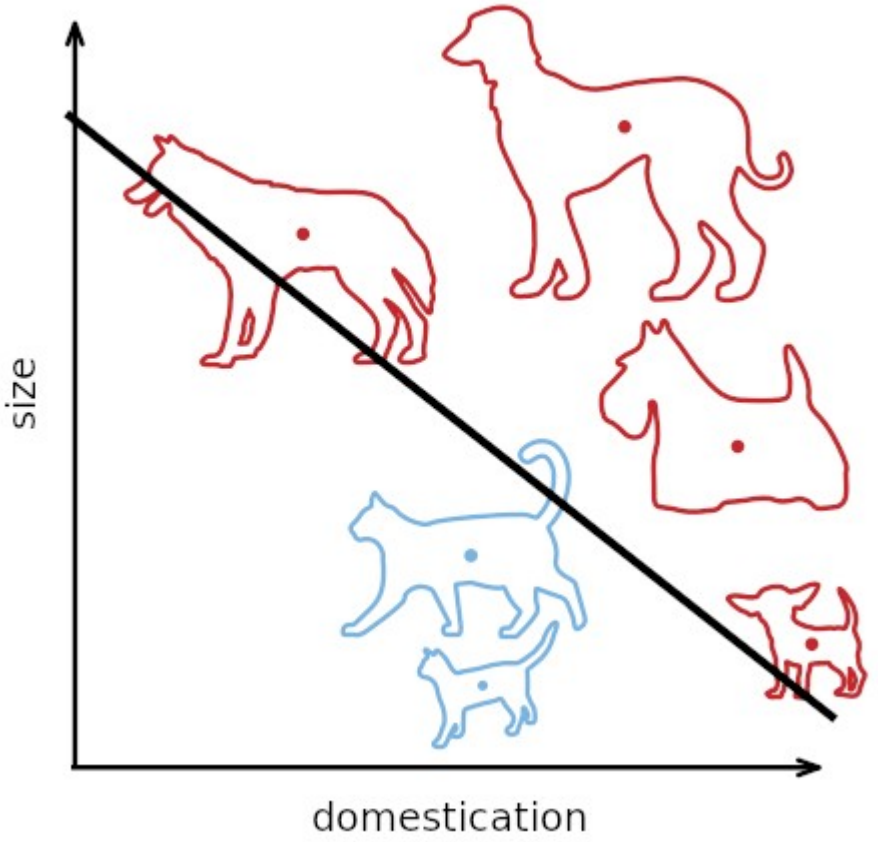
W.S. McCulloch & W. Pitts (1943). "A logical calculus of the ideas immanent in nervous activity", *Bulletin of Mathematical Biophysics*, 5, 115-137.



x_1	x_2	x_1 AND x_2	$x_1 + x_2 - 2$	$x_1 + x_2 - 2 \geq 0$
0	0	0	-2	0
0	1	0	-1	0
1	0	0	-1	0
1	1	1	0	1

L’idée générale est de représenter les objets d’intérêts par des caractéristiques numériques :

vecteur de $\mathbb{R}^n$



Types d'apprentissage

Le but des réseaux neuronaux est d'apprendre à répondre correctement à différentes entrées.

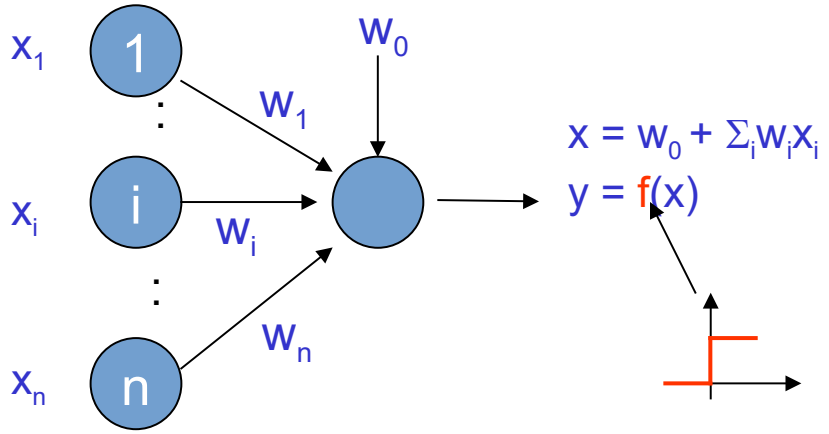
Par modification des poids par apprentissage supervisé, ou non supervisé.

- **Apprentissage supervisé**: un système "instructeur" corrige les réponses éronnées.

- **Apprentissage non supervisé:** le système apprend tout seul en formant des classes à réponses communes.

Le neurone formel

- Les x_i sont des réels appelés entrées
- Les w_i sont des réels appelés poids
- w_0 est le biais (non relié à une entrée)
- La surface $w_0 + \sum_i w_{ij} x_i = 0$ est un hyperplan dans l'espace des n variables d'entrée → le réseau réalise donc une séparation linéaire discrimante

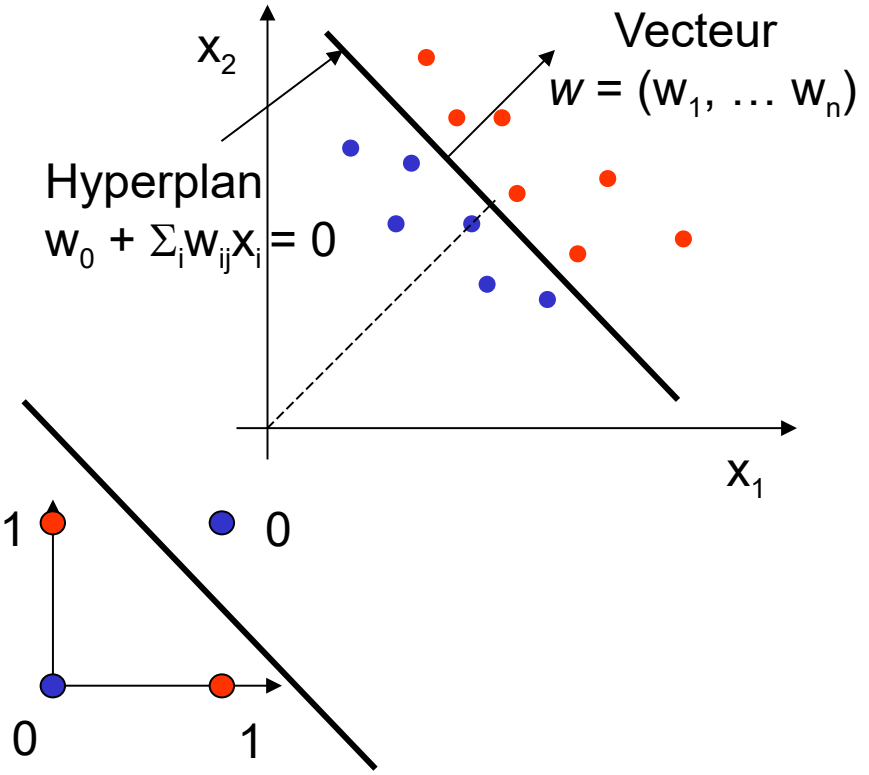


- La sortie du réseau est y=f(x) où **f** est la fonction d'activation
- L'information se propage des entrées vers la sortie (« feedforward »)

Le neurone formel

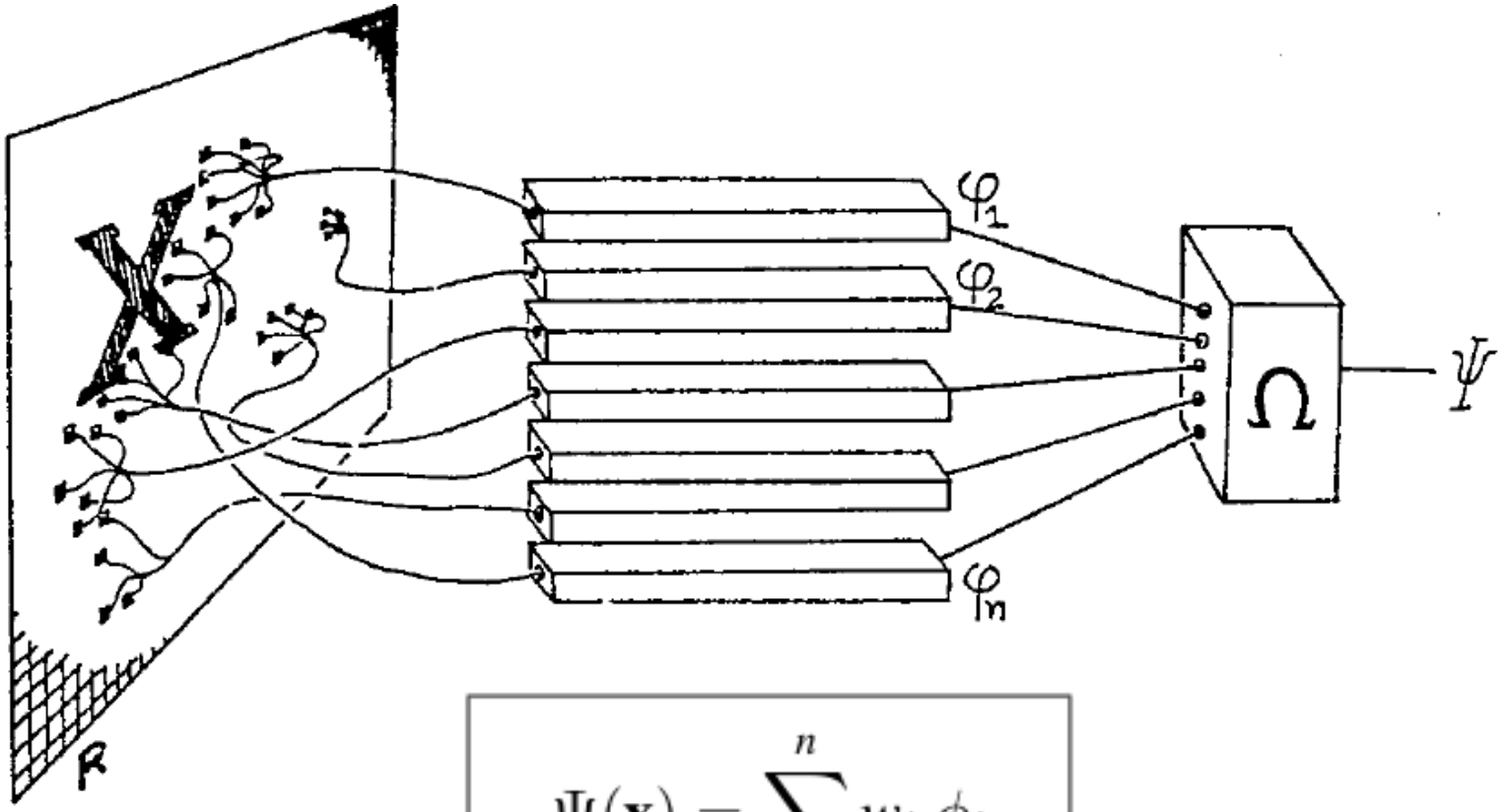
- Les x_i sont des réels appelés entrées
- Les w_i sont des réels appelés poids
- w_0 est le biais (non relié à une entrée)
- La surface $w_0 + \Sigma_i w_{ij} x_i = 0$ est un hyperplan dans l'espace des n variables d'entrée → le réseau réalise donc une séparation linéaire discrimante

- La sortie du réseau est y=f(x) où **f** est la fonction d'activation
- Remarques:
  - Le vecteur $w = (w_1, \ldots, w_n)$ est orthogonal à l'hyperplan
  - La distance de l'hyperplan à l'origine est $-w_0 / ||w||$
  - Un réseau sans couche cachée ne peut donc pas résoudre des problèmes non linéairement séparables (ex: **XOR**)



Le perceptron

- Rosenblatt (1958-1962)



$$\Psi(\mathbf{x}) = \sum_{i=1}^{n} w_i \, \phi_i$$

Algorithme d'apprentissage du perceptron

Entrée :

- un échantillon S : m couples

(entrées,sorties désirées)=($\mathbf{x}_k$={x_1,..x_n},d_k)

- un taux d'apprentissage η

Pour i = 1..n

initialiser aléatoirement les poids w_i

Répéter jusqu'à convergence

Prendre un exemple ($\mathbf{x}_k$, d_k) dans S

Calculer la sortie y pour $\mathbf{x}_k$

Pour i = 1..n

$w_i = w_i+\eta(d_k-y)x_i$ ***// Règle de Widrow-Hoff (Delta)***

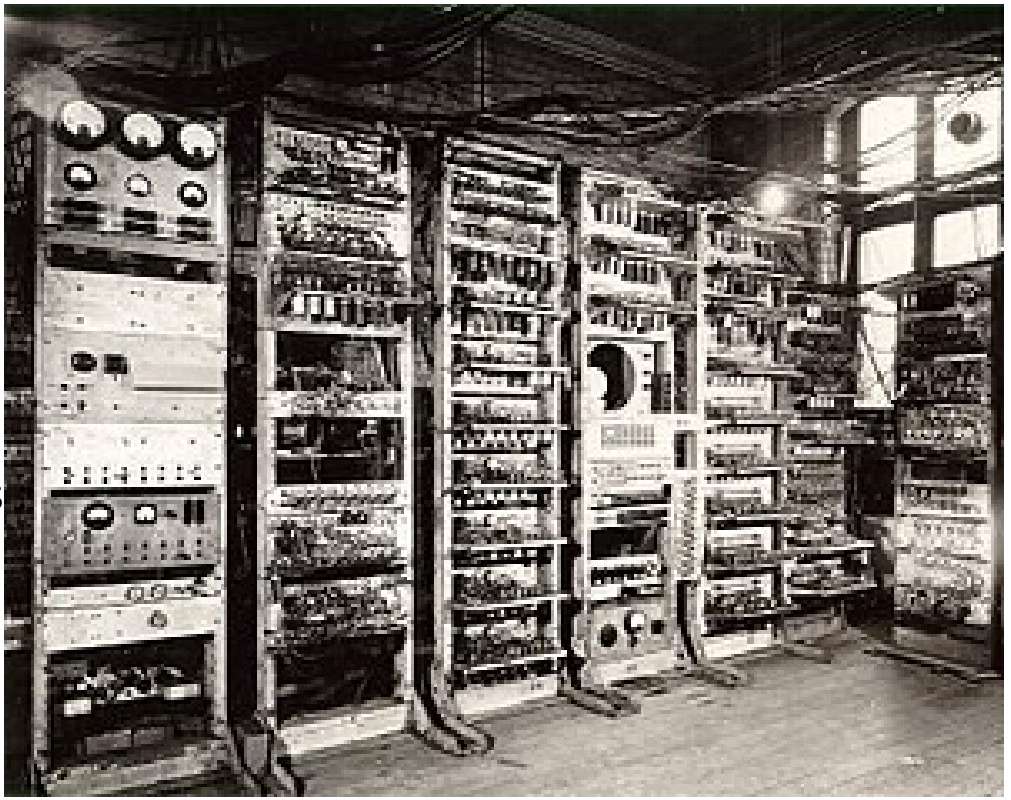
Sortie : le réseau entrainé (w_1,..w_n)

La règle originale est celle de Hebb

$w_i = w_i+\eta y x_i$

Qui s'applique uniquement
en cas de mauvais classement

1969: Minsky & Papert soulignent (exagérément !) les **limitations** des perceptrons, et stoppent la recherche pendant 10 ans...



"Mark 1 perceptron".

Machine crée pour la reconnaissance d'images

- 400 cellules photo voltaiques

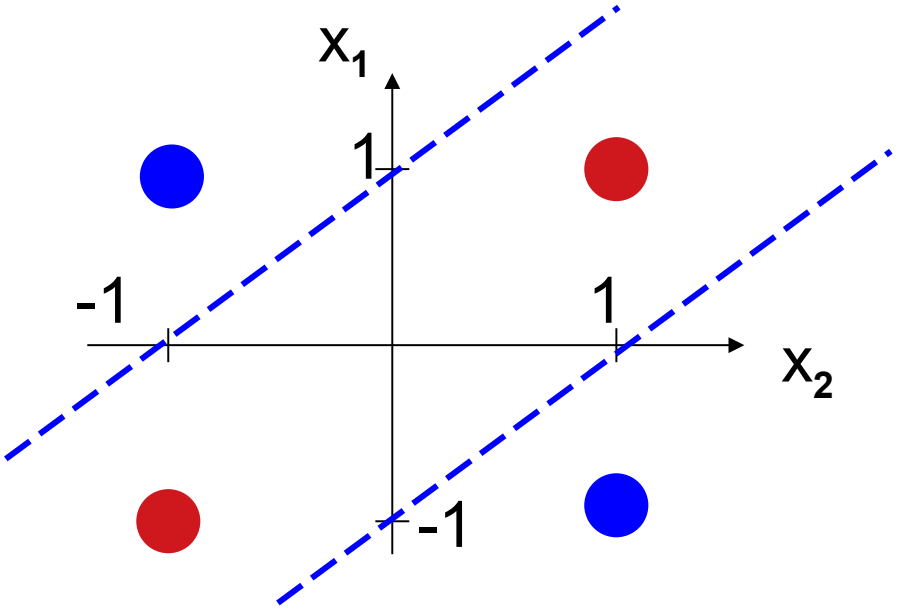
Connectées aléatoirement

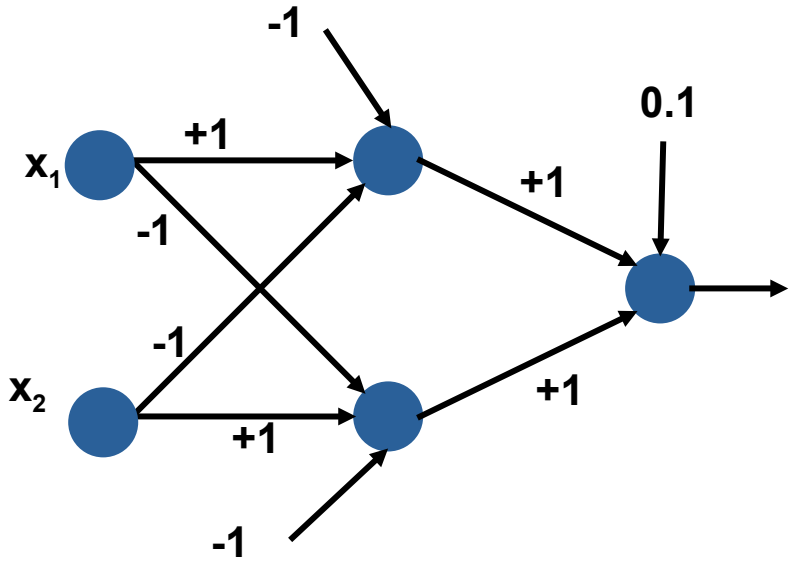
- Poids : potentiometres

- Mise à jour par des moteurs électriques!

Solution pour le XOR

x_1	x_2	x_1 xor x_2
-1	-1	**-1**
-1	1	**1**
1	-1	**1**
1	1	**-1**





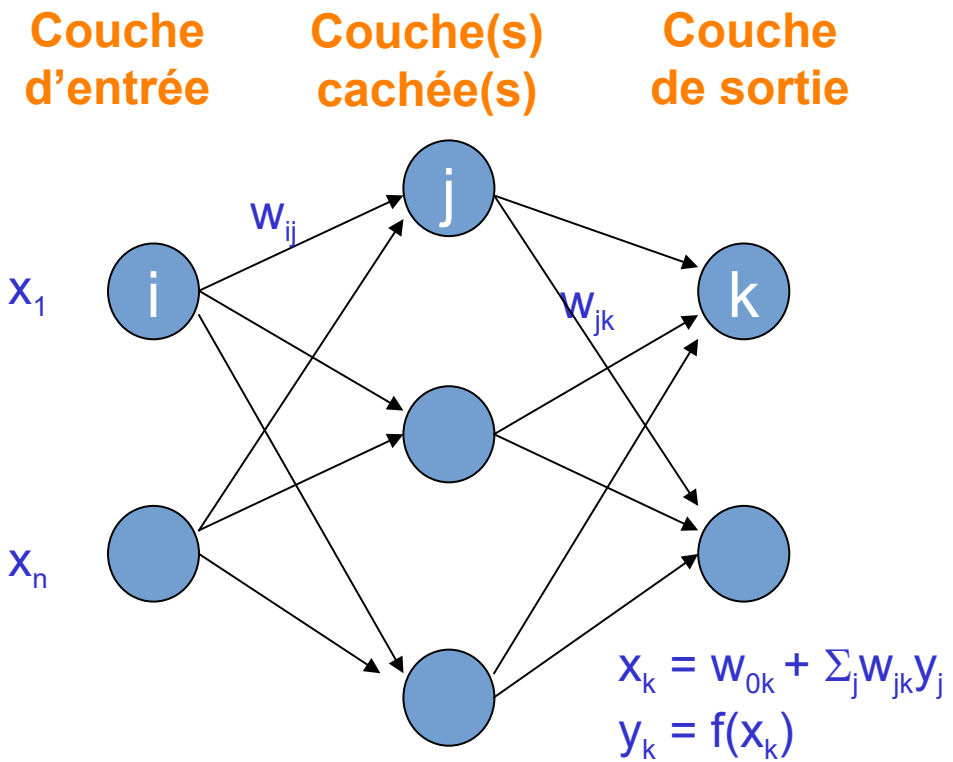
$$f(x) = \begin{cases} 1 & \text{si } x > 0 \\ -1 & \text{si } x \leq 0 \end{cases}$$

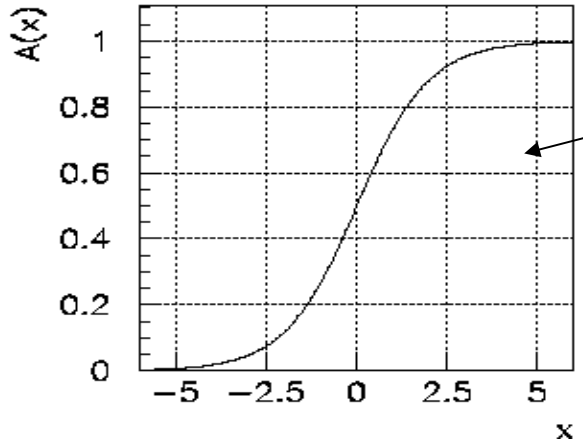
fonction d'activation
(fonction signe)

Problème : trouver automatiquement les poids !

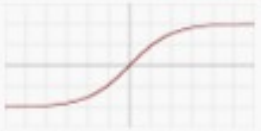
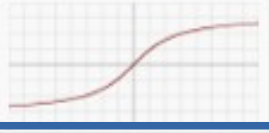
Le perceptron multi-couches

1985: l'algorithme de rétropropagation relance la recherche





- Neurones organisés en couches
  - 0, 1, 2… couches cachées
  - Réseau $n_1 – n_2 – n_3$
- Information se propage dans un seul sens (« feed-forward »)
- La fonction f(x) (fonction d'activation) est
  - linéaire ou non pour le (ou les) neurone(s) de la couche de sortie
  - **non-linéaire** pour les neurones cachés, de type $1/(1+e^{-x})$ (« sigmoïde »)

Name	Plot	Equation	Derivative
Identity		$f(x) = x$	$f'(x) = 1$
Binary step		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$f'(x) \begin{cases} 0 & \text{for } x \neq 0 \\ ? & \text{for } x = 0 \end{cases}$
Logistic (a.k.a Soft step)		$f(x) = \frac{1}{1+e^{-x}}$	$f'(x) = f(x)(1-f(x))$
TanH		$f(x) = \tanh(x) = \frac{2}{1+e^{-2x}} - 1$	$f'(x) = 1 - f(x)^2$
ArcTan		$f(x) = \tan^{-1}(x)$	$f'(x) = \frac{1}{x^2+1}$
Rectified Linear Unit (ReLU)		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Parameteric Rectified Linear Unit (PReLU) [2]		$f(x) = \begin{cases} \alpha x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Exponential Linear Unit (ELU) [3]		$f(x) = \begin{cases} \alpha(e^x - 1) & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} f(x) + \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
SoftPlus		$f(x) = \log_e(1+e^x)$	$f'(x) = \frac{1}{1+e^{-x}}$

La détermination des poids *w*

- Les paramètres ***w*** sont déterminés de façon itérative durant une phase d'apprentissage
- Méthode:
  1. Boucler sur les exemples d'un jeu d'apprentissage (= 1 epoch)
  2. Comparer pour chaque exemple la sortie du réseau et la sortie désirée (« apprentissage supervisé »)
  3. Modifier après chaque exemple les poids du réseau
  4. Recommencer jusqu'à ce que l'erreur soit « suffisament » faible

- Après la phase d'apprentissage, le réseau doit être capable de généraliser (donner une réponse correcte sur un exemple nouveau)

Modifier les poids du réseau

- La méthode pour modifier les poids du réseau est souvent appelée « méthode de la rétropropagation de l'erreur »
  - Sans doute dû à

    Learning representations by back-propagating errors, D.E.Rumelhart et al., Nature vol. 323 (1986), p. 533

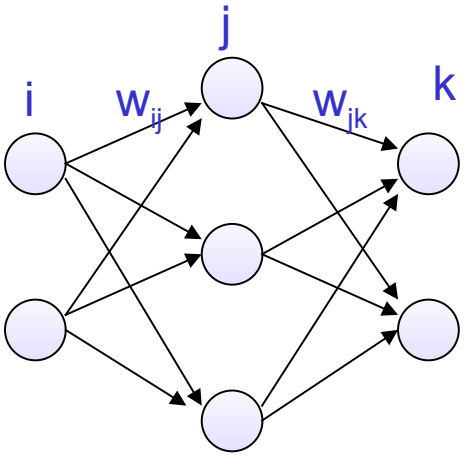
We describe a new learning procedure, back-propagation, for networks of neurone-like units. The procedure repeatedly adjusts the weights of the connections in the network so as to minimize a

La rétropropagation de l'erreur

1. Présenter un exemple au réseau, calculer la sortie y_k, comparer à la réponse attendue

$d_k \rightarrow \delta_k = f'(x_k)(y_k - d_k)$

Poids initiaux aléatoires petits autour de 0

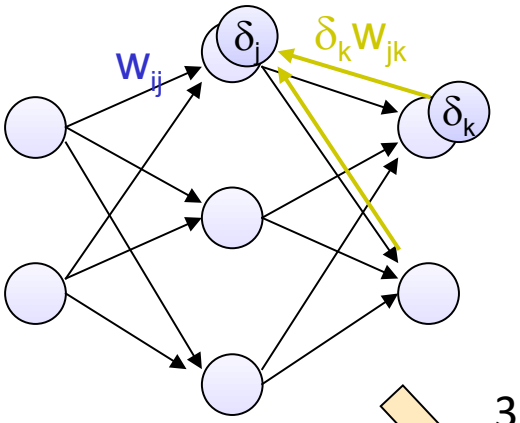


$y_k = A(x_k)$

$x_k = \Sigma_j w_{jk} y_j$

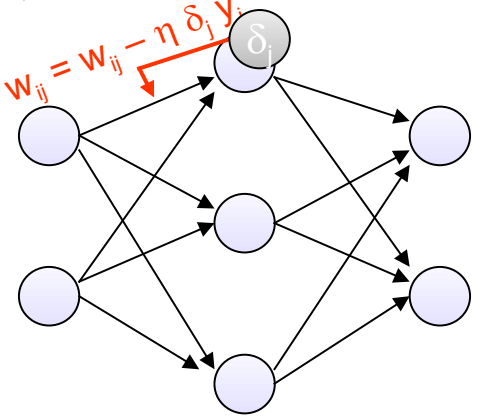
2. Calculer $\delta_j = f'(x_j) \Sigma_k \delta_k w_{jk}$

« Rétropropagation de l'erreur »



Paramètre d'apprentissage < 1, à optimiser selon le problème

3. Modifier les poids selon la formule: $w_{ij}^{\tau+1} = w_{ij}^{\tau} - \eta \, \delta_j \, y_i$

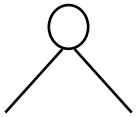
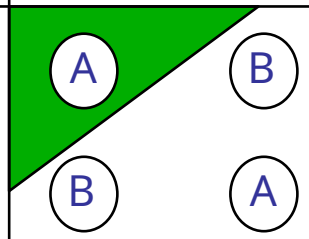
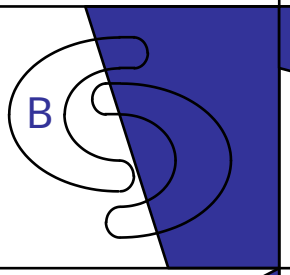
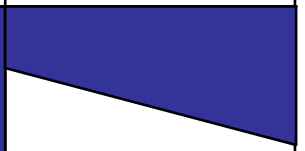
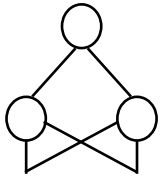
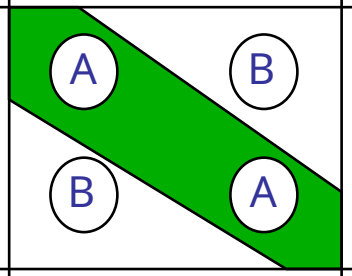
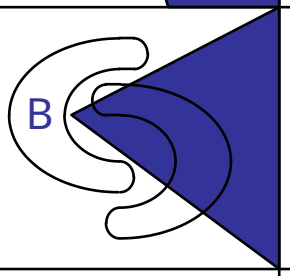
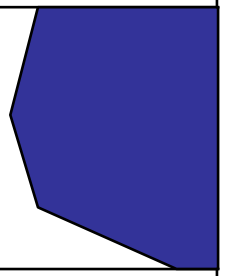
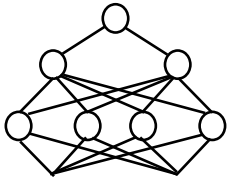
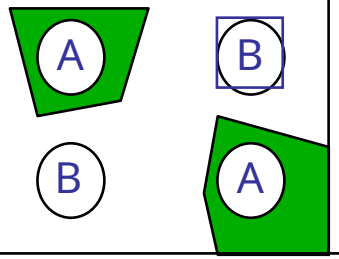
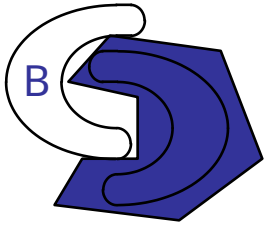
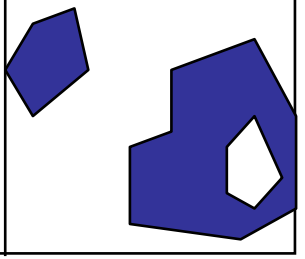


P.Werbos, thèse de l'université de Harvard (1974), publiée dans *The Roots of Backpropagation,* Wiley-IEEE (1994)

Learning representations by back-propagating errors
D.E.Rumelhart et al., Nature vol. 323 (1986), p. 533

Learning processes in an assymetric threshold network
Y. le Cun, *Disordered Systems and Biological Organization*, Springer Verlag, Les Houches, France (1986), p. 233

Capacités de discrimination des différentes architectures de RN

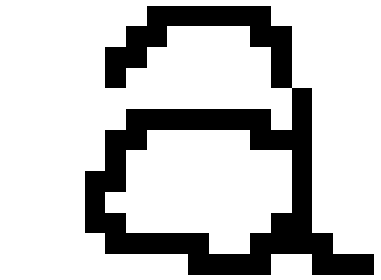
Structure	*Types of Decision Regions*	*Exclusive-OR Problem*	*Classes with Meshed regions*	*Most General Region Shapes*
Single-Layer 	*Half Plane Bounded By Hyper plane*			
Two-Layer 	*Convex Open Or Closed Regions*			
Three-Layer 	Arbitrary (Complexity Limited by No. of Nodes)			

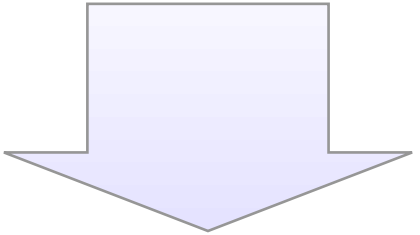
Quelle architecture ?

- Trois couches sont en théorie toujours suffisantes mais avec beaucoup de neurones
- L'algo. de rétropropagation ne garantit pas de trouver une solution….

Apprentissage supervisé (ex: OCR)

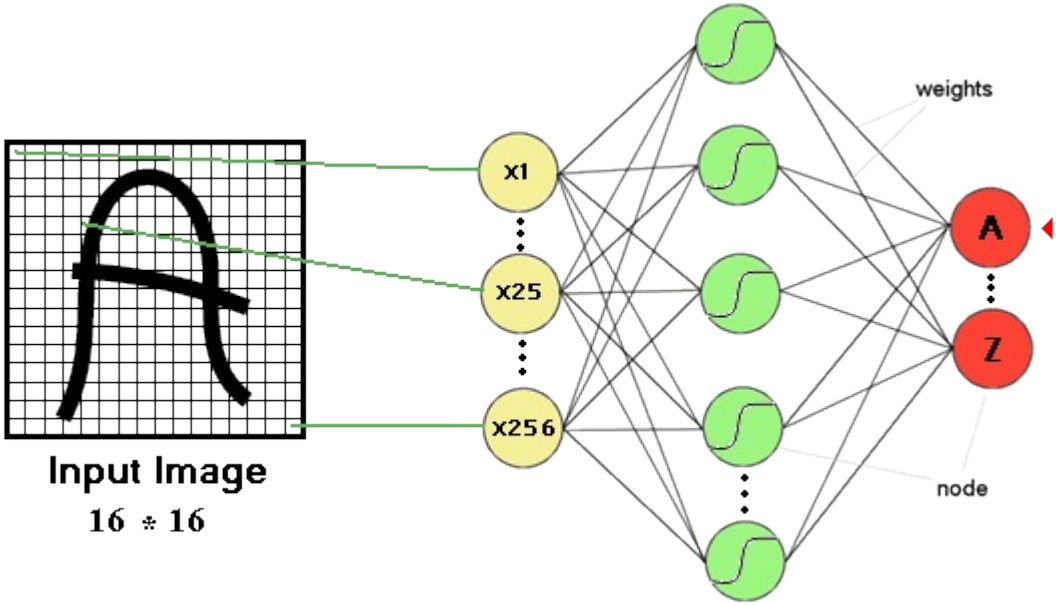
- Association imposée entre un vecteur d'entrée (forme multidimensionnelle) et un vecteur de sortie (la réponse désirée).
- L'erreur est calculée à chaque essai afin de corriger les poids.
- Les poids sont modifiés jusqu'à l'erreur minimale, voire aucune erreur.



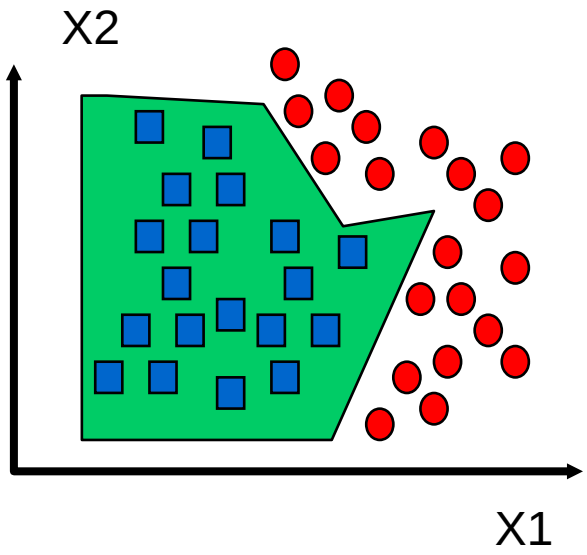


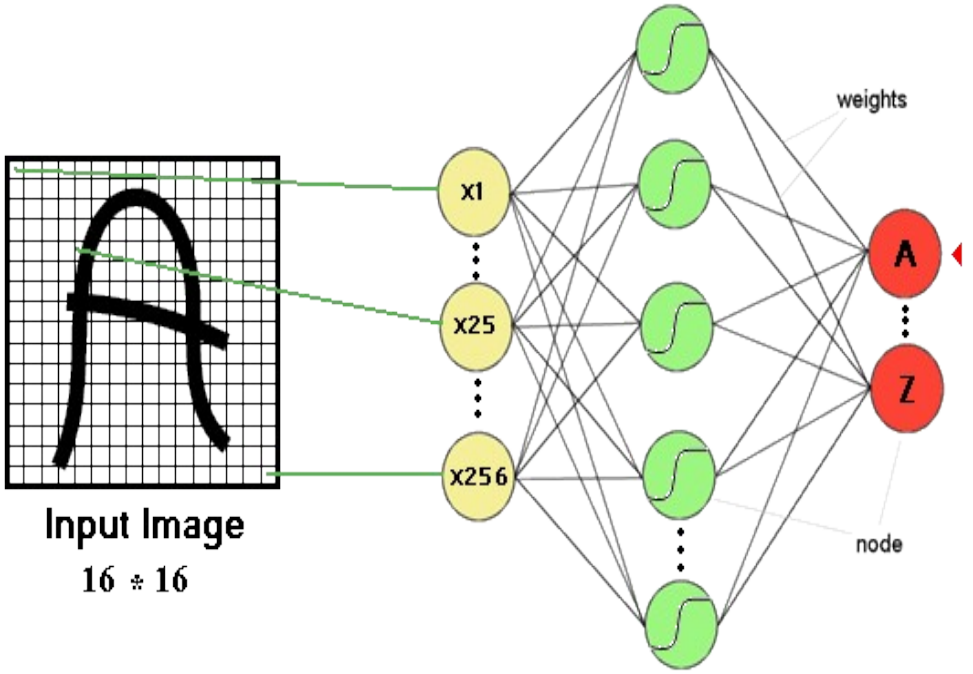
a

- La réponse attendue est le "a". Dans le cas idéal, une seule et sortie doit être activée.



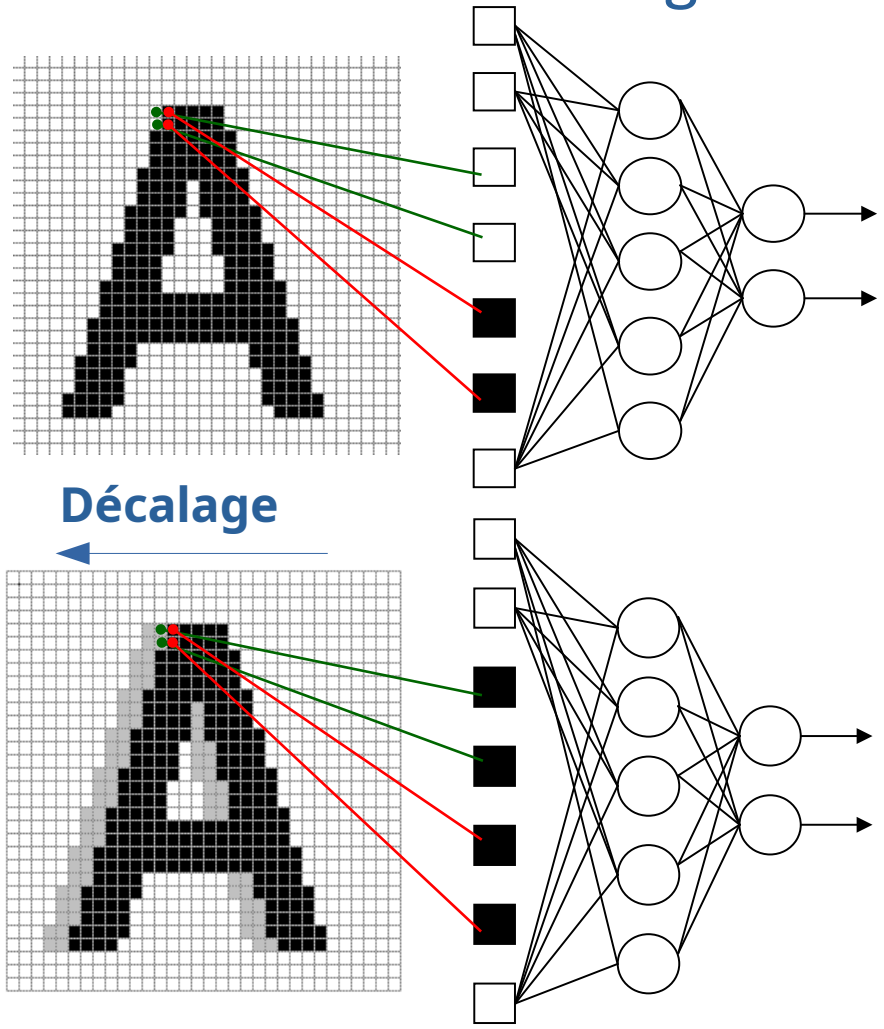
Problèmes

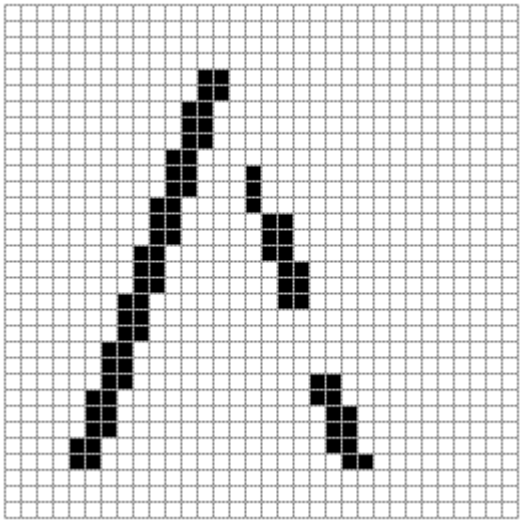




- La topologie est complétement ignorée
- Le nombre de neurones nécessaires peut devenir très important, donc le temps d'apprentissage aussi.

Peu ou pas d'invariance aux translations, rotations, changement d'échelle....





154 changements dans l'image pour 2 pixels de décalage :

77 : blanc vers noir

77 : noir vers blanc

Bilan

Un réseau complétement connecté de taille suffisante peut “apprendre” à être invariant à ces déformations.

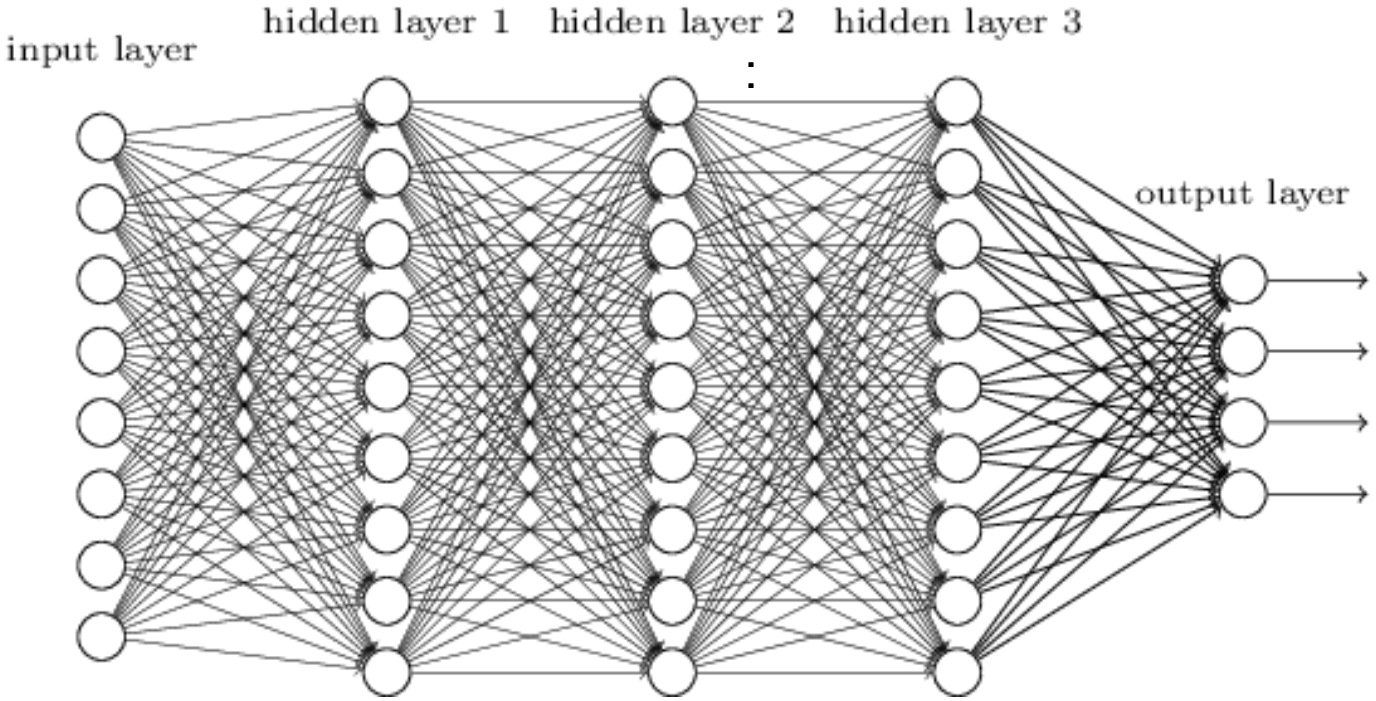
Mais :

- **Temps d’apprentissage**
- **Taille du réseau**
- **Nombre de paramètres importants**

De nouveau un ralentissement des recherches pendant 10 ans...

Convolutional Neural Networks (réseaux convolutionnels)

- Un "bon" modèle est le plus simple possible...
- A t-on réellement besoin de toutes les connexions dans un réseau ?
- Est-ce qu'on ne peut pas partager des poids ?



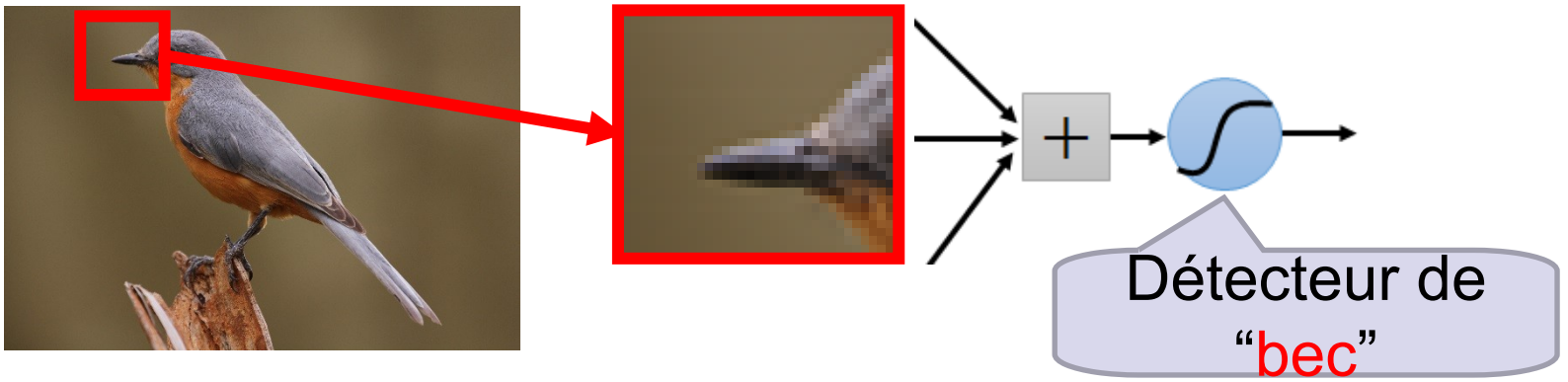
Historique

Yann LeCun, Professor of Computer Science
New York University

En 1995, Yann LeCun et Yoshua Bengio ont introduit le concept de réseaux neuronaux convolutionnels mais explosion des recherches/applications à partir de 2006

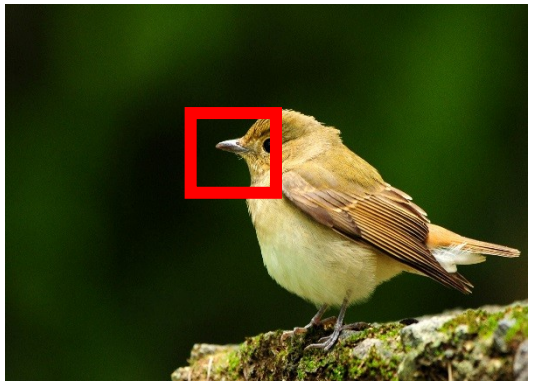
Exemple sur une image:

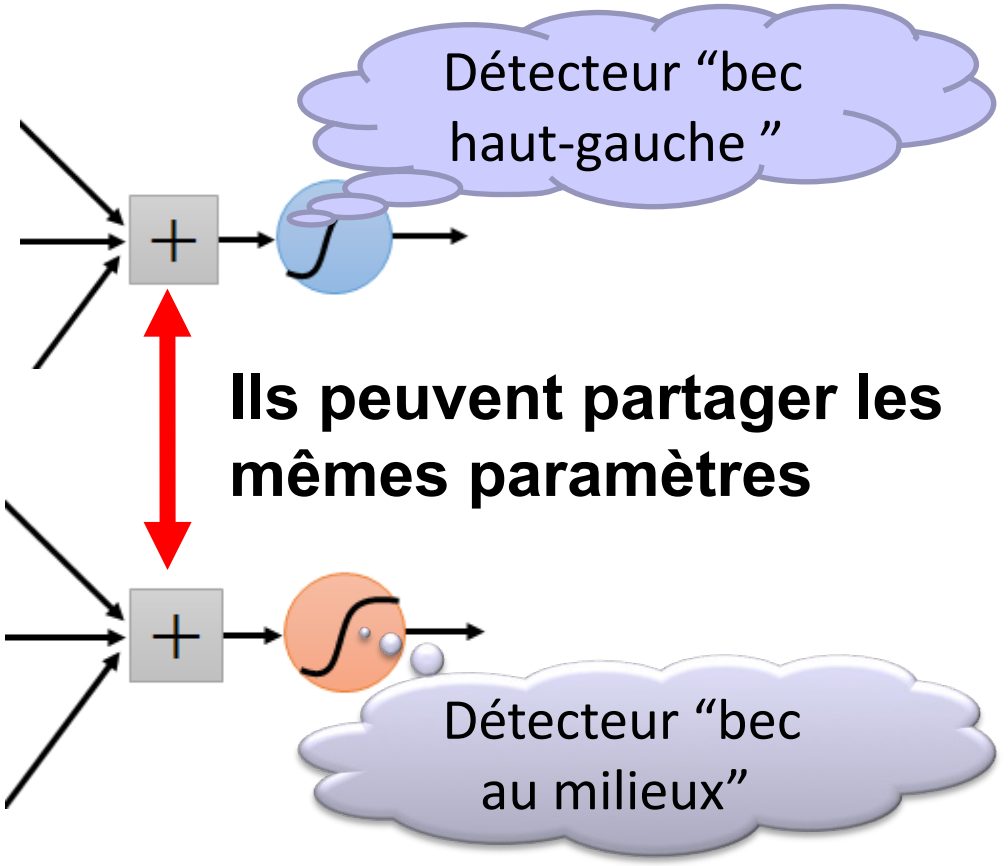
- Certains motifs sont relativement petits
  - On peut représenter une petite région avec moins de paramètres (!?)



Ceratins motifs peuvent apparaitre à différentes positions : ils peuvent être partagés

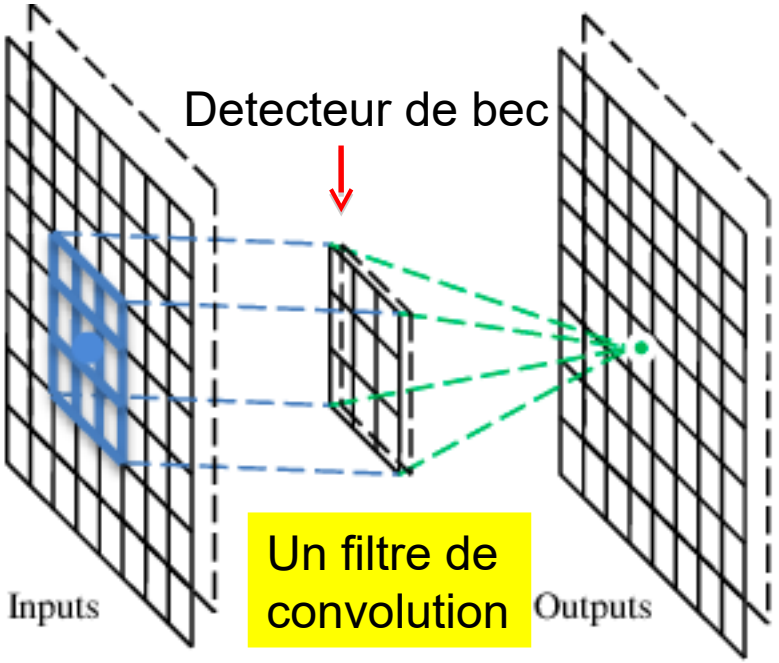
Idée : entrainer des “petits” détecteurs et leur faire parcourir l’image





Une couche convolutionnelle

Un réseau convolutionnel (**CNN - Convolutional Neural Network**) est un réseau de neurones dont certaines couches effectuent des convolutions.



Convolution

Le réseau apprend les paramètres de ces filtres.

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

Image d'entrée
6 x 6
(juste pour l'exemple !)

1	-1	-1
-1	1	-1
-1	-1	1

Filtre 1

-1	1	-1
-1	1	-1
-1	1	-1

Filtre 2

⋮ ⋮

Chaque filtre détecte un motif (3 x 3).

Convolution

Pas=1 (stride)

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

Image 6 x 6

convolution

1	-1	-1
-1	1	-1
-1	-1	1

Filtre 1

Remarque : le masque de « convolution » est utilisé comme une corrélation (i.e. pas de miroir du masque)

Convolution

Si pas=2

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

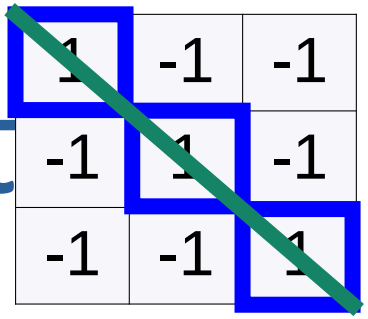
Image 6 x 6

1	-1	-1
-1	1	-1
-1	-1	1

Filtre 2

3 -3

Convolut



Filtre 1

pas=1

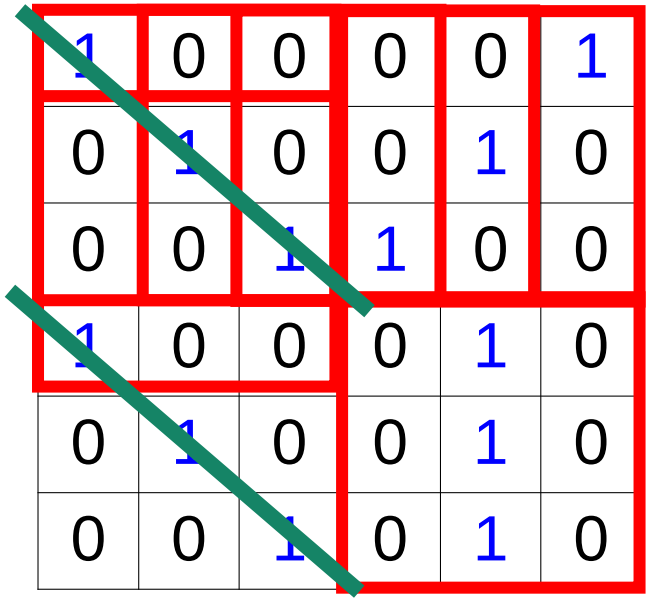
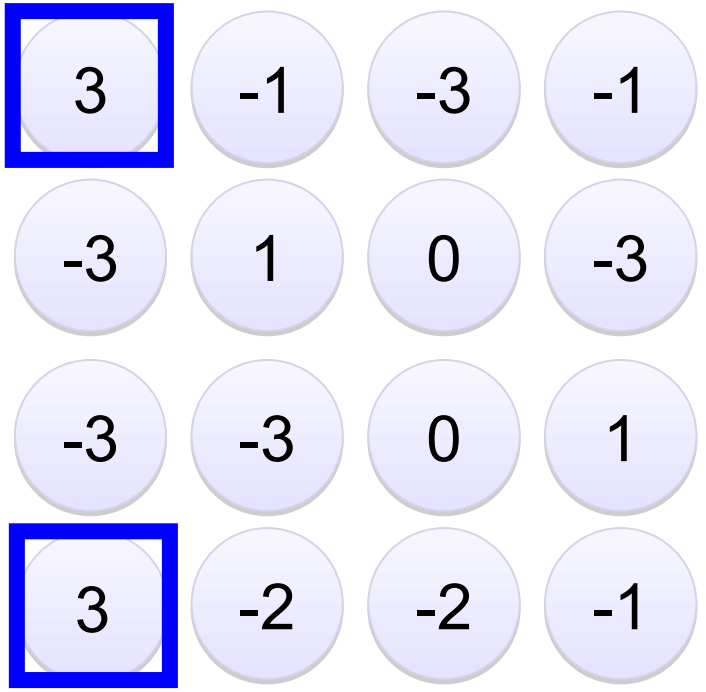


Image 6 x 6



Le motif présent dans le filtre (trait diagonal) correspond aux réponses les plus fortes

Convoluti

-1	1	-1
-1	1	-1
-1	1	-1

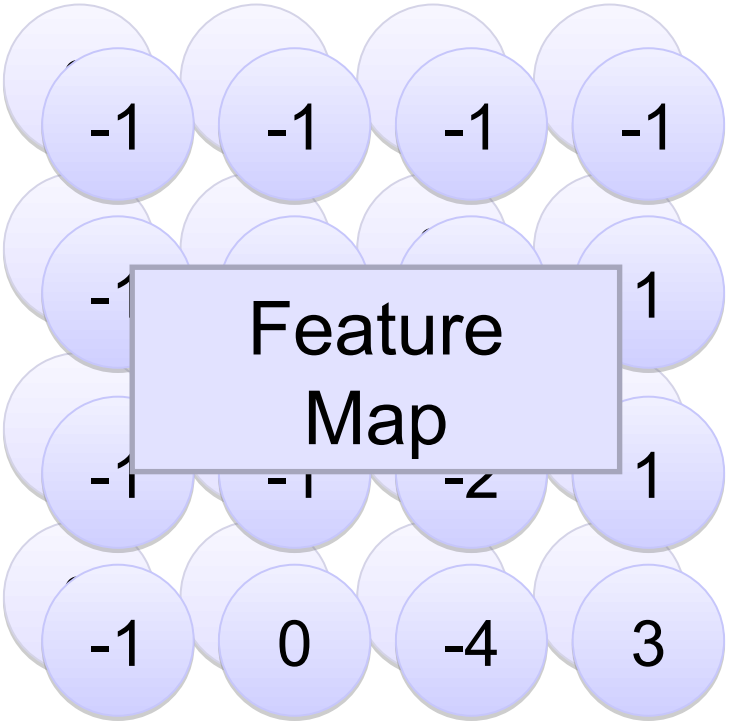
Filtre 2

Pas=1

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

Image 6 x 6

Répéter pour chaque filtre.



Deux images 4 x 4
donnant une matrice 2 x 4 x 4

Image couleur: 3 canaux RGB

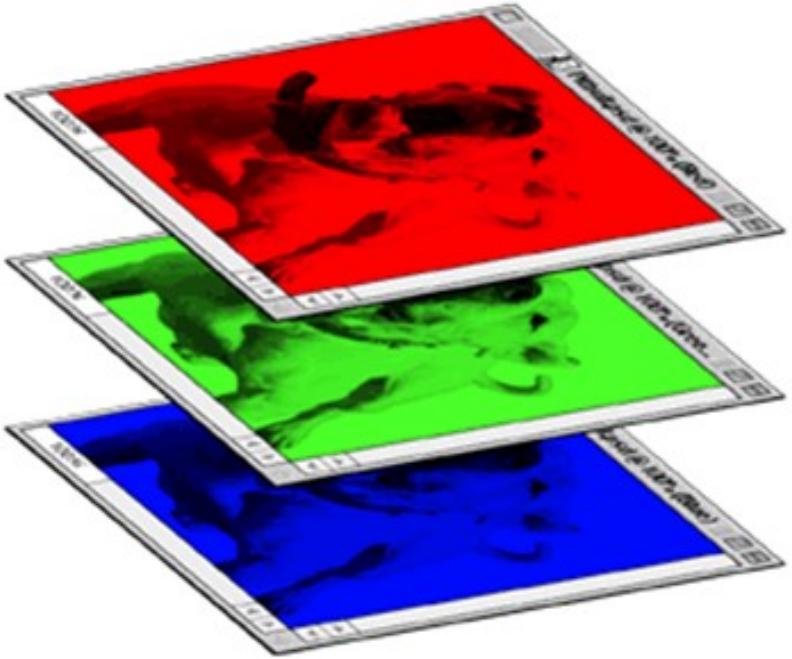
1	-1	-1
-1	1	-1
-1	-1	1

Filtre 1

-1	1	-1
-1	1	-1
-1	1	-1

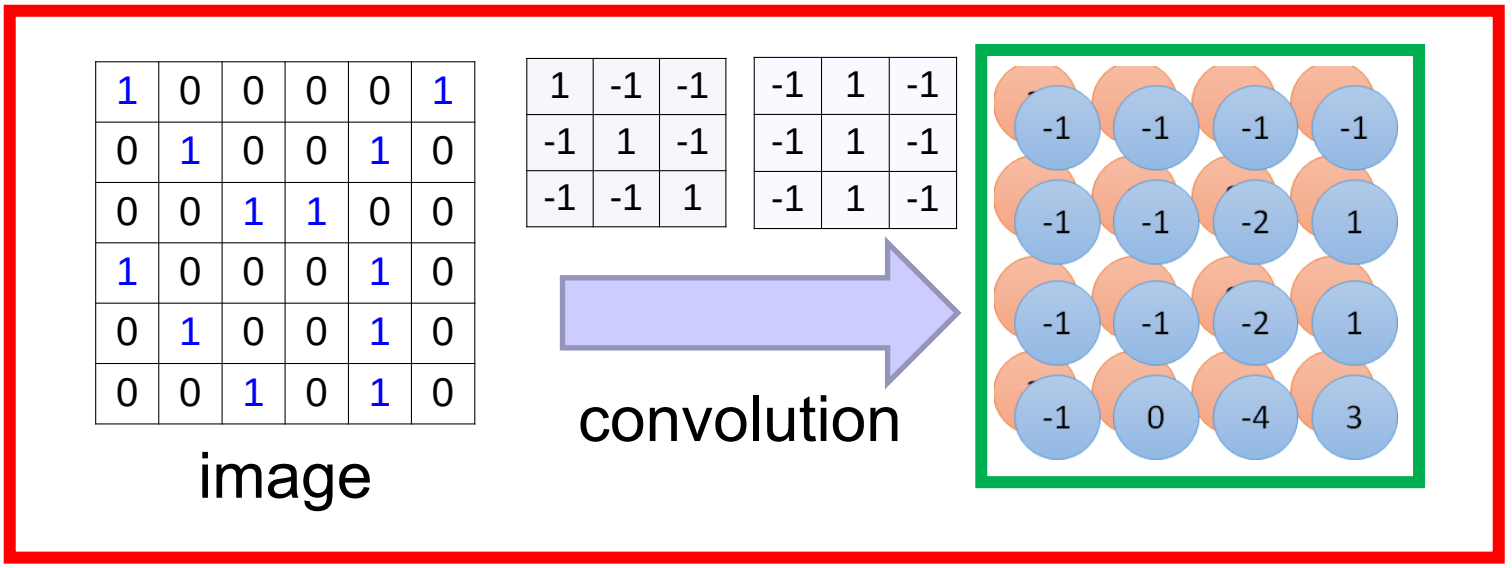
Filtre 2

Image couleur

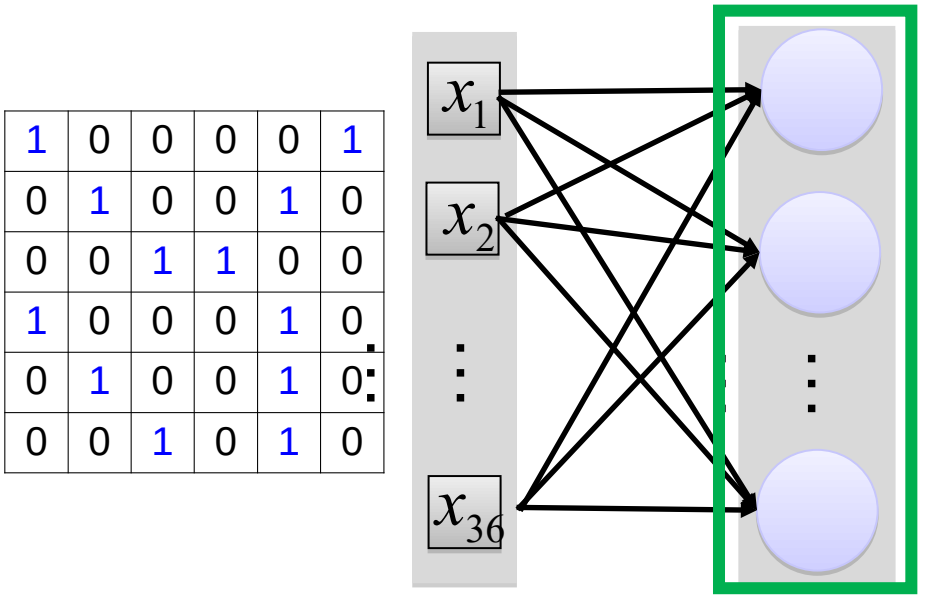


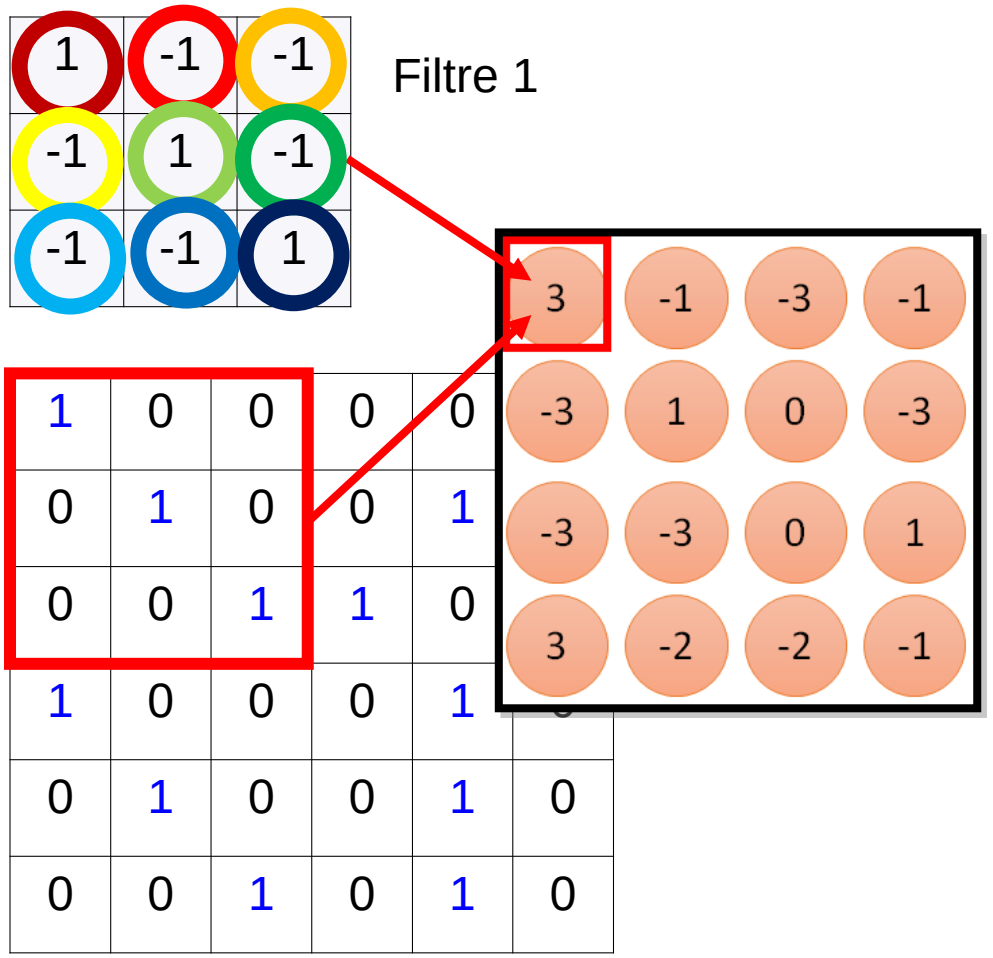
1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

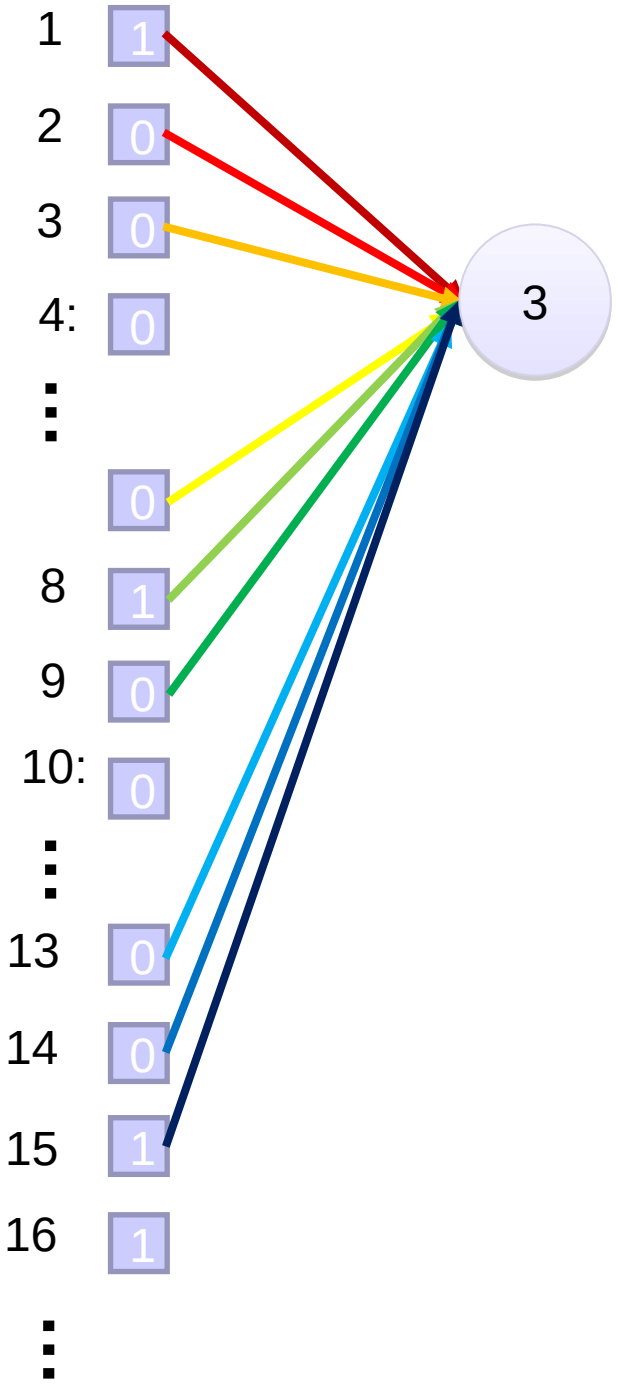
Convolution v.s. totalement connecté



Couche dense (totalement connectée)
Perceptron Multi-couches

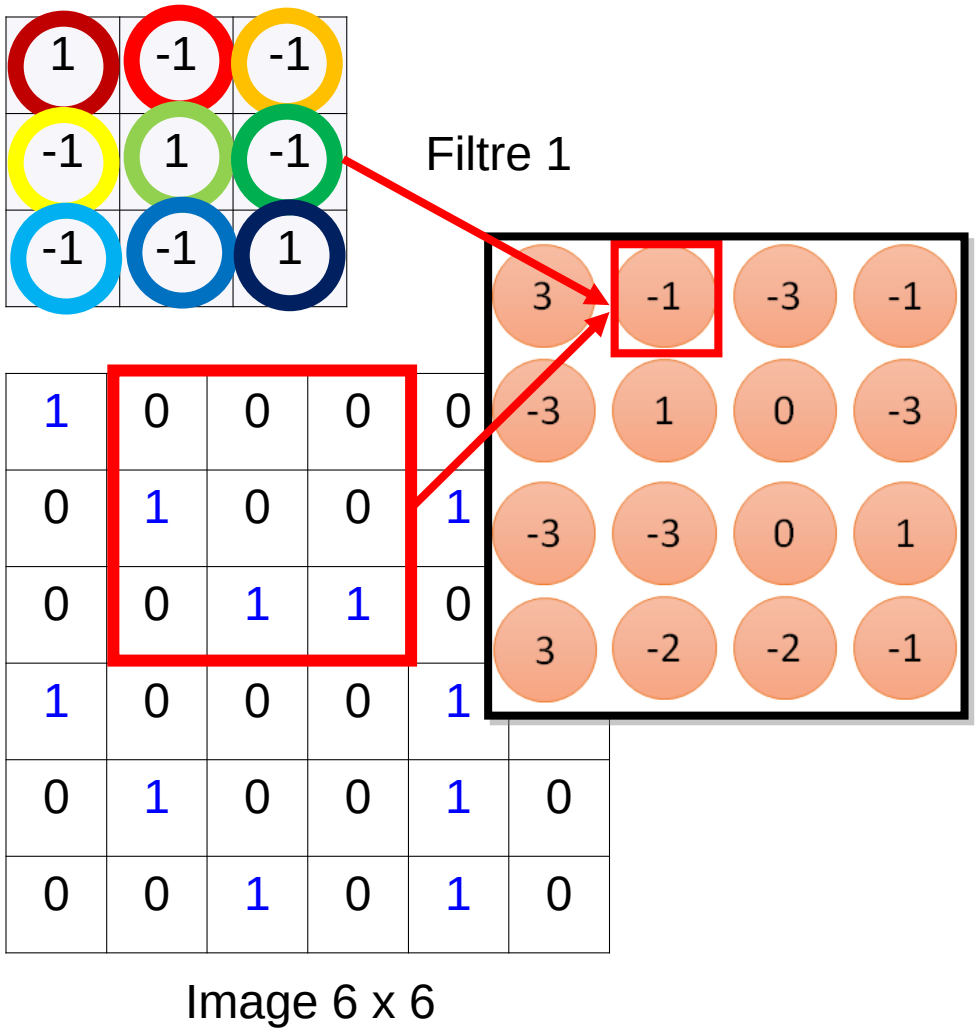


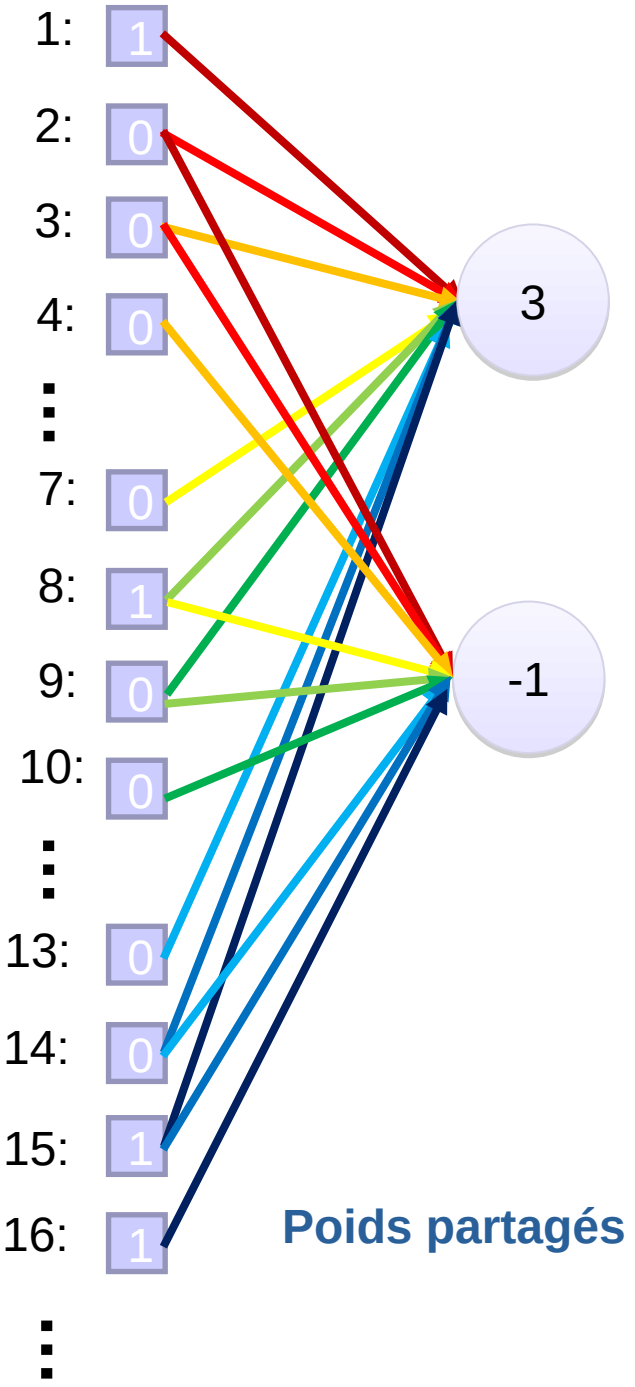




Moins de paramètres
(car partagés)

Connecté à 9 entrées ≠ dense

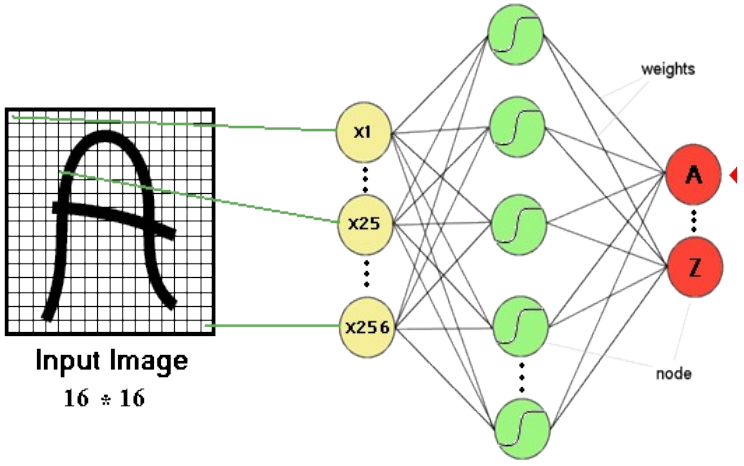


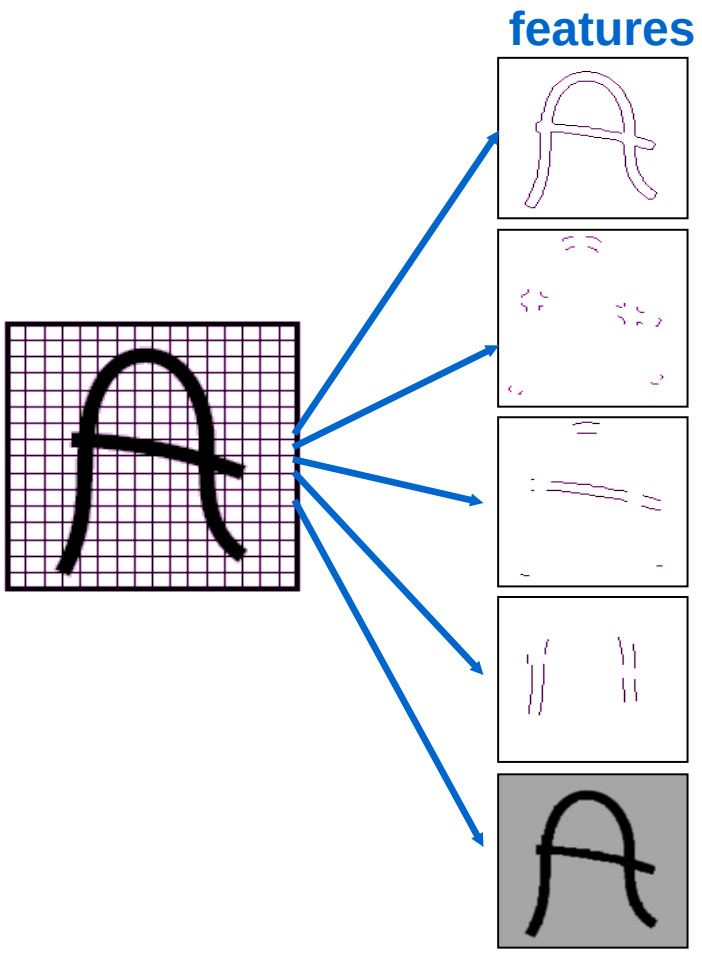


Encore moins de paramètres

Poids partagés

Détecte les mêmes attributs à des positions différentes dans l'image





Un CNN « complet »

chat chien ……

Convolution

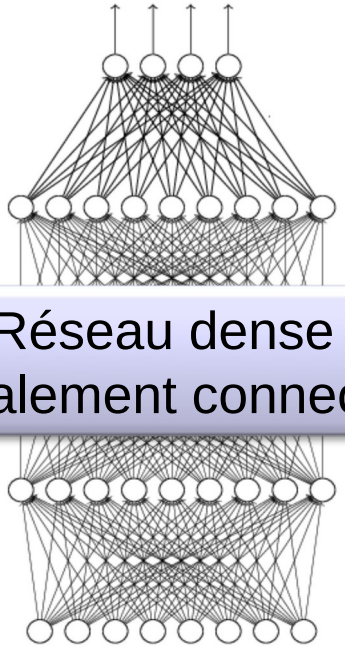
Max Pooling

Convolution

Max Pooling

Eventuellement répété

Flattened



Réseau dense
Totalement connecté

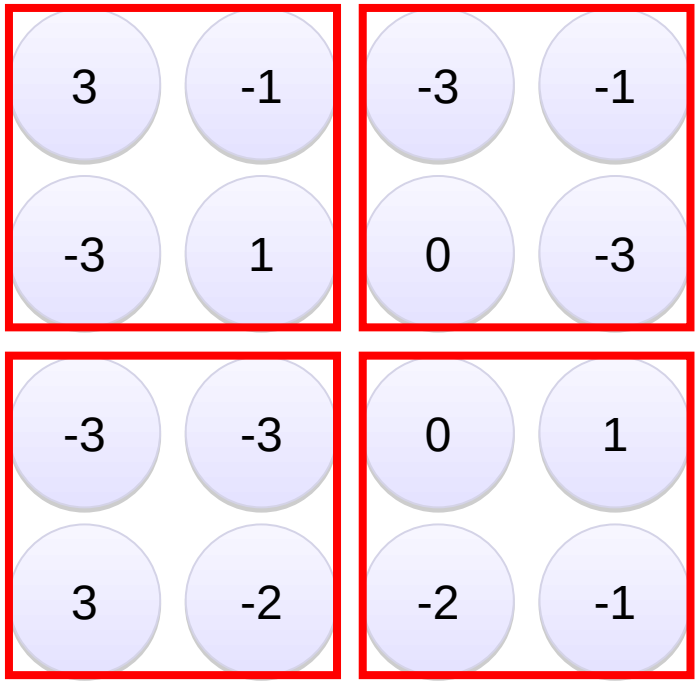
Max Pooling

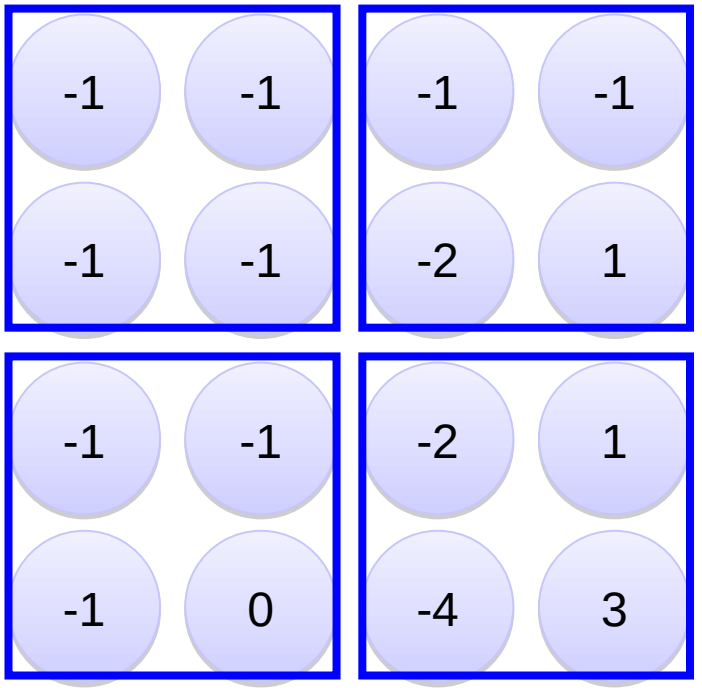
1	-1	-1
-1	1	-1
-1	-1	1

Filtre 1

-1	1	-1
-1	1	-1
-1	1	-1

Filtre 2





Pourquoi ce vote ?

- Le sous-échantillonnage ne change pas les objets.

oiseau

Sous échantillonnage

oiseau

Moins de paramètres pour caractériser les images

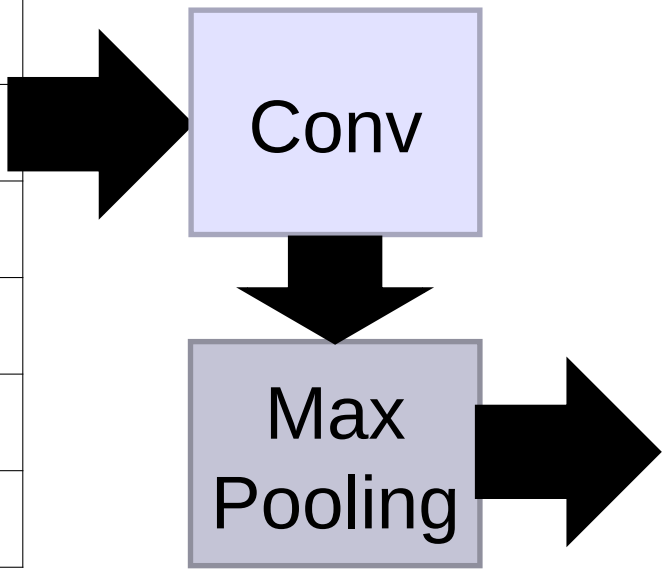
Un CNN “compresse” un réseau dense

- Réduction du nombre de connexions
- Partage des poids
- Le Max pooling réduit encore la complexité

Max Pooling

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

Image 6 x 6



Nouvelle image réduite

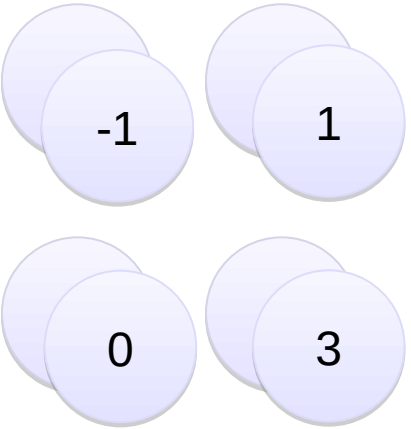


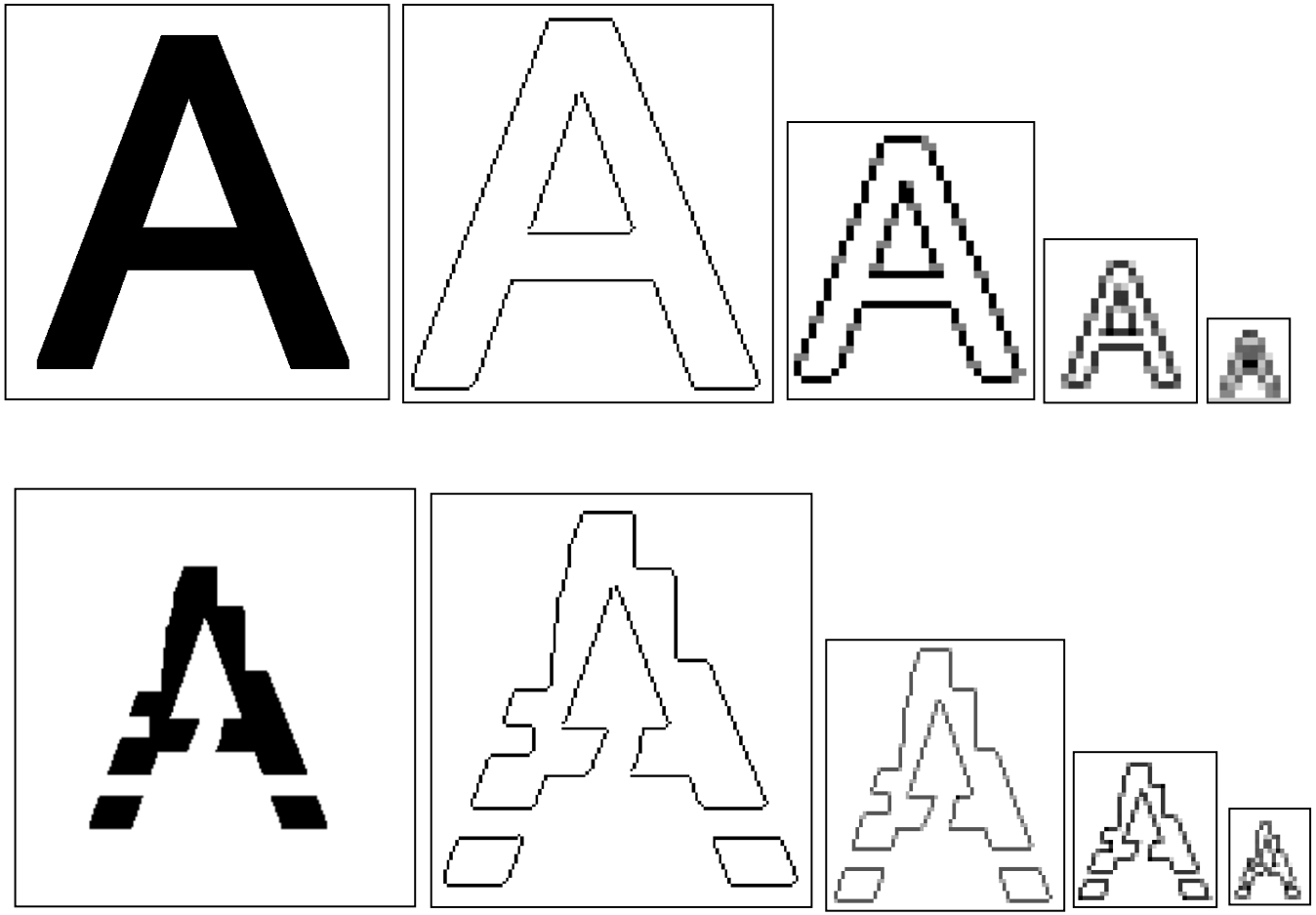
Image 2 x 2

Chaque filtre produit un canal

Sous échantillonnage

- Réduit la résolution spatiale de chaque feature map

- Permet d'obtenir une relative invariance à la distortion



Un CNN « complet »

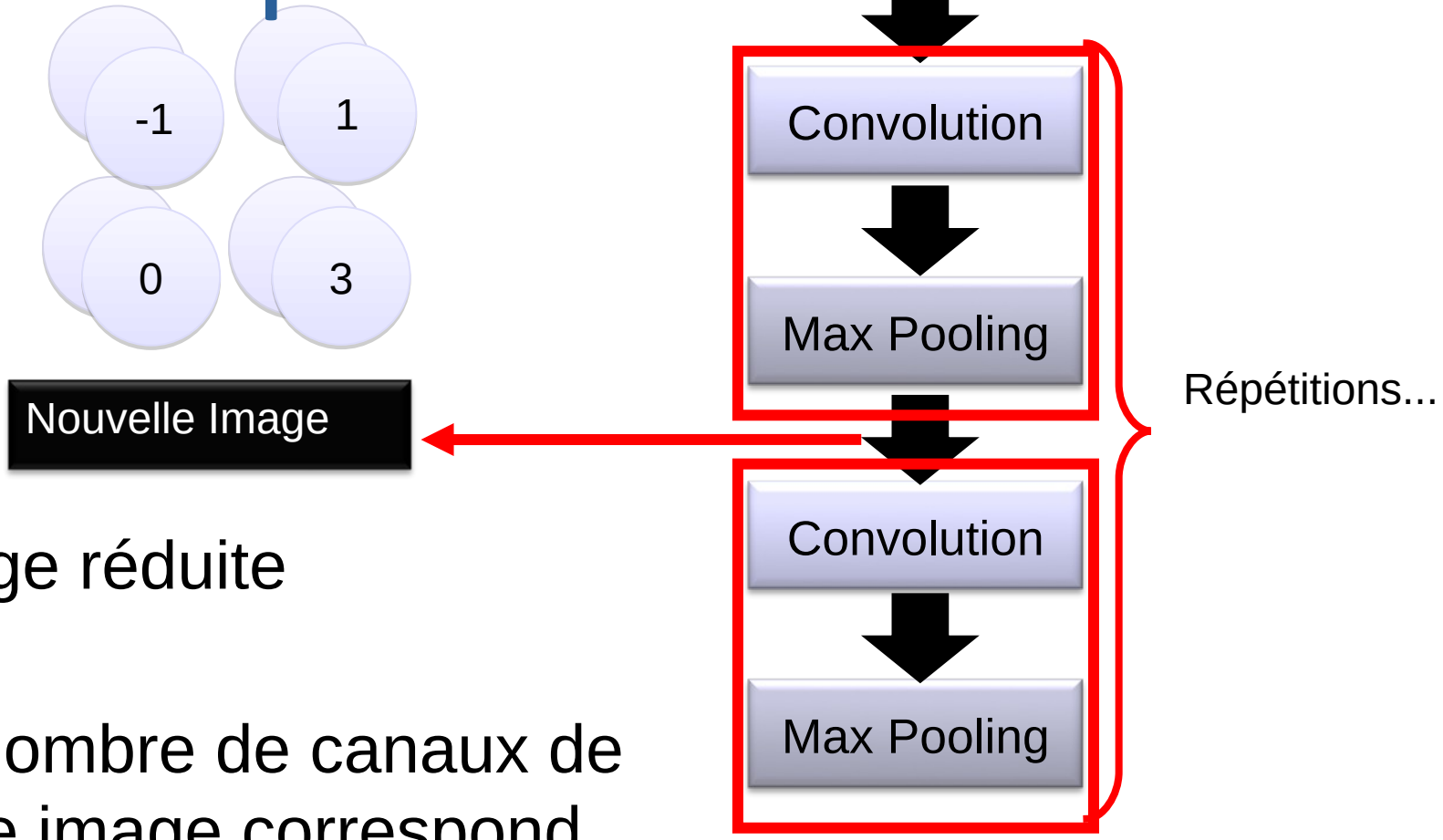
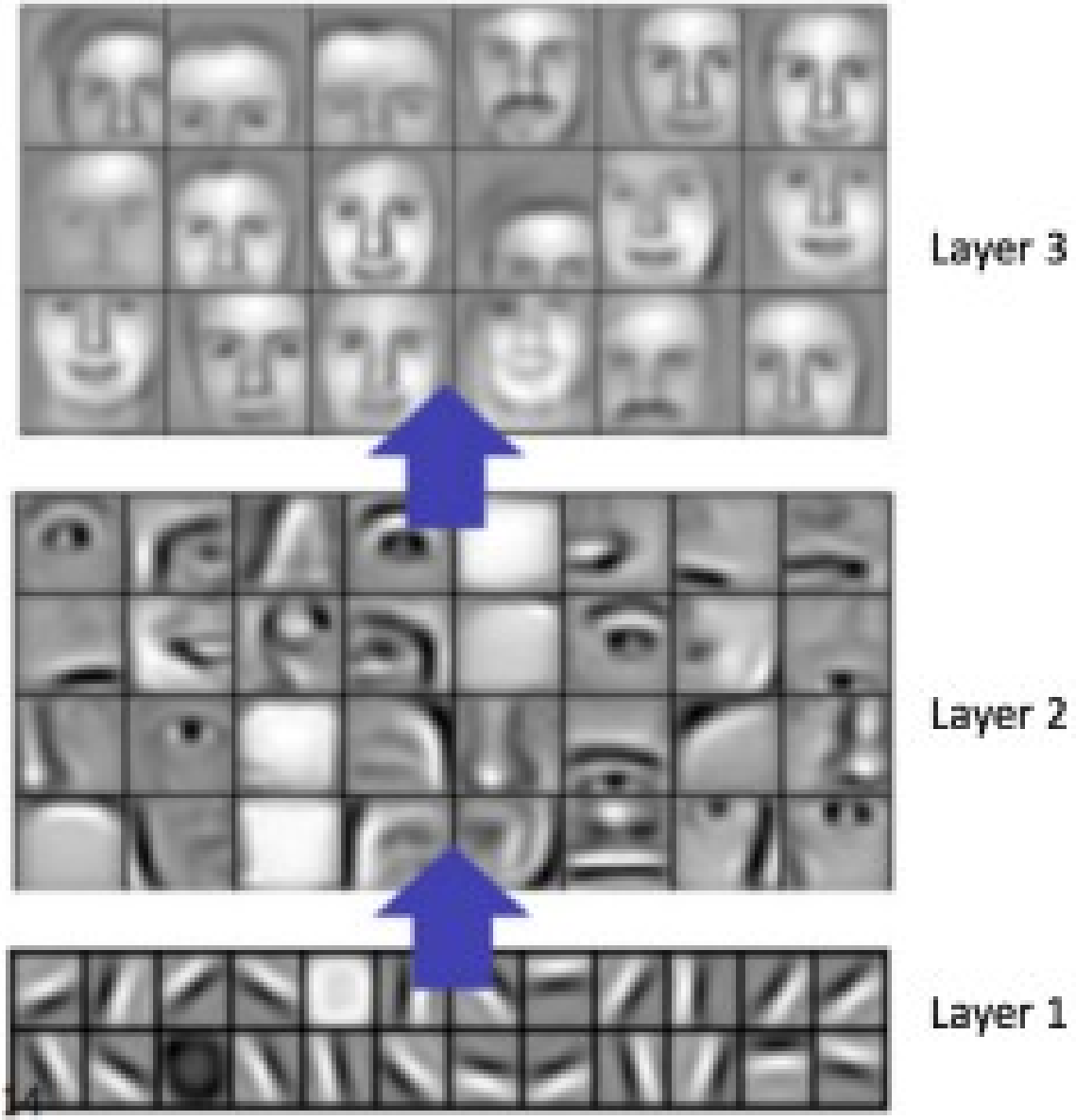


Image réduite

Le nombre de canaux de cette image correspond au nombre de filtres


Layer 3
Layer 2
Layer 1

Un CNN « complet »

Chat Chien….

Convolution

Max Pooling

Nouvelle image

Convolution

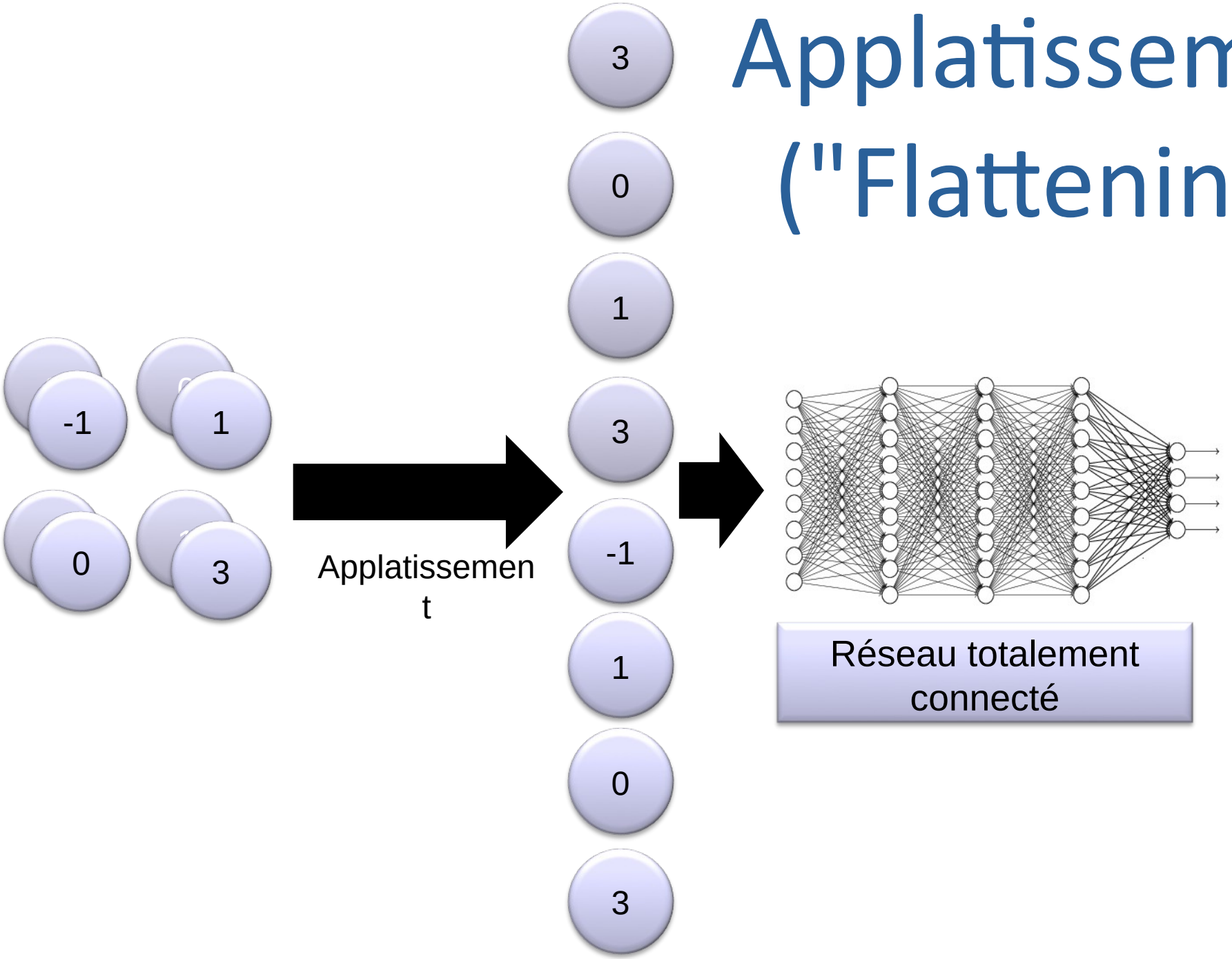
Max Pooling

Nouvelle image

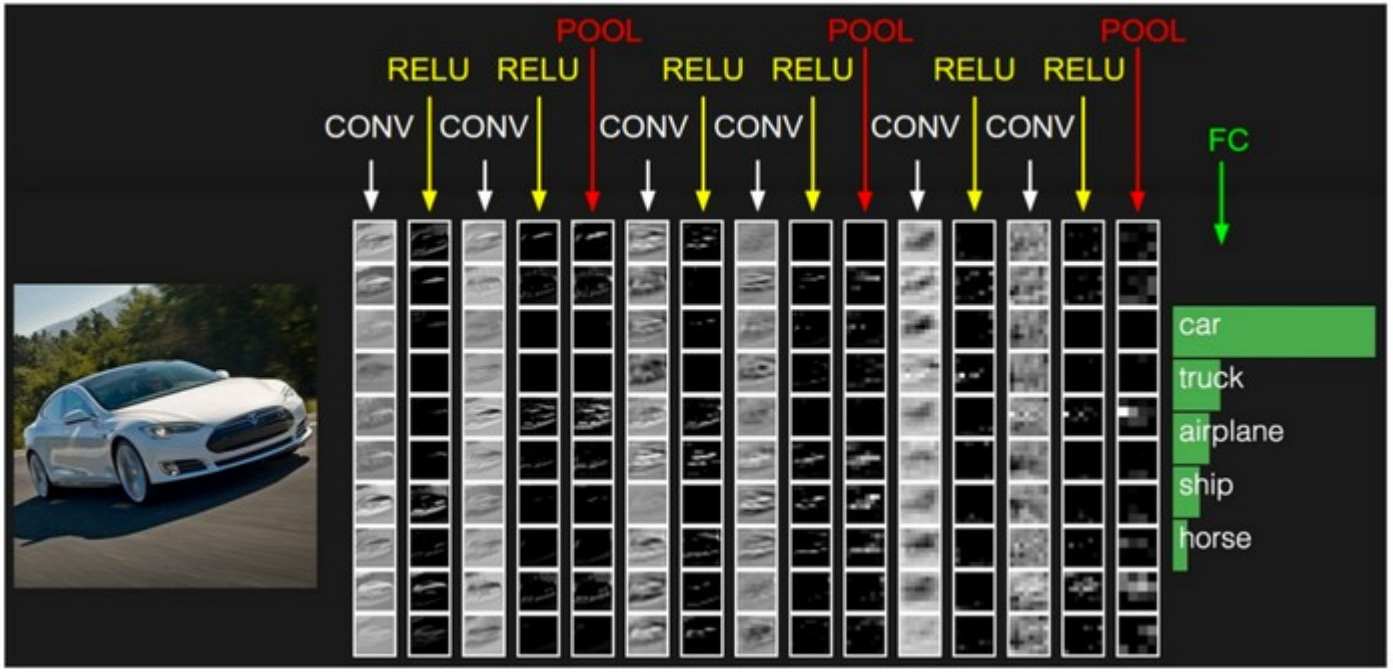
Applatissement

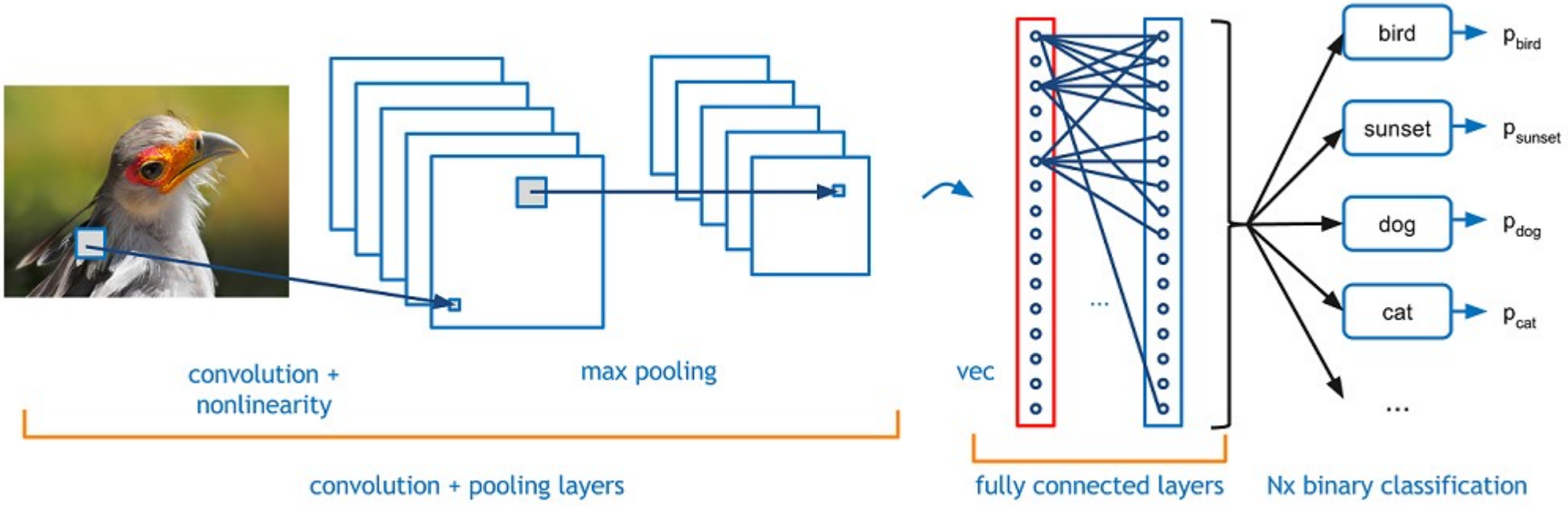
Réseau totalement connecté

Applatissement ("Flattening")



Visualisation



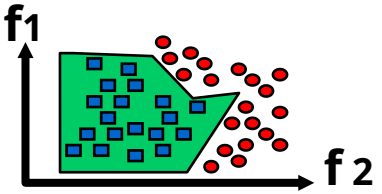

convolution +
nonlinearity
max pooling
vec
bird
sunset
dog
cat
...
p_{bird}
p_{sunset}
p_{dog}
p_{cat}
convolution + pooling layers
fully connected layers
Nx binary classification

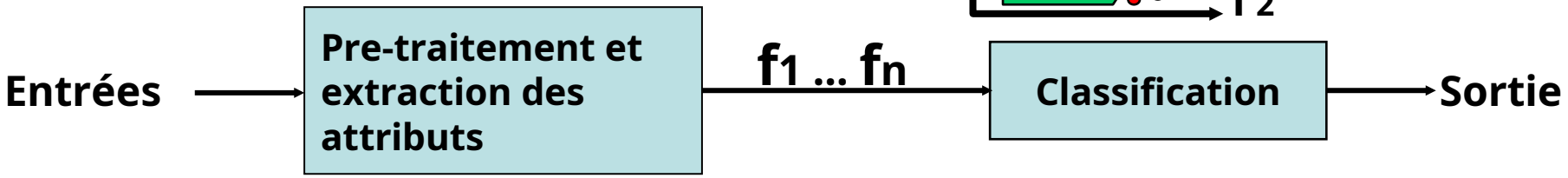
CNN

- CNN est de type feed-forward
- Inspirés par la biologie
- Repose sur la rétropropagation.
- Extraction automatique des attributs (≠ Haar par exemple)
- Reconnaissance de motifs visuels avec peu de prétraitements.
- Robustes aux changements (ex: variabilité des scripteurs).

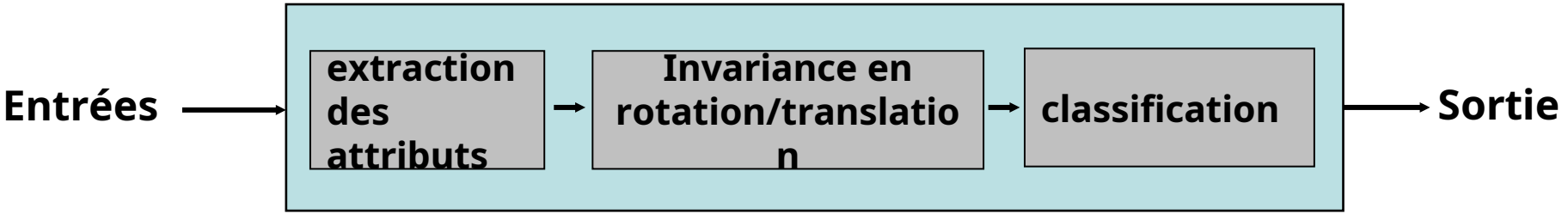
Classification

Classification





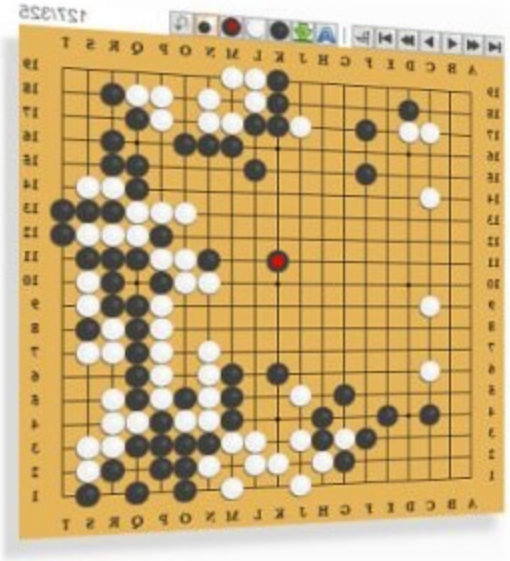
Convolutional neural network



In "Nature" 27 January 2016:

- "DeepMind's program AlphaGo beat Fan Hui, the European Go champion, five times out of five in tournament conditions..."
- "AlphaGo was not preprogrammed to play Go: rather, it learned using a general-purpose algorithm that allowed it to interpret the game's patterns."
- "...AlphaGo program applied **deep learning** in neural networks (convolutional NN) — brain-inspired programs in which connections between layers of simulated neurons are strengthened through examples and experience."

AlphaGo



Matrice 19 x 19

CNN

Mouvement suivi

Noir: 1

Blanc: -1

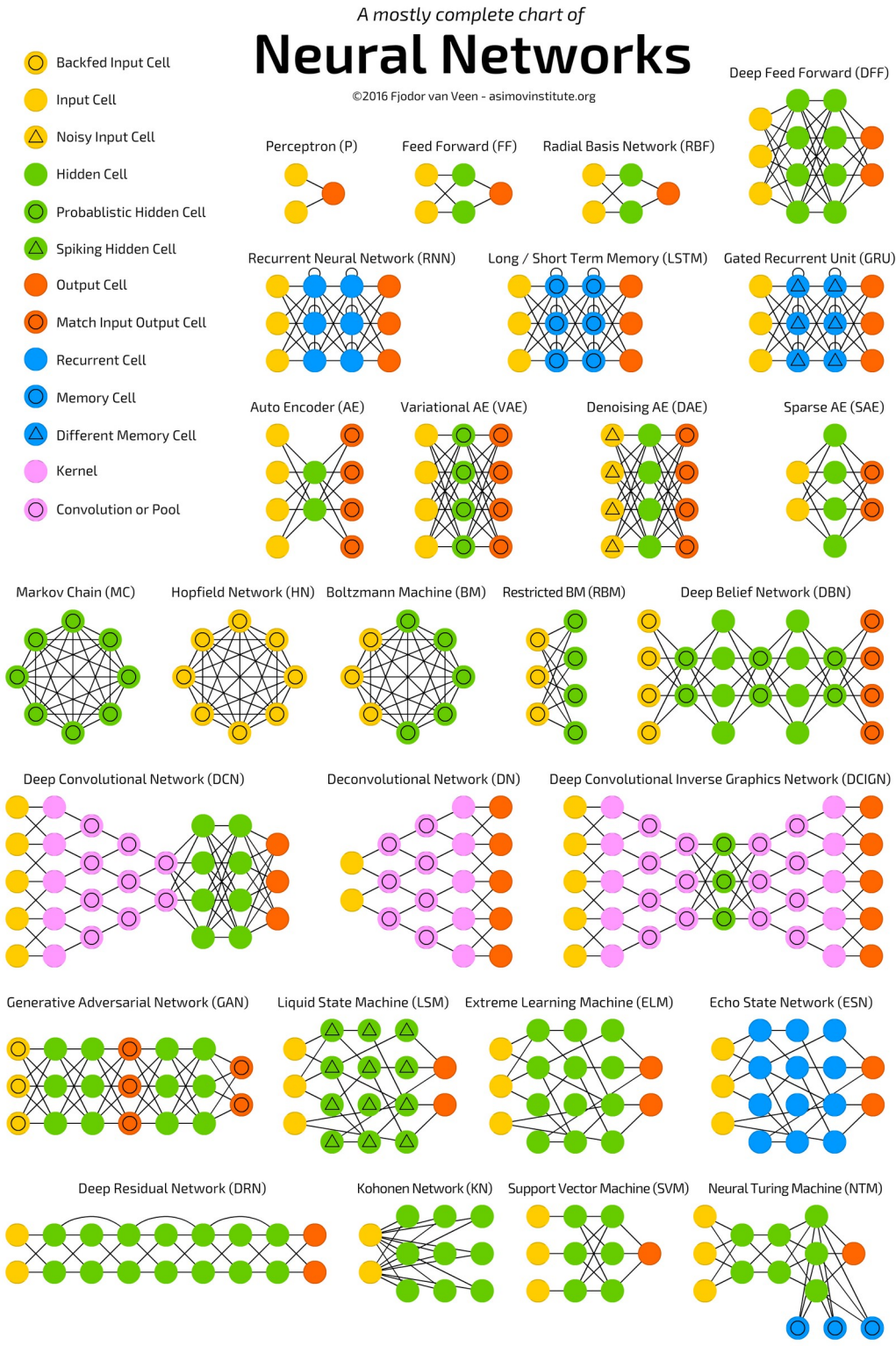
Vide: 0

L'histoire n'est pas finie !
Un humain entrainé sur les « conseils » d'une IA bat AlphaGo

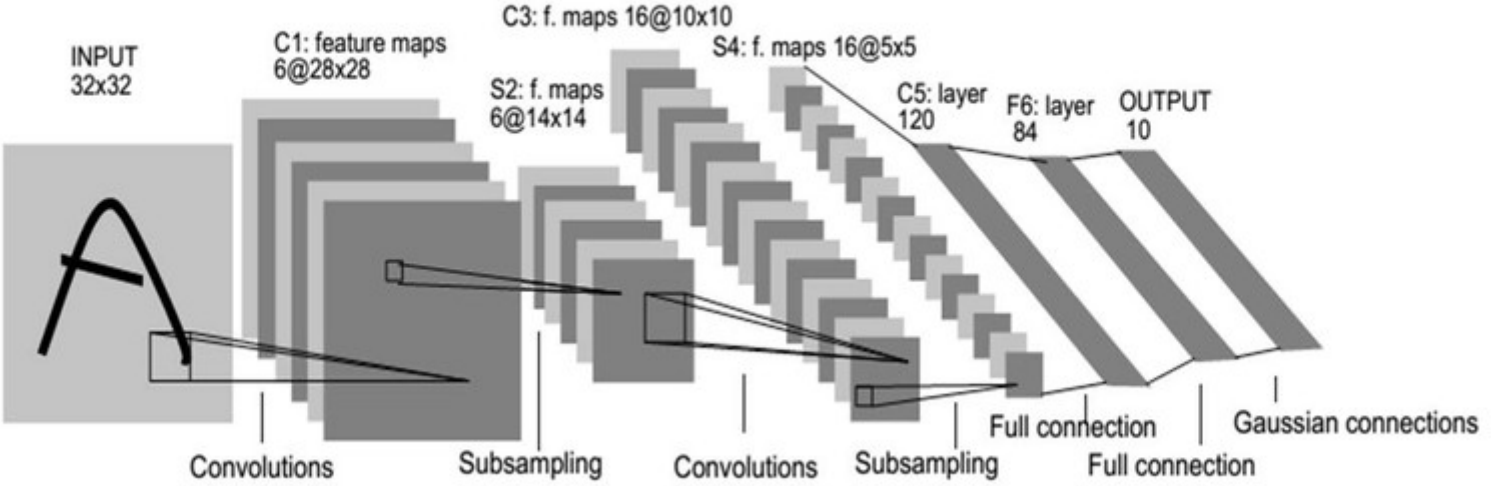
https://www.20minutes.fr/high-tech/4024717-20230221-intelligence-artificielle-perdu-jeu-go-face-humain-aide-ia

Le Zoo !

http://www.asimovinstitute.org/neural-network-zoo/



LeNet-5 (Y. LeCun - 1998)



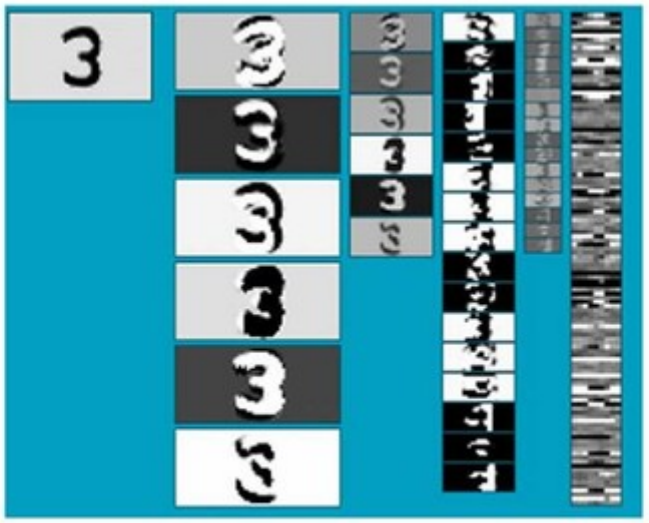


Figure from ‘
Recognition”, Y. Lecun et al Proc. IEEE, 1998 copyright 1998, IEEE

Fig. 4. Size-normalized examples from the MNIST database.

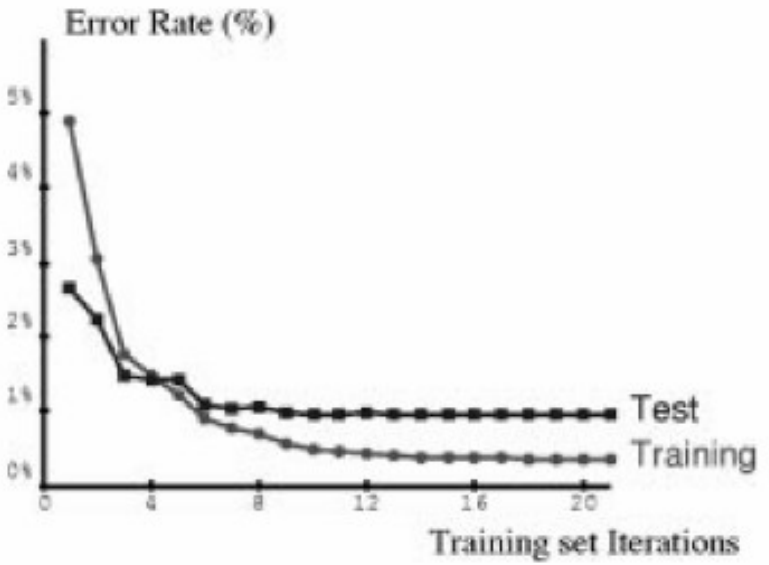


Fig. 5. Training and test error of LeNet-5 as a function of the number of passes through the 60 000 pattern training set (without distortions). The average training error is measured on-the-fly as training proceeds. This explains why the training error appears to be larger than the test error initially. Convergence is attained after 10–12 passes through the training set.

LeNet est utilisé pour classer des chiffres manuscripts.

Erreur #1% sur MNIST (60,000 images d'entrainement, 10,000 pour tester)

http://scs.ryerson.ca/~aharley/vis/conv/flat.html

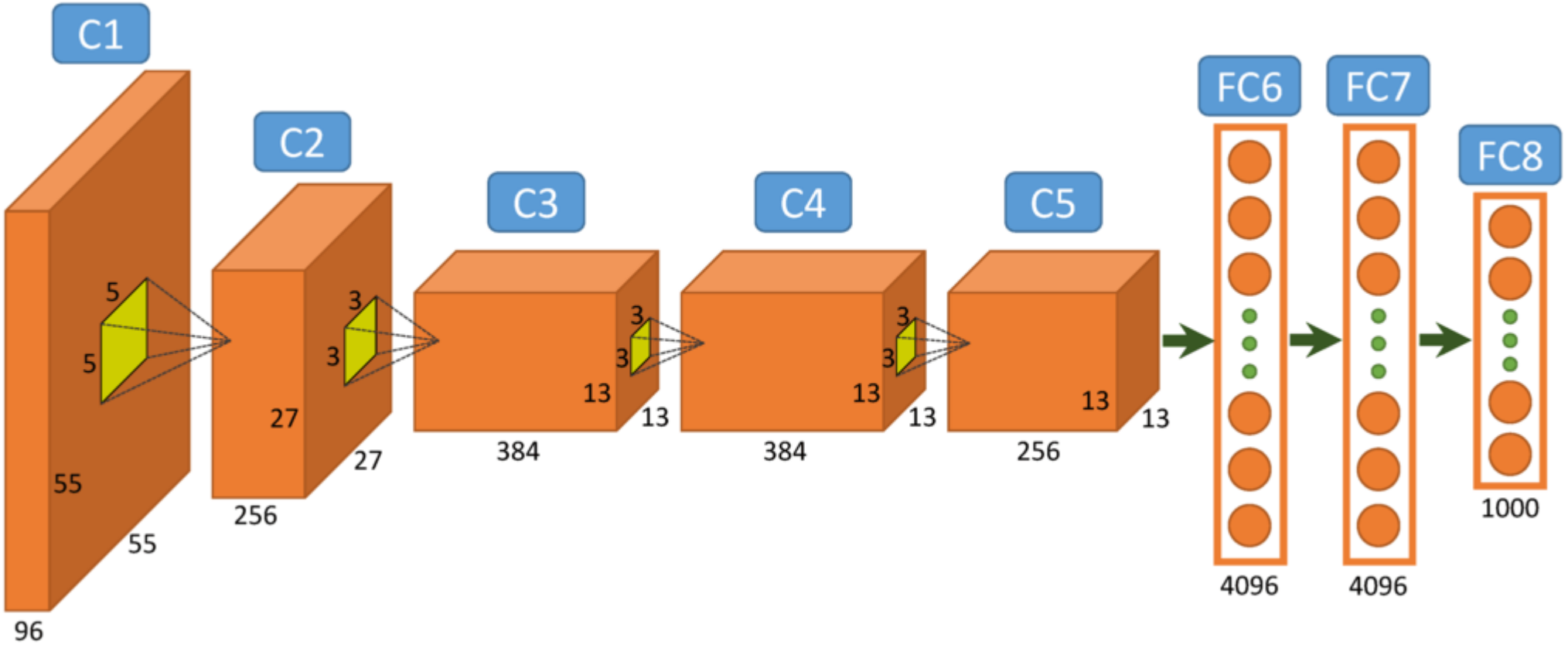
AlexNet (2012)

Remporte le **ImageNet Large Scale Visual Recognition Challenge (ILSVRC)** :

1000 catégories et 1.2 million d'images issues de Flick'r

AlexNet :
15.3% erreur, la seconde méthode (« classique ») est à 26.2% !

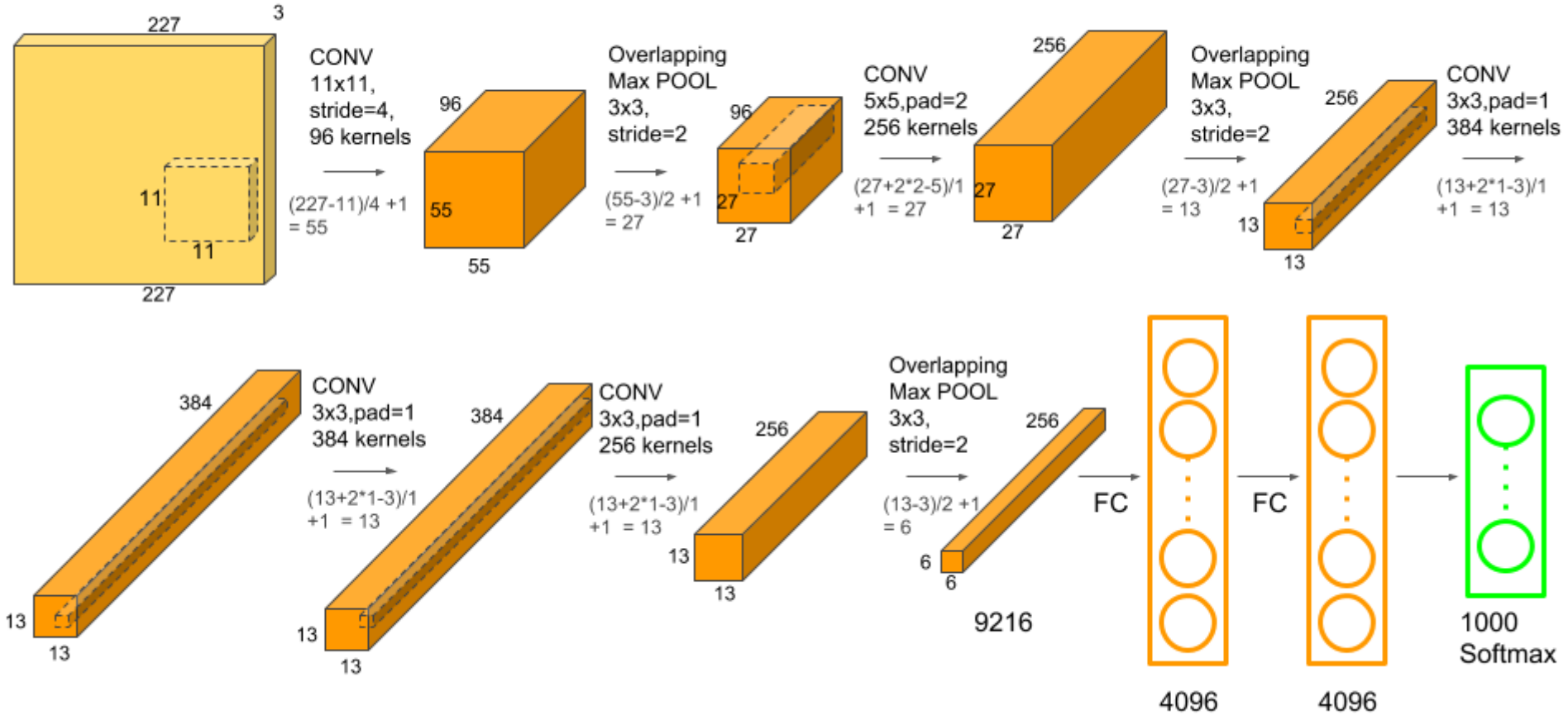
Le Deep devient incontournable !



AlexNet (2012)

Réseau très profond (pour l'époque) :

60 millions de paramètres et 650,000 neurones



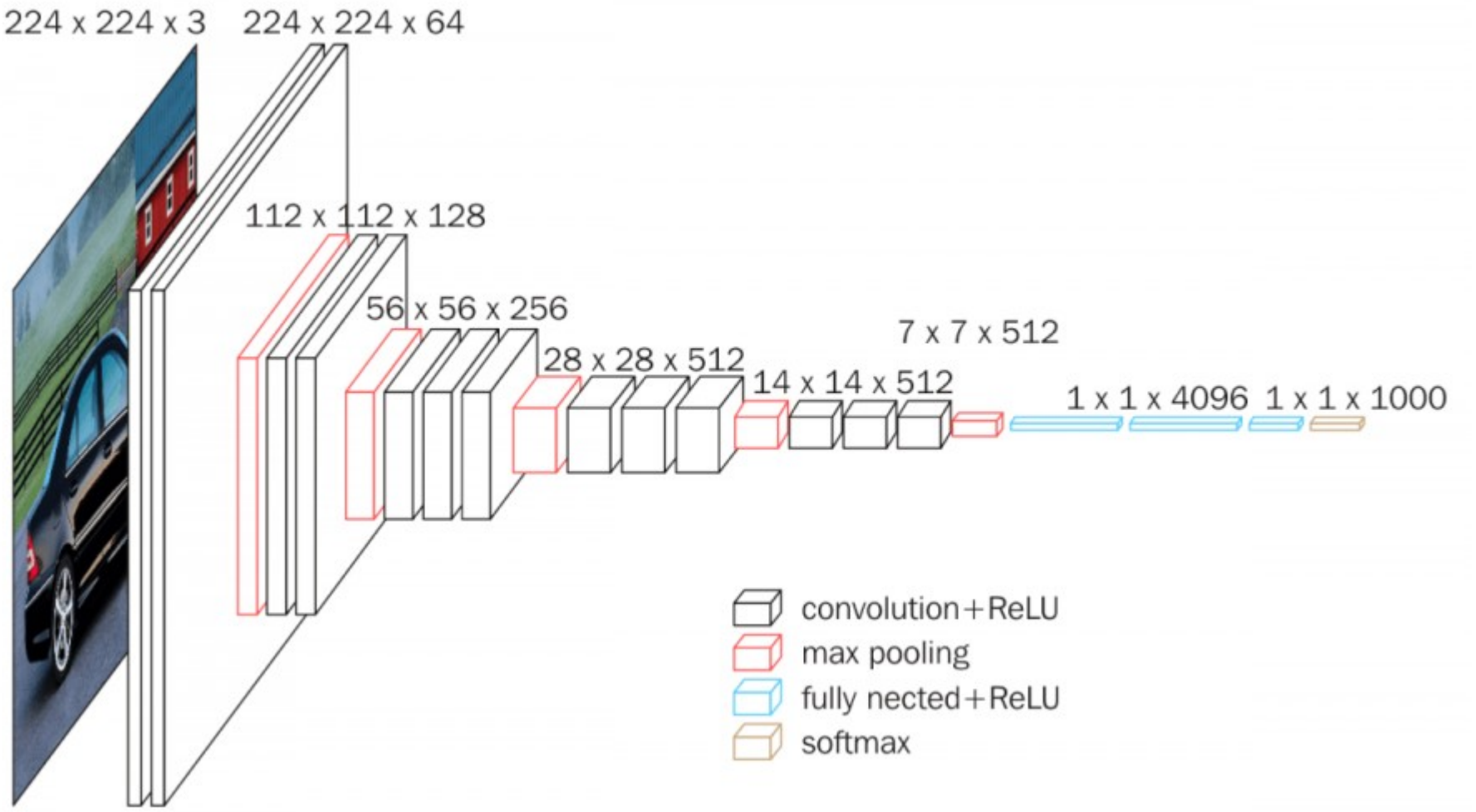
AlexNet

Toutes les images sont retaillées (#256²)

http://mcogswell.io/blog/why_cat_2/#some-features-in-a-hierarchy

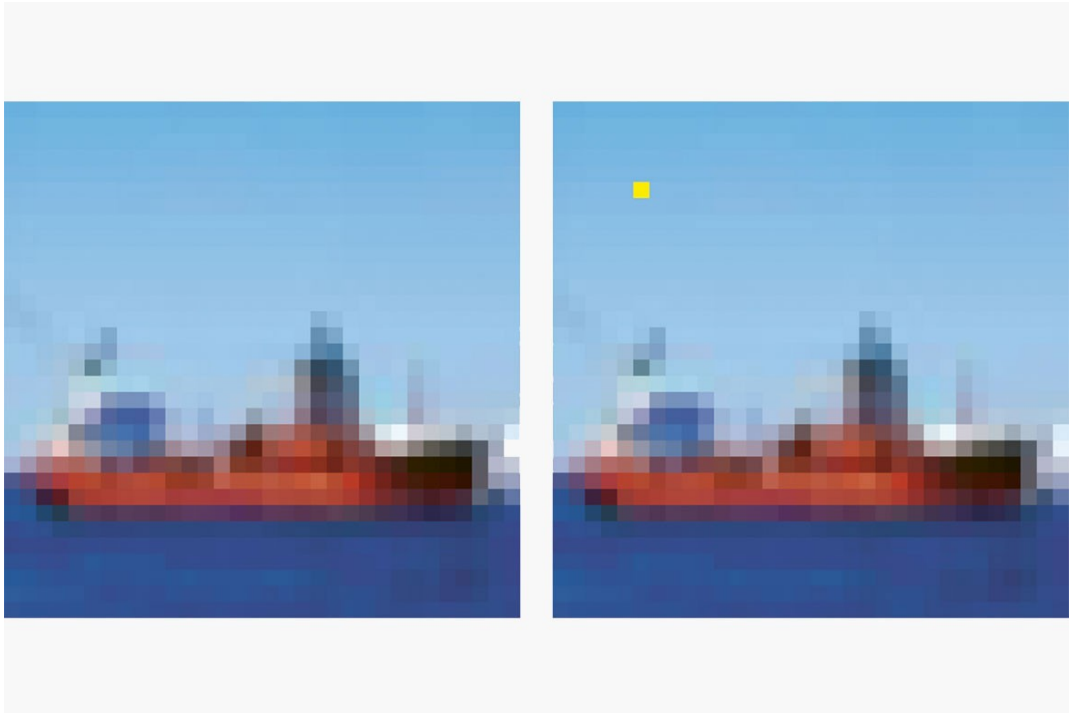
VGG-16 (2014)

Erreur 7,3 %

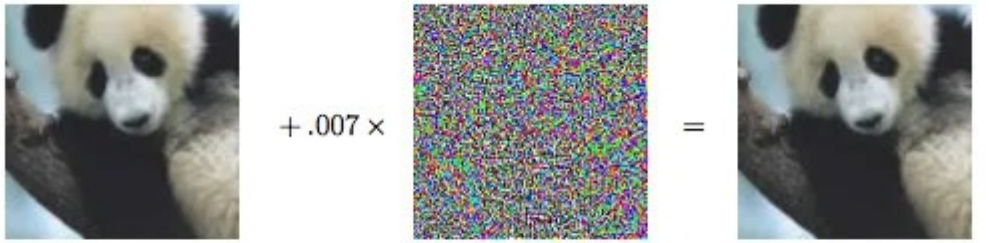


Mais des faiblesses...

La pomme seule est bien reconnue mais avec un texte ajouté elle devient un Ipad !



Un pixel rajouté dans toutes les images « chien » d'une BD entraine le classement de toutes les images avec ce pixel dans la classe « chien ». Danger !

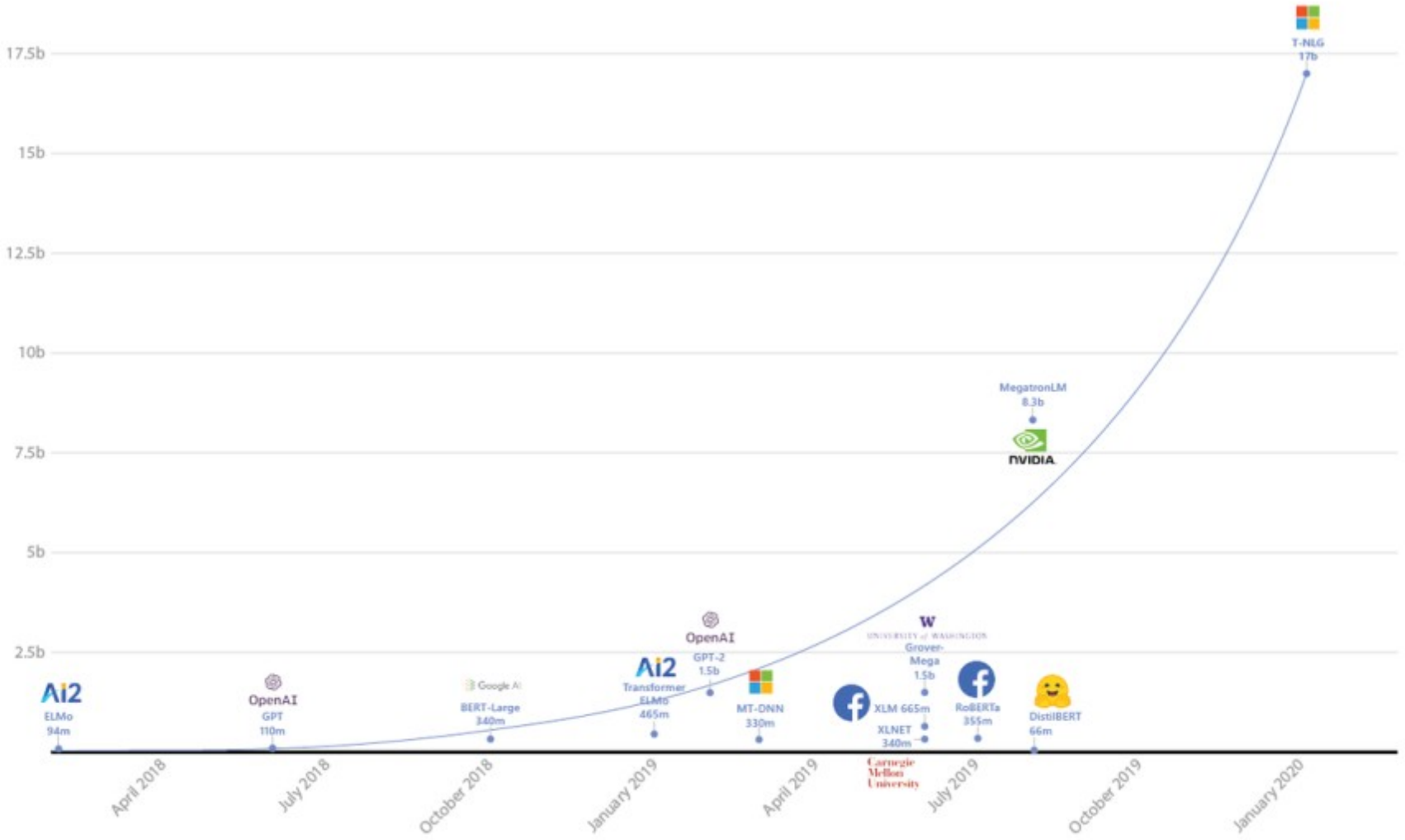


$\boldsymbol{x}$
"panda"
57.7% confidence

$\text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$
"nematode"
8.2% confidence

$\boldsymbol{x} + \epsilon\text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$
"gibbon"
99.3 % confidence

Rajouter un bruit invisible et « bien » calculé dans une image permet de la faire classer dans une classe prédéfinie...



ChatGPT : 175 Milliard de paramètres, 4 mois d'entrainement, des téraoctets (10^{12} octets) de données → exaoctets (10^{18} octets)
GPT4 : 500x plus de paramètres….
estimé en 2024 a 10^{8} paramètres (1 trillion)