

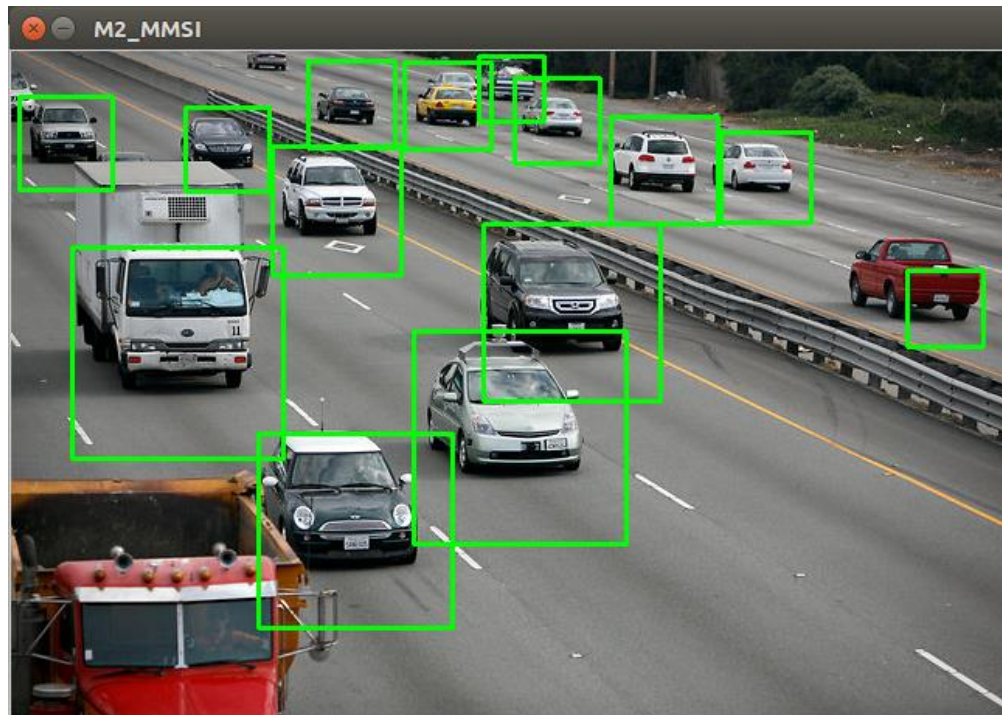


# Réseaux de Neurones et Traitement d'Images en Temps Réel

Brahim Yahiaoui  
[byahiaoui@nexyad.net](mailto:byahiaoui@nexyad.net)

# Présentation du cours

- Introduction et prise en main de OpenCV
- Les réseaux de neurones dans l'image
- Traitements des signaux en temps réel





# Les réseaux de neurones dans l'image



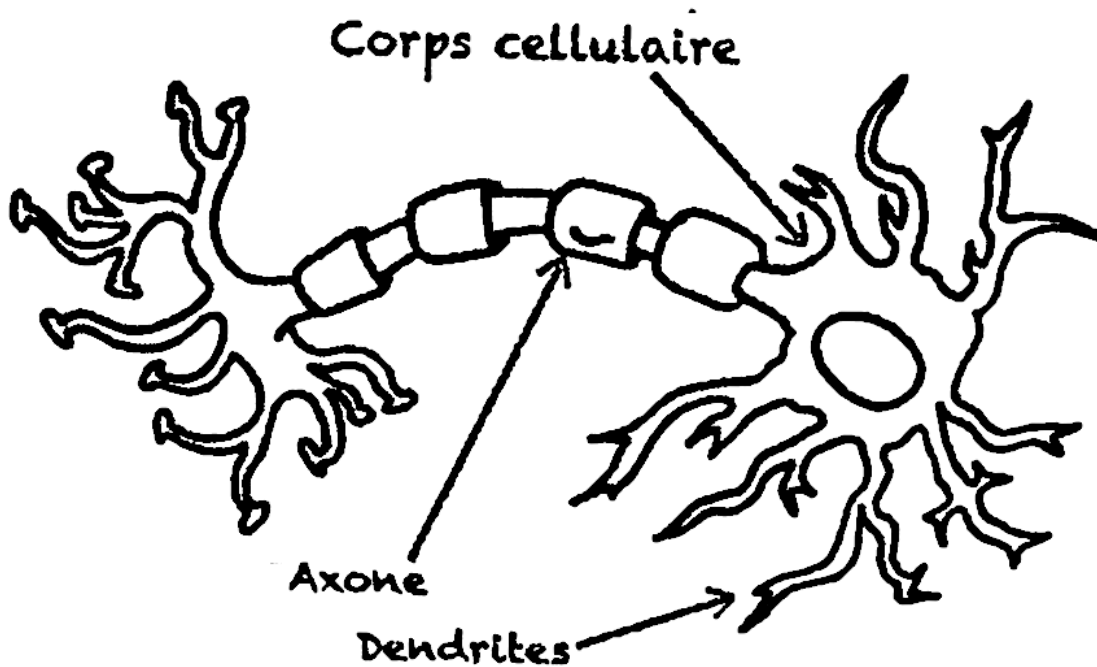
# Introduction aux réseaux de neurones

# Les réseaux de neurones

- Méthode inspirée du fonctionnement des neurones biologiques
- Utilisés pour élaborer des algorithmes capables d'apprendre
- Domaines d'application :
  - Reconnaissance optique de caractères (OCR)
  - Biologie : Classifications d'espèces animales
  - Economie : Classification d'entreprises et prédiction d'échec
  - Réseaux sociaux : Analyse sémantique de phrases sur internet pour la publicité ciblée ou la sécurité
  - Vision robotique : Détection d'obstacles et étiquetage

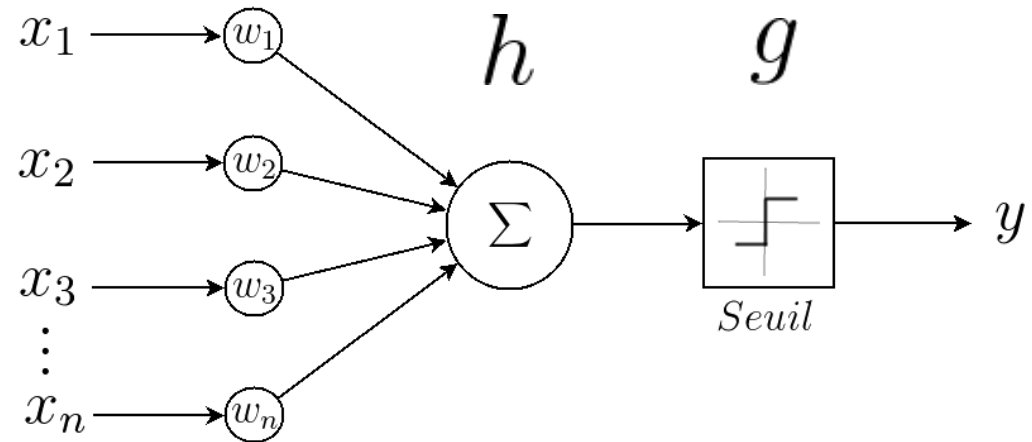
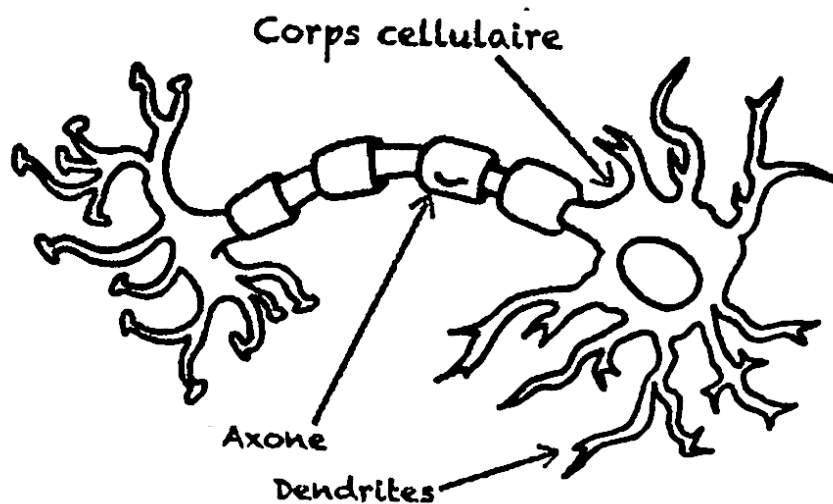
# Modèle du neurone formel

- Modélisation par McCulloch et Pitts (1943) d'un neurone



# Modèle du neurone formel

- Modélisation **mathématique** par McCulloch et Pitts (1943) d'un neurone **par un perceptron dit simple**



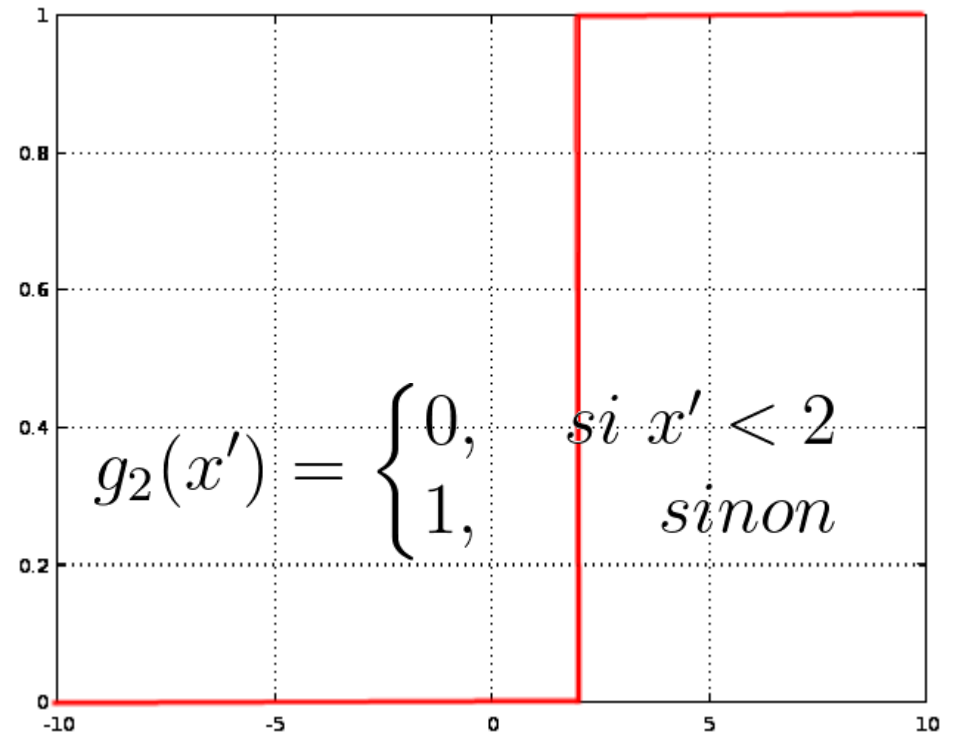
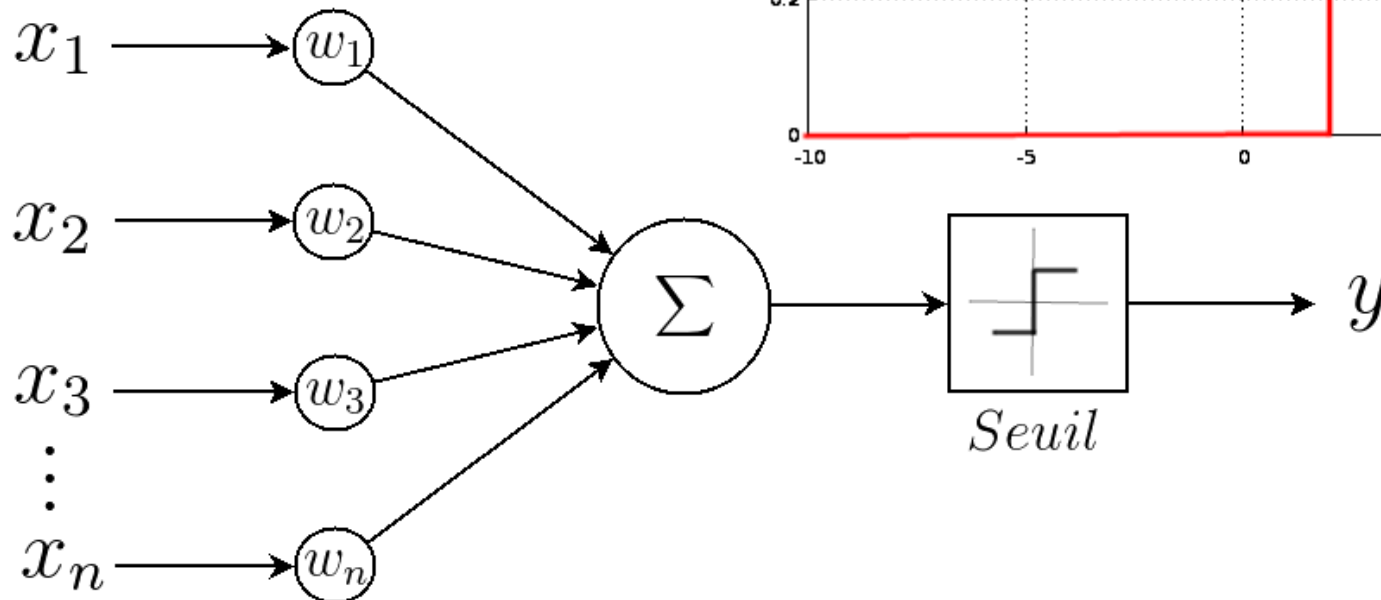
$$y = g(h(X, W)) = g \left( \sum_{i=1}^n w_i x_i \right)$$

# Fonction d'activation (seuil)

Posons  $x' = h(X, W)$

$$y = g_s(x') = g_s \left( \sum_{i=1}^n w_i x_i \right)$$

où  $g_s$  est la fonction de Heaviside, et  $s$  le seuil.



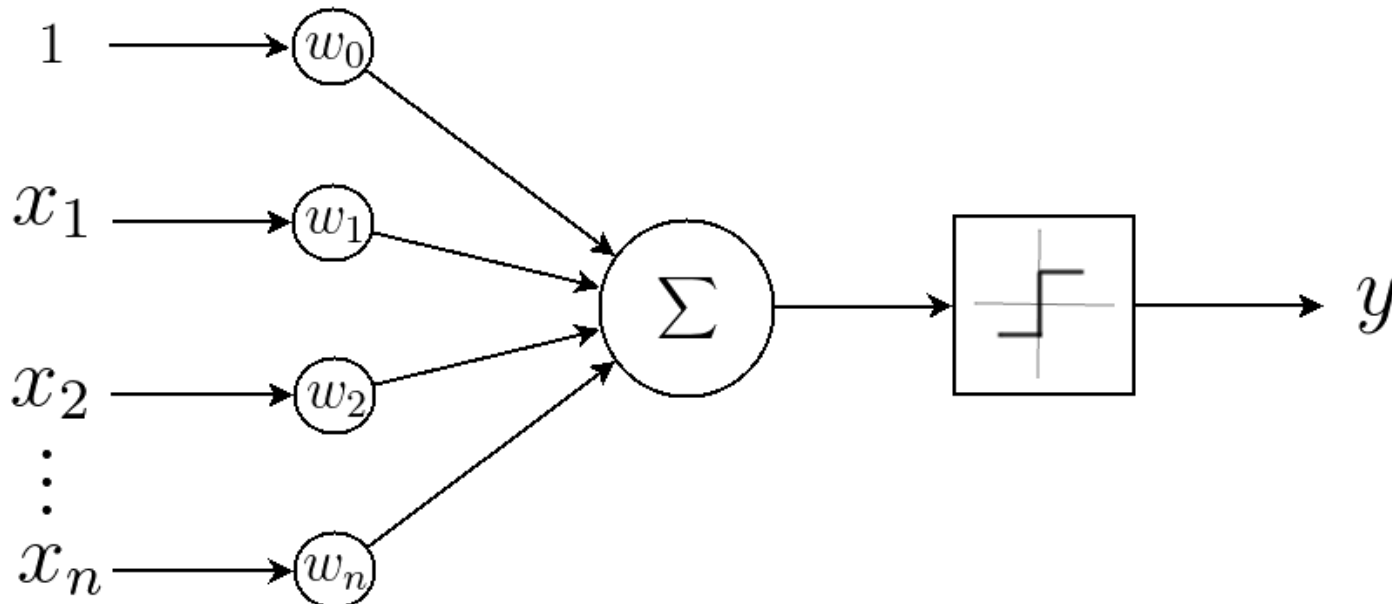


# Fonction d'activation (seuil)

$$g_s(x') = g_0 \left( \sum_{i=1}^n w_i x_i + s \right)$$

En posant  $w_0 = s$  et  $x_0 = 1$

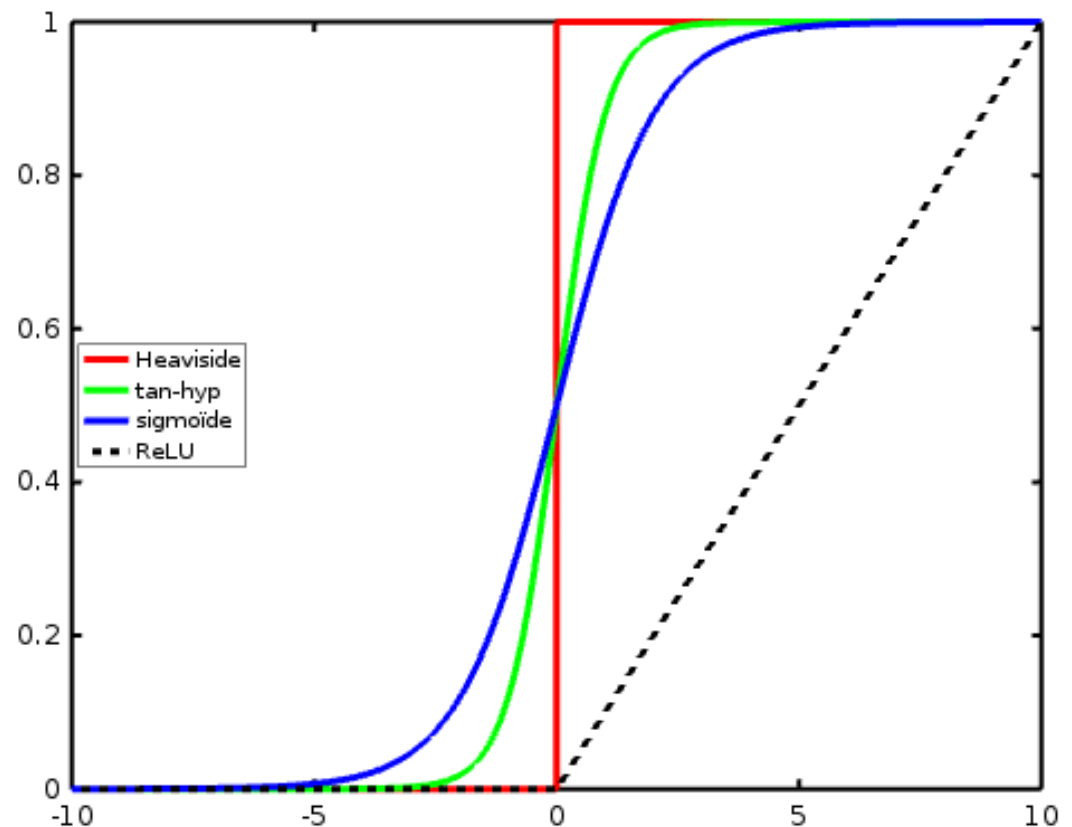
$$g_s(x') = g_0 \left( \sum_{i=0}^n w_i x_i \right)$$



# Fonction d'activation

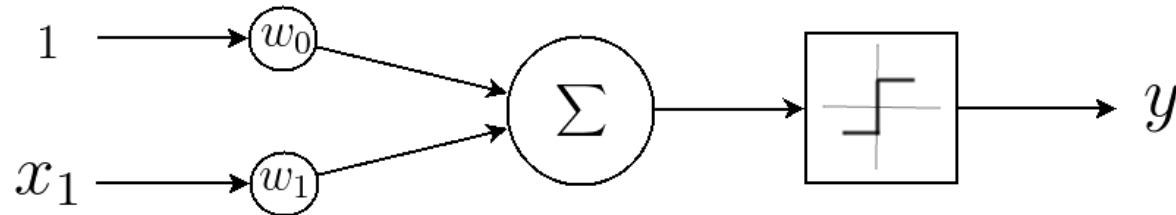
Les fonctions suivantes peuvent être utilisées comme fonction d'activation resp. par ordre de recommandation :

1. La fonction de Heaviside
2. La tangente hyperbolique
3. La fonction sigmoïde
4. La fonction rectifieur  
ReLU  $\max(0, x)$

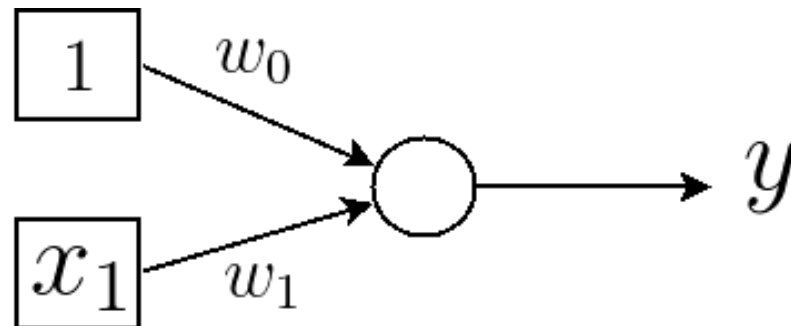


# Représentation d'un neurone en théorie des graphes

- Nous utiliserons à partir de ce slide la nouvelle représentation du neurone
- La représentation :

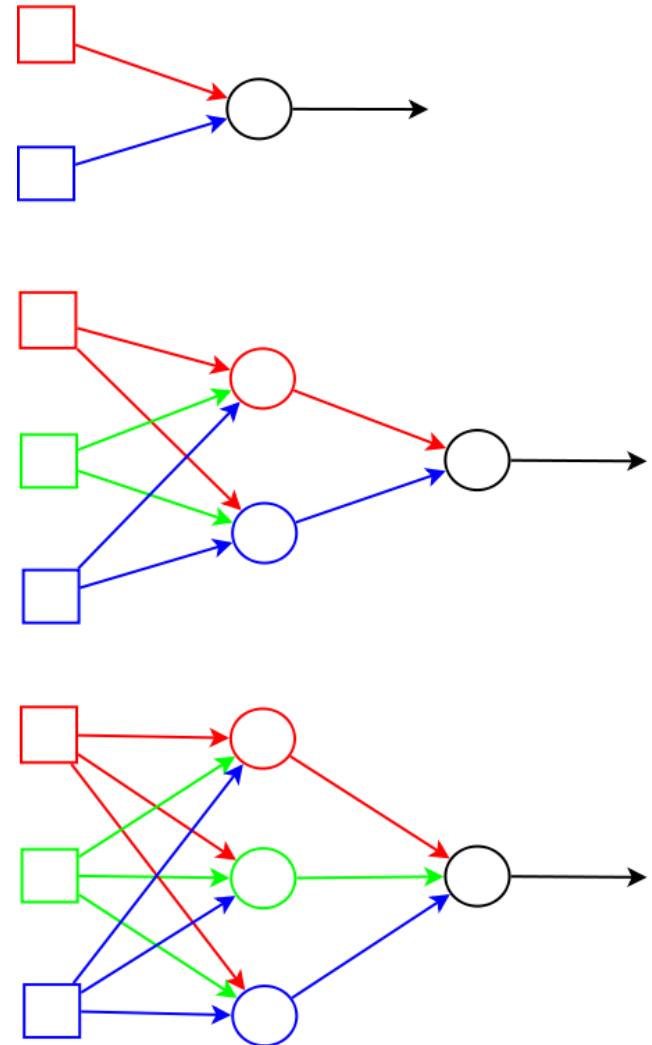


- Devient :



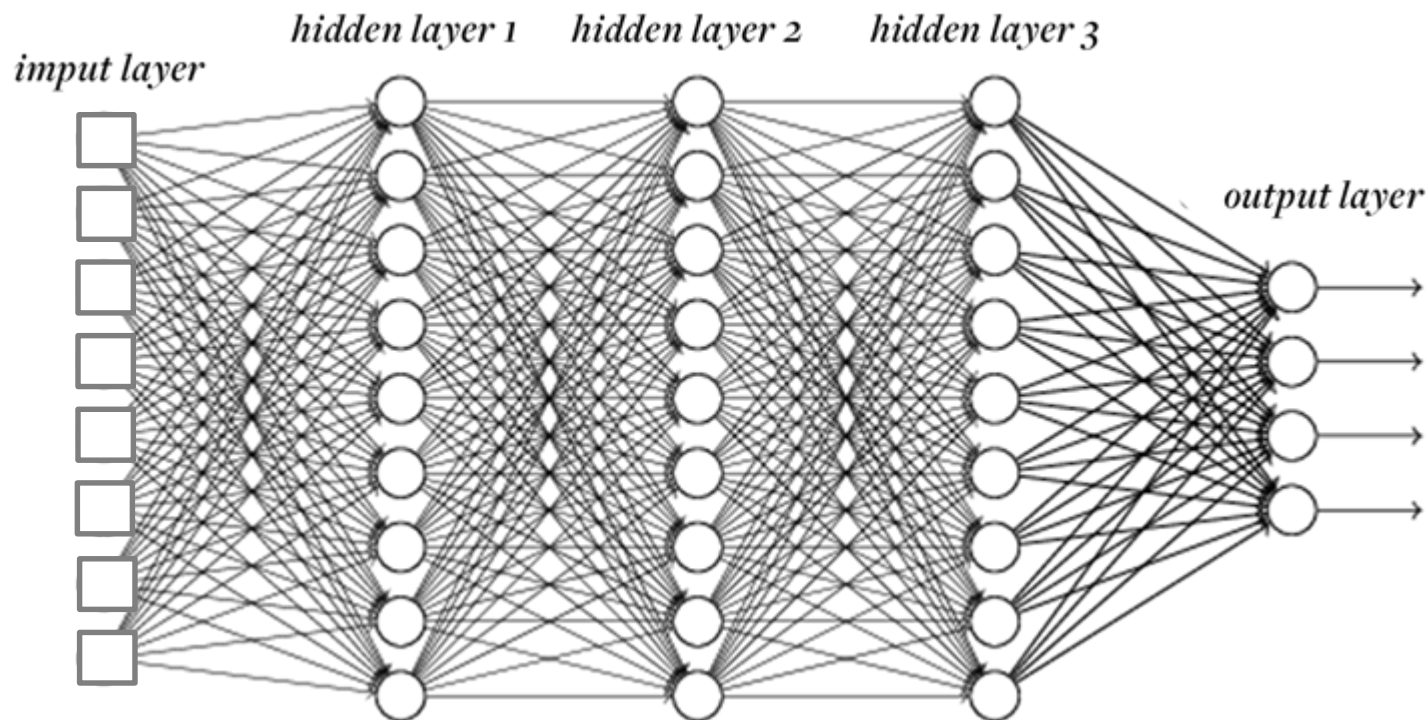
# Perceptron multi-couches

- Perceptron simple
  - Régression linéaire
  - Moindres carrés
- Perceptrons multi-couches
  - Résolution de problèmes non linéaires
  - Généralisable en réseaux de neurones à multi-couches

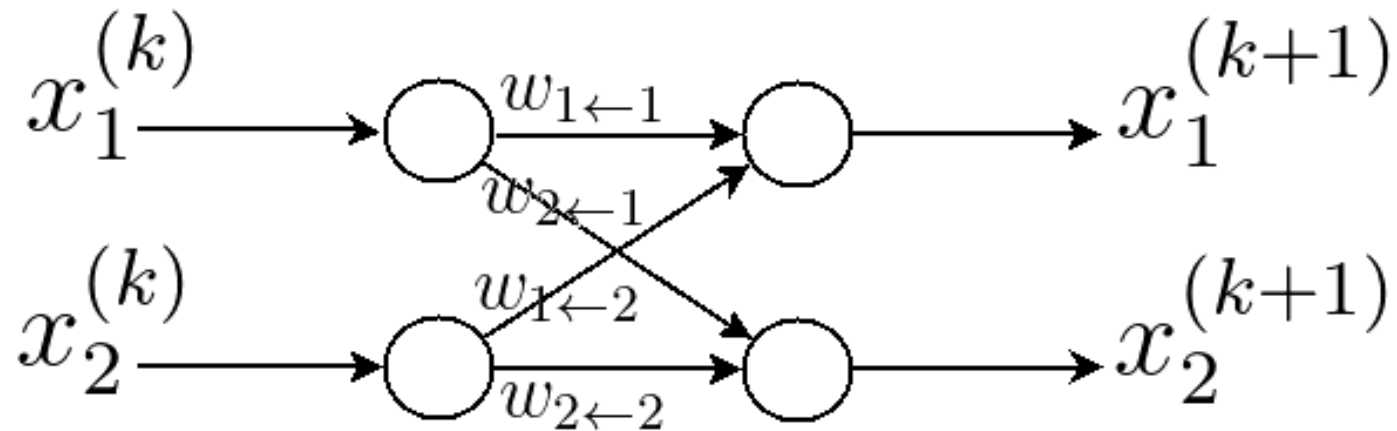


# Modèle de réseaux de neurones à multi-couches

- Exemple relativement généralisé des graphes de réseaux de neurones à multi-couches

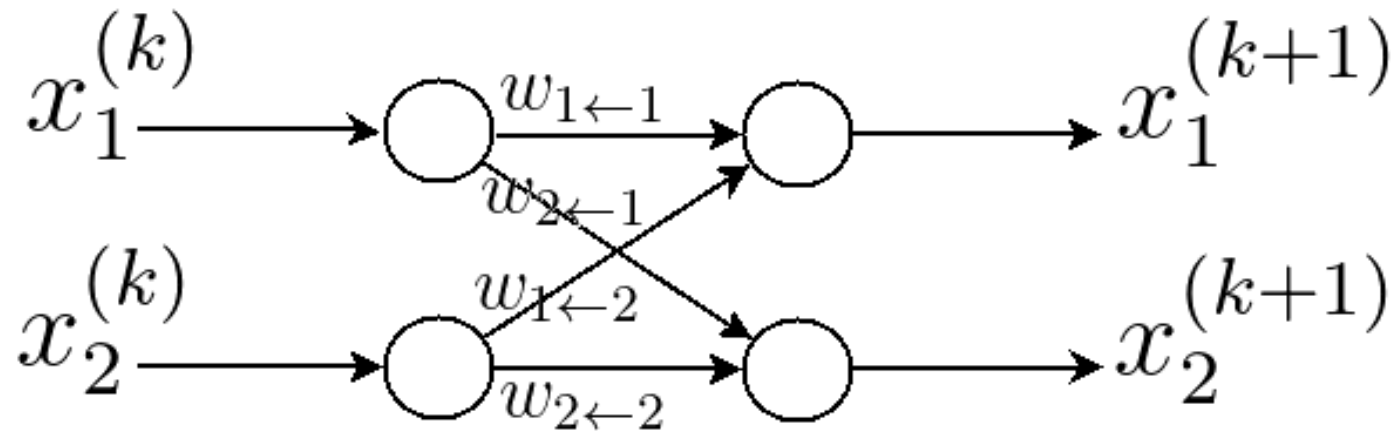


# Propagation dans un réseau de neurones



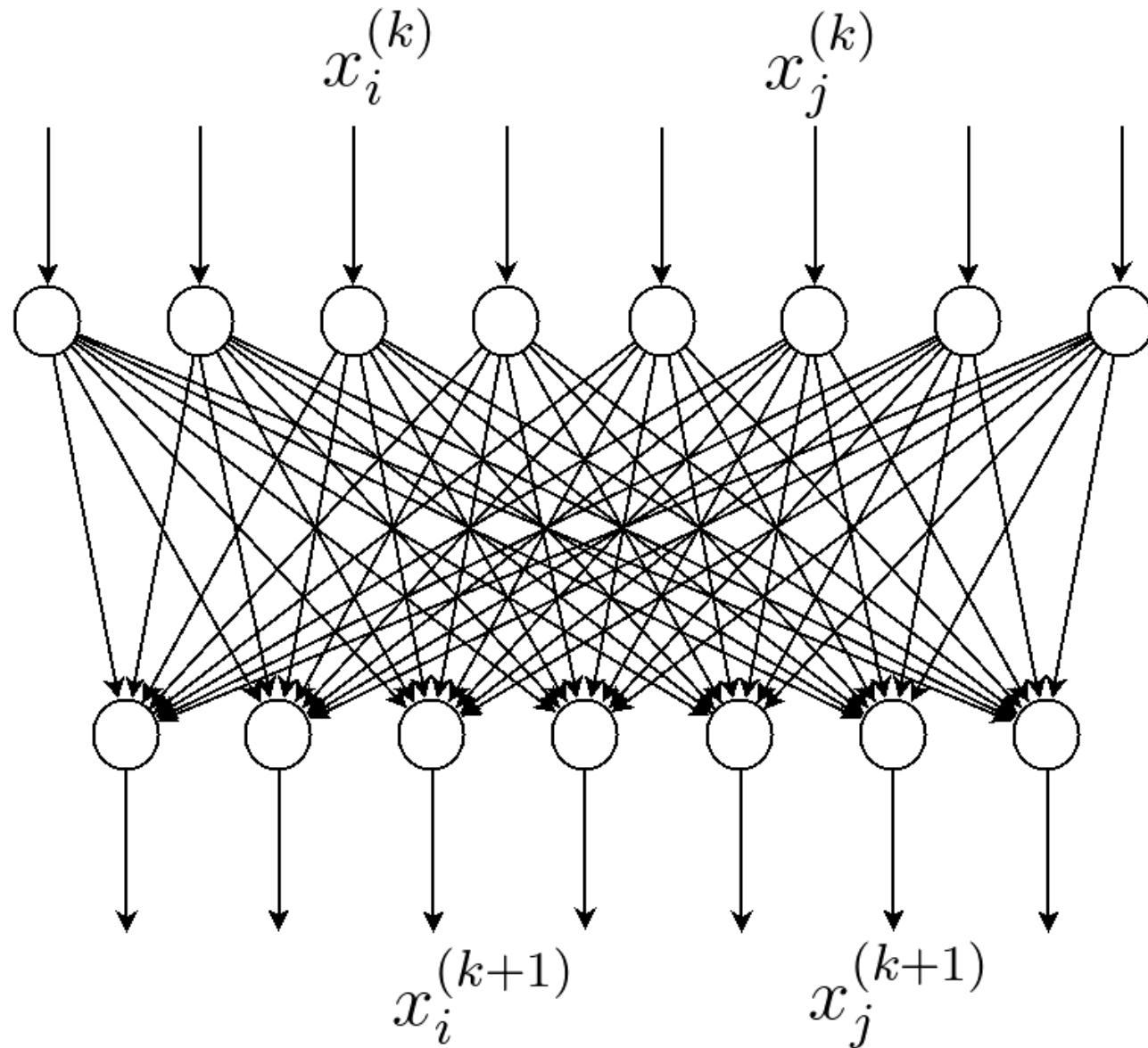
$$\begin{pmatrix} x_1^{(k+1)} \\ x_2^{(k+1)} \end{pmatrix} = g \left( \begin{pmatrix} \sum_{j=1}^2 w_{1\leftarrow j} x_j^{(k)} \\ \sum_{j=1}^2 w_{2\leftarrow j} x_j^{(k)} \end{pmatrix} \right)$$

# Propagation dans un réseau de neurones



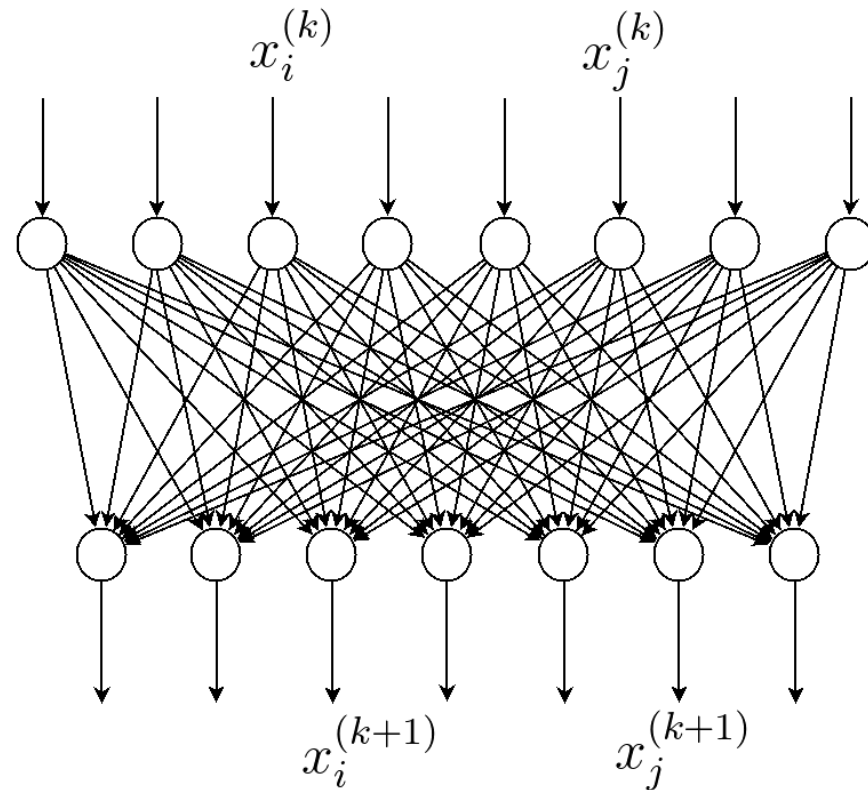
$$\begin{pmatrix} x_1^{(k+1)} \\ x_2^{(k+1)} \end{pmatrix} = g \left( \begin{pmatrix} \sum_{j=1}^2 w_{1\leftarrow j} x_j^{(k)} \\ \sum_{j=1}^2 w_{2\leftarrow j} x_j^{(k)} \end{pmatrix} \right) \\ = g \left( \begin{pmatrix} w_{1\leftarrow 1} & w_{1\leftarrow 2} \\ w_{2\leftarrow 1} & w_{2\leftarrow 2} \end{pmatrix} \begin{pmatrix} x_1^{(k)} \\ x_2^{(k)} \end{pmatrix} \right)$$

# Propagation dans un réseau de neurones





# Propagation dans un réseau de neurones



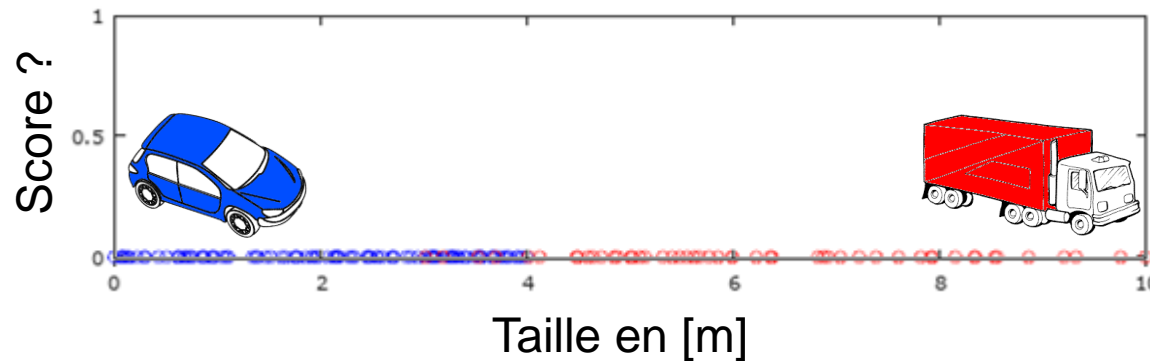
$$\begin{pmatrix} \vdots \\ x_i^{(k+1)} \\ \vdots \\ x_j^{(k+1)} \\ \vdots \end{pmatrix} = g \left( \begin{pmatrix} \vdots \\ w_{i \leftarrow i} \\ \vdots \\ w_{j \leftarrow i} \\ \vdots \end{pmatrix} \begin{pmatrix} \cdot & \dots & \dots & \dots & \dots \\ & w_{i \leftarrow j} & \dots & w_{i \leftarrow j} & \dots \\ & \vdots & \ddots & \dots & \dots \\ & w_{j \leftarrow i} & \vdots & w_{j \leftarrow j} & \dots \\ & \vdots & \vdots & \vdots & \ddots \end{pmatrix} \begin{pmatrix} \vdots \\ x_i^{(k)} \\ \vdots \\ x_j^{(k)} \\ \vdots \end{pmatrix} \right)$$

# Apprentissage et reconnaissance

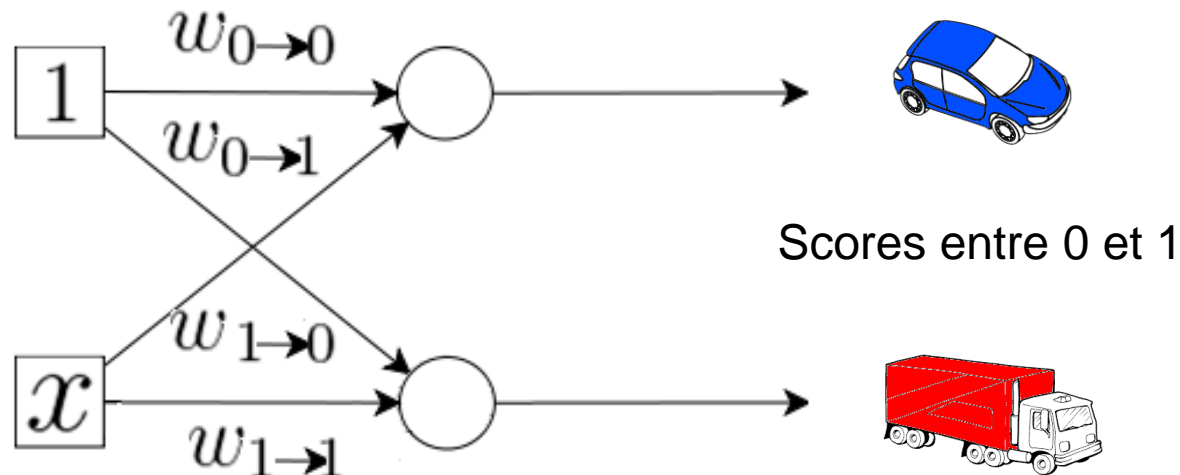
- Phase d'apprentissage supervisé
  - **Pré-requis : base de données indexés par types dite base d'apprentissage**
  - Propagation de de la donnée d'entrée
  - Comparé la donnée de sortie avec le résultat attendu
  - Rétro-propagation de l'erreur pour la correction de poids synaptiques
- Phase de validation (validation de la reconnaissance)
  - **Pré-requis : une base de données différente de la base d'apprentissage dite base de tests**
  - Evaluer l'efficience = nombre d'erreurs / nombre de tests
  - Si le pourcentage est bas. refaire l'apprentissage.

# Exemple avec une couche

- Prenons l'exemple suivant

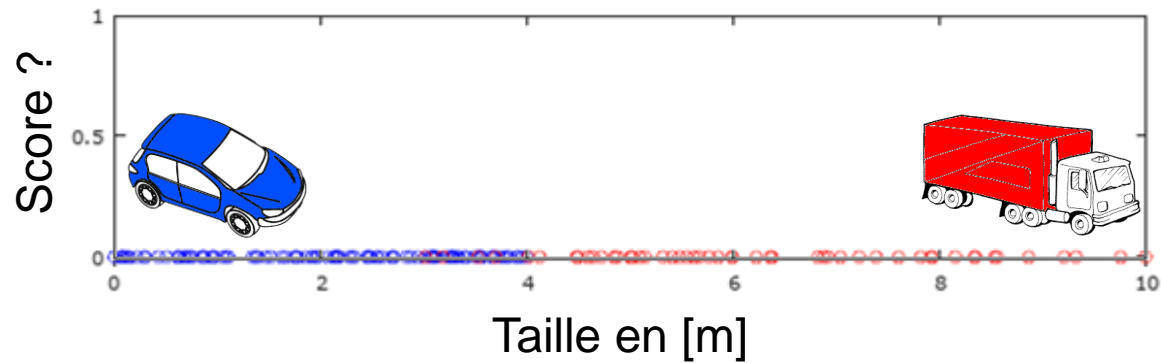


- On modélise de réseau de neurones suivant

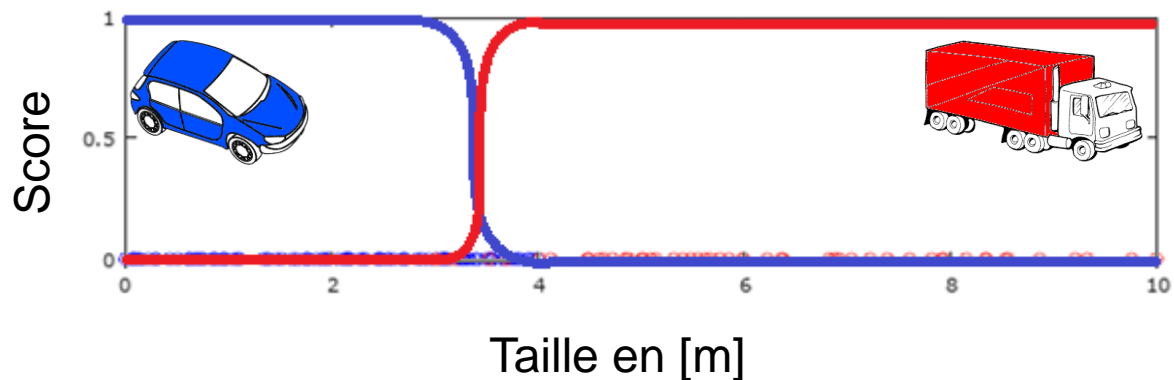


# Exemple avec une couche

- Prenons l'exemple suivant



- On obtient les sigmoïdes suivantes



# Un peu d'histoire

- 1940 : La machine de Turing
- 1943 : Le neurone formel (McCulloch & Pitts)
- 1948 : Les réseaux d'automates (Von Neuman)
- 1949 : Première règle d'apprentissage (Hebb)
- 1958-62 : Le perceptron (Rosenblatt)
- 1960 : L'adaline (Widrow & Hoff)
- 1969 : *Perceptrons* (Minsky & Papert)
  - ➔ les limites du Perceptron
  - ➔ besoin d'architectures + complexes,

Comment effectuer l'apprentissage ? On ne sait pas !
- 1974 : Rétropropagation (Werbos)
  - ➔ pas de succès !?!?
- 1986 : Rétropropagation (Rumelhart & McClelland)
- 1990 : « Société de l'Information »
  - ➔ nouvelles applications
    - recherche/filtrage d'information dans le Web
    - extraction d'information / veille technologique
    - multimedia (indexation, ...)
    - data mining

# Un peu d'histoire

- Ensuite, le Deep-Learning est arrivé

2012 Teams	%error	2013 Teams	%error	2014 Teams	%error
Supervision (Toronto)	15.3	Clarifai (NYU spinoff)	11.7	GoogLeNet	6.6
ISI (Tokyo)	26.1	NUS (singapore)	12.9	VGG (Oxford)	7.3
VGG (Oxford)	26.9	Zeiler-Fergus (NYU)	13.5	MSRA	8.0
XRCE/INRIA	27.0	A. Howard	13.5	A. Howard	8.1
UvA (Amsterdam)	29.6	OverFeat (NYU)	14.1	DeeperVision	9.5
INRIA/LEAR	33.4	UvA (Amsterdam)	14.2	NUS-BST	9.7
		Adobe	15.2	TTIC-ECP	10.2
		VGG (Oxford)	15.2	XYZ	11.2
		VGG (Oxford)	23.0	UvA	12.1

Concours ImageNet 2012-2014

# Rétro-propagation (exemple de fonction coût)

- Fonction coût qu'on souhaite minimiser pour un perceptron à  $m$  couches :

$$J^{(m)} = \frac{1}{2} \sum_{i=0}^{n^{(m)}} (x_i^{(m)} - y_i)^2$$

# Rétro-propagation (exemple de fonction coût)

- Fonction coût qu'on souhaite minimiser pour un perceptron à  $m$  couches :

$$J^{(m)} = \frac{1}{2} \sum_{i=0}^{n^{(m)}} (x_i^{(m)} - y_i)^2$$

- Résolution par une méthode de descente :

$$w_{ij} \leftarrow w_{ij} - \eta \frac{\partial J}{\partial w_{ij}}$$



# Rétro-propagation de la dernière couche

$$\frac{\partial J}{\partial w_{ij}^{(m)}} = \frac{\partial J}{\partial x_i^{(m)}} \frac{\partial x_i^{(m)}}{\partial w_{ij}^{(m)}}$$

# Rétro-propagation de la dernière couche

$$\frac{\partial J}{\partial w_{ij}^{(m)}} = \frac{\partial J}{\partial x_i^{(m)}} \frac{\partial x_i^{(m)}}{\partial w_{ij}^{(m)}}$$

$$\frac{\partial J}{\partial w_{ij}^{(m)}} = \frac{\partial J}{\partial x_i^{(m)}} \frac{\partial x_i^{(m)}}{\partial a_i^{(m)}} \frac{\partial a_i^{(m)}}{\partial w_{ij}^{(m)}} \quad \text{avec} \quad a_i^{(m)} = \sum_{k=0}^{n^{(m-1)}} w_{ik}^{(m)} x_k^{(m-1)}$$

# Rétro-propagation de la dernière couche

$$\frac{\partial J}{\partial w_{ij}^{(m)}} = \frac{\partial J}{\partial x_i^{(m)}} \frac{\partial x_i^{(m)}}{\partial w_{ij}^{(m)}}$$

$$\frac{\partial J}{\partial w_{ij}^{(m)}} = \frac{\partial J}{\partial x_i^{(m)}} \frac{\partial x_i^{(m)}}{\partial a_i^{(m)}} \frac{\partial a_i^{(m)}}{\partial w_{ij}^{(m)}}$$

avec  $a_i^{(m)} = \sum_{k=0}^{n^{(m-1)}} w_{ik}^{(m)} x_k^{(m-1)}$

$$\frac{\partial J}{\partial x_i^{(m)}} = \frac{\partial J^{(m)}}{\partial x_i^{(m)}}$$

$$(x_i^{(m)} - y_i)$$

$$J^{(m)} = \frac{1}{2} \sum_{i=0}^{n^{(m)}} (x_i^{(m)} - y_i)^2$$

# Rétro-propagation de la dernière couche

$$\frac{\partial J}{\partial w_{ij}^{(m)}} = \frac{\partial J}{\partial x_i^{(m)}} \frac{\partial x_i^{(m)}}{\partial w_{ij}^{(m)}}$$

$$\frac{\partial J}{\partial w_{ij}^{(m)}} = \frac{\partial J}{\partial x_i^{(m)}} \frac{\partial x_i^{(m)}}{\partial a_i^{(m)}} \frac{\partial a_i^{(m)}}{\partial w_{ij}^{(m)}}$$

avec  $a_i^{(m)} = \sum_{k=0}^{n^{(m-1)}} w_{ik}^{(m)} x_k^{(m-1)}$

$$\frac{\partial J}{\partial x_i^{(m)}} = \frac{\partial J^{(m)}}{\partial x_i^{(m)}}$$

$$(x_i^{(m)} - y_i)$$

# Rétro-propagation de la dernière couche

$$\frac{\partial J}{\partial w_{ij}^{(m)}} = \frac{\partial J}{\partial x_i^{(m)}} \frac{\partial x_i^{(m)}}{\partial w_{ij}^{(m)}}$$

$$\frac{\partial J}{\partial w_{ij}^{(m)}} = \frac{\partial J}{\partial x_i^{(m)}} \frac{\partial x_i^{(m)}}{\partial a_i^{(m)}} \frac{\partial a_i^{(m)}}{\partial w_{ij}^{(m)}}$$

avec  $a_i^{(m)} = \sum_{k=0}^{n^{(m-1)}} w_{ik}^{(m)} x_k^{(m-1)}$

$$\frac{\partial J}{\partial x_i^{(m)}} = \frac{\partial J^{(m)}}{\partial x_i^{(m)}}$$

$$(x_i^{(m)} - y_i)$$

$$x_i^{(m)} = g(a_i^{(m)})$$

$$g'(a_i^{(m)})$$

# Rétro-propagation de la dernière couche

$$\frac{\partial J}{\partial w_{ij}^{(m)}} = \frac{\partial J}{\partial x_i^{(m)}} \frac{\partial x_i^{(m)}}{\partial w_{ij}^{(m)}}$$

$$\frac{\partial J}{\partial w_{ij}^{(m)}} = \frac{\partial J}{\partial x_i^{(m)}} \frac{\partial x_i^{(m)}}{\partial a_i^{(m)}} \frac{\partial a_i^{(m)}}{\partial w_{ij}^{(m)}} \quad \text{avec} \quad a_i^{(m)} = \sum_{k=0}^{n^{(m-1)}} w_{ik}^{(m)} x_k^{(m-1)}$$

$$\frac{\partial J}{\partial x_i^{(m)}} = \frac{\partial J^{(m)}}{\partial x_i^{(m)}}$$

$$(x_i^{(m)} - y_i)$$

$$x_i^{(m)} = g(a_i^{(m)})$$

$$g'(a_i^{(m)})$$

$$x_j^{(m-1)}$$

# Rétro-propagation de la dernière couche

$$\frac{\partial J}{\partial w_{ij}^{(m)}} = \frac{\partial J}{\partial x_i^{(m)}} \frac{\partial x_i^{(m)}}{\partial w_{ij}^{(m)}}$$

$$\frac{\partial J}{\partial w_{ij}^{(m)}} = \frac{\partial J}{\partial x_i^{(m)}} \frac{\partial x_i^{(m)}}{\partial a_i^{(m)}} \frac{\partial a_i^{(m)}}{\partial w_{ij}^{(m)}} \quad \text{avec} \quad a_i^{(m)} = \sum_{k=0}^{n^{(m-1)}} w_{ik}^{(m)} x_k^{(m-1)}$$

Diagram illustrating the backpropagation of gradients from the output layer to the weights of the last layer:

- The term  $\frac{\partial J}{\partial x_i^{(m)}}$  is equal to  $\frac{\partial J^{(m)}}{\partial x_i^{(m)}}$ , which is boxed as  $(x_i^{(m)} - y_i)$ .
- The term  $\frac{\partial x_i^{(m)}}{\partial a_i^{(m)}}$  is equal to  $x_i^{(m)} = g(a_i^{(m)})$ , which is boxed as  $g'(a_i^{(m)})$ .
- The term  $\frac{\partial a_i^{(m)}}{\partial w_{ij}^{(m)}}$  is equal to  $x_j^{(m-1)}$ , which is boxed as  $x_j^{(m-1)}$ .

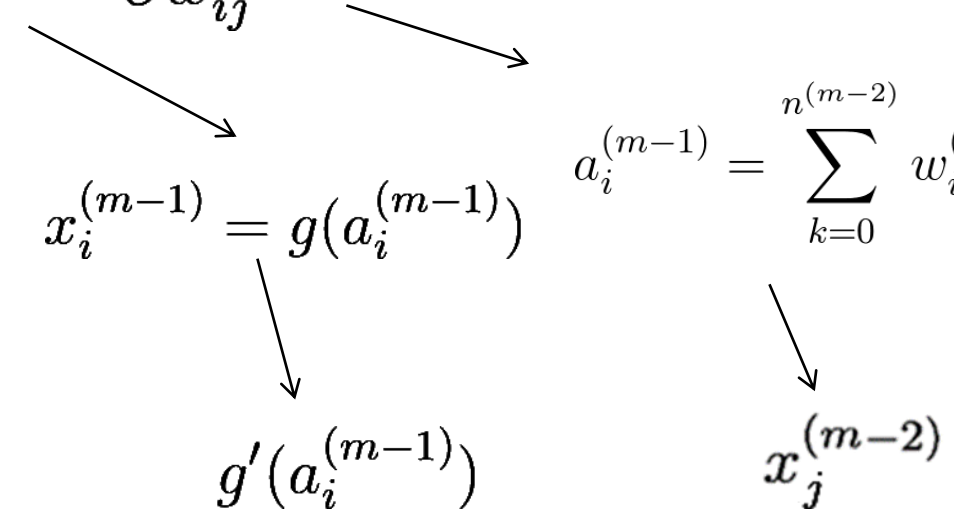
$$\frac{\partial J}{\partial w_{ij}^{(m)}} = (x_i^{(m)} - y_i) g'(a_i^{(m)}) x_j^{(m-1)}$$

# Rétro-propagation de l'avant dernière couche

$$\frac{\partial J}{\partial w_{ij}^{(m-1)}} = \frac{\partial J}{\partial x_i^{(m-1)}} \frac{\partial x_i^{(m-1)}}{\partial a_i^{(m-1)}} \frac{\partial a_i^{(m-1)}}{\partial w_{ij}^{(m-1)}}$$



# Rétro-propagation de l'avant dernière couche

$$\frac{\partial J}{\partial w_{ij}^{(m-1)}} = \frac{\partial J}{\partial x_i^{(m-1)}} \frac{\partial x_i^{(m-1)}}{\partial a_i^{(m-1)}} \frac{\partial a_i^{(m-1)}}{\partial w_{ij}^{(m-1)}}$$


The diagram illustrates the backpropagation of gradients through the second-to-last layer of a neural network. It shows the chain rule for the partial derivative of the loss  $J$  with respect to the weight  $w_{ij}^{(m-1)}$ .

The chain rule is expressed as:

$$\frac{\partial J}{\partial w_{ij}^{(m-1)}} = \frac{\partial J}{\partial x_i^{(m-1)}} \frac{\partial x_i^{(m-1)}}{\partial a_i^{(m-1)}} \frac{\partial a_i^{(m-1)}}{\partial w_{ij}^{(m-1)}}$$

The diagram shows the flow of gradients from the weight  $w_{ij}^{(m-1)}$  to the input  $x_i^{(m-1)}$  and then to the activation function  $g$ , and from the weight  $w_{ij}^{(m-1)}$  to the weighted sum  $a_i^{(m-1)}$  and then to the input  $x_j^{(m-2)}$ .

The activation function  $x_i^{(m-1)}$  is defined as:

$$x_i^{(m-1)} = g(a_i^{(m-1)})$$

The weighted sum  $a_i^{(m-1)}$  is defined as:

$$a_i^{(m-1)} = \sum_{k=0}^{n^{(m-2)}} w_{ik}^{(m-1)} x_k^{(m-2)}$$

# Rétro-propagation de l'avant dernière couche

$$\begin{aligned}
 \frac{\partial J}{\partial w_{ij}^{(m-1)}} &= \frac{\partial J}{\partial x_i^{(m-1)}} \frac{\partial x_i^{(m-1)}}{\partial a_i^{(m-1)}} \frac{\partial a_i^{(m-1)}}{\partial w_{ij}^{(m-1)}} \\
 &\searrow \quad \quad \quad \searrow \quad \quad \quad \searrow \\
 \frac{\partial J}{\partial x_i^{(m-1)}} &= \sum_{i=0}^{n^{(m)}} \frac{\partial J}{\partial a_i^{(m)}} \frac{\partial a_i^{(m)}}{\partial x_i^{(m-1)}} & x_i^{(m-1)} &= g(a_i^{(m-1)}) & a_i^{(m-1)} &= \sum_{k=0}^{n^{(m-2)}} w_{ik}^{(m-1)} x_k^{(m-2)} \\
 &\quad \quad \quad \searrow & \searrow & \searrow \\
 &\quad \quad \quad g'(a_i^{(m-1)}) & & x_j^{(m-2)}
 \end{aligned}$$

# Rétro-propagation de l'avant dernière couche

$$\frac{\partial J}{\partial w_{ij}^{(m-1)}} = \frac{\partial J}{\partial x_i^{(m-1)}} \frac{\partial x_i^{(m-1)}}{\partial a_i^{(m-1)}} \frac{\partial a_i^{(m-1)}}{\partial w_{ij}^{(m-1)}}$$

$$\begin{array}{ccc} \downarrow & \searrow & \searrow \\ \frac{\partial J}{\partial x_i^{(m-1)}} = \sum_{i=0}^{n^{(m)}} \frac{\partial J}{\partial a_i^{(m)}} \frac{\partial a_i^{(m)}}{\partial x_i^{(m-1)}} & x_i^{(m-1)} = g(a_i^{(m-1)}) & a_i^{(m-1)} = \sum_{k=0}^{n^{(m-2)}} w_{ik}^{(m-1)} x_k^{(m-2)} \\ \downarrow & \searrow & \downarrow \\ \frac{\partial J}{\partial x_i^{(m-1)}} = \sum_{i=0}^{n^{(m)}} \frac{\partial J}{\partial a_i^{(m)}} w_{ij}^{(m)} & g'(a_i^{(m-1)}) & x_j^{(m-2)} \end{array}$$

# Rétro-propagation de l'avant dernière couche

$$\begin{aligned}
 \frac{\partial J}{\partial w_{ij}^{(m-1)}} &= \frac{\partial J}{\partial x_i^{(m-1)}} \frac{\partial x_i^{(m-1)}}{\partial a_i^{(m-1)}} \frac{\partial a_i^{(m-1)}}{\partial w_{ij}^{(m-1)}} \\
 &\quad \swarrow \quad \searrow \quad \searrow \\
 \frac{\partial J}{\partial x_i^{(m-1)}} &= \sum_{i=0}^{n^{(m)}} \frac{\partial J}{\partial a_i^{(m)}} \frac{\partial a_i^{(m)}}{\partial x_i^{(m-1)}} & x_i^{(m-1)} &= g(a_i^{(m-1)}) & a_i^{(m-1)} &= \sum_{k=0}^{n^{(m-2)}} w_{ik}^{(m-1)} x_k^{(m-2)} \\
 &\quad \downarrow & \searrow & & \searrow \\
 \frac{\partial J}{\partial x_i^{(m-1)}} &= \sum_{i=0}^{n^{(m)}} \frac{\partial J}{\partial a_i^{(m)}} w_{ij}^{(m)} & g'(a_i^{(m-1)}) & & x_j^{(m-2)}
 \end{aligned}$$

$$\frac{\partial J}{\partial w_{ij}^{(m-1)}} = \left( \sum_{i=0}^{n^{(m)}} \frac{\partial J}{\partial a_i^{(m)}} w_{ij}^{(m)} \right) g'(a_i^{(m-1)}) x_j^{(m-2)}$$

# Rétro-propagation de l'avant dernière couche

$$\frac{\partial J}{\partial w_{ij}^{(m-1)}} = \frac{\partial J}{\partial x_i^{(m-1)}} \frac{\partial x_i^{(m-1)}}{\partial a_i^{(m-1)}} \frac{\partial a_i^{(m-1)}}{\partial w_{ij}^{(m-1)}}$$

$$\begin{aligned} \frac{\partial J}{\partial x_i^{(m-1)}} &= \sum_{i=0}^{n^{(m)}} \frac{\partial J}{\partial a_i^{(m)}} \frac{\partial a_i^{(m)}}{\partial x_i^{(m-1)}} & x_i^{(m-1)} &= g(a_i^{(m-1)}) & a_i^{(m-1)} &= \sum_{k=0}^{n^{(m-2)}} w_{ik}^{(m-1)} x_k^{(m-2)} \\ &\downarrow & &\downarrow & &\downarrow \\ \frac{\partial J}{\partial x_i^{(m-1)}} &= \sum_{i=0}^{n^{(m)}} \frac{\partial J}{\partial a_i^{(m)}} w_{ij}^{(m)} & g'(a_i^{(m-1)}) & & x_j^{(m-2)} \end{aligned}$$

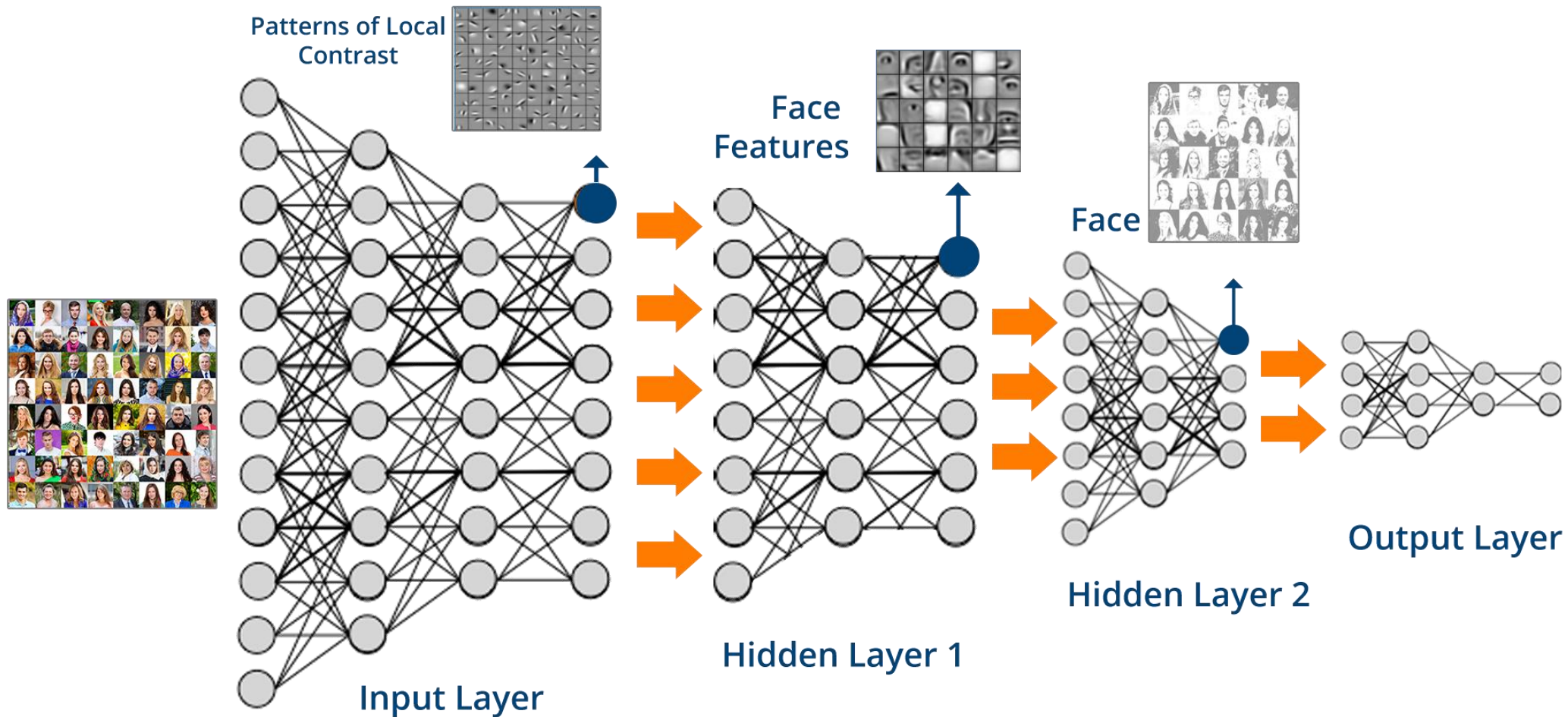
Déduire ?

$$\frac{\partial J}{\partial w_{ij}^{(m-1)}} = \left( \sum_{i=0}^{n^{(m)}} \frac{\partial J}{\partial a_i^{(m)}} w_{ij}^{(m)} \right) g'(a_i^{(m-1)}) x_j^{(m-2)}$$



# Les réseaux de neurones dans le traitement d'images -Apprentissage profond-

# Apprentissage profond



# Reconnaissance d'images en temps réel

- La réduction du temps de calcul dépend de :
  - La taille du réseau de neurones influe en fonction
    - Choix du modèle
    - Puissance de calcul
  - Des calculs dans l'image
    - L'intégrale image (notion provenant l'infographie) permet de calculer rapidement les caractéristiques

$$IImage(i, j) = \sum_{k=0}^i \sum_{l=0}^j Image(k, l)$$



# Reconnaissance d'images en temps réel

- Utile pour calculer rapidement des moyennes
- En apprentissage profond, le calcul de moyenne est souvent utilisé

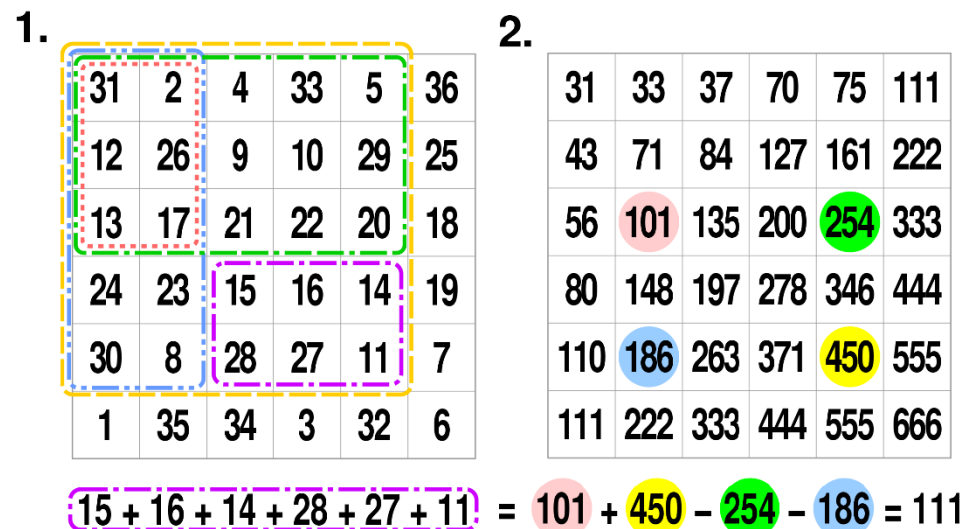


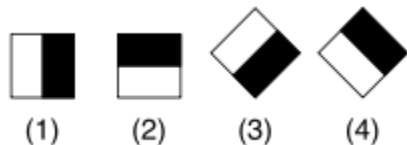
Illustration du gain de calcul avec l'intégrale image

A gauche 5 opérations, à droite 3

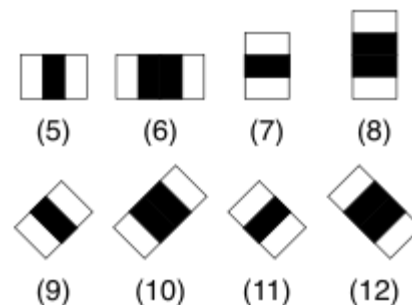
Source de l'image en.wikipedia.org Summed-area table

# Utilisation de l'intégrale image dans la méthode de Viola et Jones

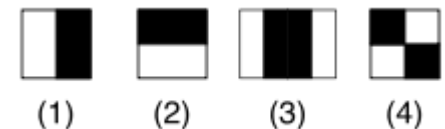
- L'intégrale image a été introduit par la méthode de Viola et Jones en 2001 pour la détection de visages. Dans cette méthode,
- La méthode utilise des caractéristiques dites pseudo-haar



Caractéristiques de  
détection de bord



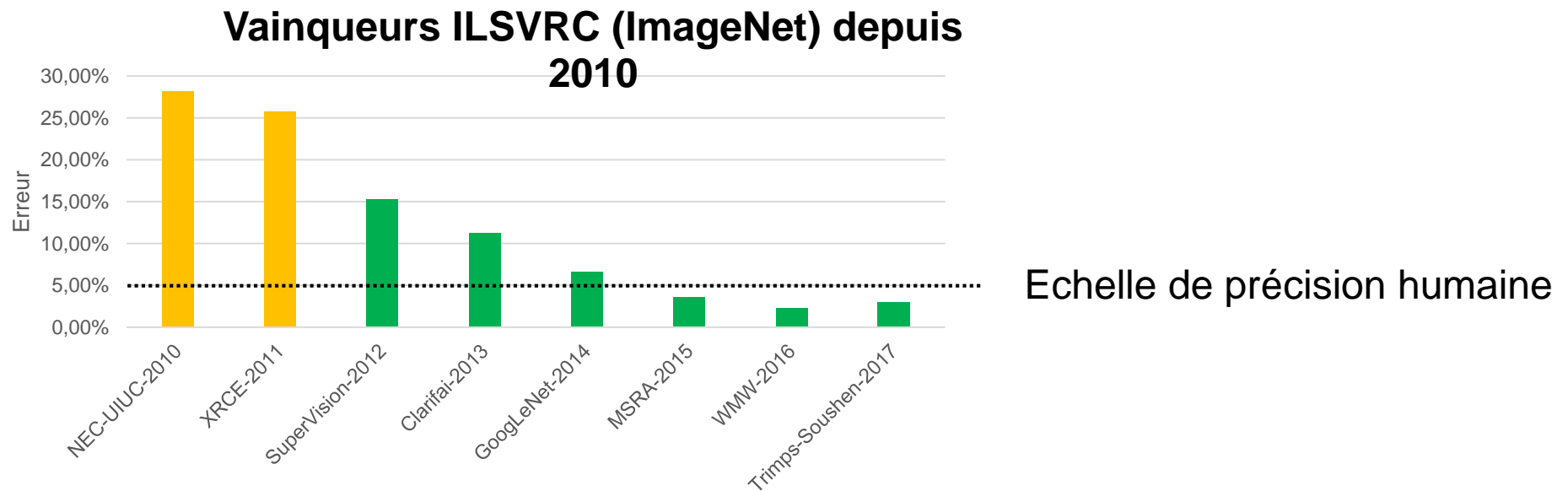
Caractéristiques de  
détection de lignes



Caractéristiques  
Viola et Jones pour  
la détection de  
visages

# Utilisation des Réseaux de Neurones Convolutifs (CNN)

- Krizhevsky et al. (2012) vainqueur de "the ImageNet object recognition challenge" avec l'algorithme AlexNet du group de recherche SuperVision



# Utilisation des Réseaux de Neurones Convolutifs (CNN)

- Krizhevsky et al. (2012) vainqueur de "the ImageNet object recognition challenge" avec l'algorithme AlexNet du group de recherche SuperVision

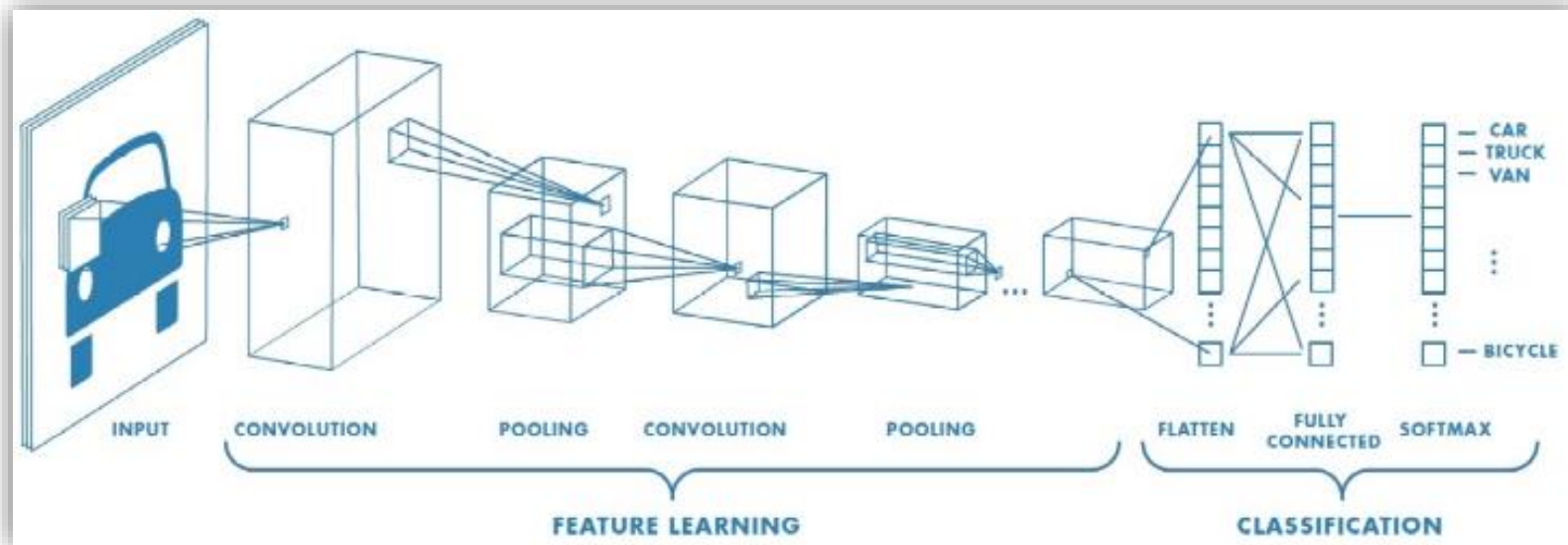
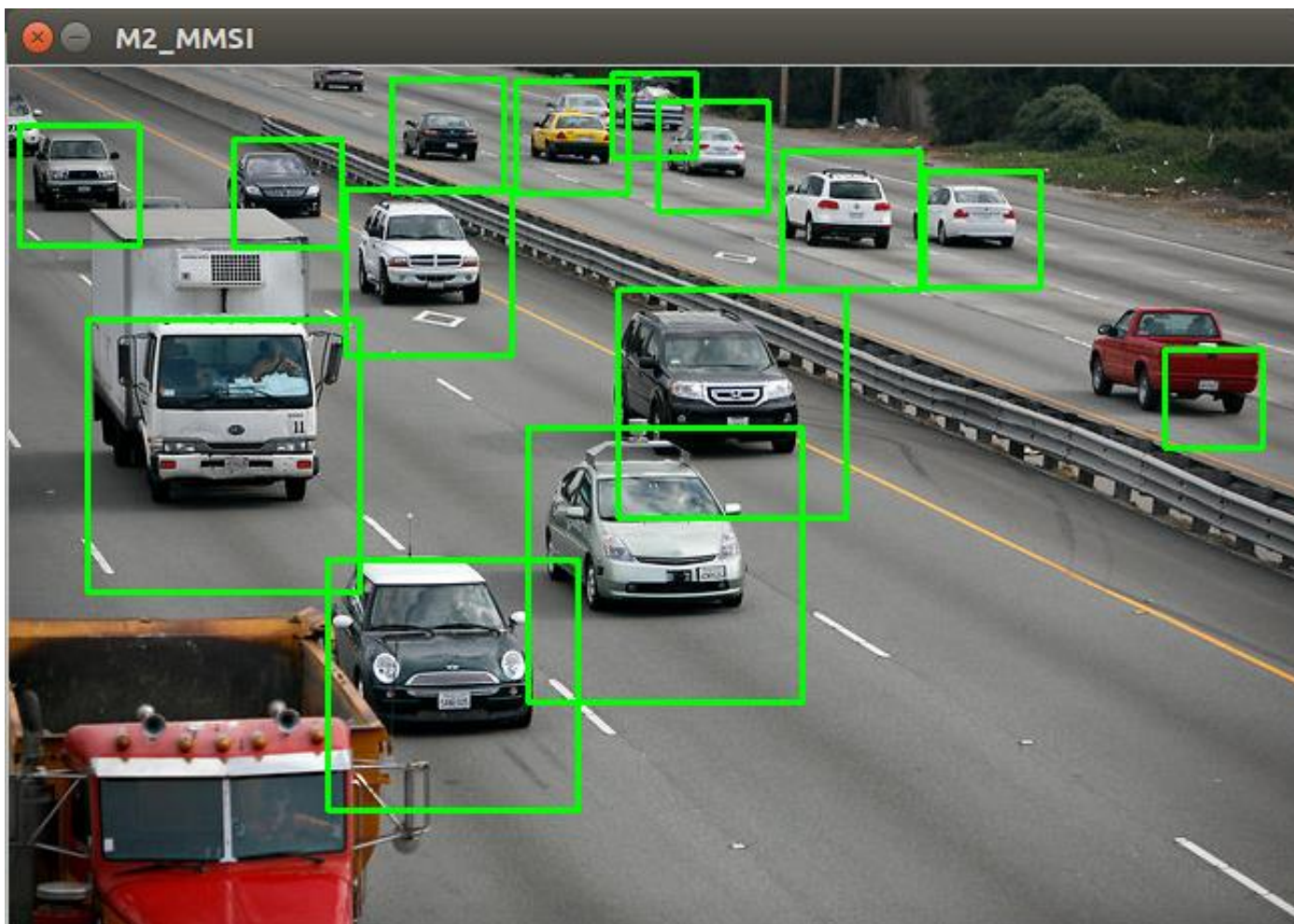


Image de : Trujillo J., Alexandra. (2018). Summarization of video from Feature Extraction Method using Image Processing and Artificial Intelligence.

# TP2 détection par pseudo caractéristiques haar



# Références

- Jean-Claude Heudin, "Comprendre le Deep Learning: Une introduction aux réseaux de neurones" , des éditions "Sciences-eBook", 2016.
- Olga Russakovsky\*, Jia Deng\*, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, 2015.
- L. Laporte, R. Flamary, S. Canu, S. Déjean and J. Mothe: Non-convex Regularizations for Feature Selection in Ranking with Sparse SVM, *IEEE Transactions on Neural Networks and Learning Systems*, 25(6) :1118 – 1130, 2014.
- Paul Viola et Michael Jones, « Robust Real-time Object Detection », *IJCV*, 2001