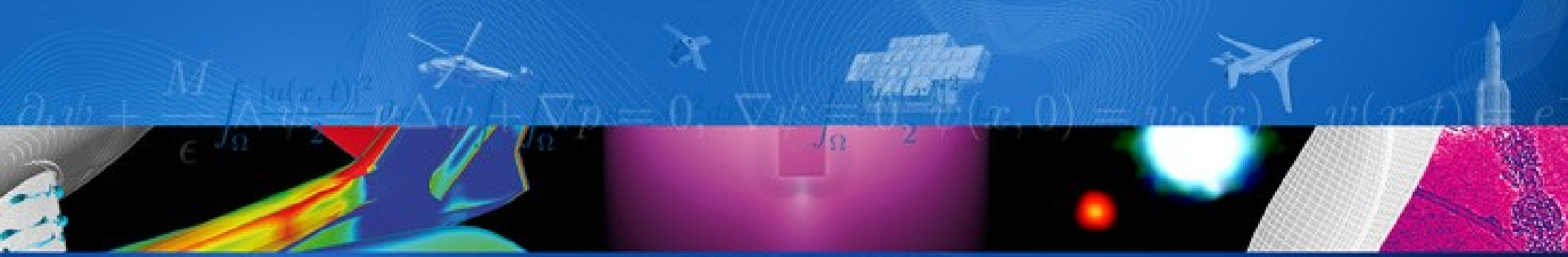


APPRENTISSAGE MACHINE & DEEP LEARNING

Deep Learning

A. Boulch, A. Chan Hon Tong, S. Herbin, B. Le Saux

ONERA

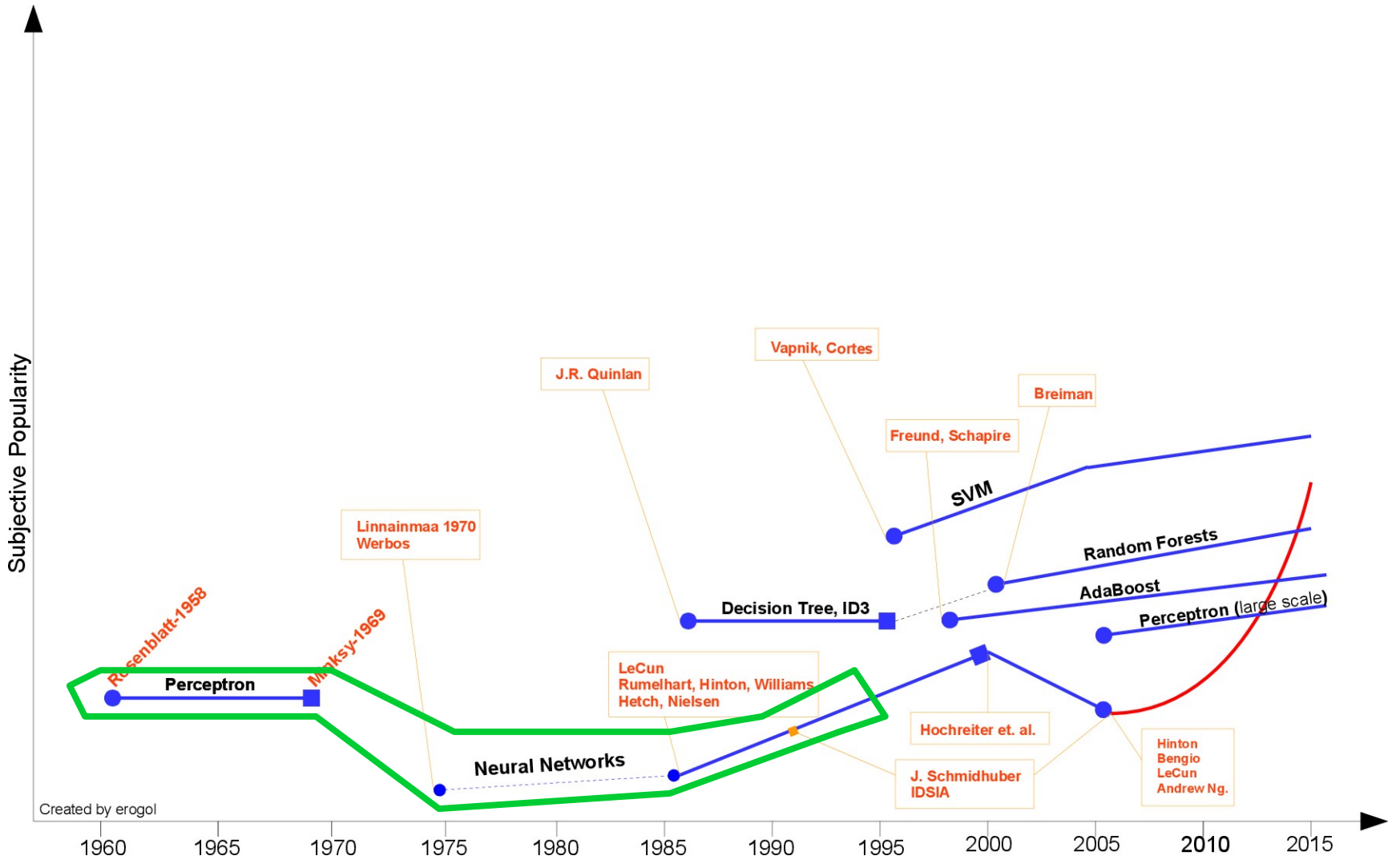
THE FRENCH AEROSPACE LAB

retour sur innovation

Deep Learning

- **Réseaux de neurones convolutifs**
- Principales difficultés
- Deep learning
  - Couches de régularisation
  - Initialisation
  - Optimiseurs
  - Les données
- Réseaux et applications

Réseaux de neurones



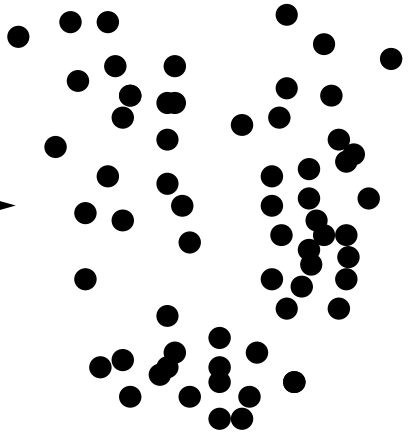
http://www.erogol.com/brief-history-machine-learning/

Réseaux de neurones

ONERA
THE FRENCH AEROSPACE LAB

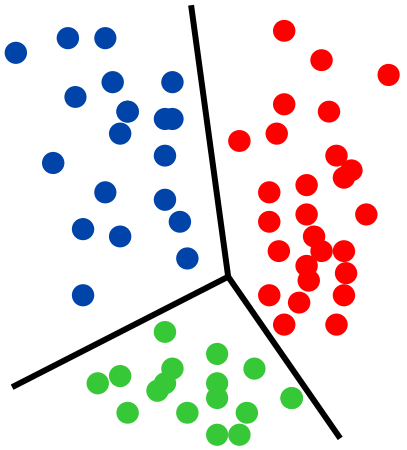
Données

Caractéristiques
bien pensées



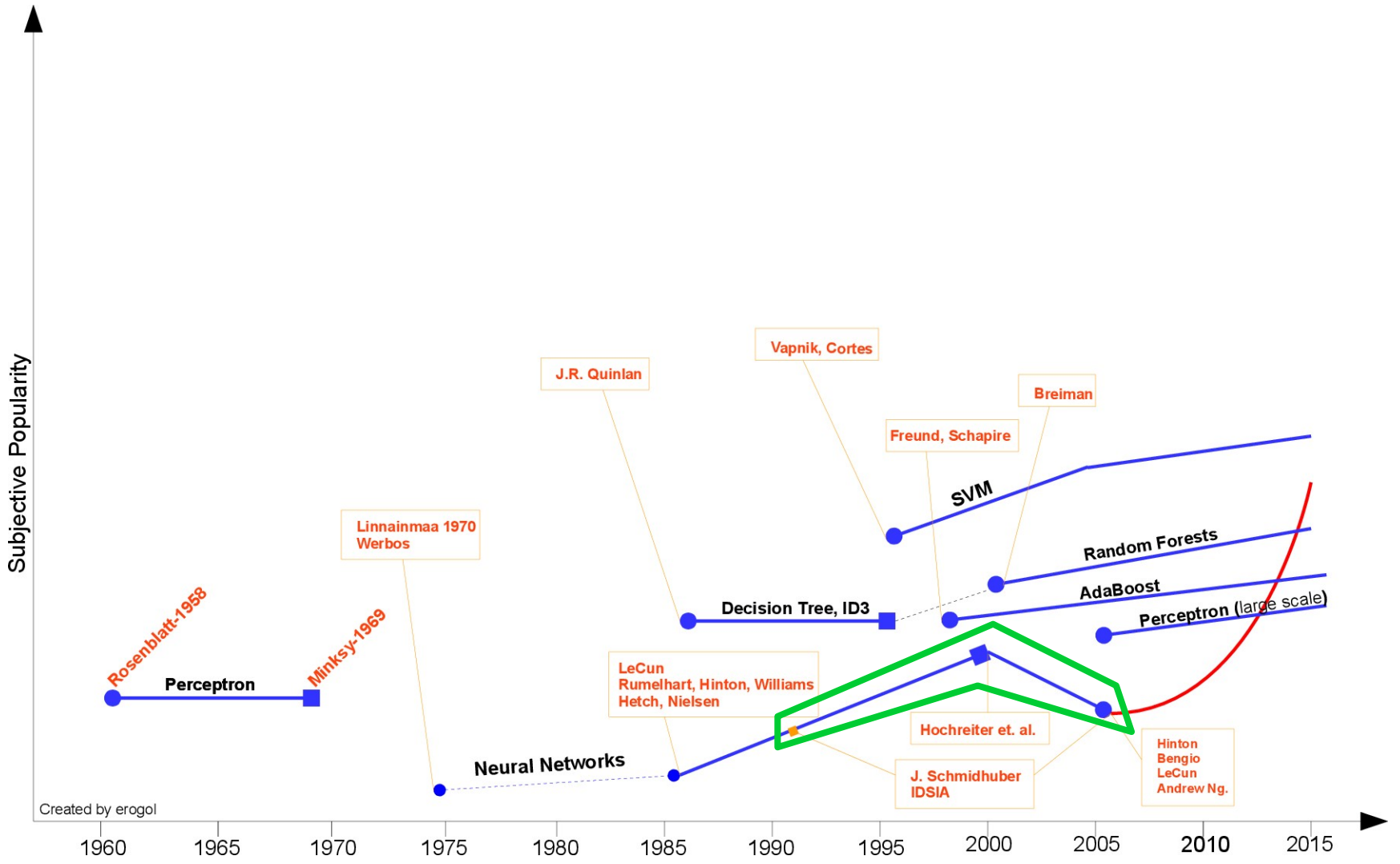
Espace
adapté

Apprentissage
Expertise



Objectif

Réseaux de neurones



http://www.erogol.com/brief-history-machine-learning/

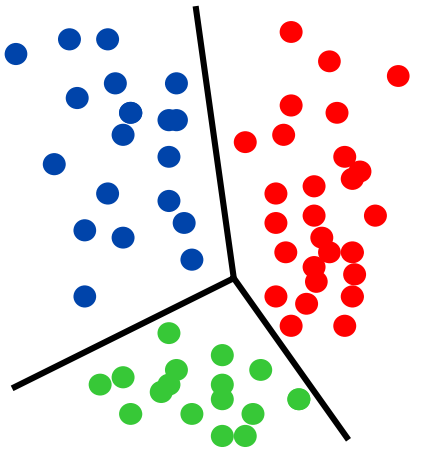
Deep learning 90's → 2005

ONERA
THE FRENCH AEROSPACE LAB

- Mise en place des premiers réseaux convolutifs

Données

Réseaux de neurones profonds



Objectif

Recherche sur les architectures, les stratégies, la mise en forme des données

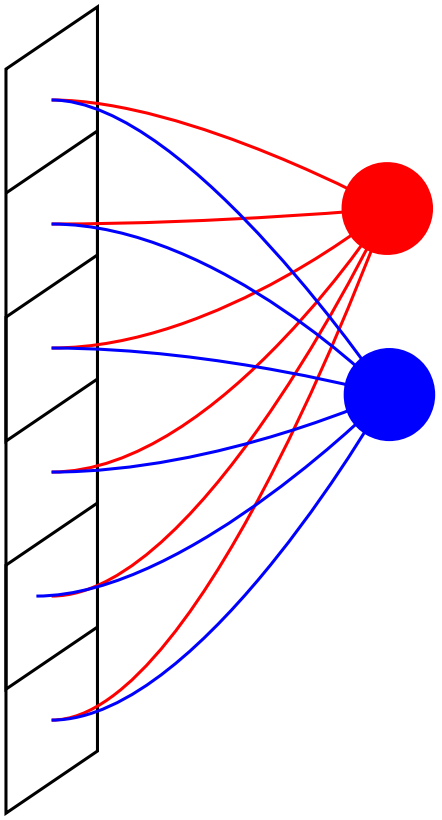
Fully connected

Perceptron

(présentation précédente)

Un neurone est connecté à toutes les entrées

Si dimension d’entrée grande, beaucoup de paramètres



Fully connected

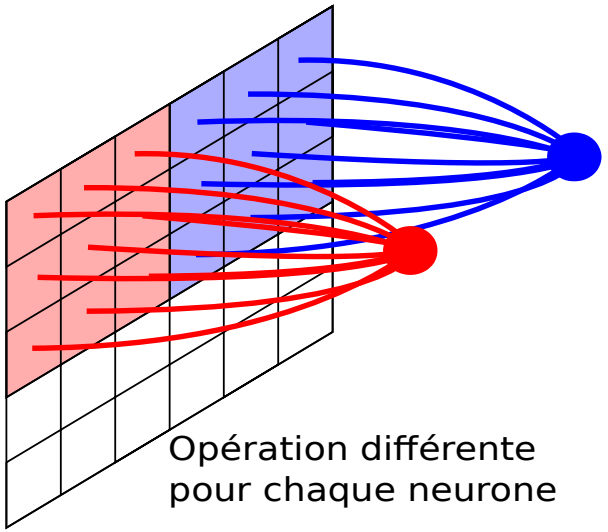
ONERA
THE FRENCH AEROSPACE LAB

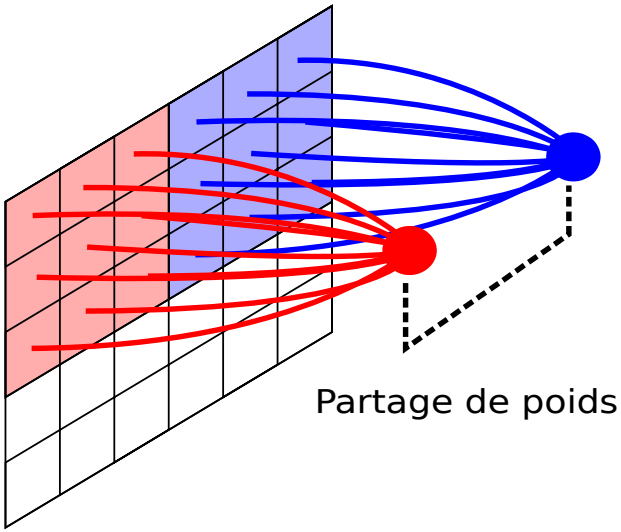
- MLP de plus en plus profonds (avant 2005)
  - Très grosses difficultés d'optimisation
  - Convergence difficile
  - Peu de données
  - Entraînement très long

    ⇒ Abandon progressif au profit des SVMs
    - Simples à utiliser
    - Preuves de convergence
    - Rapides

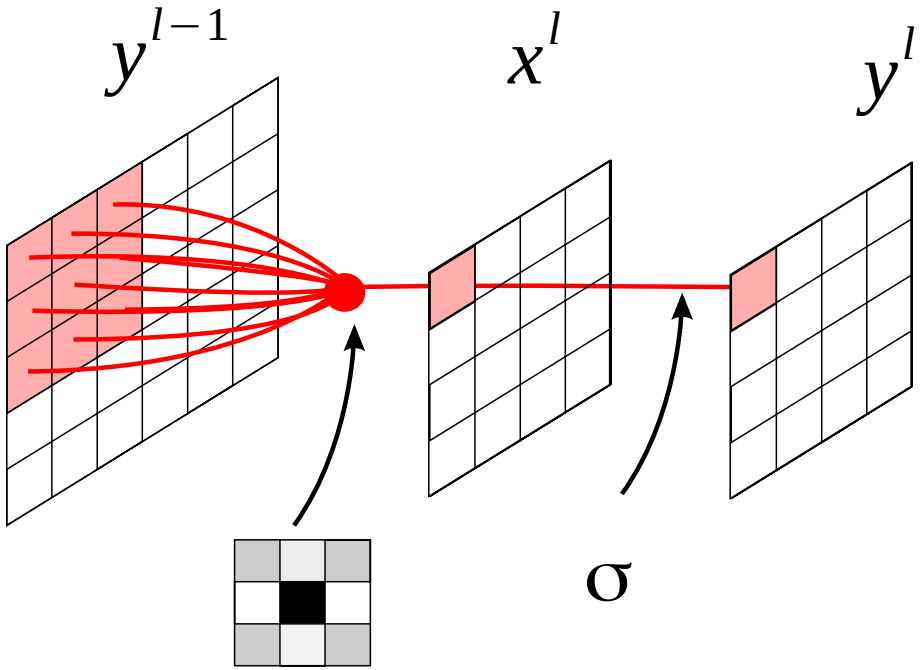
Réseaux convolutifs

Adapté aux données structurées





Convolution



Forward

$$x^{l}_{i,j} = \sum_{a} \sum_{b} \omega_{a,b}\, y^{l-1}_{i+a,j+b}$$

$$y^{l}_{i,j} = \sigma\left(x^{l}_{i,j}\right)$$

Convolution

Backward, « Delta rule » → mise à jour des poids de la convolution :

$$\frac{\partial E}{\partial \omega_{a,b}} = \sum_i \sum_j \frac{\partial E}{\partial x^l_{i,j}} \frac{\partial x^l_{i,j}}{\partial \omega_{a,b}} = \sum_i \sum_j \frac{\partial E}{\partial x^l_{i,j}} y^{l-1}_{i,j}$$

or

$$\frac{\partial E}{\partial x^l_{i,j}} = \frac{\partial E}{\partial y^l_{i,j}} \frac{\partial y^l_{i,j}}{\partial x^l_{i,j}} = \frac{\partial E}{\partial y^l_{i,j}} \sigma'\left(x^l_{i,j}\right)$$

$$\frac{\partial E}{\partial \omega_{a,b}} = \sum_i \sum_j \frac{\partial E}{\partial y^l_{i,j}} \sigma'\left(x^l_{i,j}\right) y^{l-1}_{i,j}$$

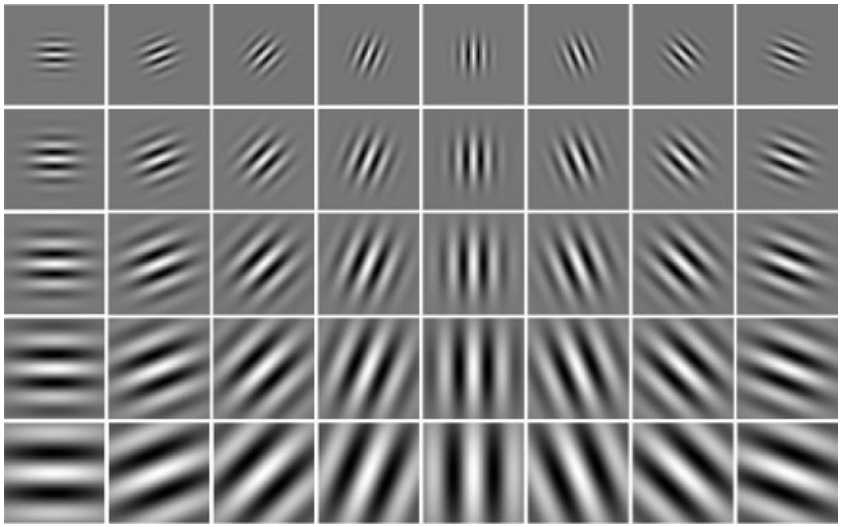
Convolution

Backward, « Chain rule » → erreur

$$\frac{\partial E}{\partial y_{i,j}^{l-1}} = \sum_a \sum_b \frac{\partial E}{\partial x_{i-a,j-b}^{l}} \frac{\partial x_{i-a,j-b}^{l}}{\partial y_{i,j}^{l-1}}$$

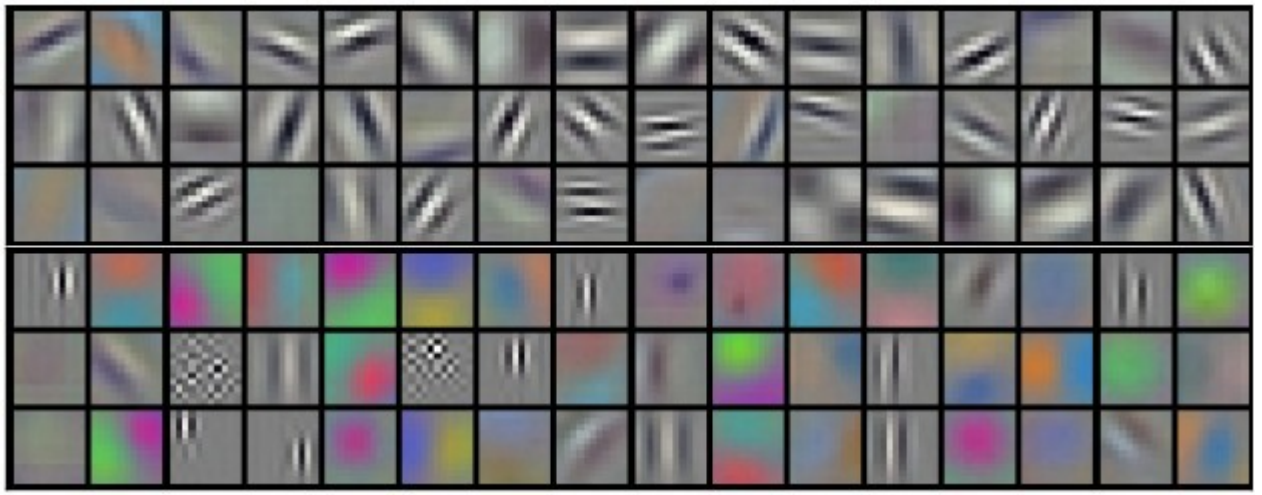
$$= \sum_a \sum_b \frac{\partial E}{\partial x_{i-a,j-b}^{l}} \omega_{a,b}$$

Couches : convolution



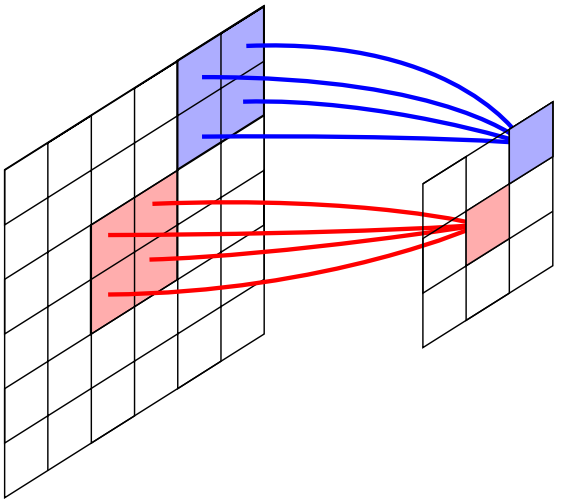
Filtres de Gabor

Exemple poids appris par la première couche de convolution d'AlexNet

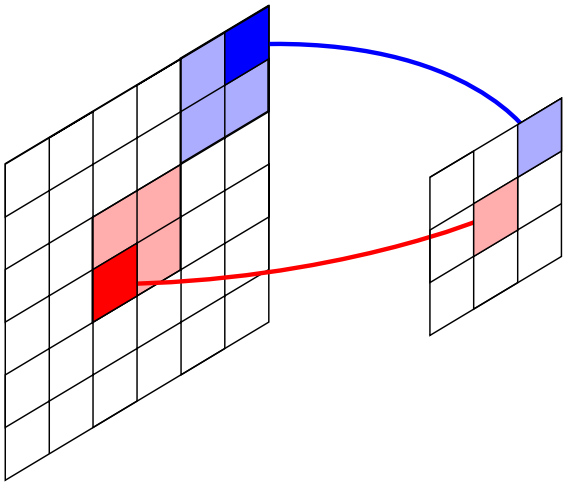


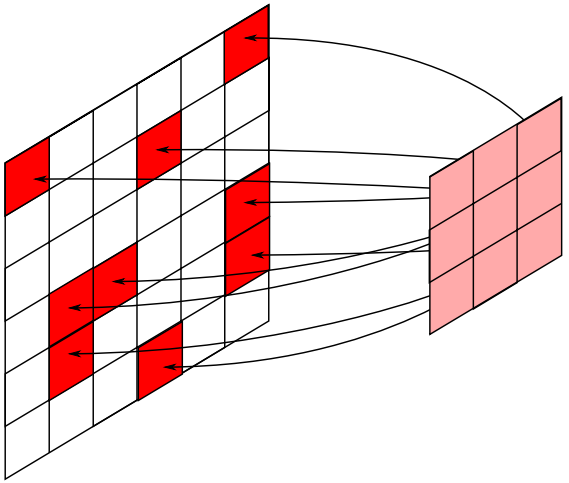
Couches : Pooling

- Réduction de dimension
- Invariabilité en translation
- Mise en correspondance des pixels voisins
  - Réduction de la profondeur



Couches : Max Pooling





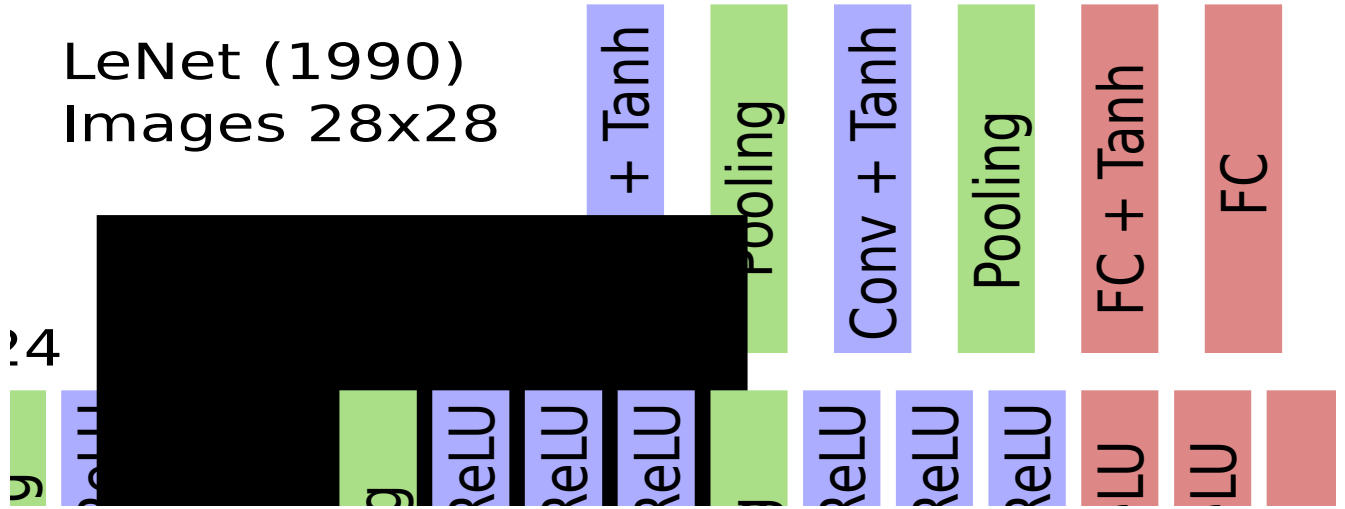
Forward

Transmission du maximum sur une fenêtre donnée

Backward

Transmission du gradient au maximum identifié, gradient nul sinon

LeNet



Gradient-based learning applied to document recognition, LeCun, Bottou, Bengio and Haffner, 1998