

# Réseaux MultiCouches (*Deep Neural Networks*)

*Pierre Chauvet*

Institut de Mathématiques Appliquées

Faculté des Sciences

Université Catholique de l'Ouest

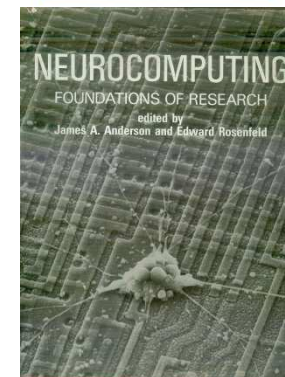


[pierre.chauvet@uco.fr](mailto:pierre.chauvet@uco.fr)



# Quelques références sur le sujet

- *Les Réseaux de Neurones Artificiels* de F. Blayo et M. Verleysen chez Que-sais-je ? (PUF)
- *Neurocomputing Foundations of Research*, J.A. Anderson & E. Rosenfeld, MIT Press
- Le site <http://deeplearning4j.org/index.html>  
Cours/Tutoriaux en ligne (aller en bas de la page d'accueil)
- Gradient-Based Learning Applied to Document Recognition, Y. LeCun, L. Bottou, y. Bengio, P. Haffner, Proc IEEE, Nov 1998,  
<http://yann.lecun.com/exdb/publis/pdf/lecun-01a.pdf>



## PLAN

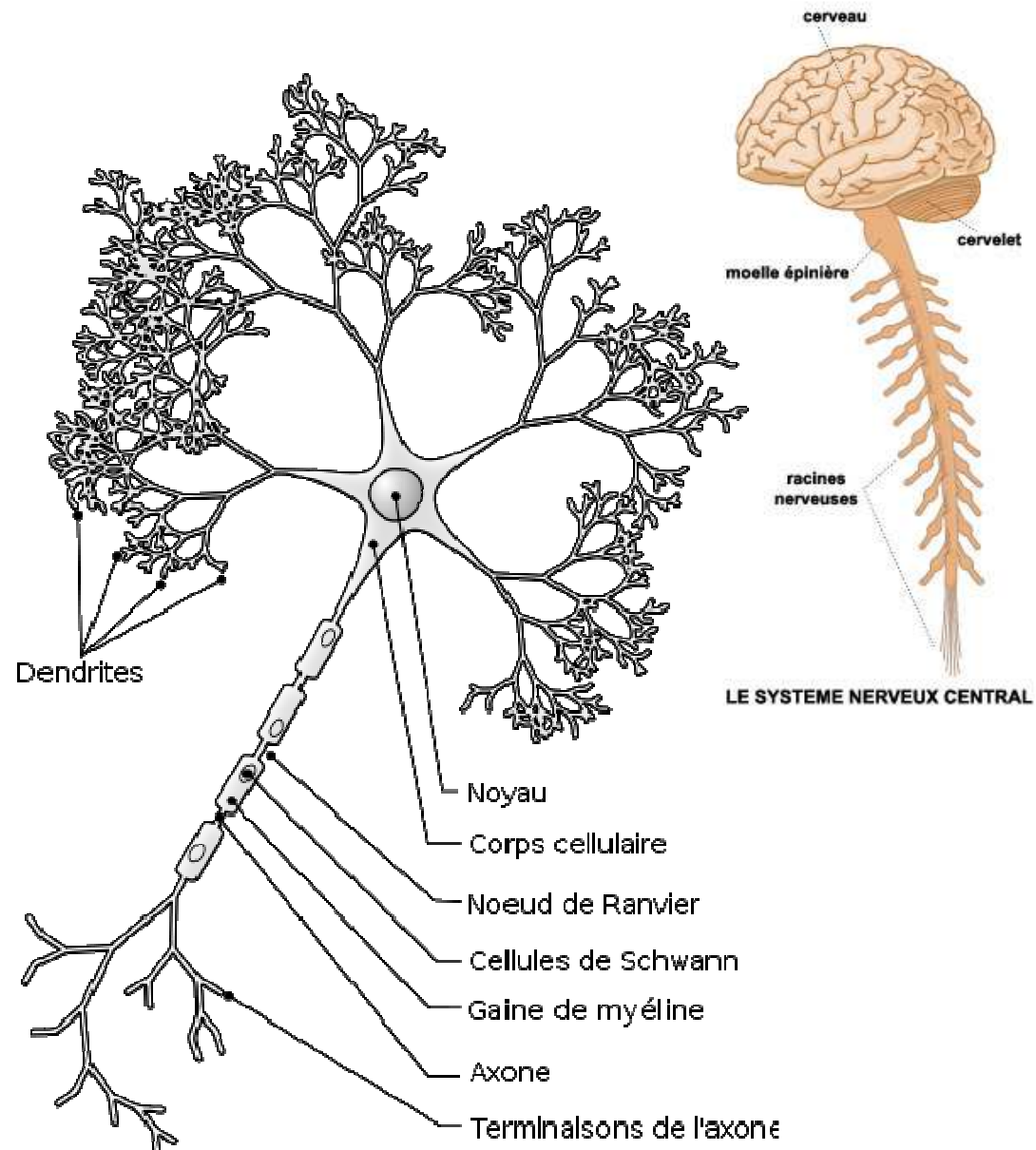
- ☐ Les Réseaux de Neurones Artificiels
- ☐ Un peu d'histoire
- ☐ Le Perceptron MultiCouche
- ☐ Les *Deep Neural Networks*
- ☐ Conclusion

# Les Réseaux de Neurones Artificiels

# Le neurone

Le neurone est la cellule qui permet la transmission de l'information et sa mémorisation (synapses):

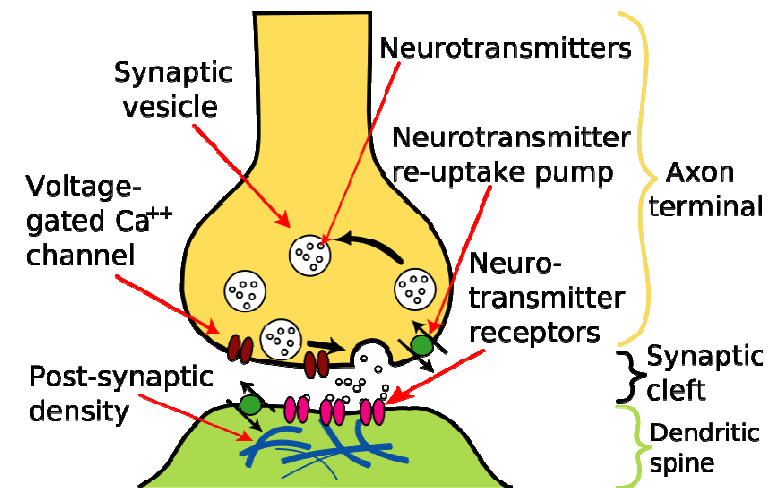
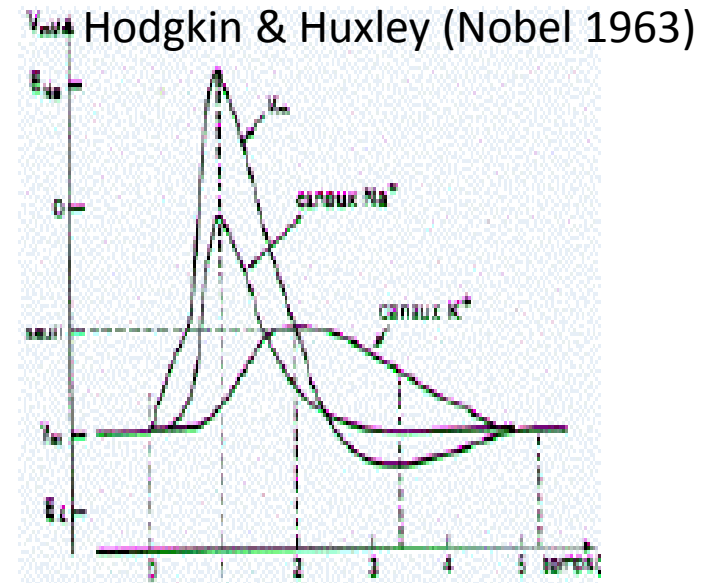
- ❑ Principale unité fonctionnelle du système nerveux
- ❑ Le système nerveux humain contient plusieurs dizaines de **milliards** de neurones



# Le potentiel d'action

Processus électriques et chimiques complexes :

- ❑ Le neurone se charge à travers ses dendrites
- ❑ Il émet un potentiel d'action (spike) qui transite le long de l'axone
- ❑ Ce potentiel arrive au niveau des synapses
  - Transite à travers des canaux ioniques
  - Modulé par les neurotransmetteurs



Eccles (Nobel 1963)

Wikipedia

# A la croisée des chemins...

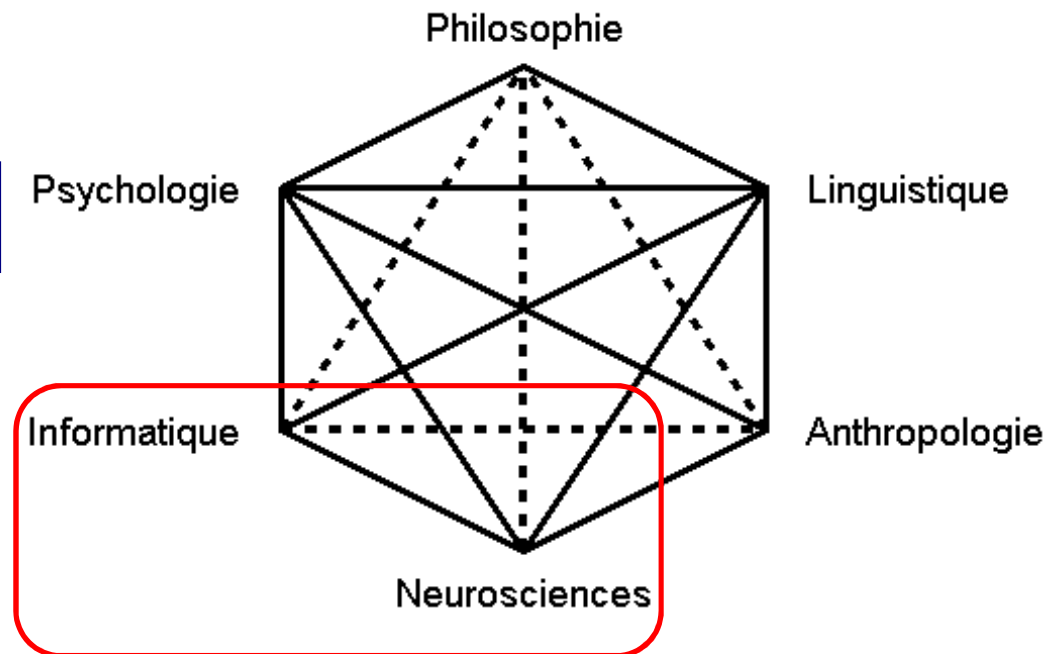
Les réseaux de neurones artificiels (notés **RNA**) s'inspirent des recherches en sciences cognitives, et plus particulièrement en neuroscience.

Idée:

- « l'intelligence », phénomène émergent du système nerveux
- l'IA peut-elle émerger d'un réseau neuro-mimétique ?



« Le connexionisme »



# Les RNA: définitions

## La définition de S. Haykin (1994)

Un réseau de neurones est un processeur massivement distribué en parallèle qui a une propension naturelle pour stocker de la connaissance **empirique** (*experiential knowledge* selon l'auteur) et la rendre disponible à l'usage. Il ressemble au cerveau sur deux aspects:

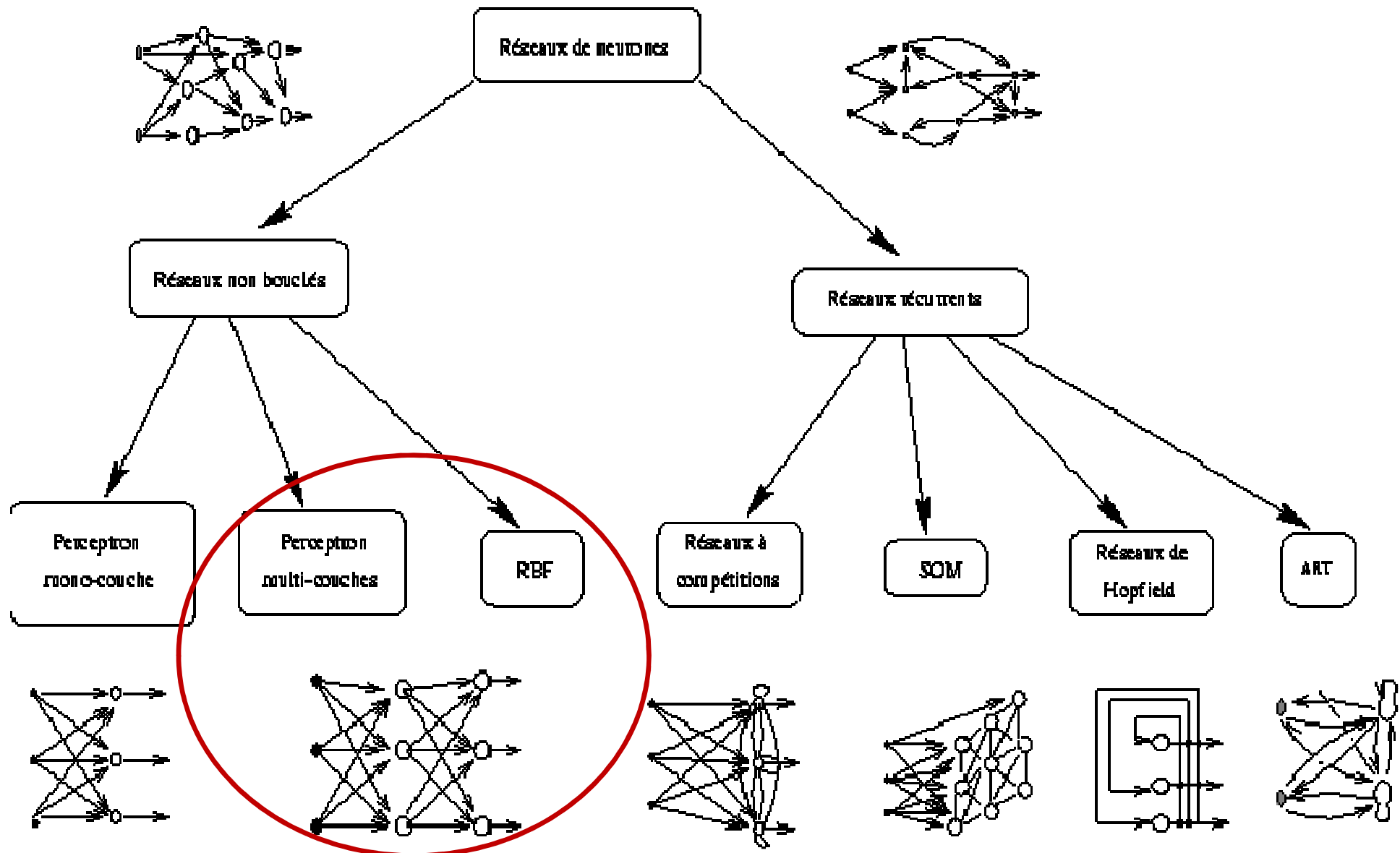
1. La connaissance est acquise par le réseau au travers d'un processus d'apprentissage
2. Les connexions entre les neurones, connues sous le nom de poids synaptiques, servent à stocker la connaissance

# Les RNA: définitions

## **La définition du DARPA Neural Network Study Group (1988)**

Un réseau de neurones est un système composé de plusieurs unités de calcul simples fonctionnant en parallèle, dont la fonction est déterminée par la structure du réseau, la solidité des connexions, et l'opération effectuée par les éléments ou nœuds.

# Les différentes architectures

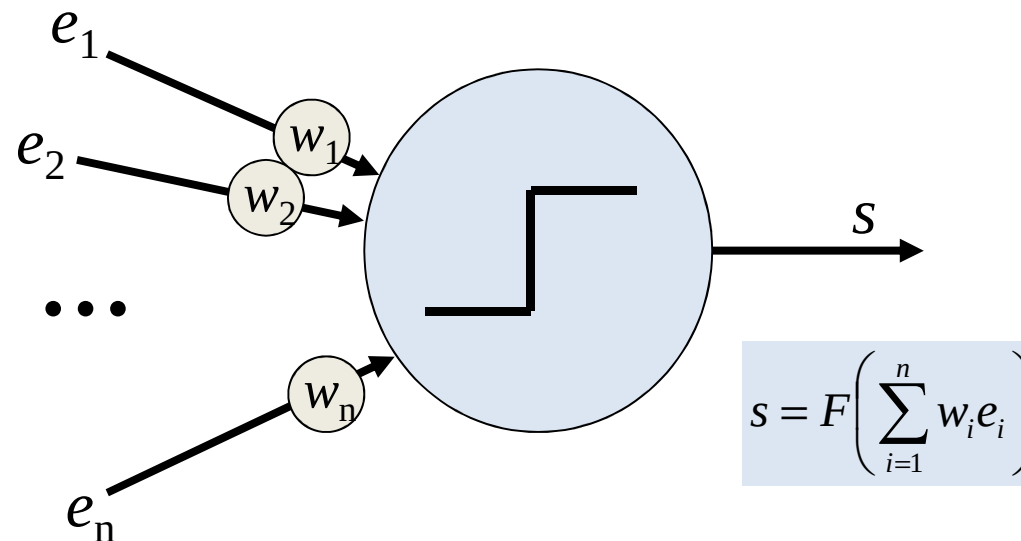
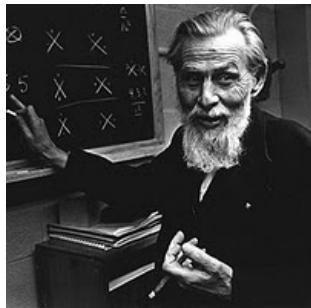


# Un peu d'histoire

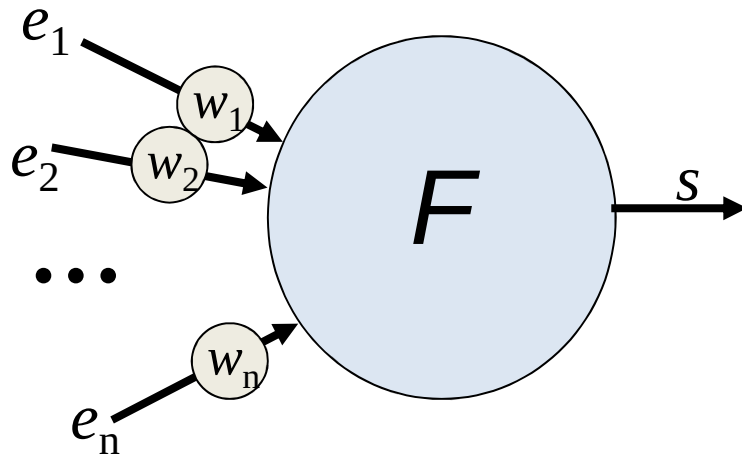
# Le neurone formel

## 1943: Modèle du neurone de J. Mc Culloch et W. Pitts

- Modèle binaire du neurone biologique, premier modèle du neurone formel.
- Les auteurs montrent que ces neurones en réseau peuvent réaliser des fonctions logiques ou arithmétiques.

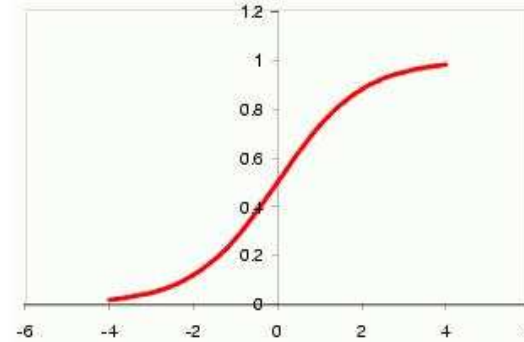


# Le neurone formel: généralisation

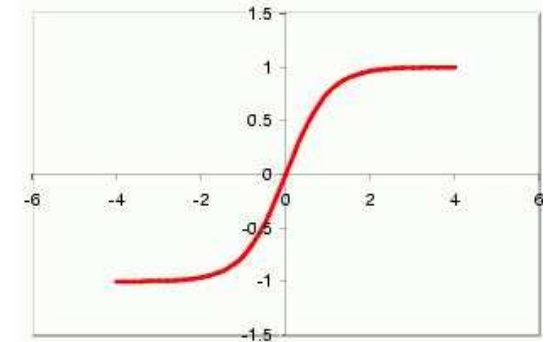


$$s = F\left(\sum_{i=1}^n w_i e_i\right)$$

$F$ : Fonction d'activation

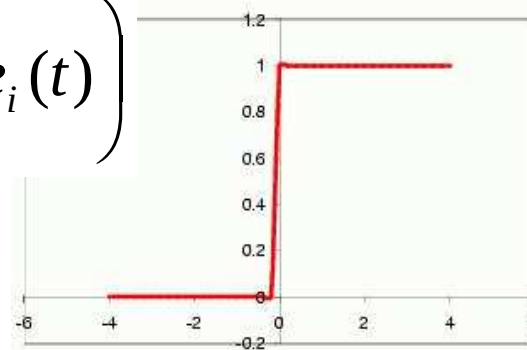


Fonction logistique

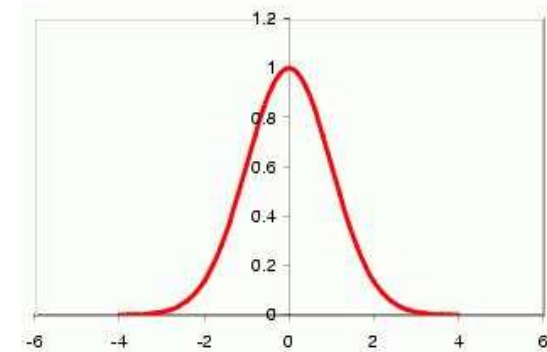


Fonction hyperbolique

$$\frac{ds}{dt}(t) = -\rho s(t) + F\left(\sum_{i=1}^n w_i e_i(t)\right)$$



Fonction à seuil

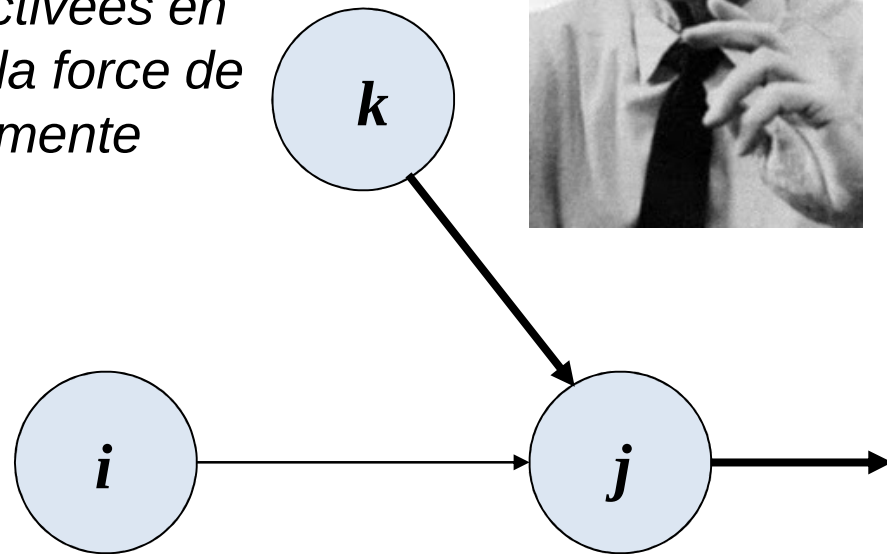
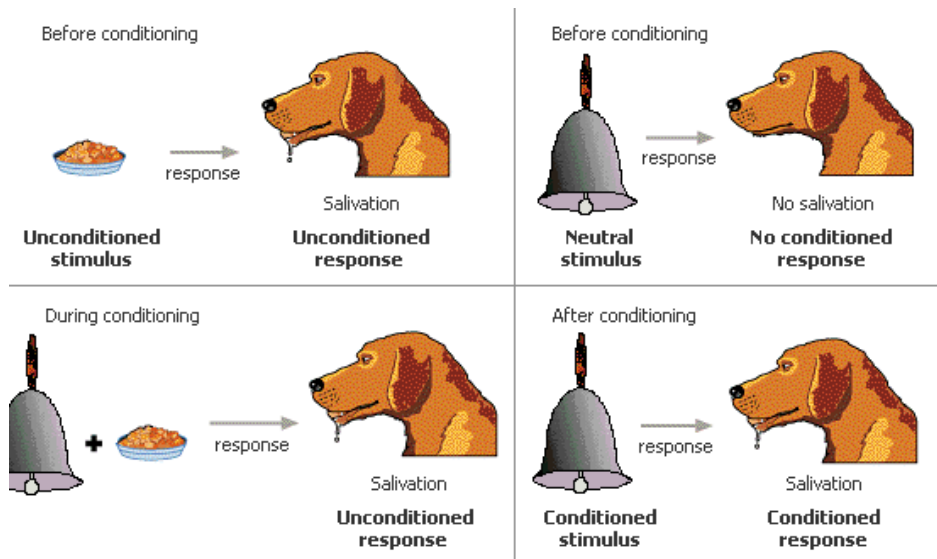
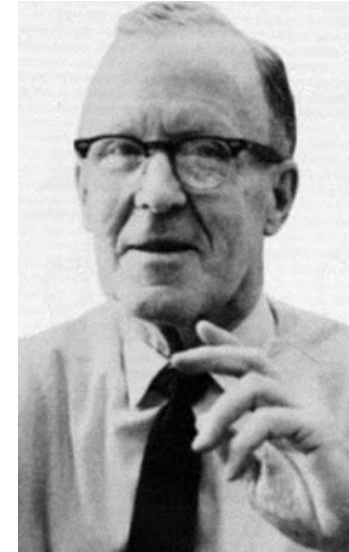


Fonction gaussienne

# La règle de Hebb (1949)

Le physiologiste **D. Hebb** invente une règle d'apprentissage qui correspond au conditionnement de Pavlov

*Si 2 cellules sont activées en même temps alors la force de leur connexion augmente*

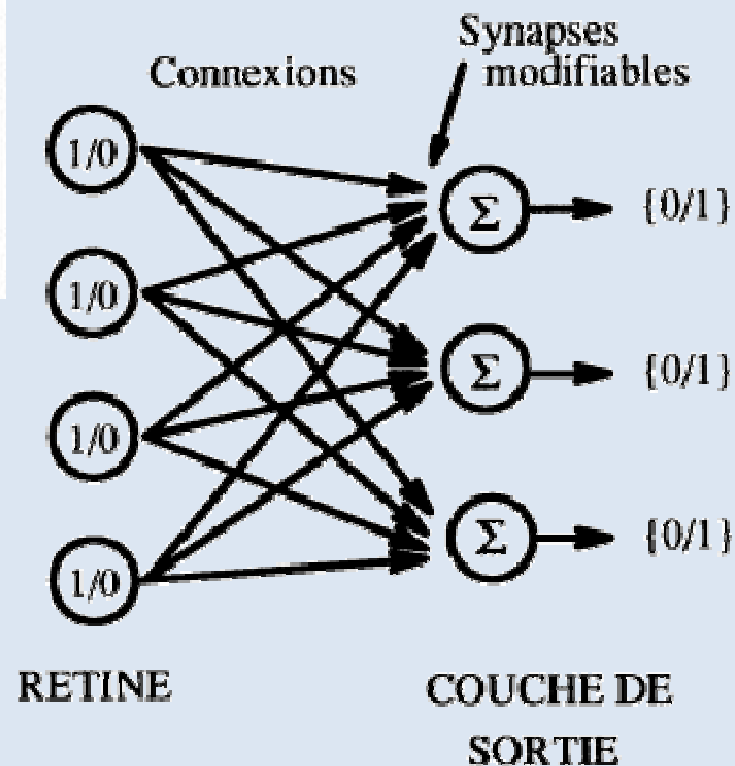


$$w_{ij}(t+1) = w_{ij}(t) + \rho s_i s_j$$

*Apprentissage non supervisé*

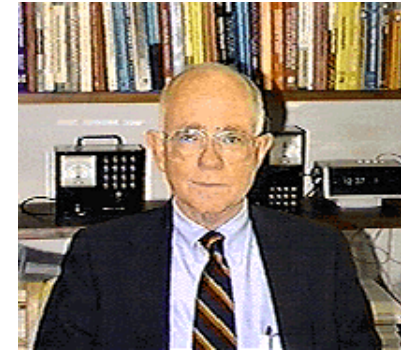
# Le perceptron (1958-1960)

**F. Rosenblatt** invente le modèle du perceptron, basé sur le principe du cortex visuel, et destiné à la reconnaissance de formes. C'est le premier réseau « multi-couche ».



# Le perceptron (1958-1960)

Le perceptron doit trouver l'ensemble des valeurs à donner aux synapses pour que les configurations d'entrée se traduisent par des réponses **voulues**. Pour cela, on utilise la *règle d'apprentissage* de Widrow-Hoff (cf modèle ADALINE):



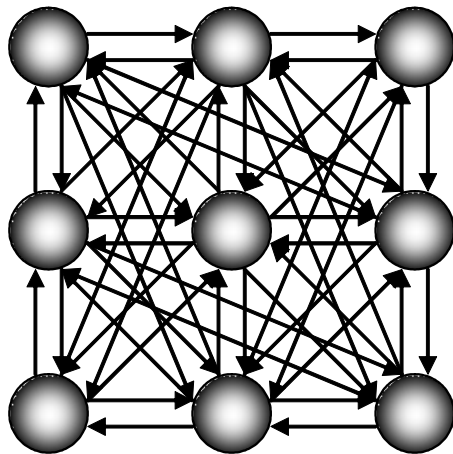
$$w_{ij}(t+1) = w_{ij}(t) + \rho s_i (s_j^* - s_j)$$

Pour apprendre, **le perceptron doit savoir qu'il a commis une erreur**, et doit connaître la réponse qu'il aurait dû donner. De ce fait, on parle *d'apprentissage supervisé*.

# Les Réseaux de Hopfield (1982)

Le physicien J.J. Hopfield invente un modèle de réseau de neurones (de type Mc Culloch et Pitt) dynamique autoassociatif.

*(Exemplaire pour la méthodologie et la rigueur mathématique)*



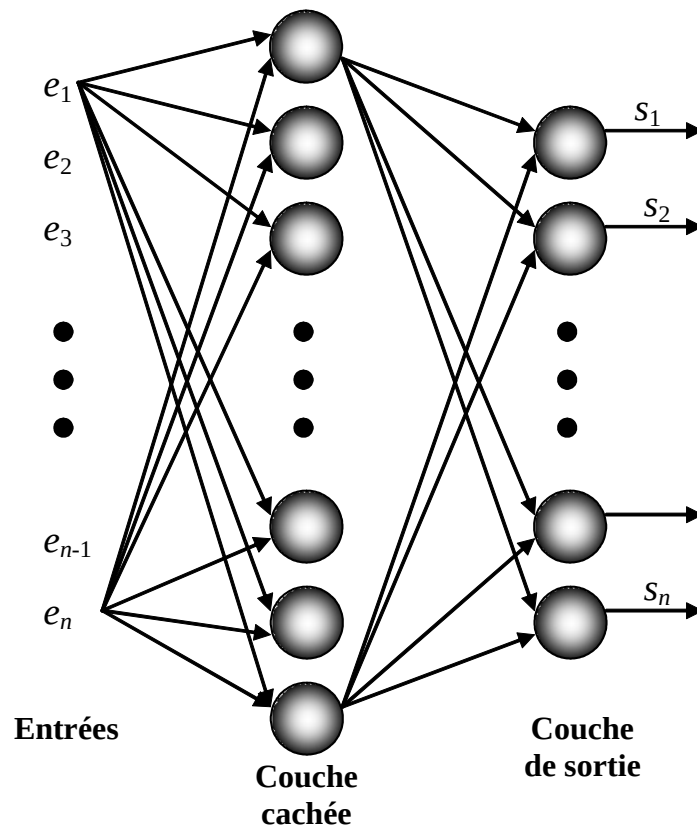
Réseau de Hopfield (« Hopfield tank »)

- tous les neurones sont connectés entre eux
- un neurone n'est pas connecté à lui-même
- les poids synaptiques sont symétriques

*Hopfield, J. J. (1982) Neural networks and physical systems with emergent collective computational properties. Proc. Nat. Acad. Sci. (USA) 79, 2554-2558.*

# La rétro-propagation du gradient (1975-1985)

P. Werbos (1975) introduit de la règle de **rétro-propagation du gradient**, obtenue par l'application de la méthode du gradient aux réseaux de neurones formels et différentiables.



⇒ Application aux  
réseaux multicouches  
**(Perceptron MultiCouche):**  
- tous les neurones d'une  
couche donnée sont  
connectés aux neurones de la  
couche suivante.

# Les Réseaux de Kohonen (1982)

Il a été observé que, dans de nombreuses zones du cortex, des colonnes voisines ont tendance à réagir à des entrées similaires.



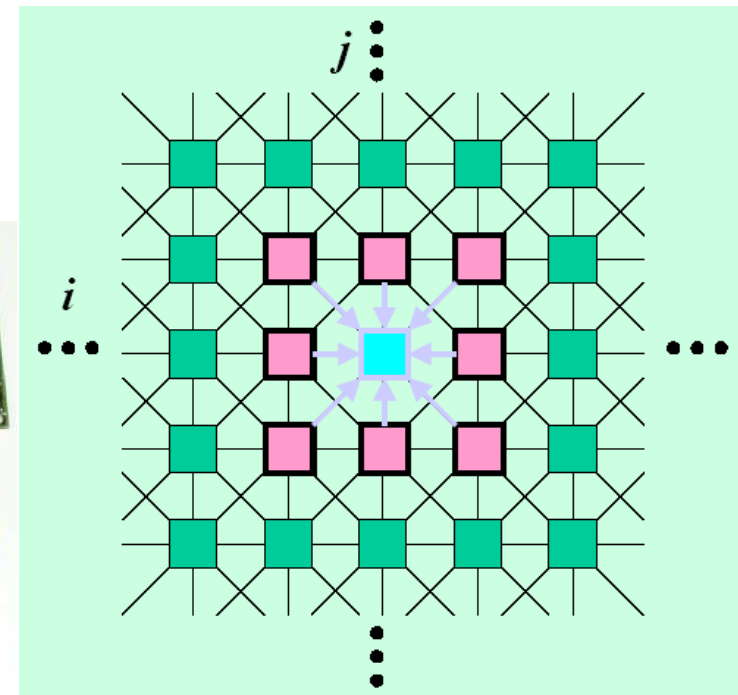
Dans les aires visuelles, par exemple, deux colonnes proches sont en correspondance avec deux cellules proches de la rétine.

Ces observations ont mené T. Kohonen à proposer un modèle de *carte topologique auto-adaptative* qui permet de coder des motifs présentés en entrée tout en conservant la topologie de l'espace d'entrée.

*Kohonen, T. (1982). Self-organized formation of topologically correct feature maps. Biological Cybernetics, 43:59-69.*

# Les Réseaux de neurones cellulaires - CNN (1988)

Cette architecture de réseau inventée par Léon O. Chua est similaire aux automates cellulaires. Il s'agit d'un réseau où chaque cellule est connectée à ses voisins (au sens de la notion de voisinage que l'on définit pour un problème donné).



L. Chua and L. Yang, "[Cellular Neural Networks: Theory](#)," *IEEE Trans. on Circuits and Systems*, 35(10):1257-1272, 1988.

# Victoire d'AlphaGo (2015)

Le programme AlphaGo bat en octobre 2015 le champion européen de Go, M. Fan Hui, puis en mars 2016 M. Lee Sedol, champion mondial

<https://deepmind.com/research/alphago/>



**Google DeepMind** (Londres)

2010: fondation de DeepMind Technologies

2013: négociations avec Facebook et Google

2014: rachat par Google -> Google DeepMind

*Silver, D. et Al. (2016). Mastering the game of Go with deep neural networks and tree search. Nature, 529:484-489.*

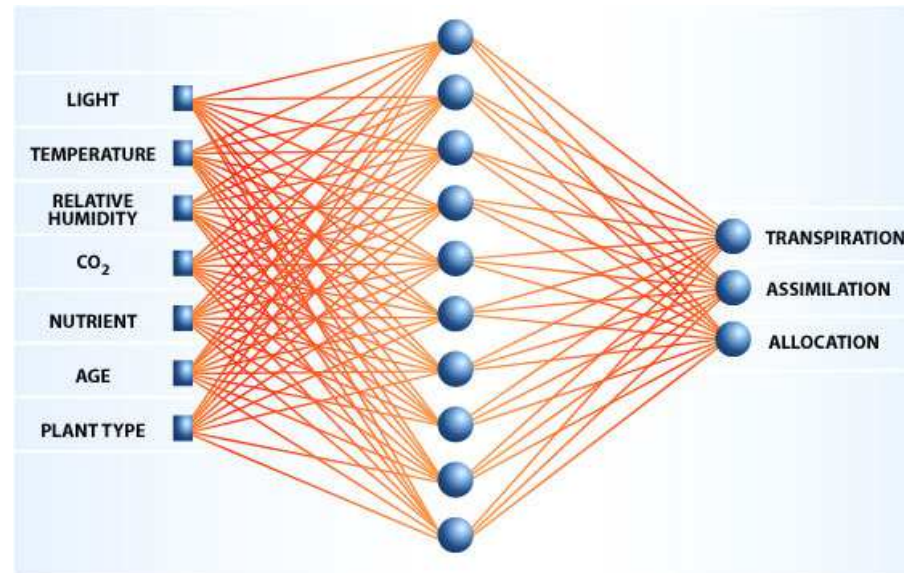
<http://www.nature.com/nature/journal/v529/n7587/full/nature16961.html>

# Mastering the game of Go with deep neural networks and tree search

**Abstract:** *The game of Go has long been viewed as the most challenging of classic games for artificial intelligence owing to its enormous search space and the difficulty of evaluating board positions and moves. Here we introduce a new approach to computer Go that uses 'value networks' to evaluate board positions and 'policy networks' to select moves. These deep neural networks are trained by a novel combination of supervised learning from human expert games, and reinforcement learning from games of self-play. Without any lookahead search, the neural networks play Go at the level of state-of-the-art Monte Carlo tree search programs that simulate thousands of random games of self-play. We also introduce a new search algorithm that combines Monte Carlo simulation with value and policy networks. Using this search algorithm, our program AlphaGo achieved a 99.8% winning rate against other Go programs, and defeated the human European Go champion by 5 games to 0. This is the first time that a computer program has defeated a human professional player in the full-sized game of Go, a feat previously thought to be at least a decade away.*



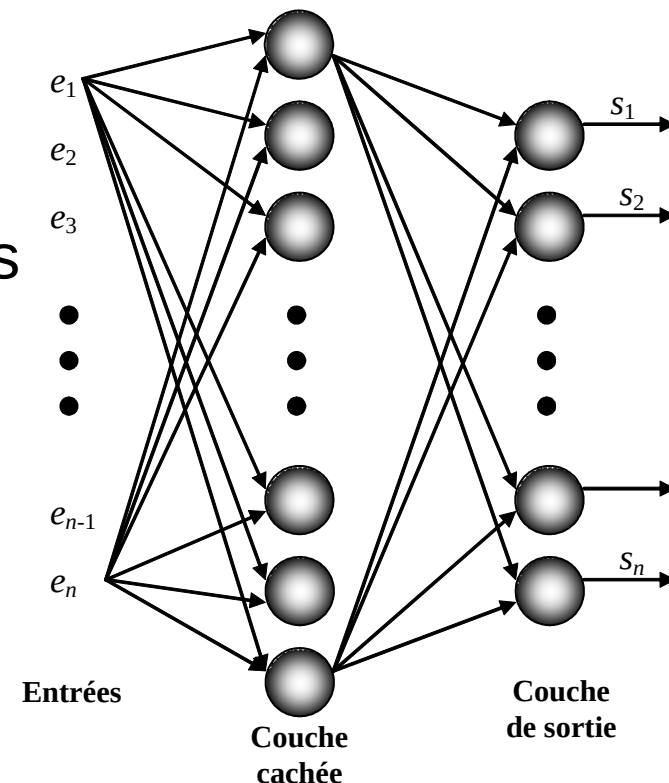
# Le Perceptron MultiCouche



# Structure du réseau

Basé sur la propagation vers « l'avant » (« *feedforward* ») des entrées à travers des couches de neurones:

- Les entrées sont connectées à tous les neurones de la première couche
- Tous les neurones d'une couche sont connectés à tous les neurones de la couche suivante
- Les sorties des neurones de la dernière couche sont les sorties du réseau



# Equations du réseau (1)

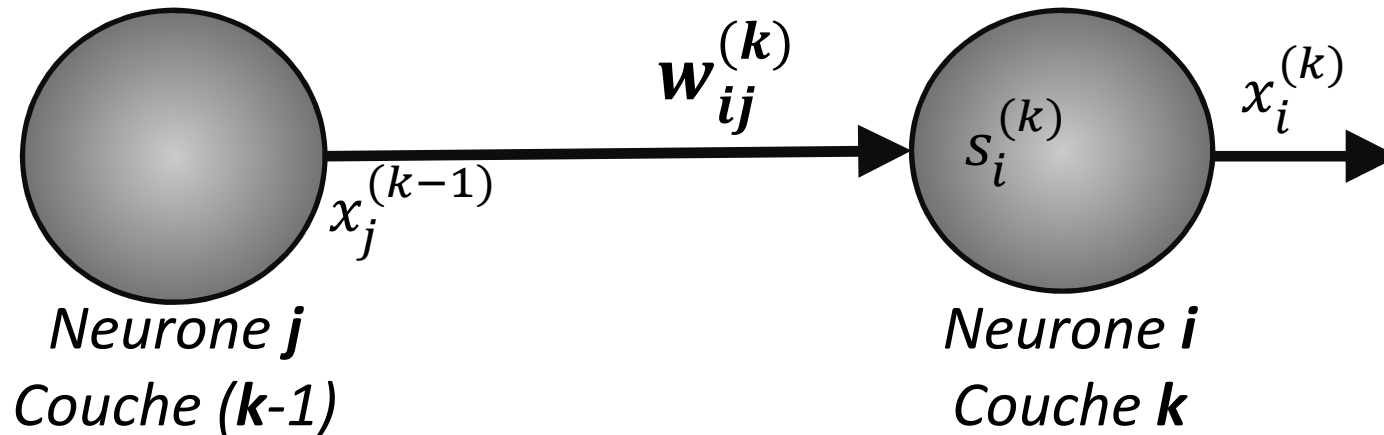
$$\left\{ \begin{array}{ll} \vec{x}^{(1)} = \vec{F}^{(1)} \left( \mathbf{W}^{(1)} \vec{e} \right) & \text{Entrées} \rightarrow 1^{\text{ère}} \text{ couche} \\ \vec{x}^{(k)} = \vec{F}^{(k)} \left( \mathbf{W}^{(k)} \vec{x}^{(k-1)} \right) & k=2 \dots N_l-1 \quad \text{Propagation} \\ \vec{o} = \vec{F}^{(N_l)} \left( \mathbf{W}^{(N_l)} \vec{x}^{(N_l-1)} \right) & \text{Sorties réseau} \end{array} \right.$$

*Couches cachées*      *Couche de sortie*

*Calcul de la sortie du  $j^{\text{ème}}$   
neurone de la  $k^{\text{ème}}$  couche*

$$x_i^{(k)} = F^{(k)} \left( \sum_{j=1}^{N^{(k-1)}} w_{ij}^{(k)} x_j^{(k-1)} \right)$$

# Equations du réseau (2)



$\mathbf{W}^{(k)}$  est la matrice des poids de la couche (k), de composantes  $w_{ij}^{(k)}$

*Calcul de la sortie du  $i^{\text{ème}}$  neurone de la  $k^{\text{ème}}$  couche*

$$x_i^{(k)} = F^{(k)} \left( \sum_{j=1}^{N^{(k-1)}} w_{ij}^{(k)} x_j^{(k-1)} \right)$$

On note  $s_i^{(k)} = \sum_{j=1}^{N^{(k-1)}} w_{ij}^{(k)} x_j^{(k-1)}$  l'état du neurone.

# Apprentissage

L'apprentissage consiste à calculer les poids  $w_{ij}^{(k)}$  de manière à ce que le réseau modélise (au sens de la régression) des couples d'entrées-sorties extraits d'une base de donnée.

→ Modèle boîte noire



Méthode: minimisation d'un critère de moindre carré

$$J(\mathbf{W}) = \frac{1}{N_p} \sum_{p=1}^{N_p} \left\| \vec{o}(\mathbf{W}, \vec{e}^{(p)}) - \vec{o}^{(p)} \right\|^2$$

# Apprentissage

- Pour chaque couple (Vecteur d'entrées , Vecteur de sorties) de la base de données :
  - calculer le vecteur des sorties du réseau pour le vecteur d'entrées de la base de données
  - calculer sa distance avec le vecteur des sorties de la base de données (→ erreur entre sortie du réseau et sortie désirée)
- Calculer la moyenne de toutes les erreurs

$$J(\mathbf{W}) = \frac{1}{N_p} \sum_{p=1}^{N_p} \left\| \vec{o}(\mathbf{W}, \vec{e}^{(p)}) - \vec{o}^{(p)} \right\|^2$$

Problème: trouver l'ensemble des poids synaptiques  $\mathbf{W}$  (les paramètres) qui minimise ce critère

# La rétropropagation du gradient (Werbos, Rumelhart, ...)

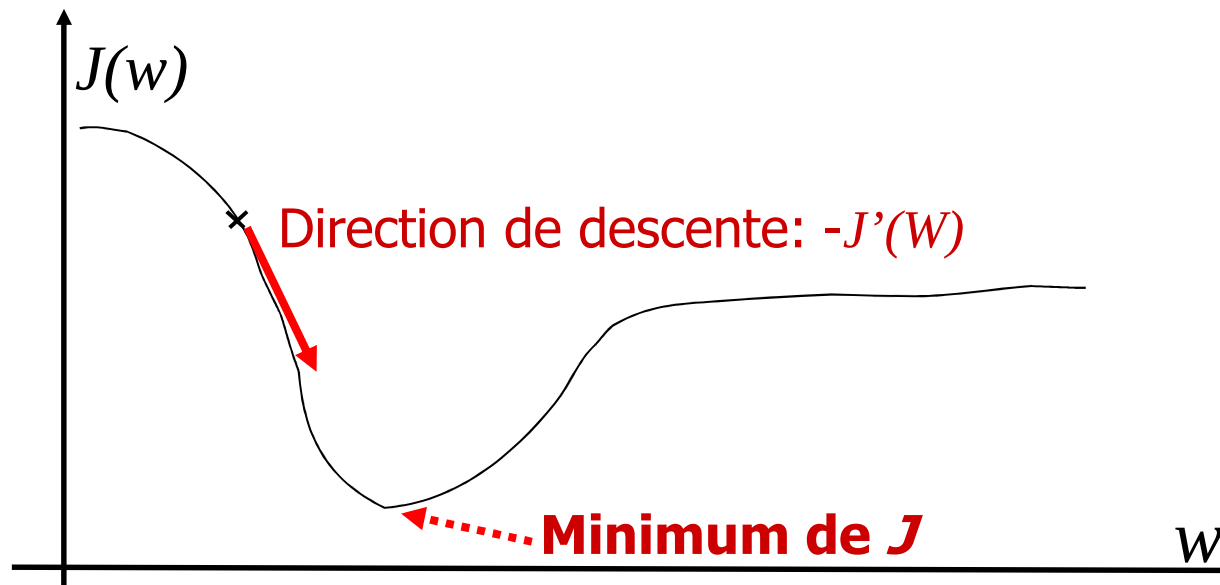
- Méthode la plus utilisée car :
  - S'adapte à un nombre quelconque de couche, à un nombre quelconque de neurones par couche, à n'importe quelle fonction de transfert dérivable
  - Efficace en terme d'implémentation
  - Facilement modifiable pour différentes heuristiques (méthode Levenberg-Marquardt, quick-prop, ...)
- Basée sur la méthode du gradient :

$$\mathbf{W}(t+1) = \mathbf{W}(t) - \rho \frac{dJ}{d\mathbf{W}}(\mathbf{W}(t))$$

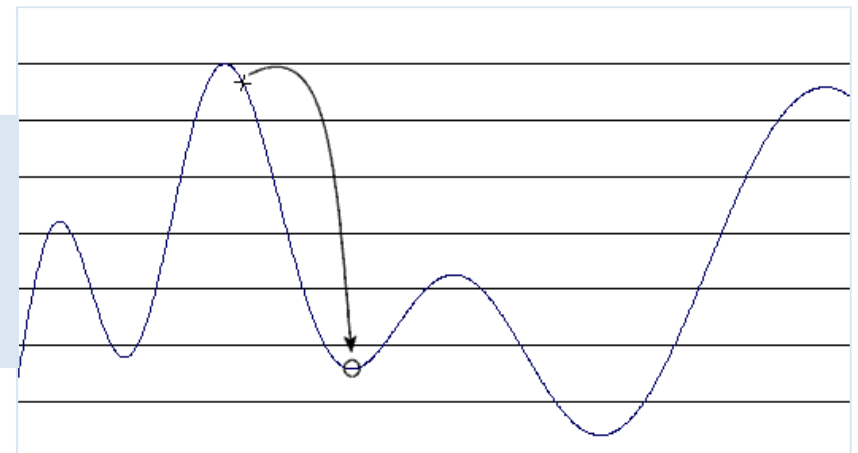
Pas d'apprentissage

# Algorithme du Gradient

Illustration dans le plan  $(J(w), w)$  : « la descente de gradient »



Problème: c'est une méthode locale -> trouve le minimum « le plus proche » dans la direction de descente



# Rétropropagation *on-line* et *batch*

Deux façons d'appliquer l'algorithme de rétropropagation :

□ *batch* :

mise à jour des poids après la présentation de tous les exemples

- calculs et stockage plus lourds si trop d'exemples

□ séquentiel : (*on-line*, stochastique)

mise à jour des poids après chaque exemple

- besoin de tirer l'exemple au hasard
- problèmes de convergence

# Calcul du Gradient – *on-line* (1)

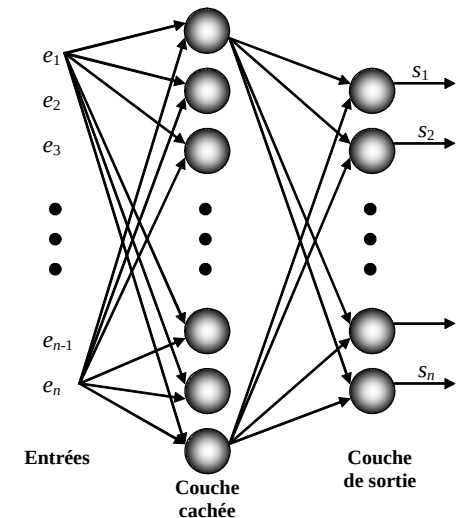
Basé sur la dérivée de la composée (règle de « chaînage »):

si  $J(w) = J(o(s(w)))$  alors

$$\frac{dJ}{dw} = \frac{dJ}{do} \frac{do}{ds} \frac{ds}{dw}$$

$J$  = Erreur entre sorties du réseaux et sorties voulues:

$$J(\mathbf{w}) = \sum_{l=1}^N (o_l - o_l^*)^2$$



Pour les poids de la **couche de sortie**  $w_{ij}^{(Nl)}$ , en notant  $w_{ij} \equiv w_{ij}^{(Nl)}$ , nous avons :

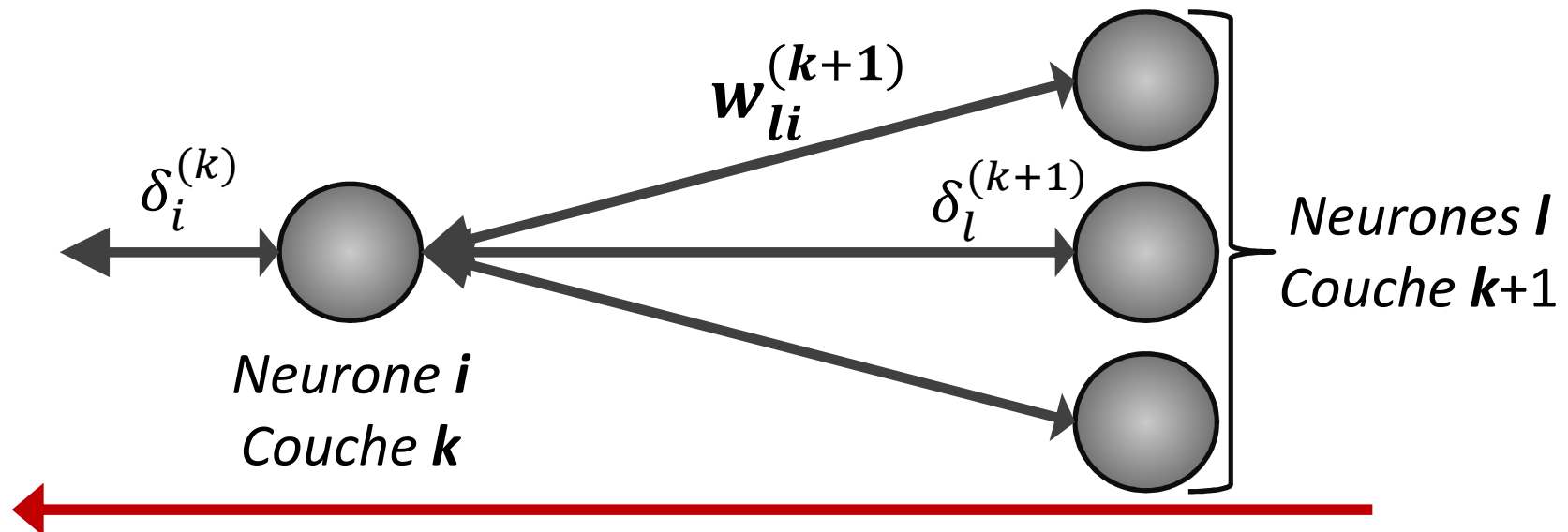
$$\frac{\partial J}{\partial w_{ij}} = \frac{\partial J}{\partial o_i} \frac{\partial o_i}{\partial s_i} \frac{\partial s_i}{\partial w_{ij}} = 2(o_i - o_i^*) F' \left( s_i^{(Nl)} \right) x_j^{(Nl-1)} = \delta_i^{(Nl)} x_j^{(Nl-1)}$$

Widrow-Hoff

# Calcul du Gradient – *on-line* (2)

On ***rétropropage*** l'erreur  $(\vec{o}(\mathbf{W}, \vec{e}) - \vec{o}^*)$  de la couche de sortie vers la première couche cachée :

- Pour chaque neurone  $i$  de la couche de sortie ( $Nl$ ) calculer  $\delta_i^{(Nl)} = 2 \left( o_i^{(Nl)} - o_i^* \right) F^{(Nl)'} \left( s_i^{(Nl)} \right)$
- Pour chaque neurone  $i$  de la couche  $0 < k < Nl$  calculer  $\delta_i^{(k)} = \left( \sum_{l=1}^{N^{(k+1)}} \delta_l^{(k+1)} w_{li}^{(k+1)} \right) F^{(k)'} \left( s_i^{(k)} \right)$



# Un peu de méthode...

Une démarche classique en modélisation des données:

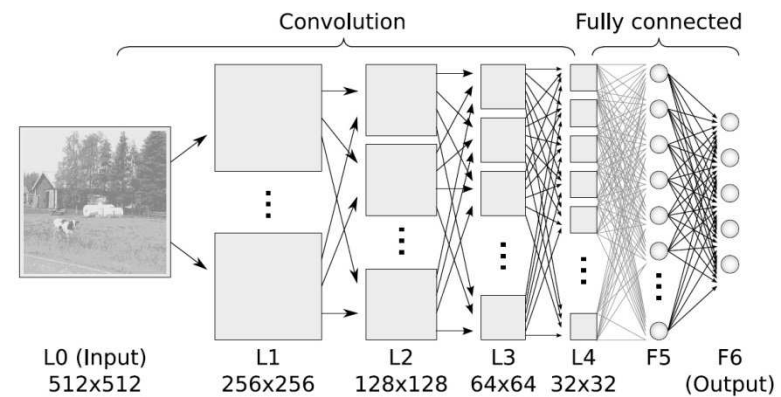
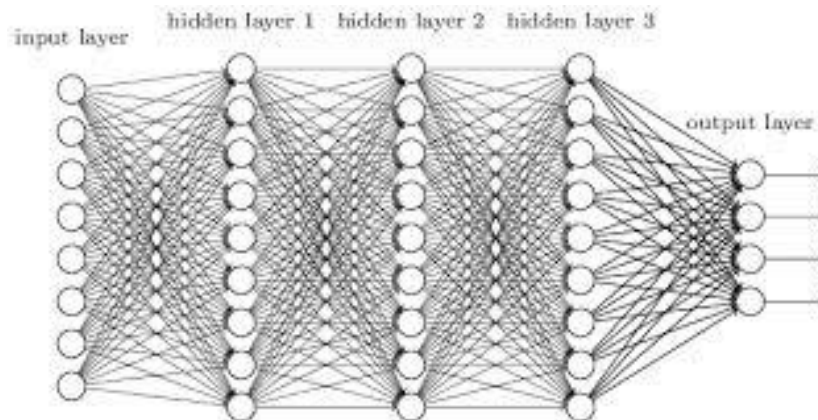
## □ Préparer les données

- Quantification des variables qualitatives
- Suppression des données incomplètes ou non pertinentes
- Normalisation, équilibrage
- Détermination variables explicatives / à expliquer
- Construction de 3 ensembles: données d'apprentissage, données de validation, données de test

## □ Appliquer un processus itératif et incrémental

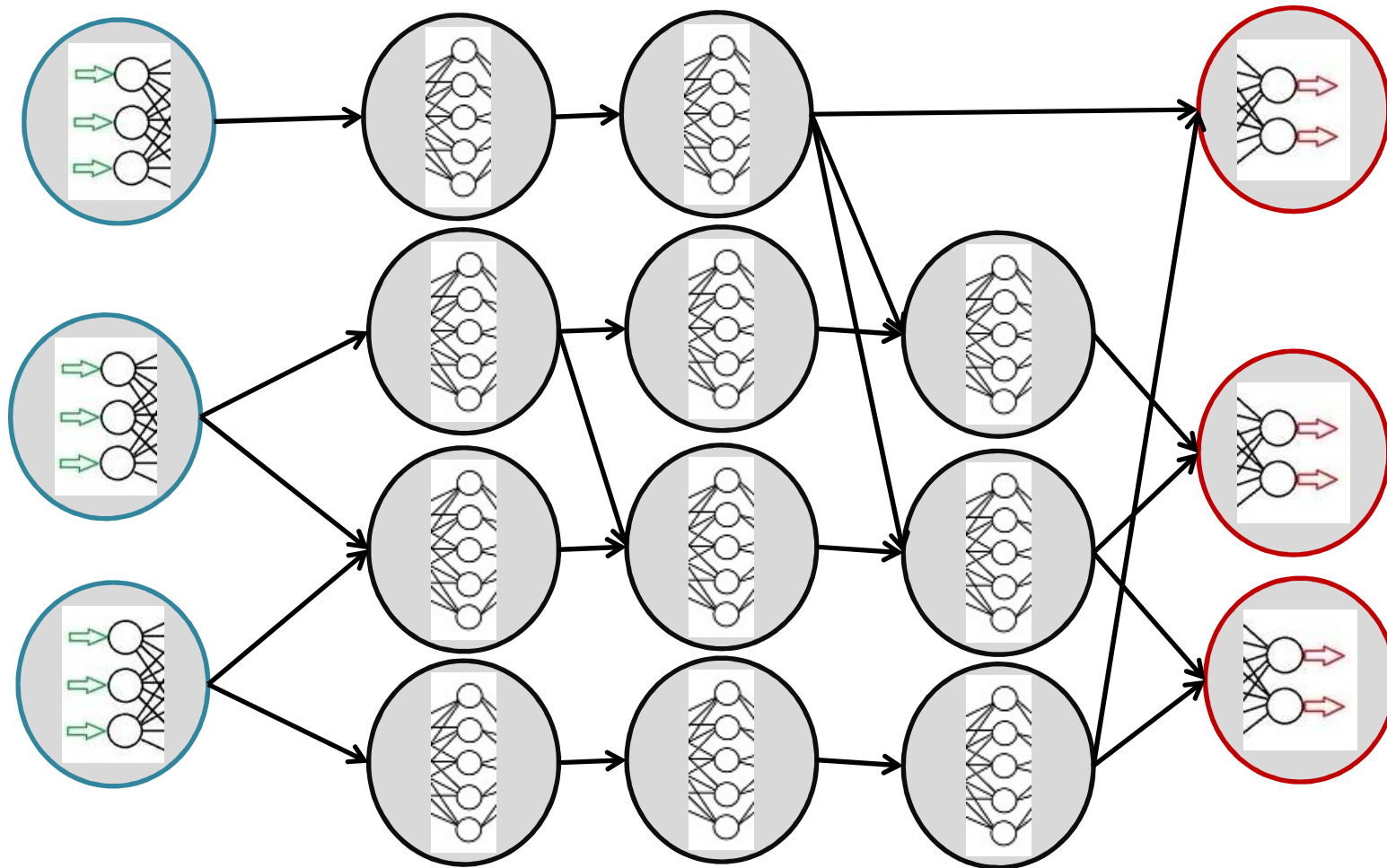
- Choix et préparation des échantillons
- Choix de la structure du réseau
- Apprentissage
- Validation/Test

# *Les Deep Neural Networks*



<https://www.caa.tuwien.ac.at/cvl/teaching/diplomarbeiten/flow-cytometry-minimal-residual-disease-assessment-using-advanced-deep-neural-networks/>

# Graphe et PMC (1)



Graphe orienté non cyclique où:

- Nœud = Couche de neurones
- Arc = Les sorties du nœud source sont transmises à tous les neurones du nœud puits

# Graphe et PMC (2)

La méthode de rétropropagation de l'erreur (et donc du gradient) fonctionne quelque soit le nombre de « nœuds » (i.e. de couches neuronales).

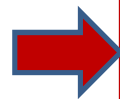
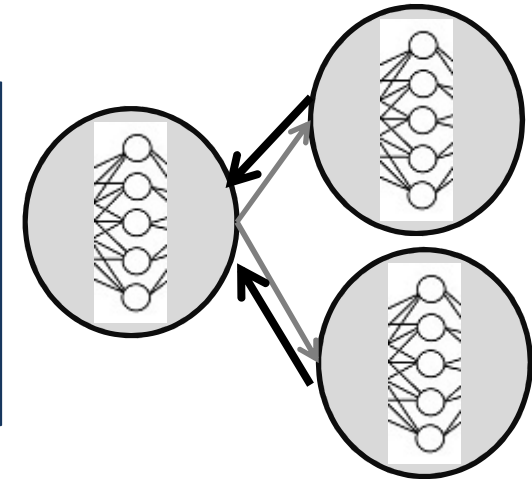
❑ Toute couche dispose de ces informations:

- Liste de « successeurs »
- Liste de « prédecesseurs »

❑ L'erreur  $\delta_i^{(\text{Noeud } k)}$  est calculée à partir des erreurs  $\delta_i^{(\text{SUCCESSEUR}(\text{Noeud } k))}$

**Deep Neural Network** = Perceptron MultiCouche avec de nombreuses couches

➡ **Deep Learning** = Méthode de calcul du gradient basé sur la rétropropagation et appliquée à un *Deep Neural Network*



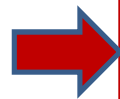
Mais pourquoi ne pas imaginer **d'autres types de couches ??**  
-> Chaîne de traitements / Processus de traitement en série

# Réseau Neuronale à Convolution

## *Convolutional Neural Network*

- ❑ Initialement appliqué au traitement d'images:
  - Reconnaissance automatique (OCR, biométrie)
  - Détection de formes (présence: oui/non)
- ❑ Suite de traitements:
  - Traitements d'image classique utilisant des **filtres** de convolution
    - Simplification (faire ressortir les contours, supprimer le bruit,...)
    - Réduction (diminuer taille et profondeur couleur)
  - Classification par Perceptron multicouche

*Filtres paramétrés  
(cf LeNet-5) ou non*

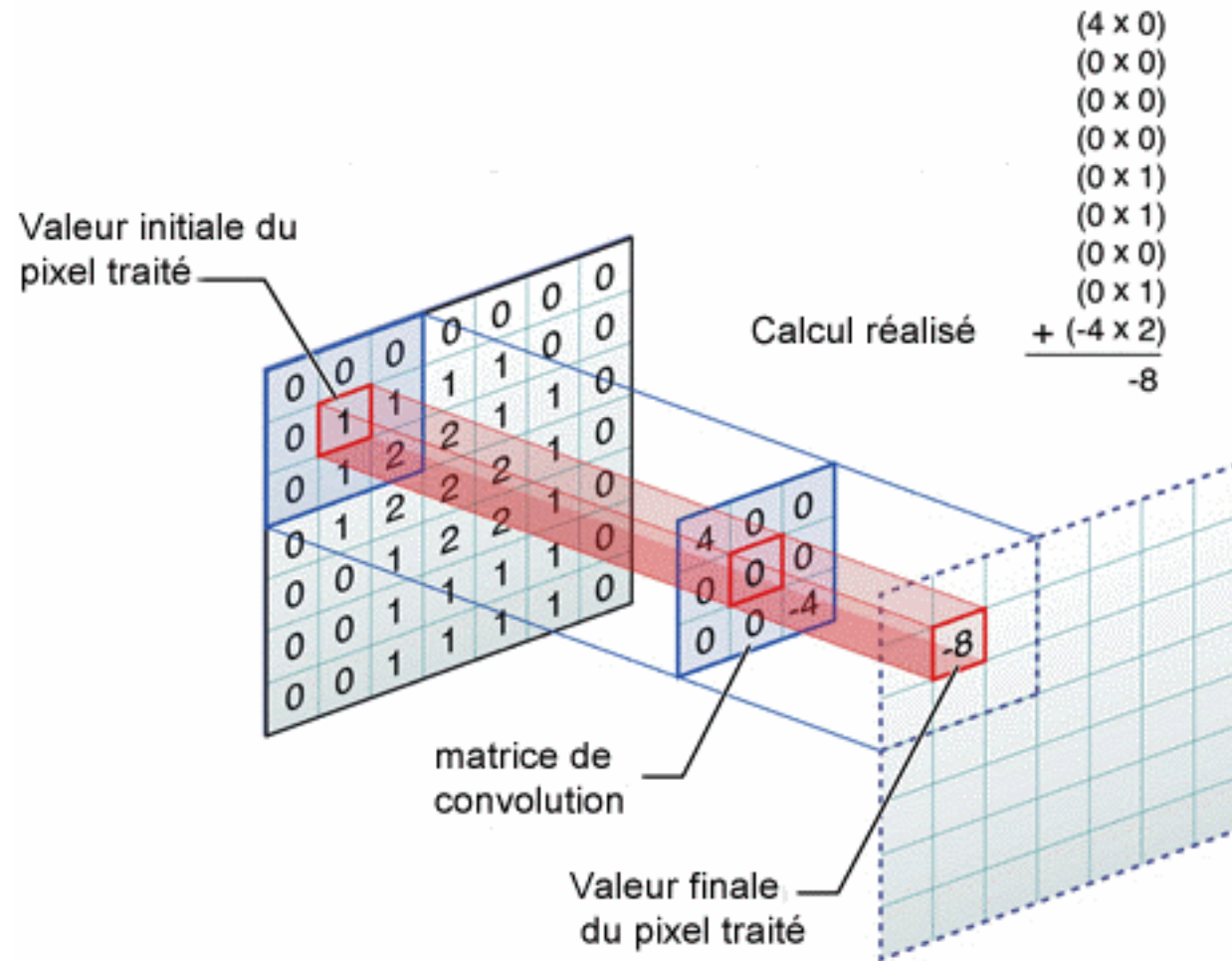


*Convolutional Neural Network: couche successives de modèles/processus par ailleurs « classiques »*

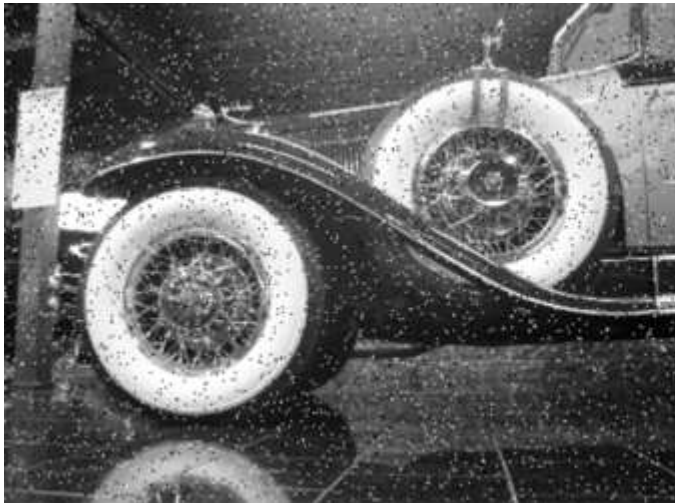
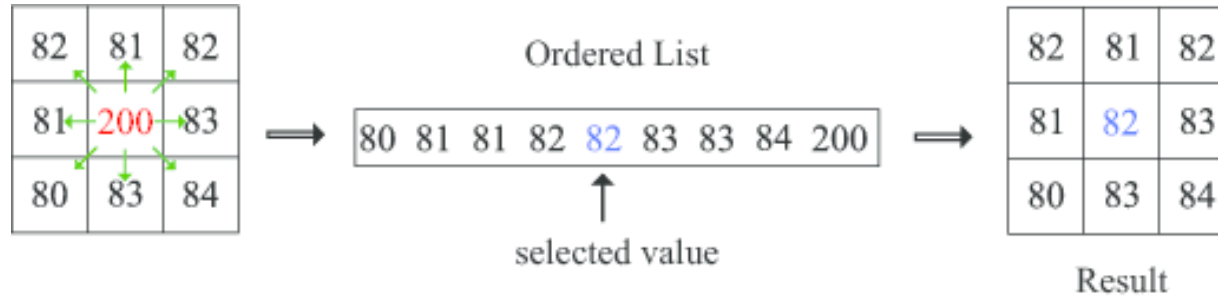
Sur internet:

<http://deeplearning4j.org/convolutionalnets.html>

# Rappel: Filtres de convolution

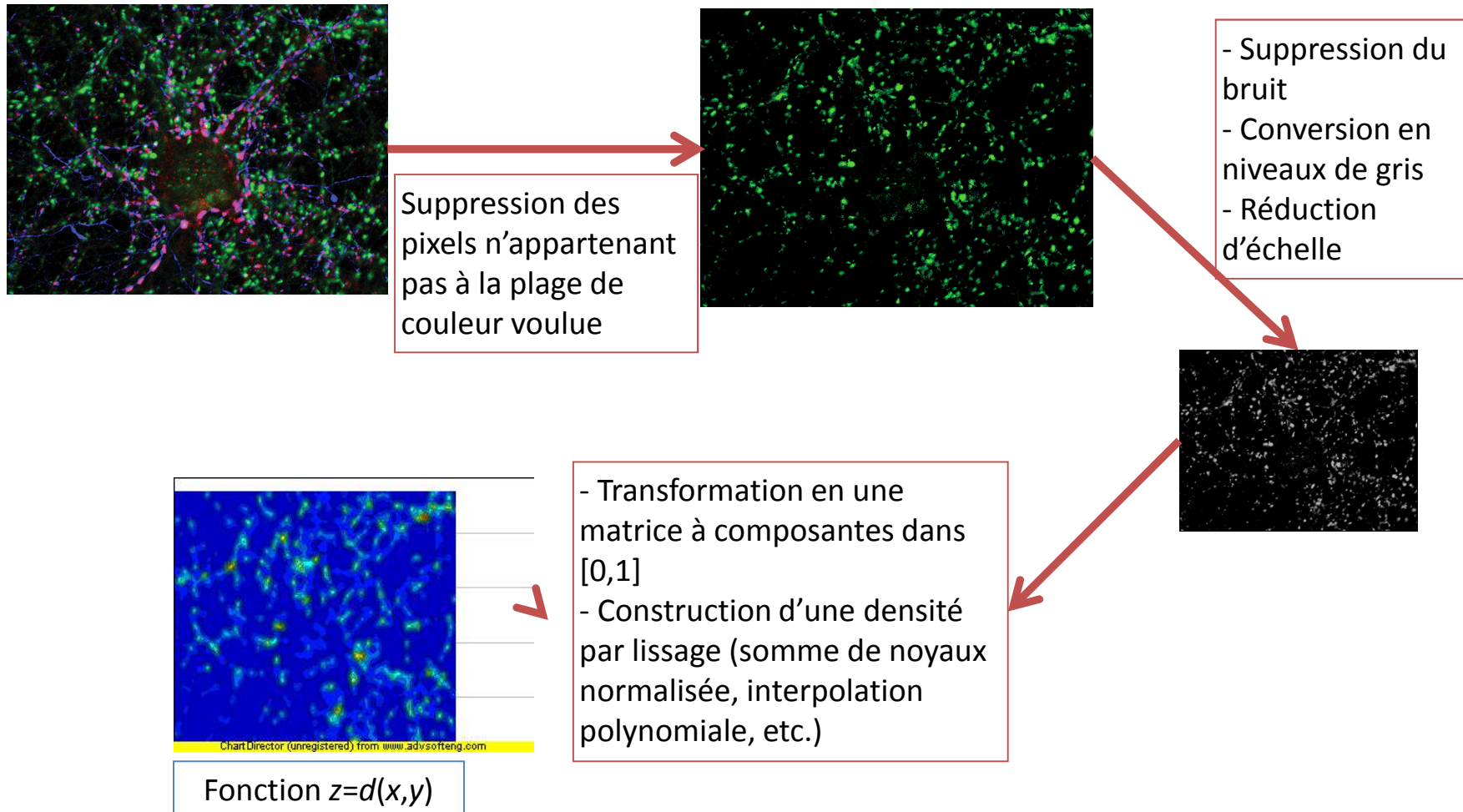


# Exemple: Filtre Median



Utiliser pour le « Débruitage »

# Exemple: calcul densité de récepteurs synaptiques

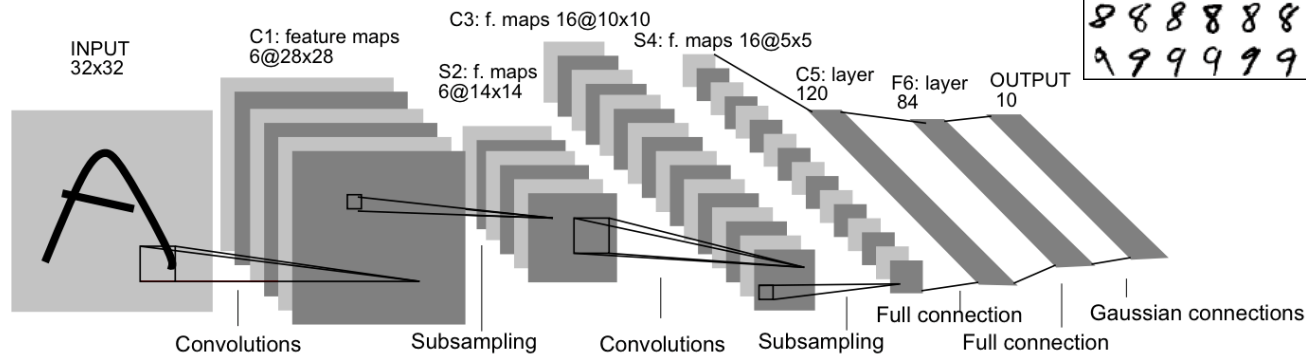


# LeNet-5

(<http://yann.lecun.com/exdb/lenet/>)

- ❑ Réseau convolutif mis au point par Y. LeCun, L. Bottou, Y. Bengio et P. Haffner en 1997-1998 pour la reconnaissance de caractères manuscrits
- ❑ Repose sur la base de données MNIST  
(<http://yann.lecun.com/exdb/mnist/>)

PROC. OF THE IEEE, NOVEMBER 1998



*Exemple base MNIST*

Base complète:  
60000+60000 images

Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

# LeNet détaillé...

PROC. OF THE IEEE, NOVEMBER 1998

C=Convolutional layer  
S=Sub-sampling layer  
F=Fully-connected layer

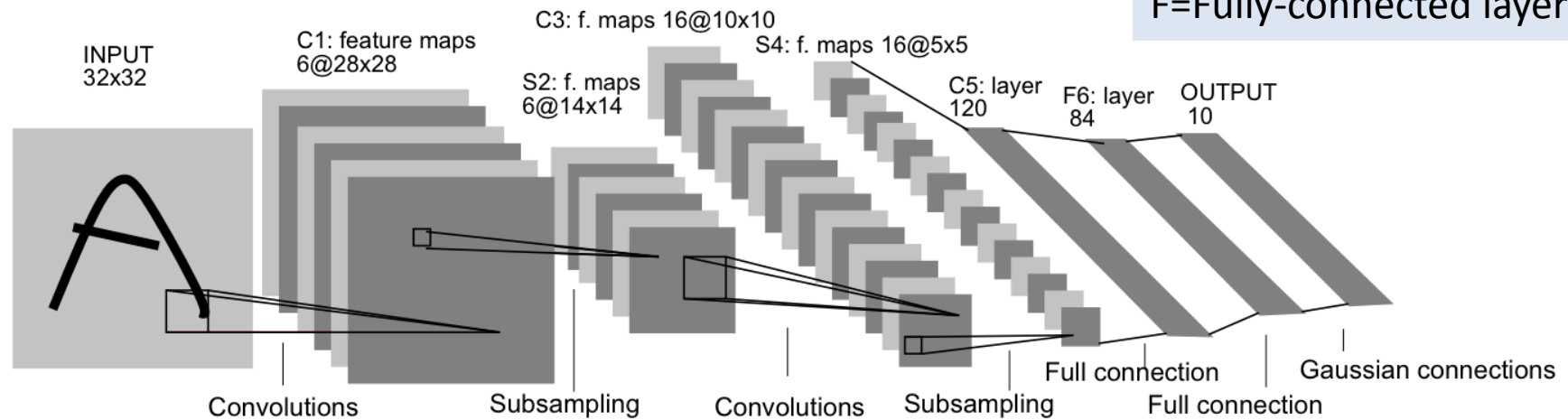
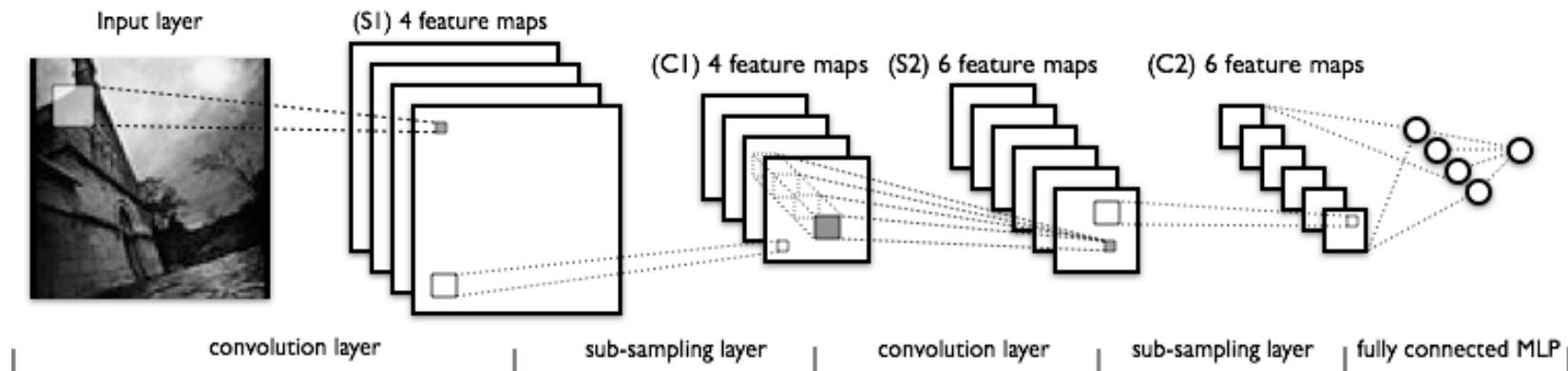


Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.



<http://deeplearning.net/tutorial/lenet.html>

# Autre exemple

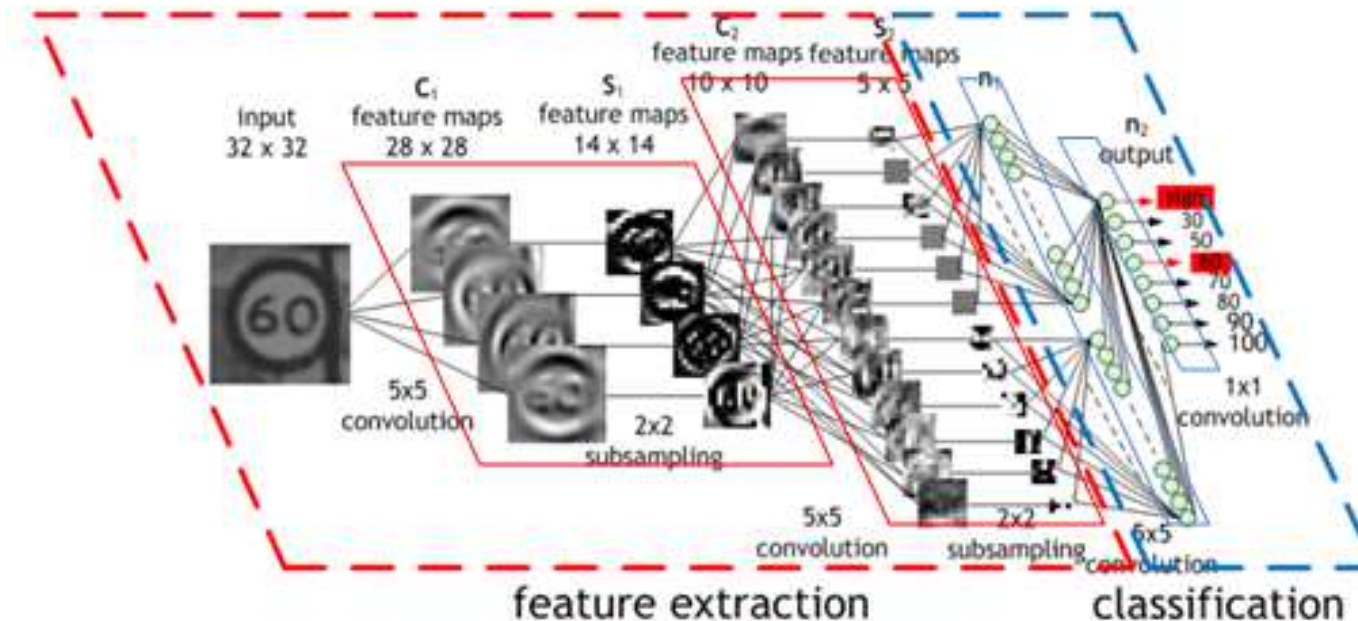
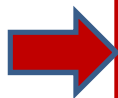


Figure 5: An image of a traffic sign is filtered by 4 5x5 convolutional kernels which create 4 feature maps, these feature maps are subsampled by max pooling. The next layer applies 10 5x5 convolutional kernels to these subsampled images and again we pool the feature maps. The final layer is a fully connected layer where all generated features are combined and used in the classifier (essentially logistic regression).

<https://devblogs.nvidia.com/parallelforall/deep-learning-nutshell-core-concepts/>

# Avantages sur le PMC classique

- ❑ Une connectivité locale et répliquée:
  - Moins de paramètres libres (et même beaucoup moins...)
  - Complexité qui n'explose pas avec la taille de l'image (y compris avec espace des couleurs)
  - Apprentissage facilitée des invariants
- ❑ Un mélange de couches à paramètres adaptatifs ou non
  - Architecture qui facilite l'introduction de connaissances a priori
  - La rétro-propagation du gradient s'adapte facilement, et peut se superposer à d'autres algorithmes



*Versatilité du modèle qui permet de fusionner approches neuronales et modèles « classiques »*

# Généralisation

## « orientée objet » et graphe

- ❑ Convolutional Neural Network = graphe acyclique de couches avec arcs paramétrés (généralisation du PMC)
- ❑ Chaque couche (ou groupe de couches) est généralement:
  - Une couche de convolution (notion de champs récepteurs)
  - Une couche de sous-échantillonnage (« pooling », « max-pooling », « min-pooling », ...)
  - Une couche de transformation/correction (ReLU, Tanh, ... -> connectivité locale)
  - Un perceptron ou un PMC, à connectivité totale
  - Une couche de sortie ( $\neq$  couche de sortie du PMC) => décision / perte
- ❑ Possibilité de traitements en parallèle

# Domaines d'application

## ❑ Deux types de réseaux à considérer:

- Convolutional Neural Networks (CNN): Généralisation du PMC avec des couches « non neuronales »
- Deep Neural Networks (DNN): PMC avec un grand nombre de couches

## ❑ Deux types d'entrées à considérer:

- Entrées sans topologie (une liste de variables explicatives)
  - DNN
  - Régression/Classification non linéaire pour des jeux de données avec un grand nombre de variables (domotique, marketing, actuariat, ...)
- Entrées avec une topologie (une image, un signal – la position relative des pixels/valeurs temporelles porte l'information)
  - CNN
  - Détection/Classification de patterns dans des images et des signaux (reconnaissance de forme, reconnaissance de la parole, interface homme-machine/EEG,...)

# Conclusion

# Limites de l'exposé

- ❑ Limité à l'apprentissage supervisé « basique » :
  - La fonction à optimiser peut être plus compliquée
  - Adaptation « assez simple » à l'apprentissage non supervisé – le modèle PMC ou DNN reste valable – cf *Restricted Boltzmann Machine*
  
- ❑ Généralisation avec des boucles
  - Boucles courtes (neurone connecté à lui-même)
    - > mémoire locale -> gérer l'oubli
  - Boucles longues (d'une couche avant vers une couche arrière)
    - > système dynamique -> gérer les problèmes de convergence
  - Adapter la rétro-propagation du gradient

# Quelques réflexions

## □ Une reconnaissance progressive:

- Au début, objet pour la physiologie, les sciences cognitives, puis l'IA
- Accélération avec l'appropriation par les statisticiens (le Machine-Learning) – exemple: inclusion dans R du PMC
- Nouvelle accélération avec l'appropriation par les grandes entreprises (GAFA, assurance/finance) – le POC AlphaGo

## □ Avec une tendance à « réinventer la roue »:

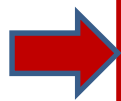
- Le « connexioniste » (informaticien, psychologue) touche à tout, sans connaître les méthodes d'autres domaines
- Exemple: méthodes d'apprentissage = méthodes classiques des numériciens, mais adaptées
- *Magie du RNA*, basée sur la conviction de l'IA faible: phénomène d'émergence

# Question ouverte

Un PMC est une boîte noire, un DNN encore plus !

Comment expliquer la décision du réseau ?

La non réponse à cette question est un problème lors de la mise en œuvre comme outil d'aide à la décision (frein à l'utilisation dans les entreprises...)



*Champ de recherche naissant... Le DNN pourrait bien être étudié comme le cerveau biologique !*

