Apprentissage Statistique

Du Perceptron au *Deep Learning*

PHILIPPE BESSE

INSA de Toulouse
Institut de Mathématiques

Intelligence Artificielle

- 1943 que Mc Culloch et Pitts *neurone formel*
- 1959 Rosenblatt *perceptron*
- 1970 Approche *symbolique* *vs* *connexioniste*
- Connaissance localisée *vs* répartie

Systèmes experts

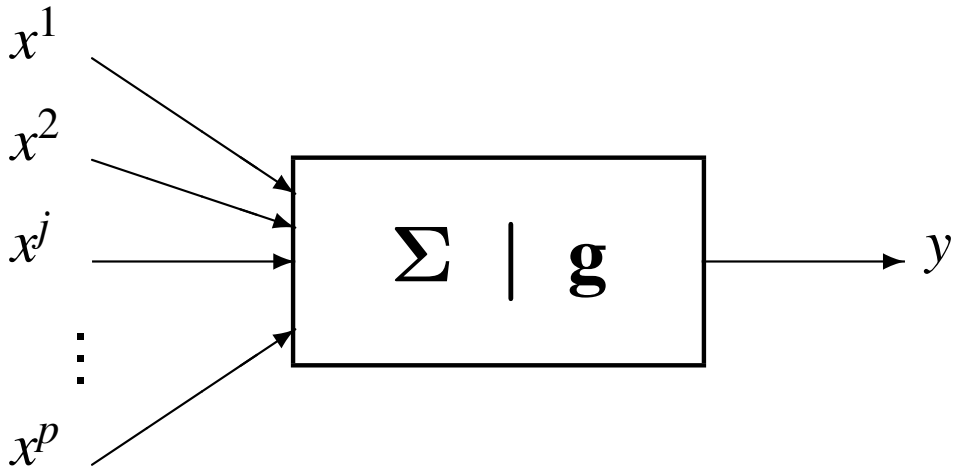
- Base de connaissance, base de faits
- Moteur d'inférence

Réseaux de neurones

- Années 80 :
  - Algorithme de rétropropagation de l'erreur
  - Modèle markovien d'apprentissage
  - Développement considérable
- Années 90 : mise en veilleuse (boosting, SVM...)
- 2010 ; le retour : *deep learning*

Définition

- Un réseau est un graphe de *neurones formels* se distinguant par le type des neurones et l'architecture
- Analogie biologique avec les axones, dendrites et noyaux.



Représentation d'un neurone formel

Notations

- $s = h(x_1, \ldots, x_p) = g\left(\alpha_0 + \sum_{j=1}^{p} \alpha_j x_j\right) = g(\alpha_0 + \boldsymbol{\alpha}'\boldsymbol{x})$
- $[\alpha_0, \ldots, \alpha_p]$: vecteur de poids
- Mémoire ou connaissance répartie du réseau

Fonction d'activation d'un neurone

Linéaire $g(x) = x$ (identité)

Seuil $g(x) = \mathbf{1}_{[0,+\infty[}(x)$

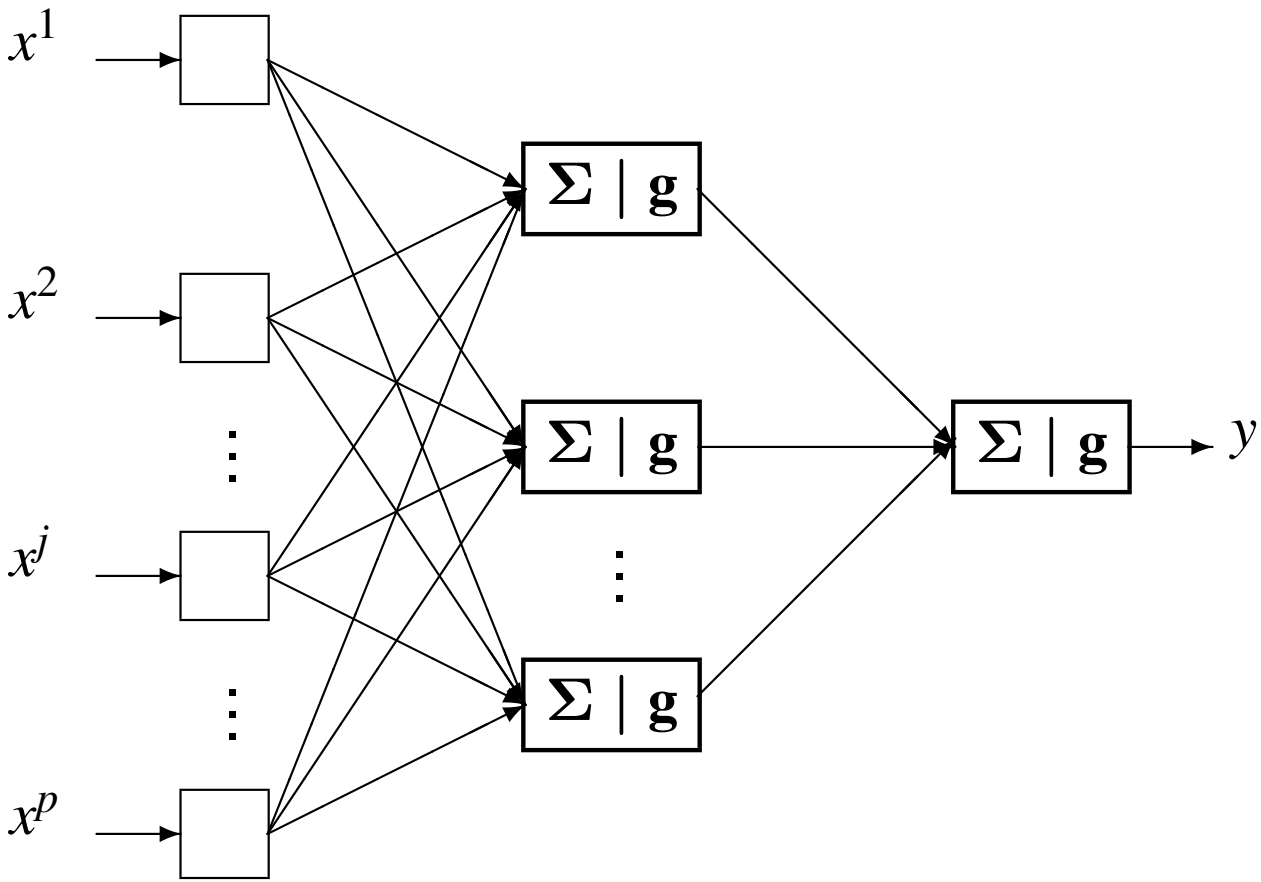
Sigmoïde $g(x) = 1/(1+e^x)$

ReLu $g(x) = \max(0, x)$ (REctified Linear unit)

softmax $g(x)_j = \frac{e^{x_j}}{\sum_{k=1}^{K} e^{x_k}}$ pour tout $k \in \{1 \dots K\}$

Radiale $g(x) = \sqrt{1/2\pi} e^{-x^2/2}$

Stochastique $g(x) = 1$ avec probabilité $1/(1+e^{-x/H})$
sinon $g(x) = 0$



Perceptron élémentaire avec une couche cachée et une couche de sortie.

Fonction de transfert du réseau

- $Y = f(X^1, \ldots, X^p; \boldsymbol{\alpha})$
- $\boldsymbol{\alpha}$: $\alpha_{jk\ell}$ *poids* (paramètre) de la $j$ème entrée du $k$ème neurone de la ℓème couche.
- $X^1, \ldots, X^p$: *entrées* (variables explicatives)
- Y : *sortie* (variable à expliquer) ou cible du modèle
- *apprentissage* = estimation
- Théorème d'approximation universelle
- Y est quantitative, qualitative à une ou plusieurs classes
- Exemple en régression

$$\begin{aligned} y = f(\boldsymbol{x}; \boldsymbol{\alpha}, \boldsymbol{\beta}) &= \beta_0 + \boldsymbol{\beta}' \boldsymbol{z} \\ \text{avec} \quad z_k &= g(\alpha_{k0} + \boldsymbol{\alpha_k}' \boldsymbol{x}); k = 1, \ldots, q \end{aligned}$$

Estimation par moindres carrés

- Observations : $(x_i^1, \ldots, x_i^p; y_i) \quad i = 1, \ldots, n$
- $Q(\alpha, \beta) = \sum_{i=1}^{n} Q_i = \sum_{i=1}^{n} [y_i - f(\boldsymbol{x}_i; \boldsymbol{\alpha}, \boldsymbol{\beta})]^2$
- Avec $\boldsymbol{\alpha}_{j=0,p;k=1,q}$ et $\boldsymbol{\beta}_{k=0,q}$
- Minimisation de Q ou d'une autre fonction (entropie, nombre de mal classés)

Calcul du gradient par rétropropagation

- Soit $z_{ki} = g(\alpha_{k0} + \boldsymbol{\alpha}_k' \boldsymbol{x}_i)$ et $\boldsymbol{z}_i = \{z_{1i}, \ldots, z_{qi}\}$

$$\begin{aligned} \frac{\partial Q_i}{\partial \beta_k} &= -2(y_i - \phi(x_i))(\boldsymbol{\beta}' \boldsymbol{z}_i) z_{ki} = \delta_i z_{ki} \\ \frac{\partial Q_i}{\partial \alpha_{kj}} &= -2(y_i - \phi(x_i))(\boldsymbol{\beta}' \boldsymbol{z}_i) \beta_k g'(\boldsymbol{\alpha}_k' \boldsymbol{x}_i) x_{ip} = s_{ki} x_{ip}. \end{aligned}$$

- δ_i et s_{ki} : termes d'erreur en sortie et sur chaque neurone
- avec $s_{ki} = f'(\boldsymbol{\alpha}_k' \boldsymbol{x}_i) \beta_k \delta_i$
- Évaluation en deux passes, avant puis retour

Algorithmes d'optimisation

- Algorithme itératif :

$$\beta_k^{(r+1)} = \beta_k^{(r)} - \tau \sum_{i=1}^{n} \frac{\partial Q_i}{\partial \beta_k^{(r)}}$$

$$\alpha_{kp}^{(r+1)} = \alpha_{kp}^{(r)} - \tau \sum_{i=1}^{n} \frac{\partial Q_i}{\partial \alpha_{kp}^{(r)}}.$$

- Taux d'appentissage : τ
- Second ordre : BFGS, Levenberg-Marquardt, gradient conjugué
- Variantes avec inertie, adaptative...
- Attention à la convergence : optimum local
- Possibilité : *batch* d'observations à chaque itération

Algorithme de rétropropagation du gradient

- **Initialisation**
- Poids $b_{jk\ell}$ uniforme sur $[0, 1]$
- Normaliser dans $[0, 1]$ $x^1, \ldots, x^p; y$
- **tant que** $Q >$ errmax ou niter $<$ itermax
  - Ordre aléatoire de l'échantillon d'apprentissage
  - **pour** $i = 1 \ldots n$
    - $\varepsilon(i) = y_i - \phi(x_i^1, \ldots, x_i^p; (b)(i-1))$
    - $b_{jk\ell}(i) = b_{jk\ell}(i-i) + \Delta b_{jk\ell}(i)$ pour tout j, k, l
  - **fin pour**
- **fin tant que**

Paramètres et complexité du modèle

- Architecture du réseau : nombre de paramètres
- Nombre maximum d'itérations ou erreur maximum tolérée
- Coefficient de pénalisation (*decay*) : $Q(\boldsymbol{\theta}) + \gamma \|\theta\|^2$
- Taux d'apprentissage et stratégie d'évolution
- Taille des *batchs*
- ...

Utilisation

- Champs d'application nombreux
- Principales critiques
  - Difficultés d'apprentissage
  - Taille et temps de l'apprentissage
  - Boîte noir

Cancer du sein / carte visa

- Matrice de confusion pour l'échantillon test

	benign	malignant
FALSE	83	1
TRUE	3	50

	FALSE	TRUE
FALSE	110	16
TRUE	27	47

- Taux d'erreur estimée à 3% et 21,5%

Concentration d'ozone

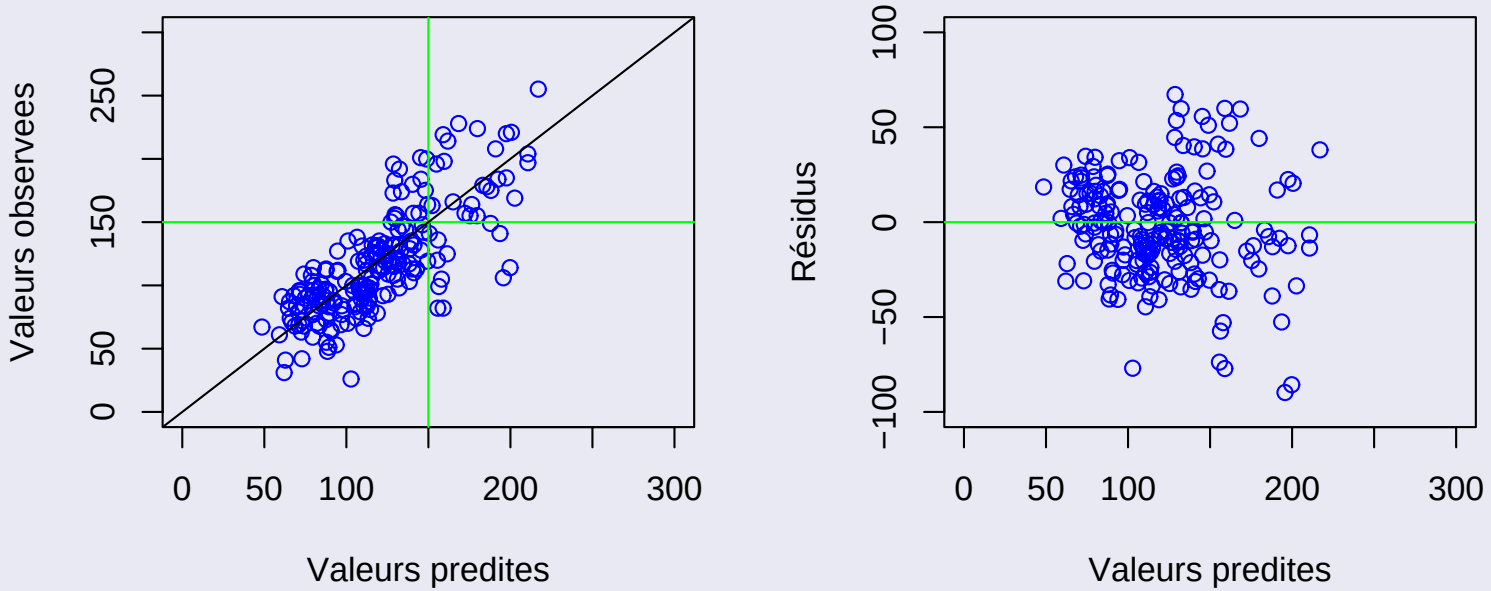
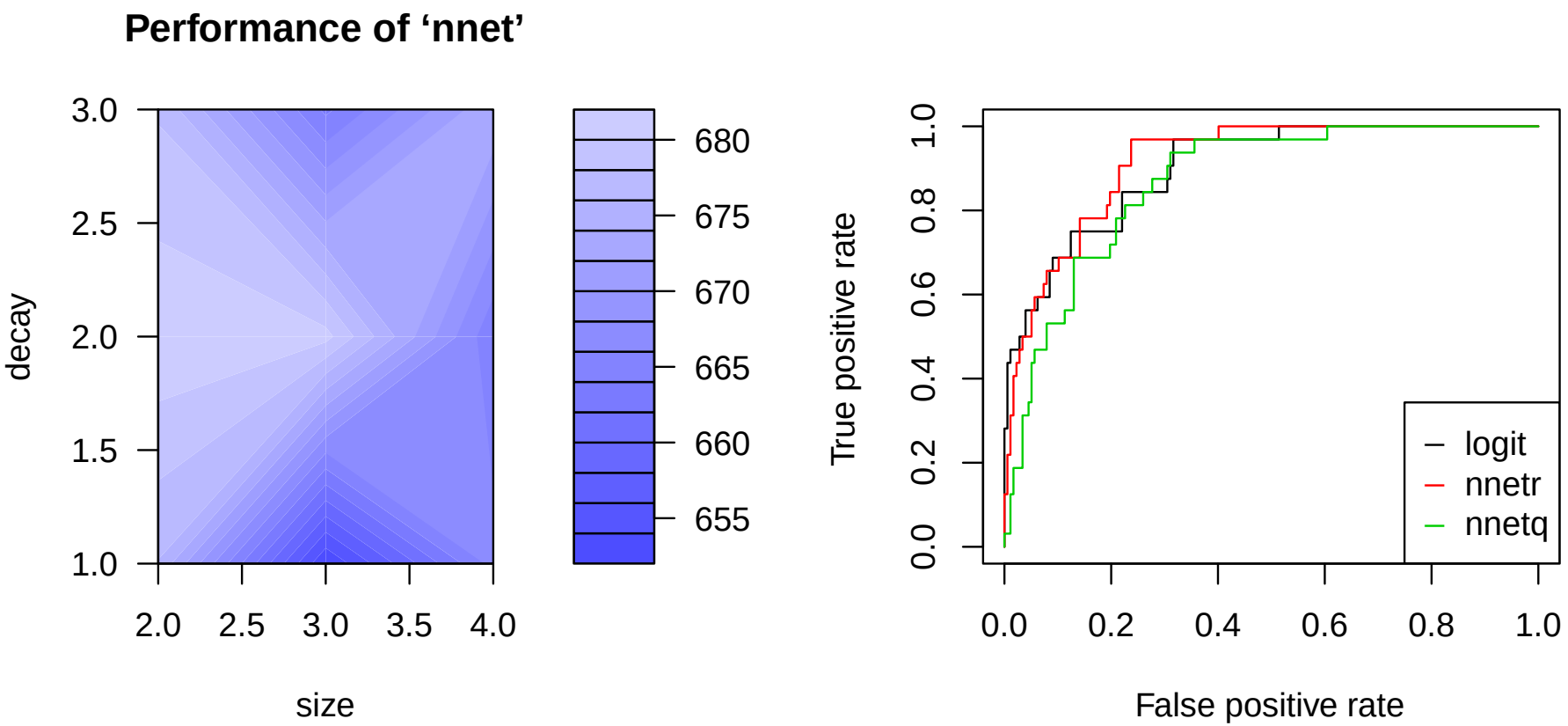


FIGURE – Ozone : Valeurs observées et résidus de l'échantillon test

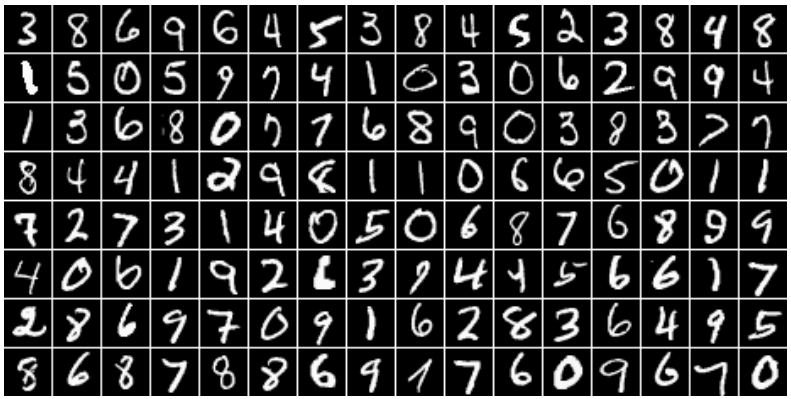
Taux d'erreur de 14,4% (quantitatif) et 15,6% (qualitatif).



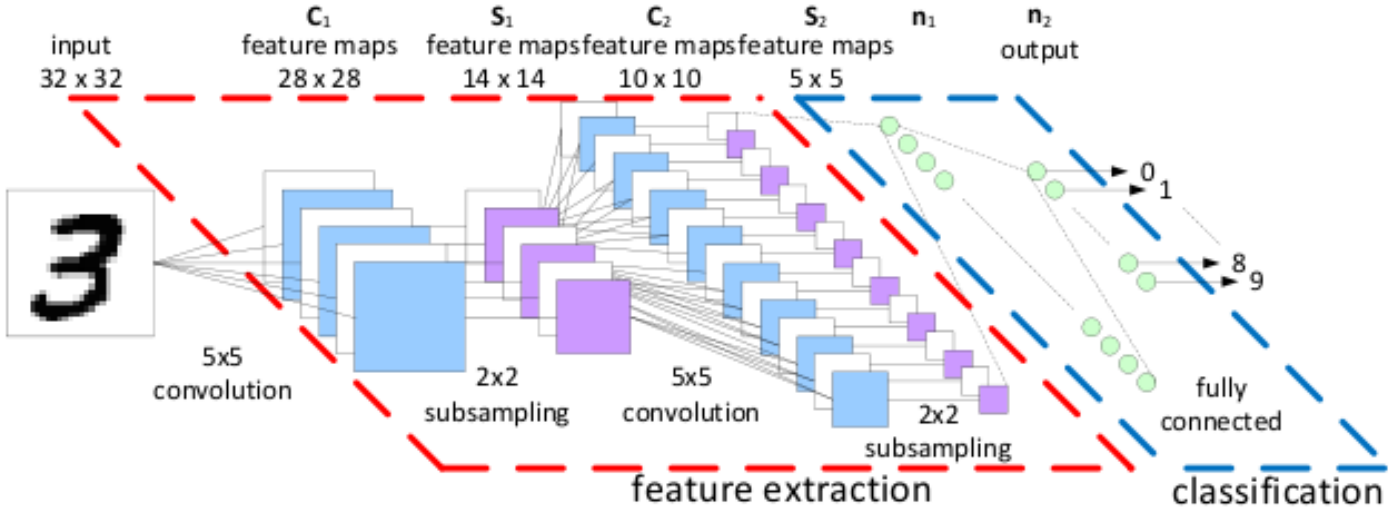
Ozone : optimisation des réseaux et courbes ROC

Pourquoi *deep learning*

- Yan le Cun : MNIST, de 12% (1989) à 0.3% (2012)
- *Convolutional neural network* (ConvNet)
- Bases de données
- Puissance de calcul (*GPU*)
- Logiciels *Caffe, Torch, Tensorflow, Theano, Keras...*
- Applications vedettes des réseaux :
  ConvNet, LSTM, AutoEncoder...
  - Traitement d'images : reconnaissance d'objets
  - Signal : reconnaissance de la parole
  - Traduction automatique



MNIST Database (Le Cun, 1989)



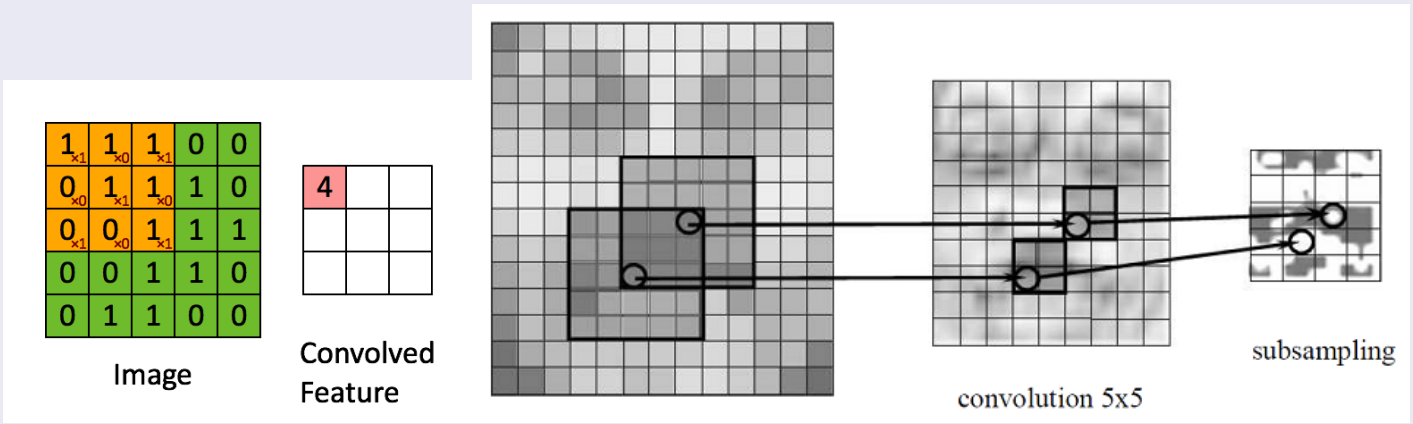
Couches de réseau de convolution

ImageNet Database

- Concours chaque année depuis 2010
- 15 millions d'images avec labels, 22000 catégories
- Sous-ensemble : 1,2 millions d'images, 1000 catégories
- 50 000 images de validation, 150 000 de test

Couche de convolution

- Propriétés d'invariance
- Translation, rotation, homothétie,
- *Scattering* et bases d'ondelettes (S. Mallat)



Premiers succès

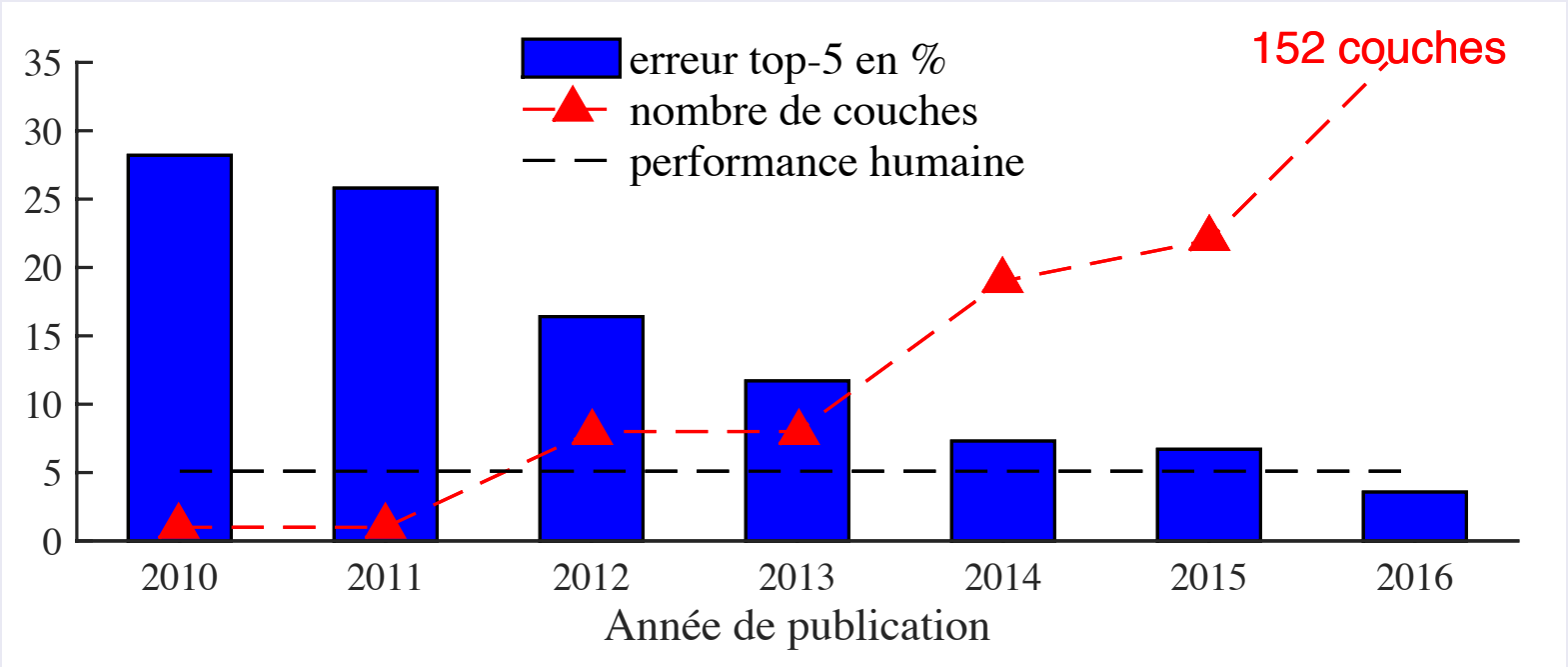
2012 Teams	%error
Supervision (Toronto)	15.3
ISI (Tokyo)	26.1
VGG (Oxford)	26.9
XRCE/INRIA	27.0
UvA (Amsterdam)	29.6
INRIA/LEAR	33.4

2013 Teams	%error
Clarifai (NYU spinoff)	11.7
NUS (singapore)	12.9
Zeiler-Fergus (NYU)	13.5
A. Howard	13.5
OverFeat (NYU)	14.1
UvA (Amsterdam)	14.2
Adobe	15.2
VGG (Oxford)	15.2
VGG (Oxford)	23.0

2014 Teams	%error
GoogLeNet	6.6
VGG (Oxford)	7.3
MSRA	8.0
A. Howard	8.1
DeeperVision	9.5
NUS-BST	9.7
TTIC-ECP	10.2
XYZ	11.2
UvA	12.1

Y. le Cun, tutoriel StatLearn

Mieux que l'expert humain



Principales couches

- *Fully connected* : *softmax*
- *Convolution* neural network
- *Max pooling*
- *Normalisation*
- *Drop out*
- *LSTM* ou réseau récurrent
- ...

Utilisation rudimentaire

- Sans base de données démesurée ni *google cloud*
- Choisir un réseau proche de l'objectif et déjà appris
- *Inception* de *tensorFlow* ou *AlexNet* de *Caffe*
- Supprimer la dernière couche *fully conected*
- Apprendre la dernière couche sur le nouveau problème ou
- Lui substituer une autre méthode d'apprentissage