

# GLO-4030/7030 APPRENTISSAGE PAR RÉSEAUX DE NEURONES PROFONDS

Plan de cours

Introduction

Fonctions d'activation

Hiver 2020

Philippe Giguère

# Plan de cours

# Bienvenue à bord!

- 3ème édition!
- L'équipe de cette année :



Frédéric Paradis



Gaël Letarte

# Séminaire départemental

- Vendredi 17 janvier 2020
- Apprentissage de réseaux de neurones à activations binaires avec garanties statistiques  
Gaël Letarte  
*Étudiant au doctorat, membre du GRAAL*
- Heure: 13h30  
Local: PLT-3775
- Résumé: Malgré les exploits empiriques des réseaux de neurones profonds, il existe peu de garanties théoriques solides expliquant leurs performances. Nous allons présenter nos récents travaux portant sur l'analyse des réseaux de neurones profonds avec activations binaires, pour lesquels nous pouvons obtenir des garanties statistiques sur leur taux d'erreur. Il en découle une approche originale permettant d'entraîner un modèle composé d'une agrégation de tels réseaux. Cette analyse se base sur la théorie PAC-Bayésienne, qui s'est illustrée récemment comme un outil fructueux pour analyser la capacité de généralisation des réseaux de neurones profonds.



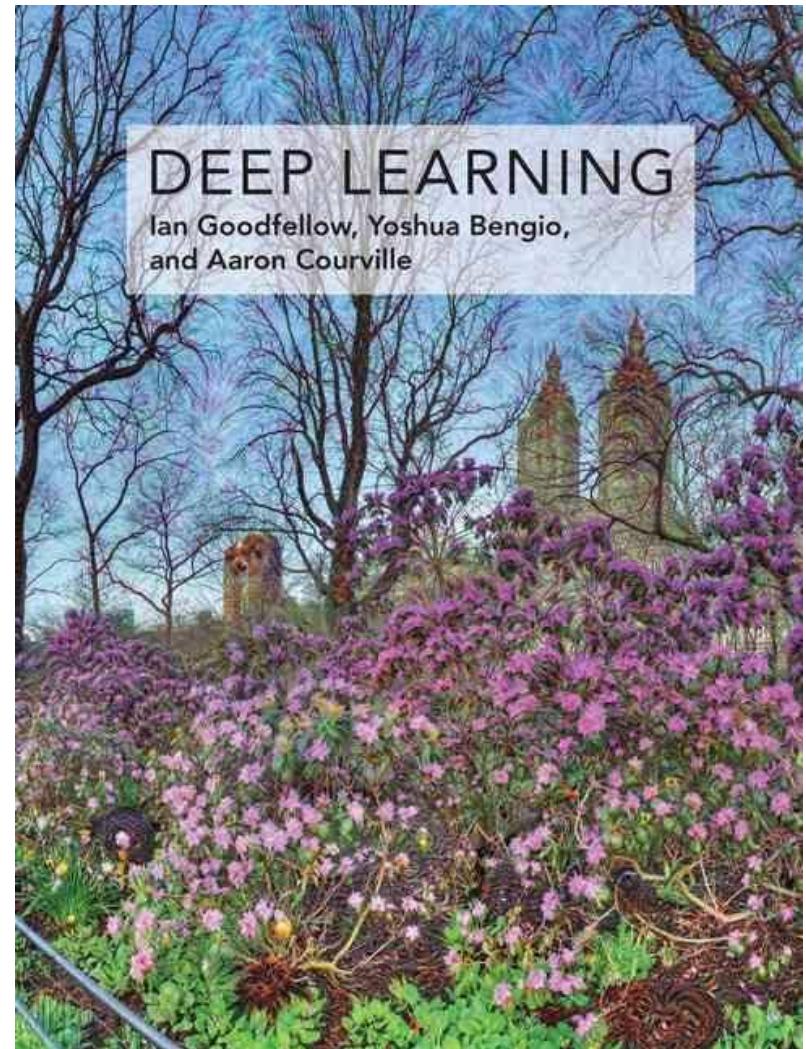
**En cours de mise-en-oeuvre**

- Accès à des GPU via Jupyter notebook
  - NVIDIA K20 ou K80
- <http://jupyterhub.helios.calculquebec.ca>,
- Au besoin, lisez le Wiki
  - <https://wiki.calculquebec.ca/w/JupyterHub>
- Cœurs réservés pour les laboratoires :  
réservation glo4030
- Gros merci à
  - Florent Parent
  - Félix-Antoine Fortin

# Manuel (obligatoire)

- Bonne référence
- Version html gratuite

<http://www.deeplearningbook.org/>



# Pré-requis

- Python (laboratoire, TP)
- Certaine connaissance de l'apprentissage machine
- Probabilité
- Algèbre linéaire
- Dérivée
- Un peu d'optimisation

# Contenu du cours

- Un peu de théorie, oui...
- ... mais aussi de la pratique...
- ... et de la collection de timbres
  - nombre de techniques, trucs
  - 20+ exemples d'architecture
  - grand nombre de papiers
    - 70+ présentations orales
- Donner un aperçu du domaine, vous aider à démarrer dans la recherche



"All science is either physics or stamp collecting"  
–E. Rutherford

# Aperçu 1<sup>ère</sup> moitié

- Connaissances de base (vertical)
  - Introduction, neurone, fonctions d'activation
  - Graphes de calculs, fonctions de perte, rétro-propagation
  - Méthodes d'entraînement par optimisation, *batch norm*, initialisation des poids, trucs et astuces
  - Techniques de régularisation
  - Réseaux à convolution I
  - Réseaux à convolution II
  - Examen intra

# Aperçu 2<sup>ème</sup> moitié

- Concepts avancés (horizontal) :
  - Word embeddings
  - Autoencodeurs
  - Réseaux récurrents (RNN, LSTM et GRU)
  - Modèles d'attention, proposition de régions d'images, réseaux à mémoire
  - Apprentissage multitâches, pertes auxiliaires
  - Distillation (compression) des réseaux
  - Réseaux génératifs type GAN
  - Et +

# Examen

- Mi-Session (25 février 12h30 à 15h20)
  - 35% pour GLO-4030
  - 33% pour GLO-7030
- Final (21 avril de 12h30 à 14h20)
  - GLO-4030 seulement
  - Examen de 2 heures, 20 %
  - Séance de présentations orales pour GLO-7030
- Pas de documents permis

# Travaux pratiques

- 2 travaux
- Total de **20 %**
- En python ou PyTorch
- Interdit d'utiliser :
  - github public
  - pastebin
  - etc.

# Projets

- Équipe de 1 à 2
- GLO-4030 : **25 %**
- Pour GLO-7030 : **35 %**
  - comme pas d'examen, projet devra être ambitieux (proche de publiable)
  - bonne méthodologie
- Trouvez un jeu de données proche de votre recherche / laboratoire / programme
- Vous pouvez utiliser langage et librairie de votre choix (TensorFlow, PyTorch, etc)
- Rapport sous format d'article scientifique

# Mon projet en 180 s (GLO-7030)

- Présentation chronométrée de 3 minutes de votre projet de recherche
- 14 avril
- 1 seule acétate, sans animation, format pdf ratio 16:9
- 1 seul membre de l'équipe
- Jugé sur la clarté et la qualité, et non sur le mérite technique
- Note attribuée à l'équipe entière (**4 %**)

# Présentations orales GLO-7030

- Article plus récent que **1<sup>er</sup> juin 2018**
- Uniquement des conférences suivantes : NIPS, CVPR, ICLR, RSS, ICCV, ECCV, EMNLP, ACL, NAACL et COLING
- Présentation **10 minutes**, style conférence
- M'envoyer votre choix le plus rapidement possible (pour éviter les doublons)
- Compte pour **12 %**

# Présentations orales GLO-7030

- 72 inscriptions (horaire chargé!)
- Date de votre présentation idéalement en lien avec le contenu du cours
- Horaire approximatif :

Date	nombre prés.
Vendredi 21 février	8
Vendredi 20 mars	10
Vendredi 3 avril	10
Vendredi 10 avril	10
Vendredi 17 avril	10
Mardi 21 avril	14
Vendredi 24 avril	10

- Si vous voulez présenter sur les CNN, m'envoyer votre suggestion d'article pour pouvoir présenter le 21 février

# Sites web

- Site du cours :

<https://ulaval-damas.github.io/glo4030/>

- Site pour le forum :

- site monPortail GLO-7030
  - les étudiants GLO-4030 ont plein accès au forum, en tant qu'utilisateur invité du cours

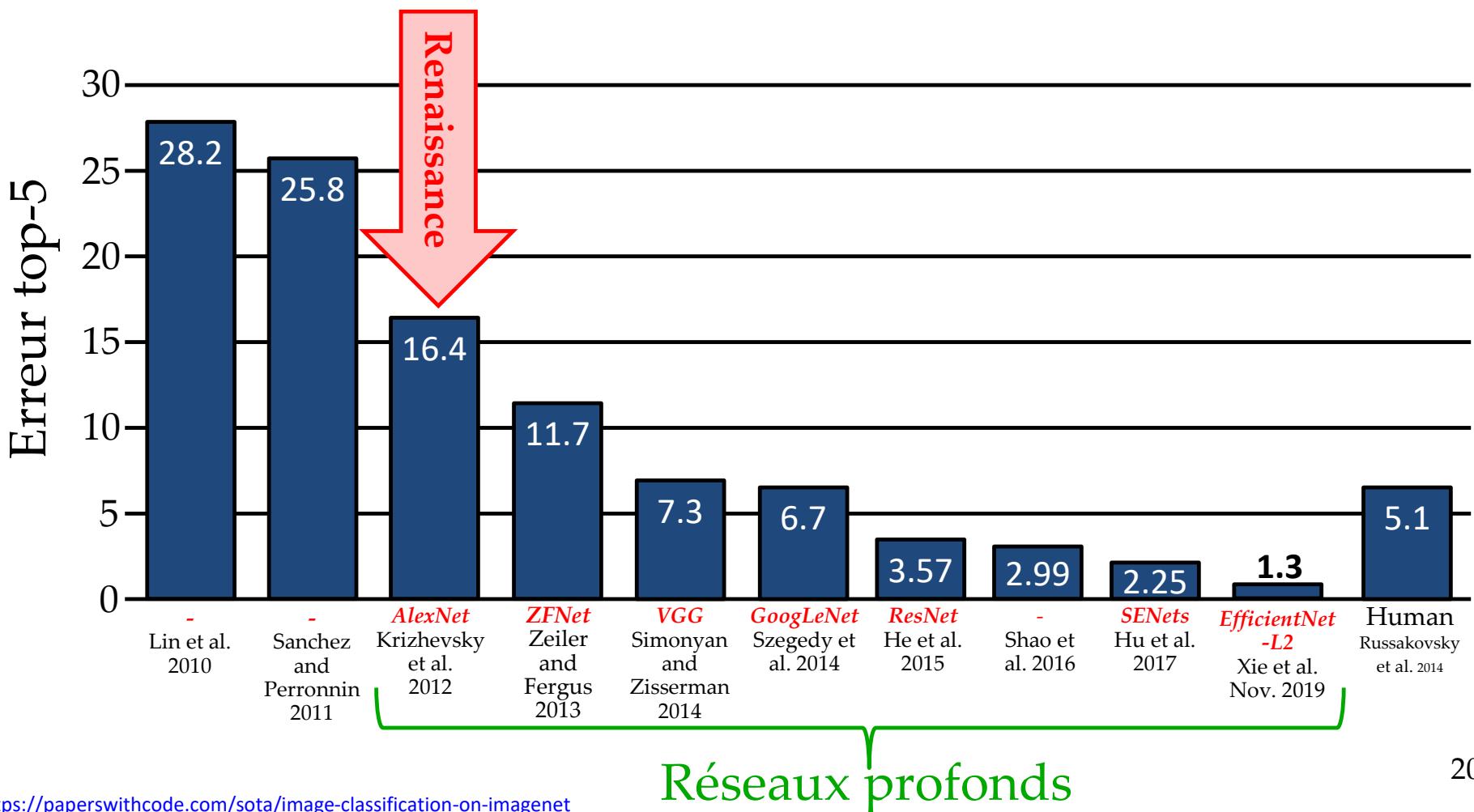
# Librairie utilisée : PYTORCH 1.?

- Recommandation unanime des experts locaux (ils ont utilisés Theano, TensorFlow, Torch, Keras)
- Python, et non pas LUA
- Facile à débugger
  - Vous pouvez extraire les données du GPU en tout temps
- Dérivation automatique autograd
- Support GPU pour Ndarray
- Package d'optimisation par descente de gradient inclus (SGD, RMSprop, ADAM, etc.)
- Beaucoup d'utilitaires (data loading, entraînement, data augmentation, torchvision etc.)
- Facile d'obtenir des modèles pré-entraînés

# Introduction

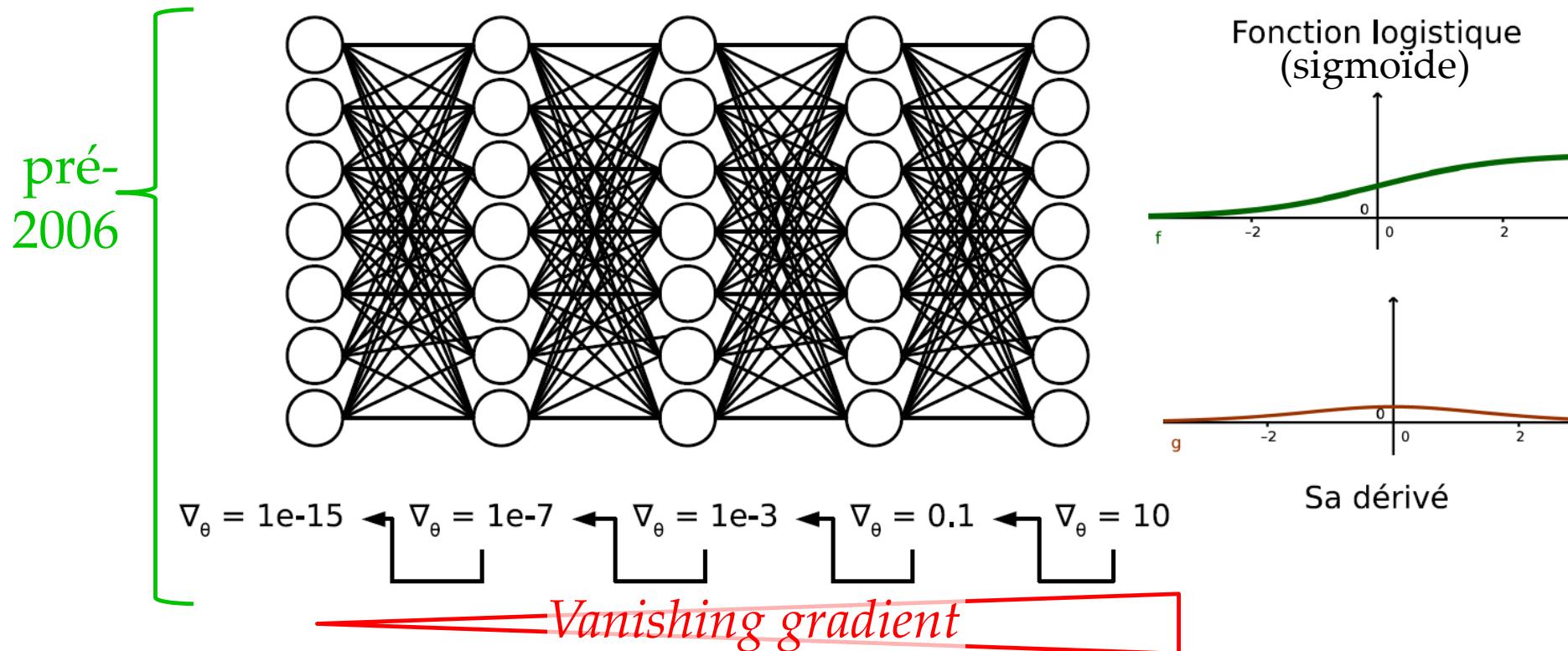
# Large Scale Visual Recognition Challenge

- *Image Classification Challenge :*
  - 1,000 classes d'objets
  - 1,431,167 images



# Causes de la renaissance #1

## Nouvelles fonctions d'activations



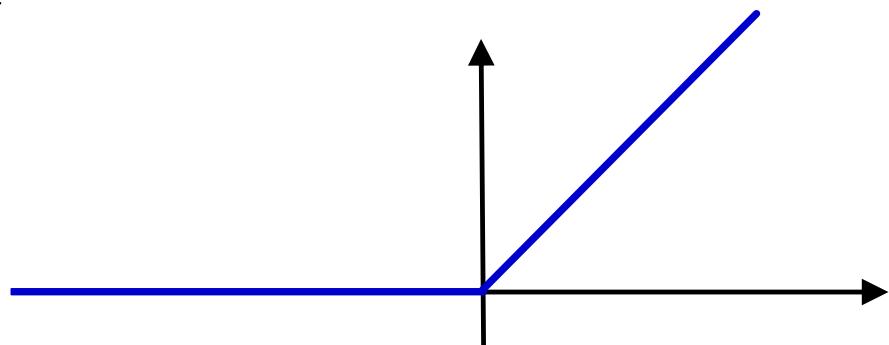
Venait limiter en pratique la profondeur

# Causes de la renaissance #1

## Nouvelles fonctions d'activations

- ReLU : Rectifier Linear Unit
- Introduite en 2010 par Nair et Hinton

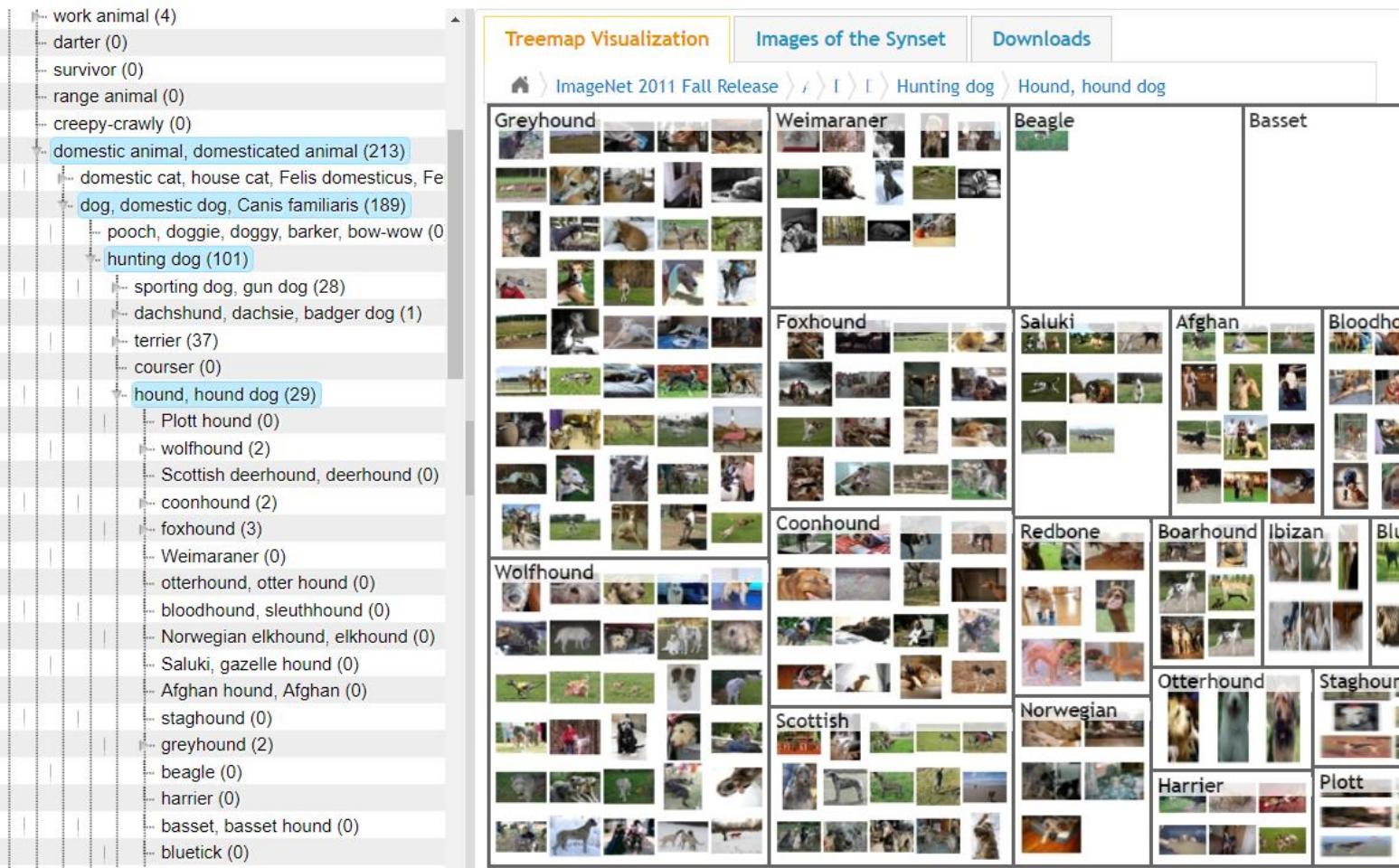
$$f(x) = \begin{cases} x & \text{si } x \geq 0 \\ 0 & \text{sinon} \end{cases}$$



- Se calcule très rapidement : `max(input, 0)`
- Beaucoup moins de *vanishing gradient*, car pente = 1 dans la partie active

# Causes de la renaissance #2

- Grands jeux de données
  - [www.image-net.org](http://www.image-net.org)
  - 14 millions images, 22 000 catégories



# Causes de la renaissance #3

## Puissance de calcul via GPU



Juin 2005



2,999 USD  
**110 TFlops**

Rank	System	Cores	Rmax (TFlop/s)	Rpeak (TFlop/s)	Power (kW)
1	BlueGene/L - eServer Blue Gene Solution , IBM DOE/NNSA/LLNL United States	65,536	136.8	183.5	716
2	BGW - eServer Blue Gene Solution , IBM IBM Thomas J. Watson Research Center United States	40,960	91.3	114.7	448
3	Columbia - SGI Altix 1.5 GHz, Voltaire Infiniband , HPE NASA/Ames Research Center/NAS United States	10,160	51.9	61.0	
4	Earth-Simulator , NEC Japan Agency for Marine-Earth Science and Technology Japan	5,120	35.9	41.0	3,200

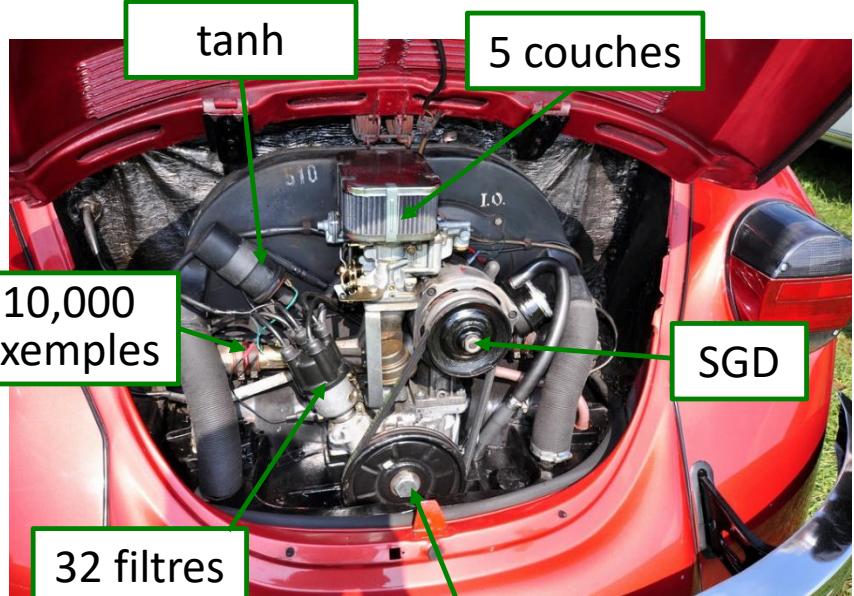
# Continuation de la renaissance

- Progrès très rapide via arXiv.org
- Plus d'une centaine de soumission par jour

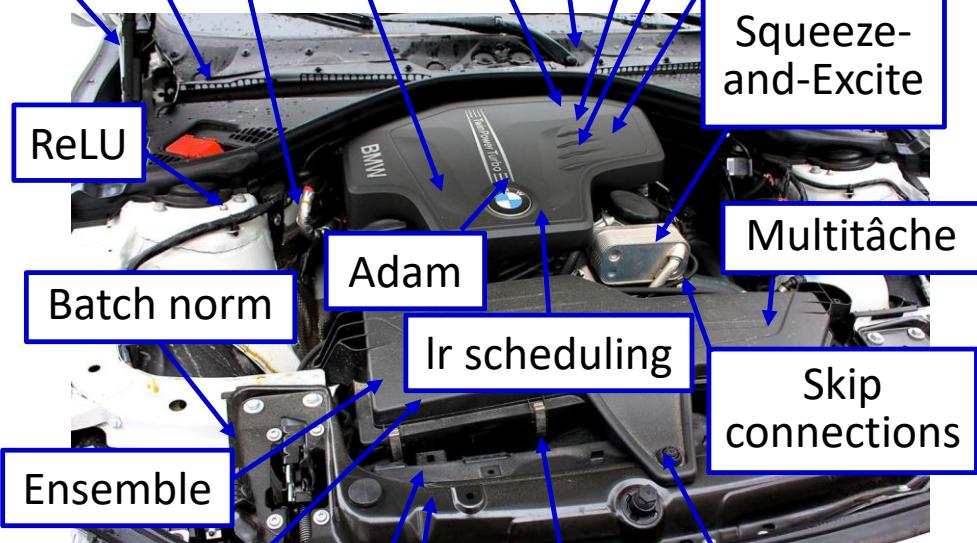
**Certain danger : beaucoup de  
« bruit » dans les publications**

- Un article est souvent périmé lorsque présenté en conférence
- D'où la motivation pour vos présentations orales (GLO-7030)

# Évolution des réseaux



ca. 1990



ca. 2019

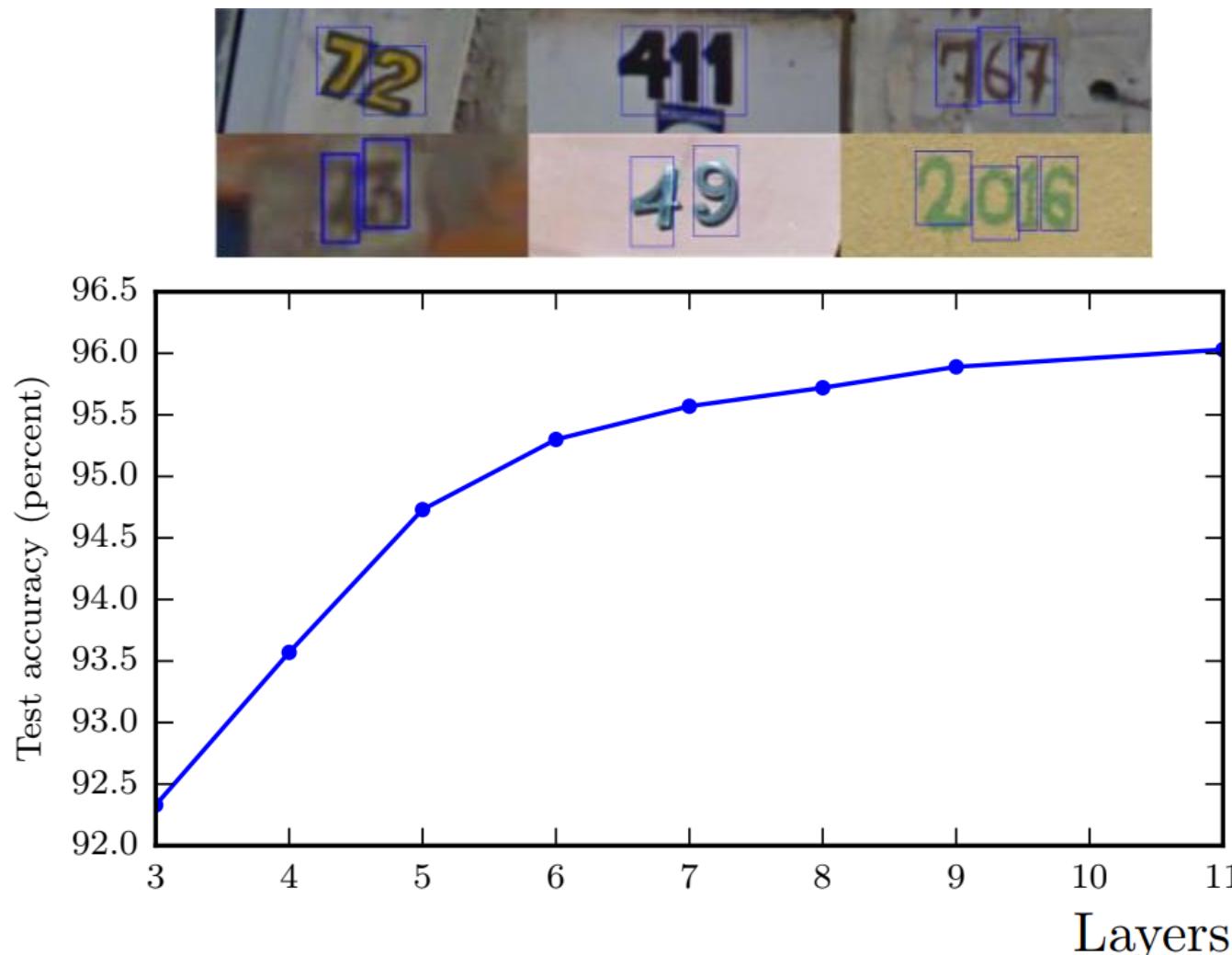
# Pourquoi le Deep?

- Fonction mathématique extrêmement flexible et puissante (millions de paramètres)
- Théorème d'approximation universelle :
  - Peut approximer n'importe quelle fonction\* avec un niveau de précision arbitraire
  - <http://neuralnetworksanddeeplearning.com/chap4.html>
- Réseaux peu profonds vont :
  - demander beaucoup de neurones (exponentiel)

\*continue sur des sous-ensembles compacts de  $R^n$

# Généralisation vs. profondeur

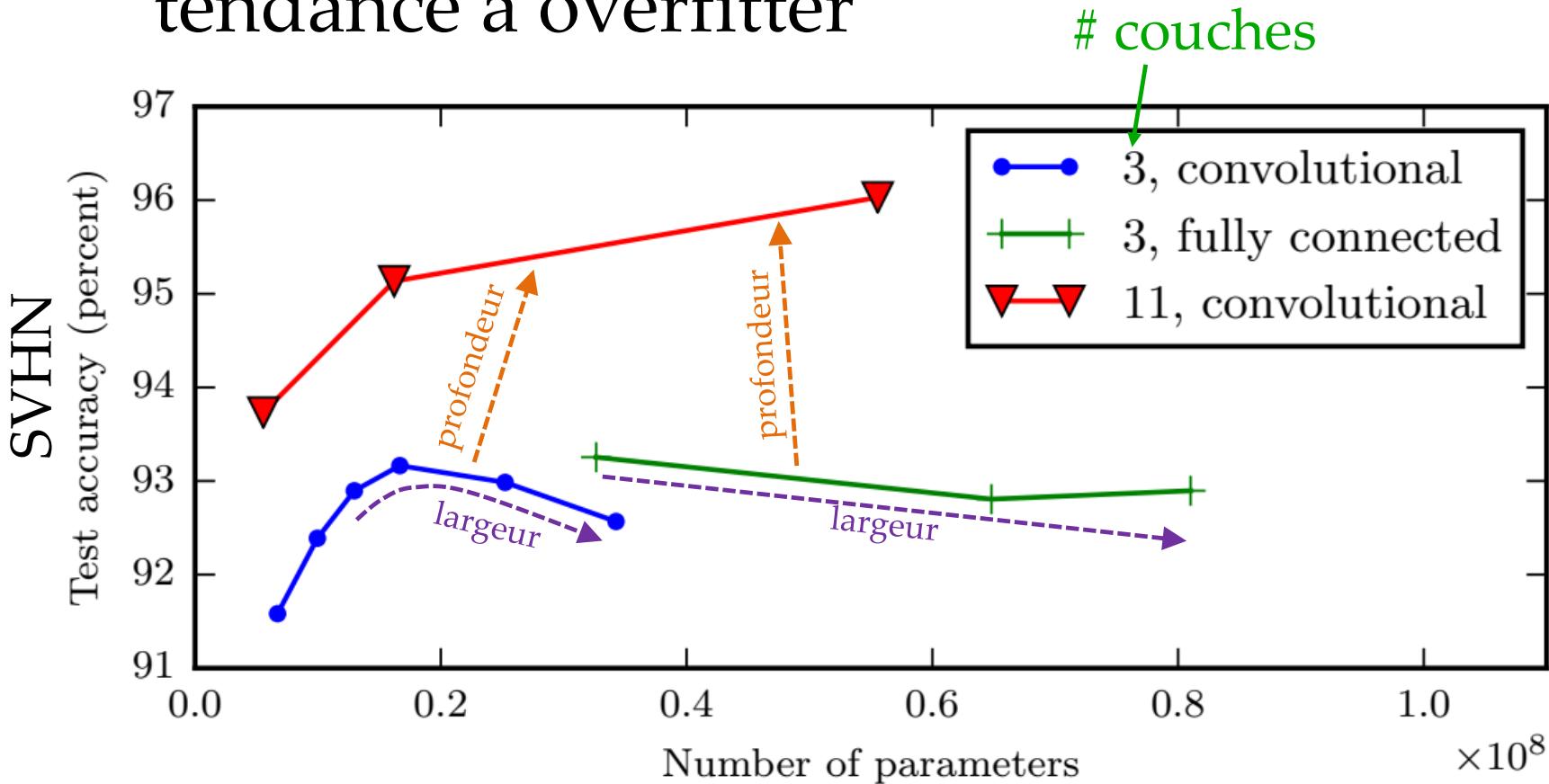
- Street View Home Numbers SVHN



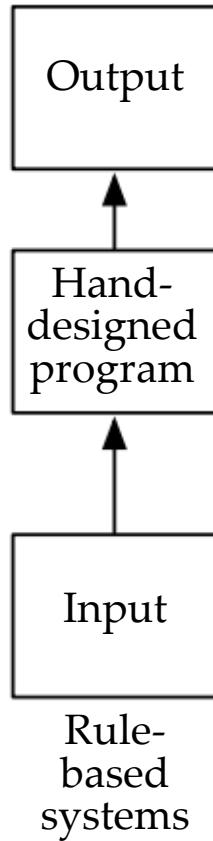
Goodfellow et al., Multi-digit Number Recognition from Street View Imagery  
using Deep Convolutional Neural Networks, ICLR 2014.

# Meilleure généralisation

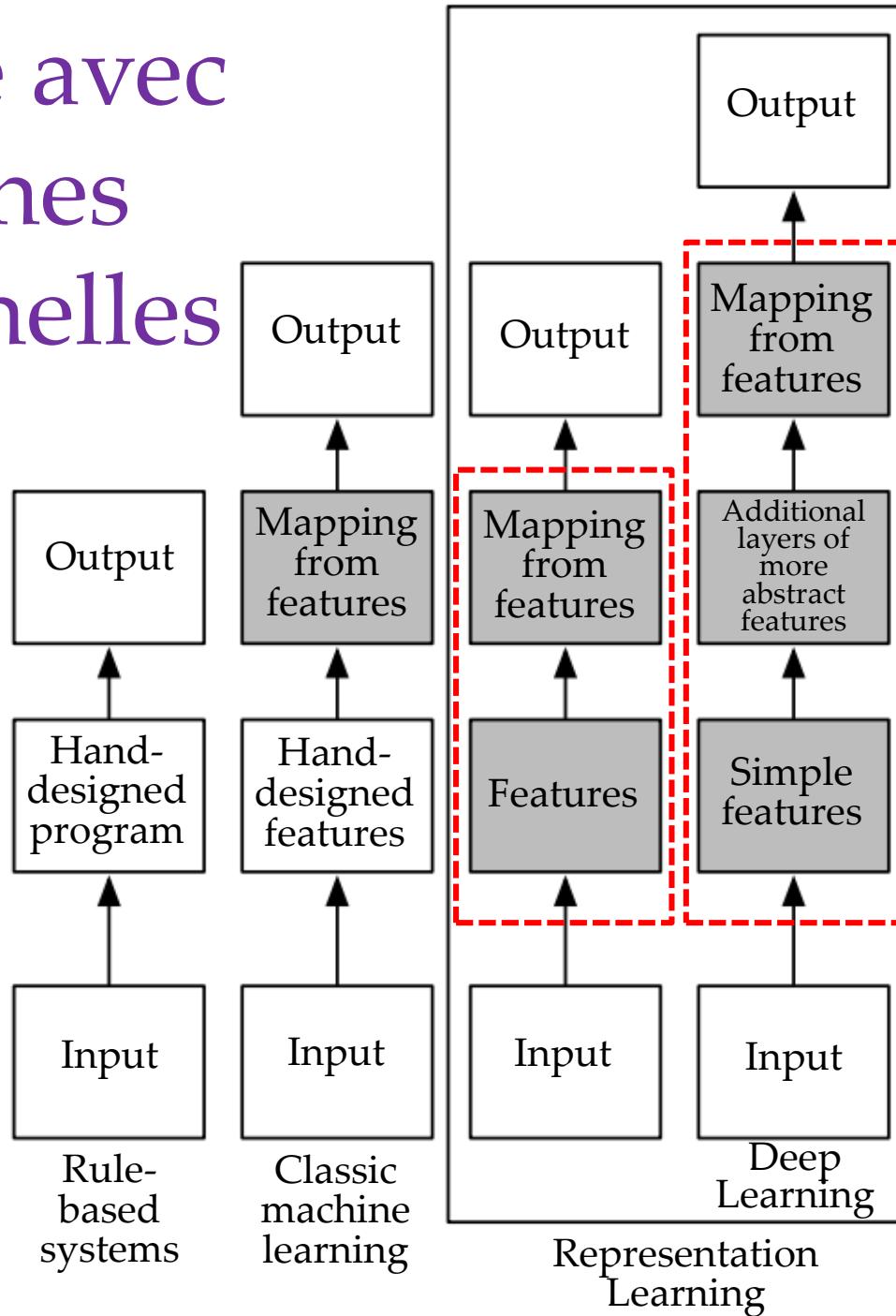
- Modèles larges ou peu profonds ont tendance à overfitter



# Contraste avec approches traditionnelles



# Contraste avec approches traditionnelles



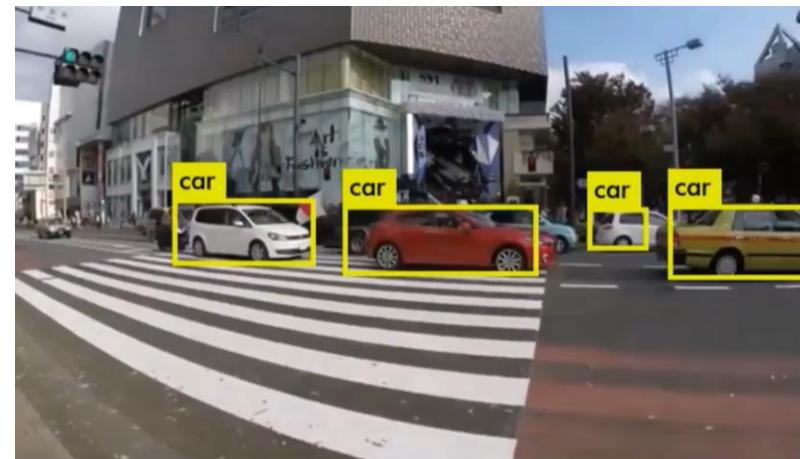
Appris  
conjointement

# Architecture = a priori

- Façon d'injecter de l'information a priori, via l'architecture
- Peut être vu comme un prior (parfois infiniment) fort
- Par exemple :
  - CNN (localité dans les images)
  - Maxpooling (invariance à la position)
  - RNN (dépendance temporelle)
  - Attention (régions plus informatives)
  - Spatial transformer network (déformation affines)
  - Softmax (appartenance à une seule classe)

# Pourquoi le Deep en industrie ?

- Applicable à des solutions industrielles
- Entraînement (long) se fait en différé
- Tourne en temps réel sur machine GPU ou TPU (Tensor Processor Unit, Google)
- Le temps d'exécution dépend peu du nombre de classes
  - YOLO9000 : détection de **9000 objets**, 40 FPS
  - comment : un **seul pipeline commun** d'extraction des features



<https://www.youtube.com/watch?v=uG2UOasIx2I>

# Pourquoi le Deep en industrie ?

- Si on découvre des nouveaux cas problématiques, on les ajoute dans la banque d'entraînement
  - facile à expliquer à un non-expert
- La quantité de données d'entraînement n'influe pas sur le temps d'inférence\*
- Systèmes experts (*explicite*) deviennent fragiles avec nombre croissant de règles
  - réseaux neurones : *implicite*

\*bien entendu, on pourrait augmenter la taille du réseau si on a beaucoup de données, afin d'en profiter

# « Deep Learning is eating software »

The pattern is that there's an existing software project doing data processing using explicit programming logic, and the team charged with maintaining it find they can replace it with a deep-learning-based solution. I can only point to examples within Alphabet that we've made public, like upgrading search ranking, data center energy usage, language translation, and solving Go, but these aren't rare exceptions internally. What I see is that almost any data processing system with non-trivial logic can be improved significantly by applying modern machine learning.

This might sound less than dramatic when put in those terms, but it's a radical change in how we build software. Instead of writing and maintaining intricate, layered tangles of logic, **the developer has to become a teacher, a curator of training data and an analyst of results**. This is very, very different than the programming I was taught in school, but what gets me most excited is that it should be far more accessible than traditional coding, once the tooling catches up.

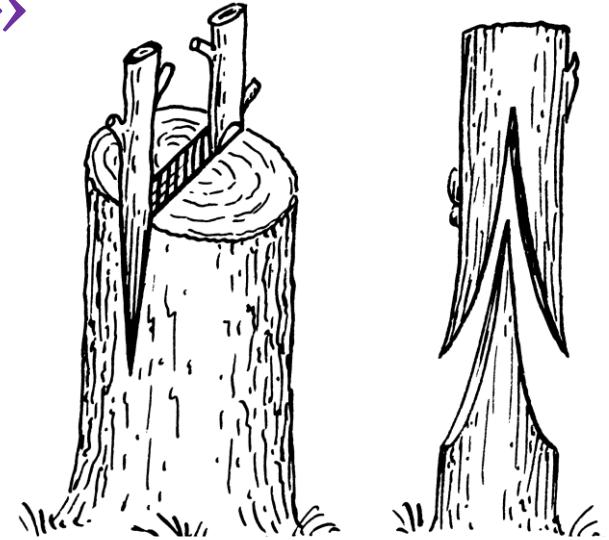
<https://petewarden.com/2017/11/13/deep-learning-is-eating-software/>

# Transfert d'innovation

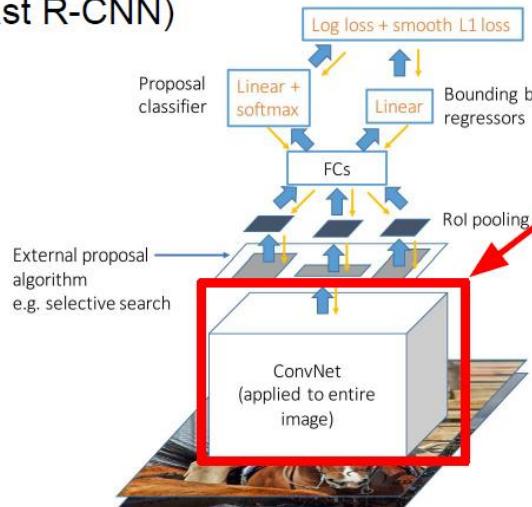
- Les innovations dans les architectures de réseaux faits pour une tâche X ont tendance à aussi aider pour de nombreuses autres tâches
- Synergies et emballages

# « Bouturage »

- Combiner des réseaux pré-entraînés sur des tâches différentes



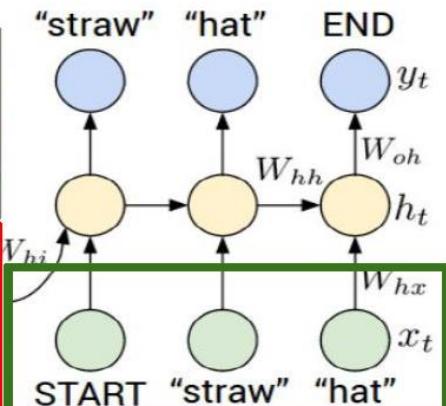
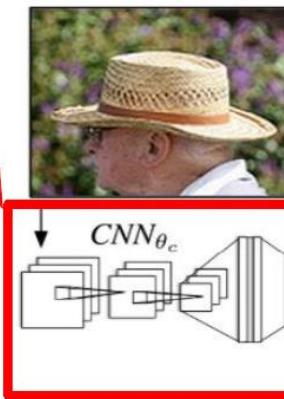
Object Detection  
(Fast R-CNN)



CNN pretrained  
on ImageNet

Image Captioning: CNN + RNN

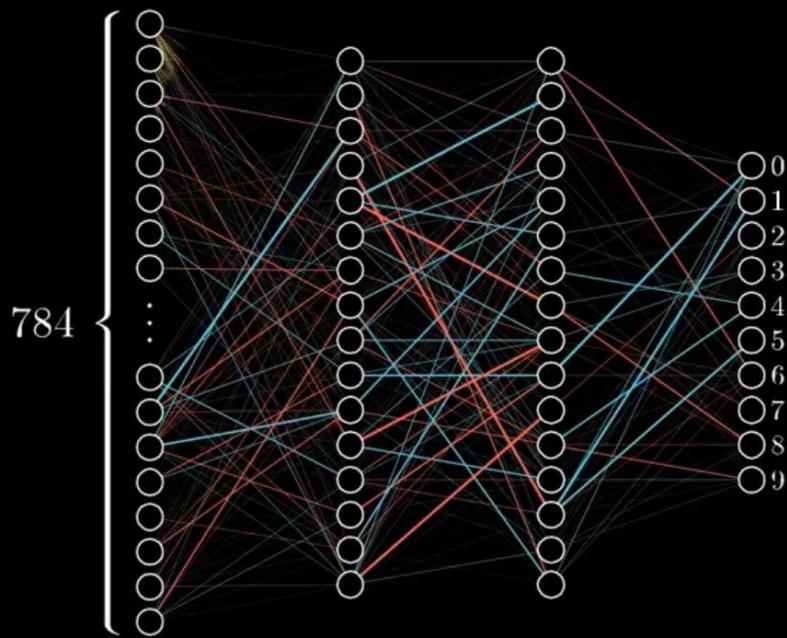
Word vectors pretrained  
with word2vec



# Le gradient = sève

- Pas de gradient = pas d'apprentissage

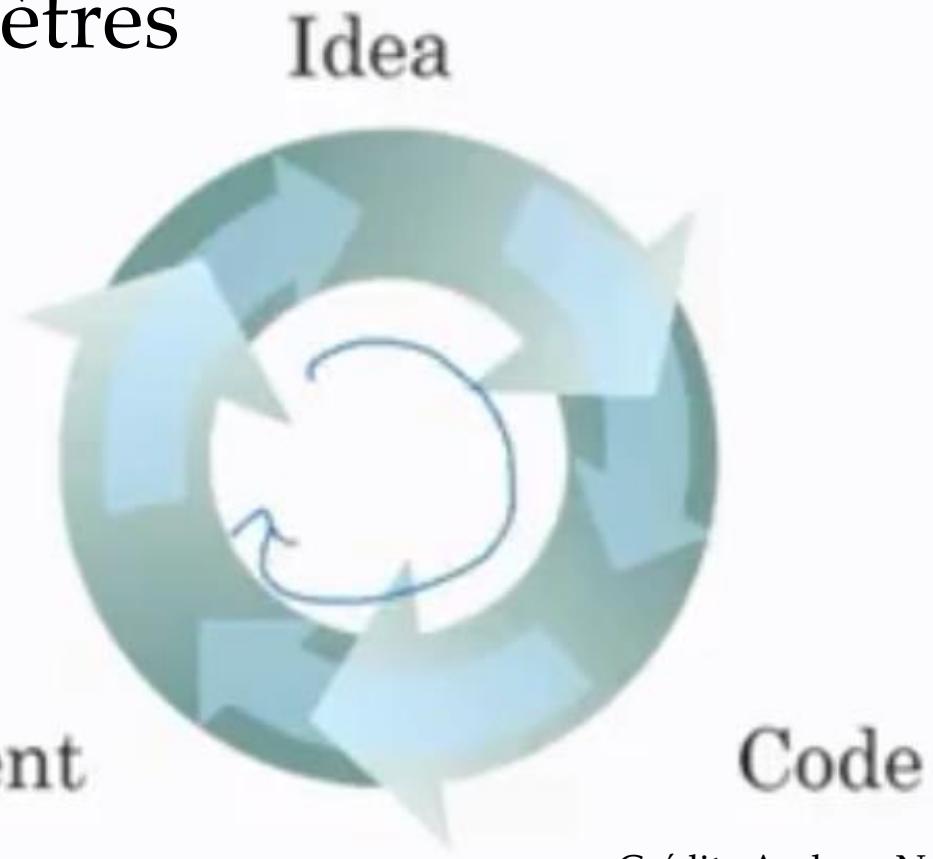
Training in  
progress. . .



# Processus itératif

*Les voies du réseaux sont impénétrables*

- Beaucoup de paramètres et de techniques
- Important de développer une intuition



Crédit : Andrew Ng

# Optimisation vs. gradient

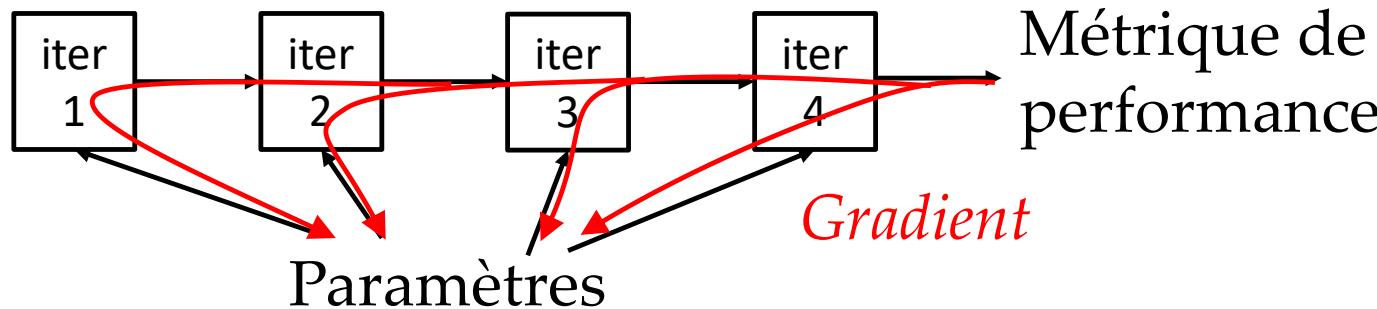
- Théorème d'approximation universelle ne dit pas comment trouver cette fonction
- Relation incestueuse entre les architectures développées (dérivable *end-to-end*) et les méthodes d'optimisation
- Autres approches (Hebbian), mais on n'en parlera pas dans le cours

# Toujours besoin de beaucoup de données?

- Non, si l'on passe par des réseaux pré-entraînés
- Fantastiques extracteurs de caractéristiques
- Résultats dépassent souvent l'état de l'art pré-Deep

# Paradigme du graphe de calcul

- Plus qu'un *à-côté* des recherches en *Deep*
- Nouvelle manière de voir les algorithmes
  - Prend algo traditionnel
  - Rend pleinement différentiable (*soft*)
    - choix binaire 0-1 → sigmoïde (NetVLAD)
  - « Dérouler » l'exécution



# Deep Learning : appel à la prudence

- Excellente présentation NIPS 2017 d'Ali Rahimi : *Deep Learning is alchemy*
- <https://youtu.be/Qi1Yry33TQE>
- Doit chercher plus de rigueur, comprendre ce qui se passe

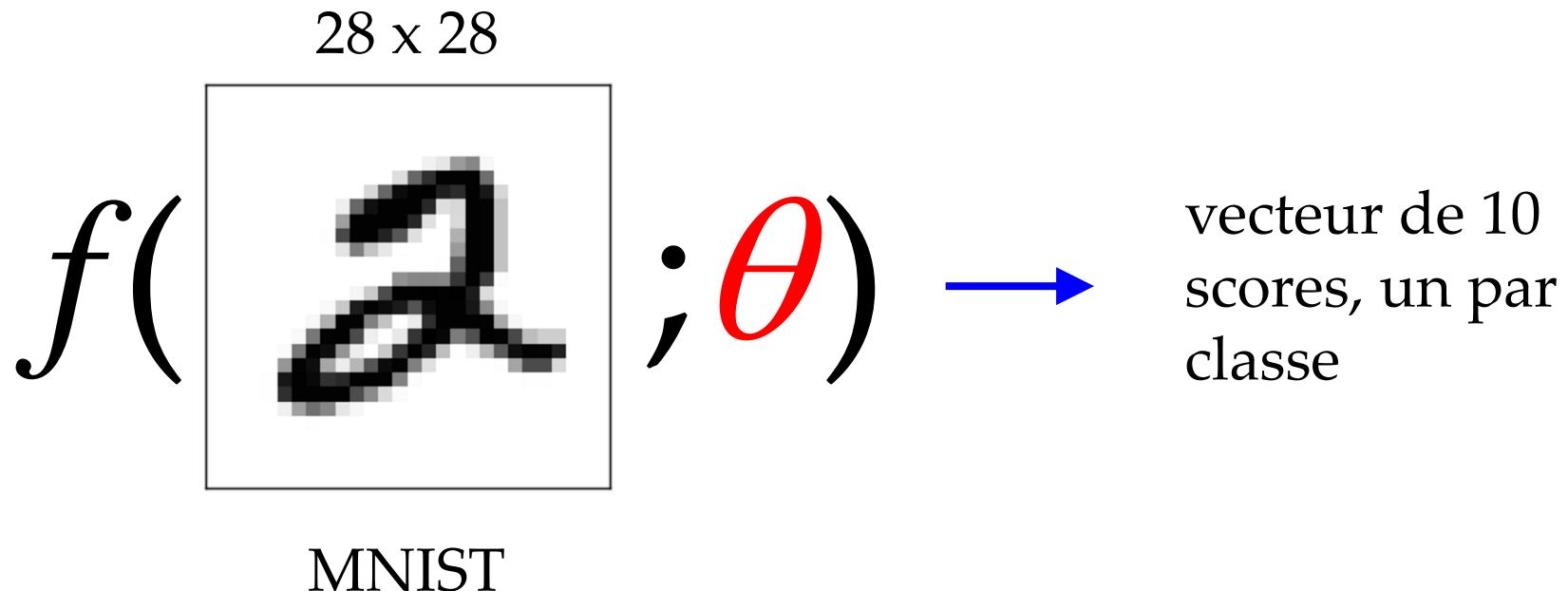
# Deep Learning : appel à la prudence

## Troubling Trends in Machine Learning Scholarship, Lipton et al., 2018

1. *Failure to distinguish between explanation and speculation.*
2. *Failure to identify the sources of empirical gains, e.g. emphasizing unnecessary modifications to neural architectures when gains actually stem from hyper-parameter tuning.*
3. *Mathiness: the use of mathematics that obfuscates or impresses rather than clarifies, e.g. by confusing technical and non-technical concepts.*
4. *Misuse of language, e.g. by choosing terms of art with colloquial connotations or by overloading established technical terms.*

# Exemples d'applications

# Reconnaissance de caractères



(Labo 1)

$\theta$ : paramètres de la fonction

# Reconnaissance d'images

224 x 224 x 3

$$f(\quad ; \theta)$$



vecteur de 1000 scores, un par classe

ImageNet

Note : les fonctions seront différentes à l'autre d'un exemple à l'autre

$\theta$  : paramètres de la fonction

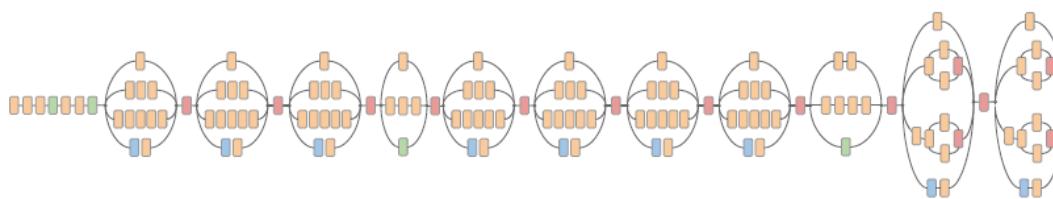
# Dermatologist-level classification of skin cancer with deep neural networks

Andre Esteva<sup>1\*</sup>, Brett Kuprel<sup>1\*</sup>, Roberto A. Novoa<sup>2,3</sup>, Justin Ko<sup>2</sup>, Susan M. Swetter<sup>2,4</sup>, Helen M. Blau<sup>5</sup> & Sebastian Thrun<sup>6</sup>

Skin lesion image

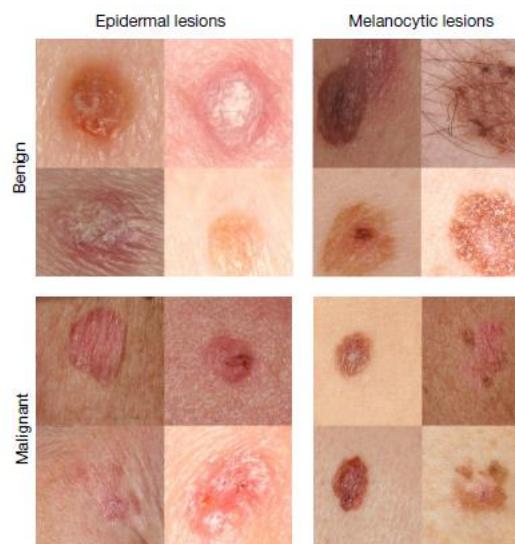
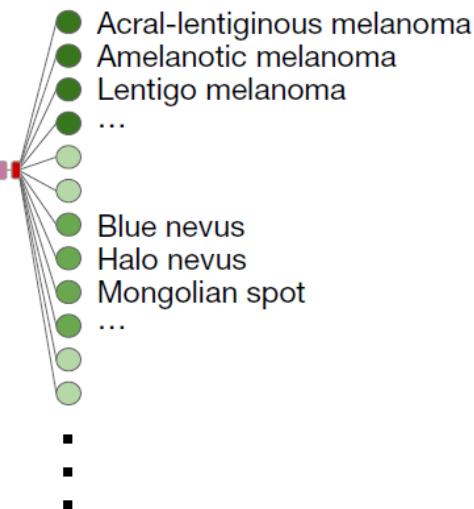


Deep convolutional neural network (Inception v3)



- Convolution
- AvgPool
- MaxPool
- Concat
- Dropout
- Fully connected
- Softmax

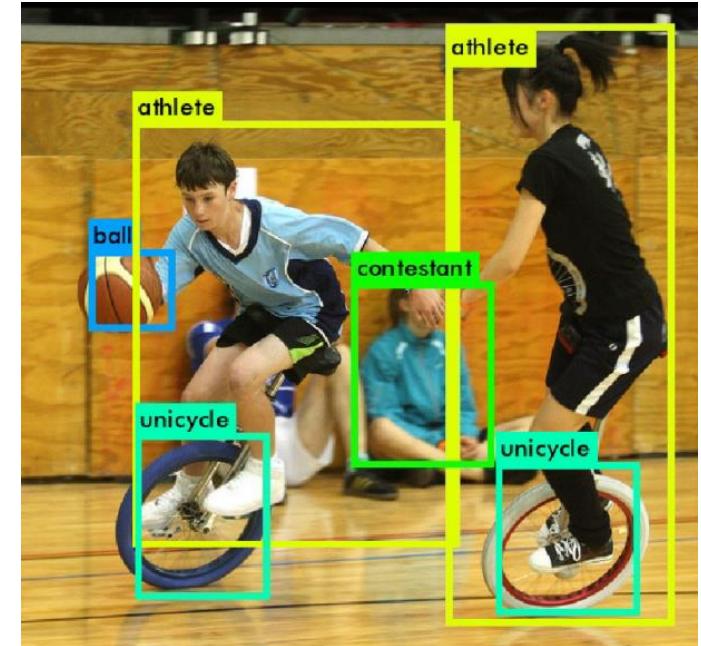
Training classes (757)



130,000 images  
d'entraînement

# Détection d'objets

$f(\cdot; \theta)$  →



# Description d'images

$$f(\quad ; \theta)$$



Construction  
worker in orange  
safety vest is  
working on road

# Qui est cet acteur?



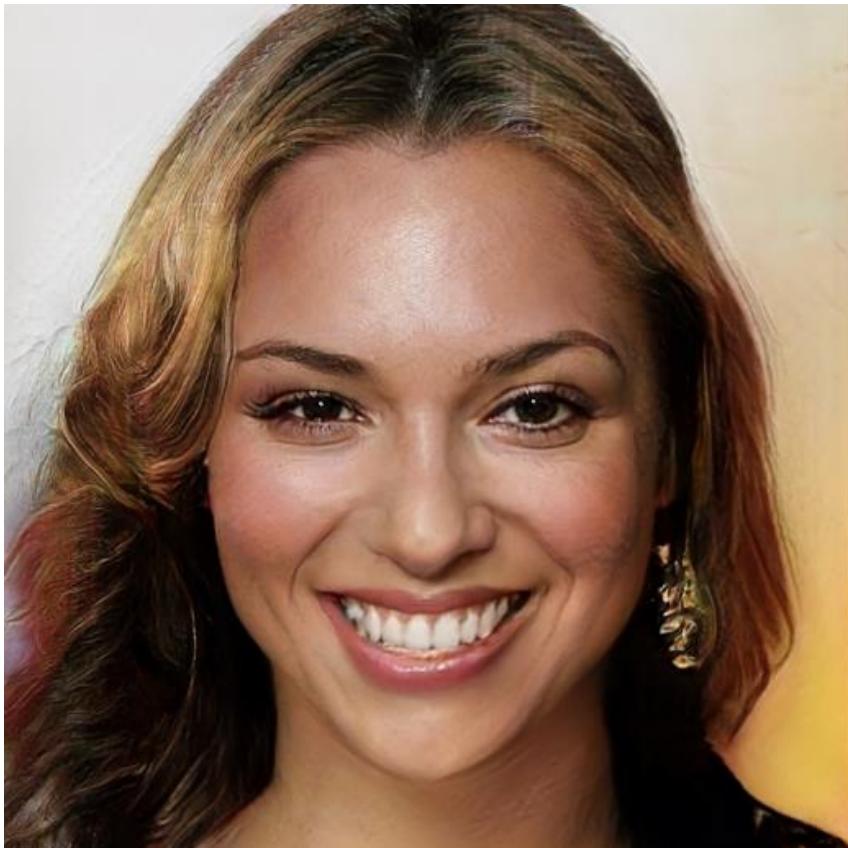
# Génération d'images

$$f(\begin{array}{|c|c|c|}\hline & & \\ \hline & & \\ \hline\end{array}; \theta) \rightarrow$$



Karras et al., Progressive Growing of GANs for improved quality, stability and variation, ICLR 2018.

# Génération d'images (NVIDIA)



# Reconnaissance voix

$$f(\text{}; \theta) \rightarrow \text{Ok Google, where is my car}$$

# Génération de voix : *Tacotron 2*

$f($  She earned a  
doctorate in  
sociology at  
Columbia  
University ;  $\theta$ ) \rightarrow



Note : exemple d'entraînement :



« George Washington was the first President of the United States »

A:

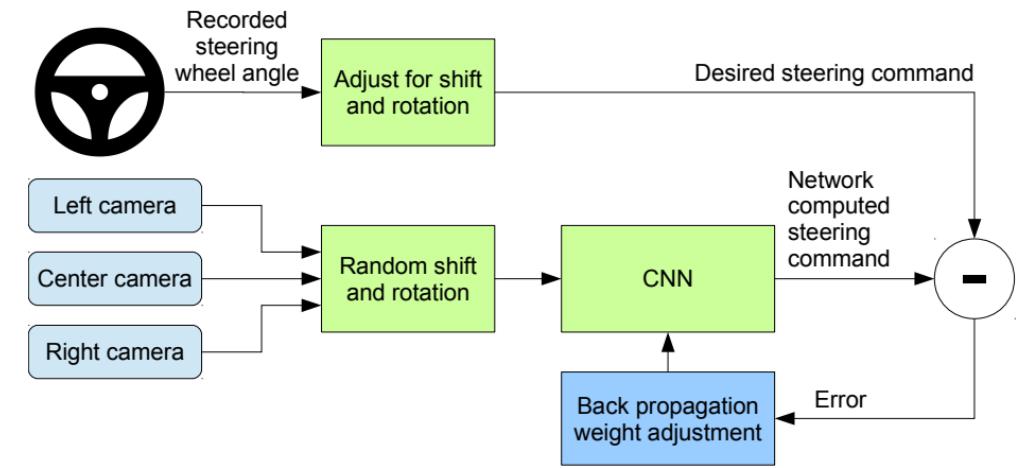
B:

Natural TTS Synthesis by Conditioning WaveNet on Mel Spectrogram Predictions  
[Jonathan Shen](#), arXiv:1712.05884, dec 2017.

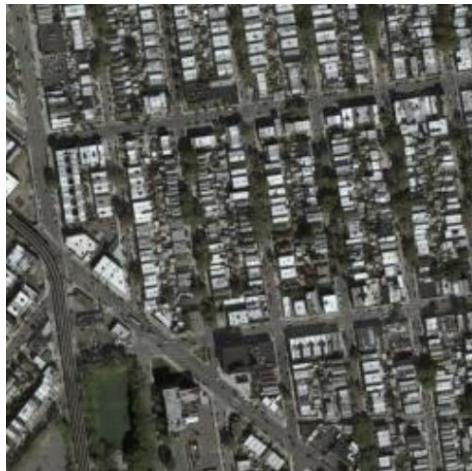
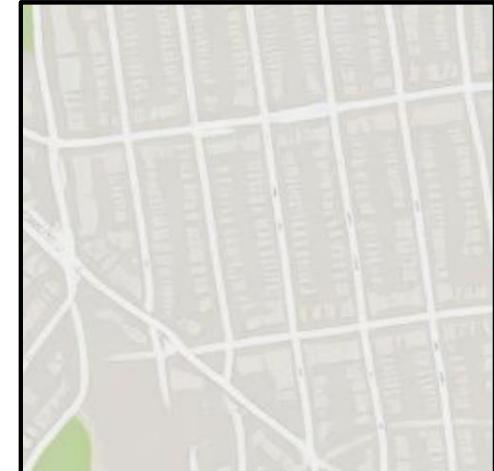
# Traduction automatique

$f($  I think,  
therefore  
I am.  $; \theta)$  → Je pense  
donc je  
suis.

# Apprentissage visuomoteur

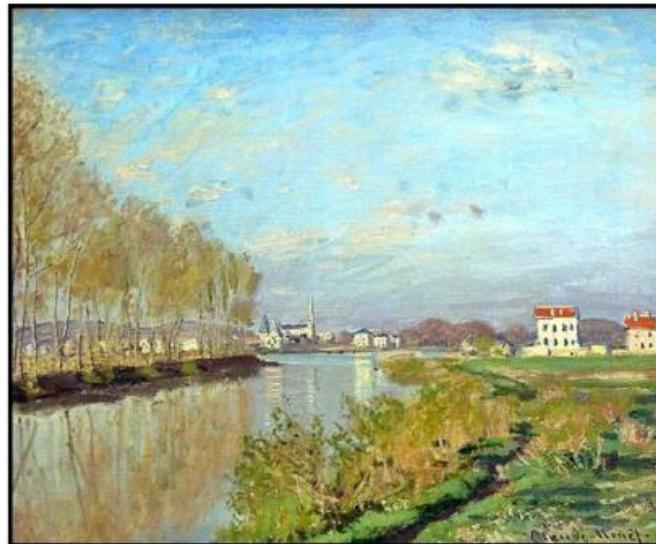


# Transfert de style

 $G($  $; \theta) \rightarrow$  $F($  $; \theta) \rightarrow$ 

# Transfert de style

Monet  Photos



Monet → photo

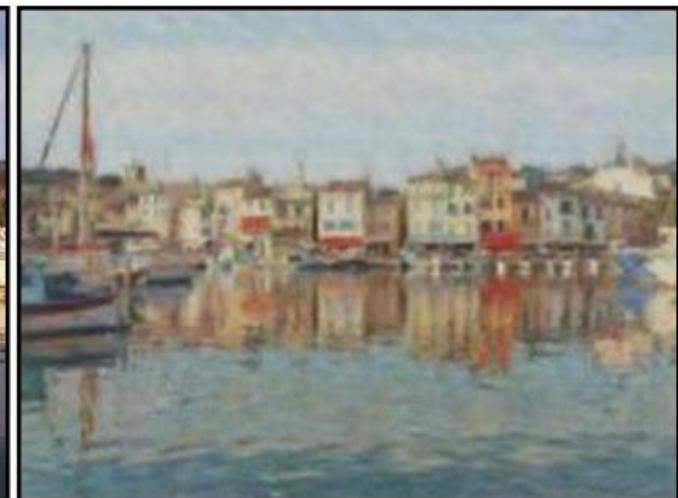
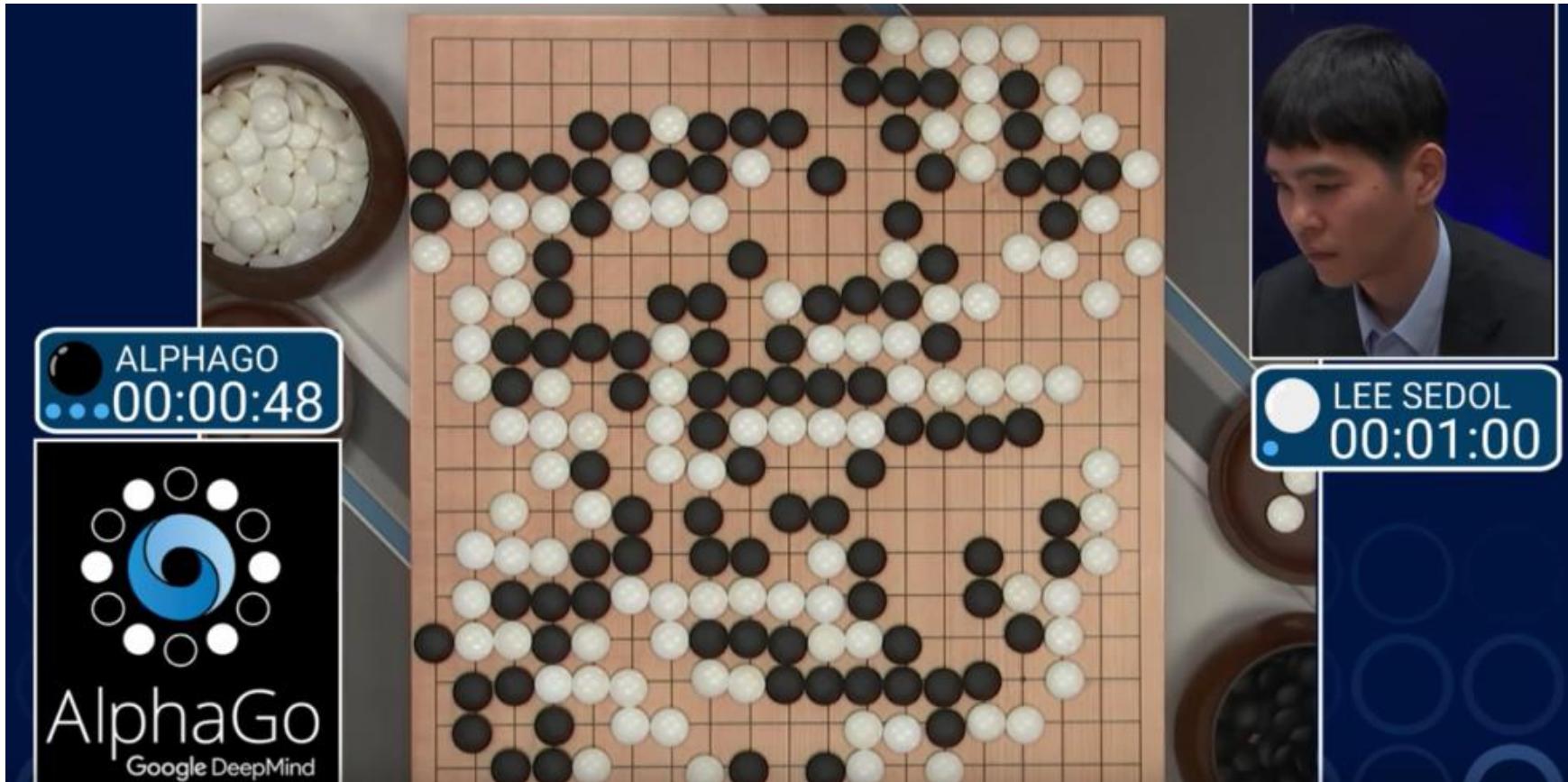


photo → Monet

# Jeu de Go (2017)



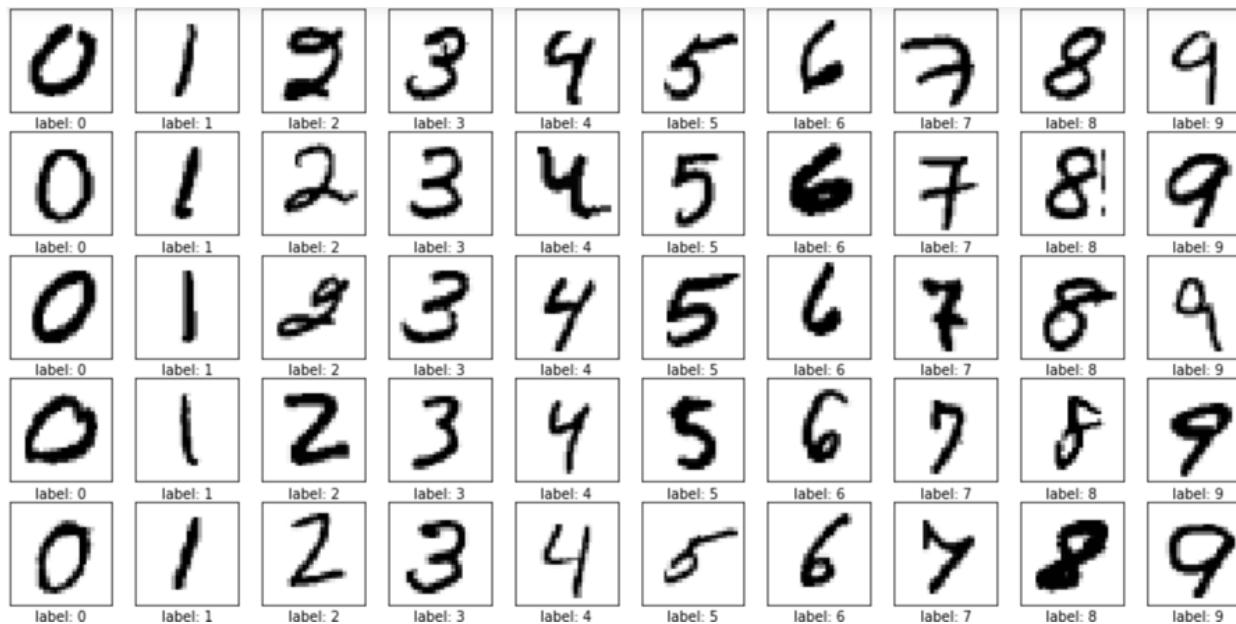
# Jeu Starcraft (2019)

- Programme AlphaStar de Deepmind
- Niveau Grand maître pour Starcraft
  - meilleur que 99.8% des joueurs humains
- Basé sur des approches d'apprentissage génériques
  - réseaux pour les stratégies et contre-stratégies

# Réseaux classiques peu profonds : classification d'images

# MNIST

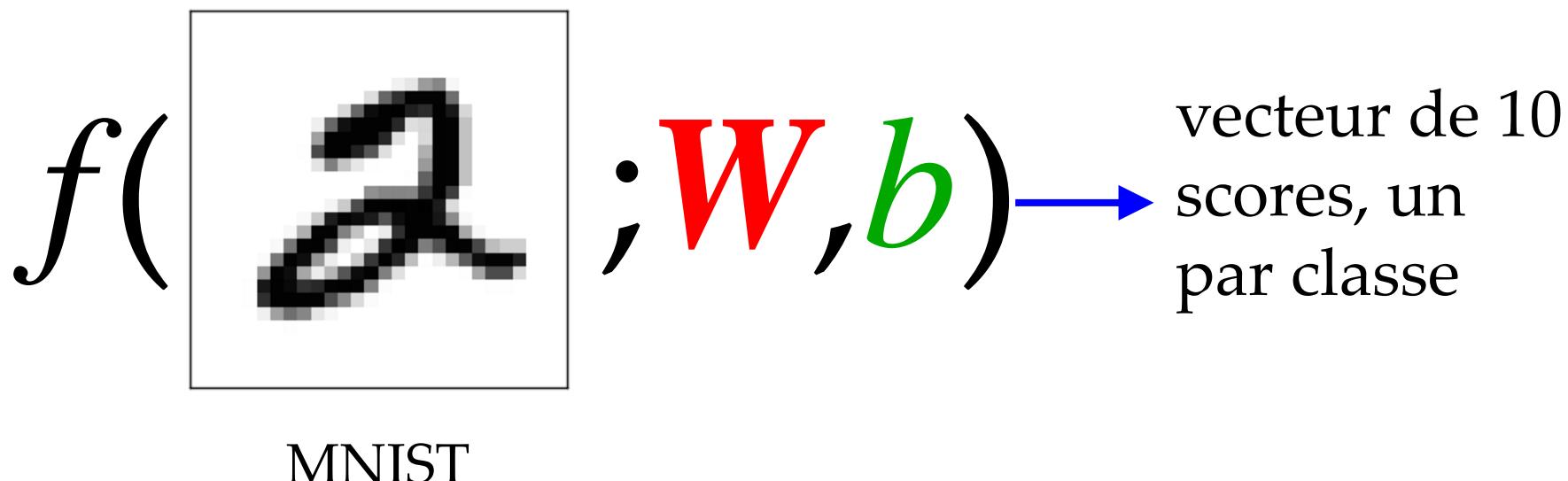
- 10 chiffres
- 70,000 images de  $28 \times 28$  pixels



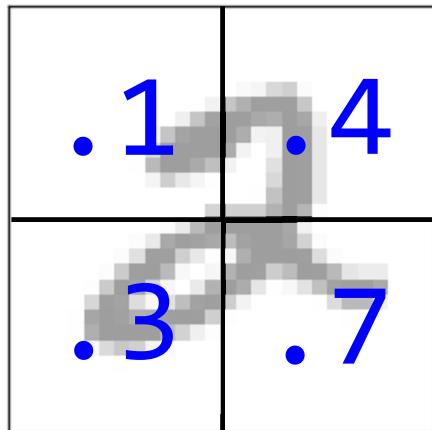
Source : <http://blog.welcomege.com/mnist-database/>

# Classificateur linéaire sur MNIST

- Le plus simple possible : **linéaire**
- Paramètres  $\mathbf{W}$  (taille  $c \times n$ ) + biais  $\mathbf{b}$  (taille  $c \times 1$ )
  - $n$  : nombre de pixels
  - $c$  : nombre de classes



# Exemple simplifié



« Flatten »

$$f = Wx + b$$

$W$

$$\begin{matrix} .7 & - .3 & .2 & .3 \\ -.5 & 1.7 & 1.5 & .4 \\ .7 & 1.1 & -.4 & -.1 \end{matrix} \quad \begin{matrix} .1 \\ .4 \\ .3 \\ .7 \end{matrix}$$

$b$  Score

$$+ \begin{matrix} 1 \\ 1.2 \\ -.4 \end{matrix} = \begin{matrix} 1.22 \\ 2.56 \\ -.08 \end{matrix}$$

Chiffre 1

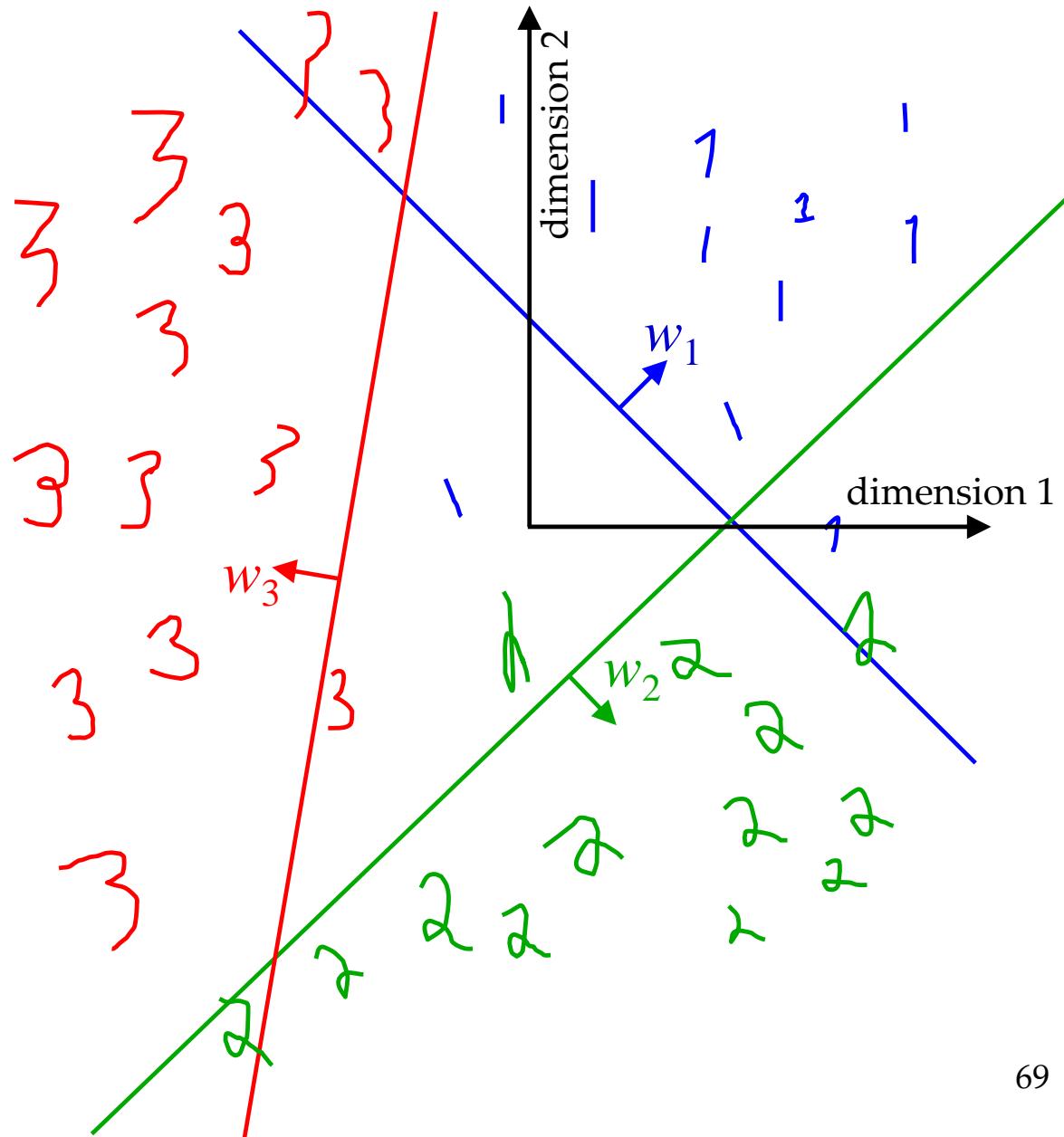
Chiffre 2

Chiffre 3

# Interprétation classificateur linéaire

$$f = Wx + b$$

	$w_1$	$w_2$	$w_3$
	.7	-.3	.2
	-.5	1.7	1.5
	.7	1.1	-.4



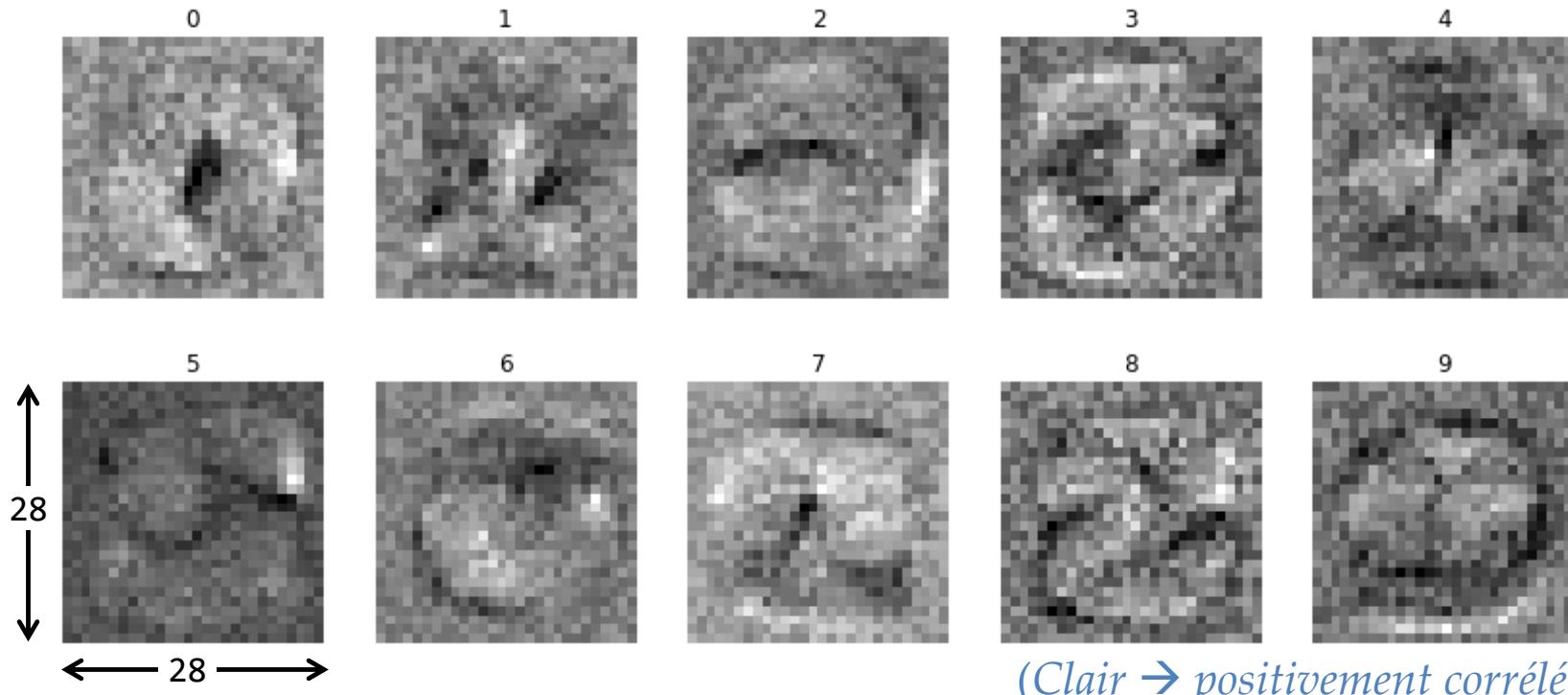
# Réseau 1 couche linéaire

- Image 28x28 → 784x1
- Matrice  $W$  10x784
- Biais  $b$  10x1
- Initialisé au hasard
- Entrainement SGD sur perte *multiclass hinge loss*
- Train set : 60,000 exemples
- Test set : 10,000 exemples (non vues à l'ent.)
- Résultat ~92% précision sur test

```
class SimpleLinear(nn.Module):  
    def __init__(self):  
        super(SimpleLinear, self).__init__()  
        self.fc1 = nn.Linear(784, 10)  
  
    def forward(self, x):  
        x = x.view(-1, 784) # Flatten  
        x = self.fc1(x)  
        return x
```

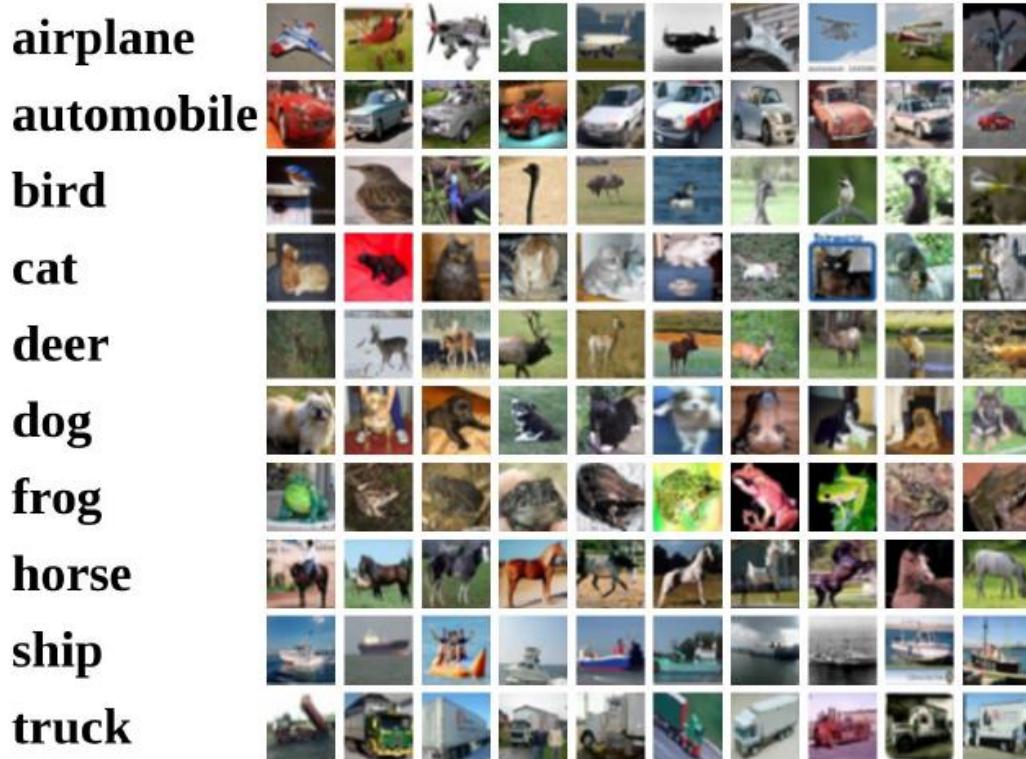
$$L = \frac{1}{m} \sum_{i \neq cible} \max(0, 1 - \underbrace{\text{sortie}(cible)}_{+ petits} + \underbrace{\text{sortie}(i)}_{+ grand})$$

# Poids $W$ appris : patrons $28 \times 28$

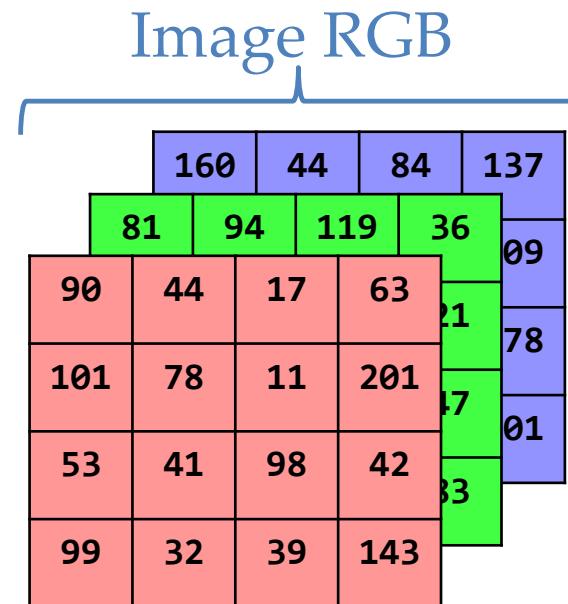


- On voit corrélation spatiale sur pixels voisins
- Réseau doit la découvrir : doit avoir plus d'exemples d'entraînement (*sample efficiency*)
- Si applique permutation identique sur les pixels, même précision

# Sur images couleur CIFAR-10



$$32 \times 32 \times 3 = 3072$$



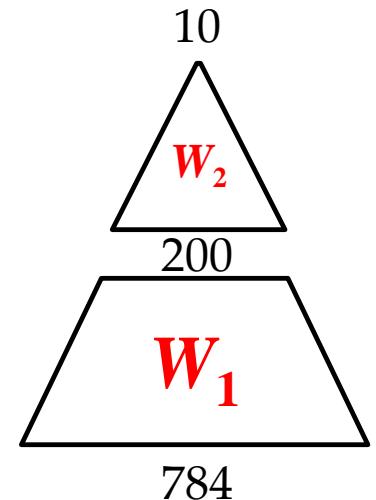
Exemple de W trouvé



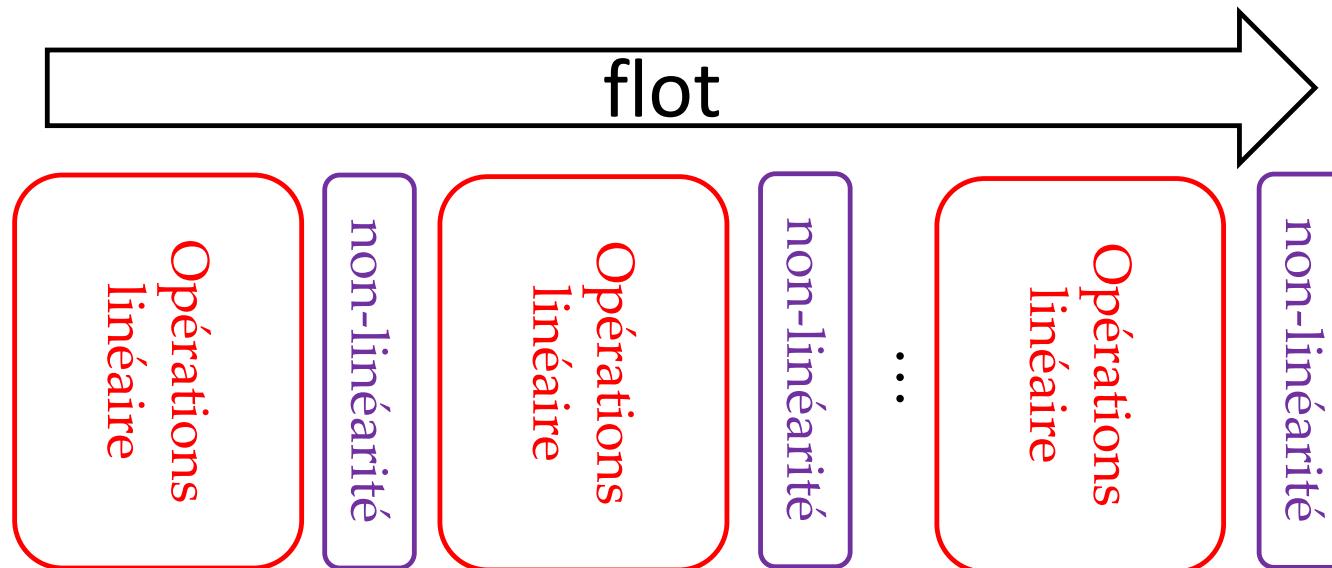
(Note : biais dans le dataset : l'image est centrée sur l'objet)

# Réseau 2 couches linéaires

- Matrice  $W_1$  200x784, bias  $b_1$  200x1
- Matrice  $W_2$  10x200, biais  $b_2$  10x1
- # paramètres  $\sim 160,000$ 
  - précédent  $\sim 8,000$  paramètres
- Résultats? Encore  $\sim 92\%$ !
- Pas plus expressif que linéaire 1 couche, car se simplifie :
$$W_2(W_1x + b_1) + b_2 = Wx + b$$
- Ajout de non-linéarités pour augmenter la puissance d'expression

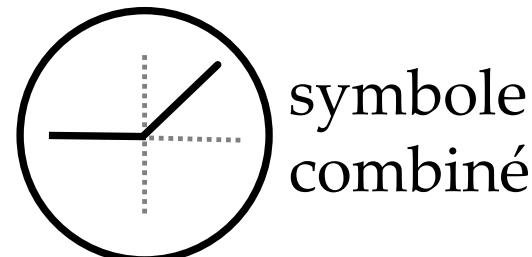
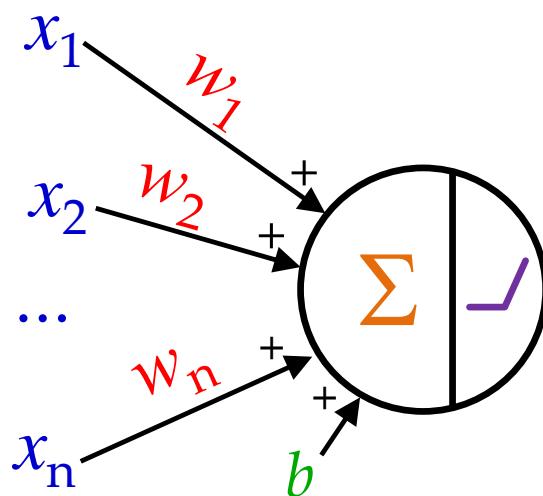


# Ajout de non-linéarité



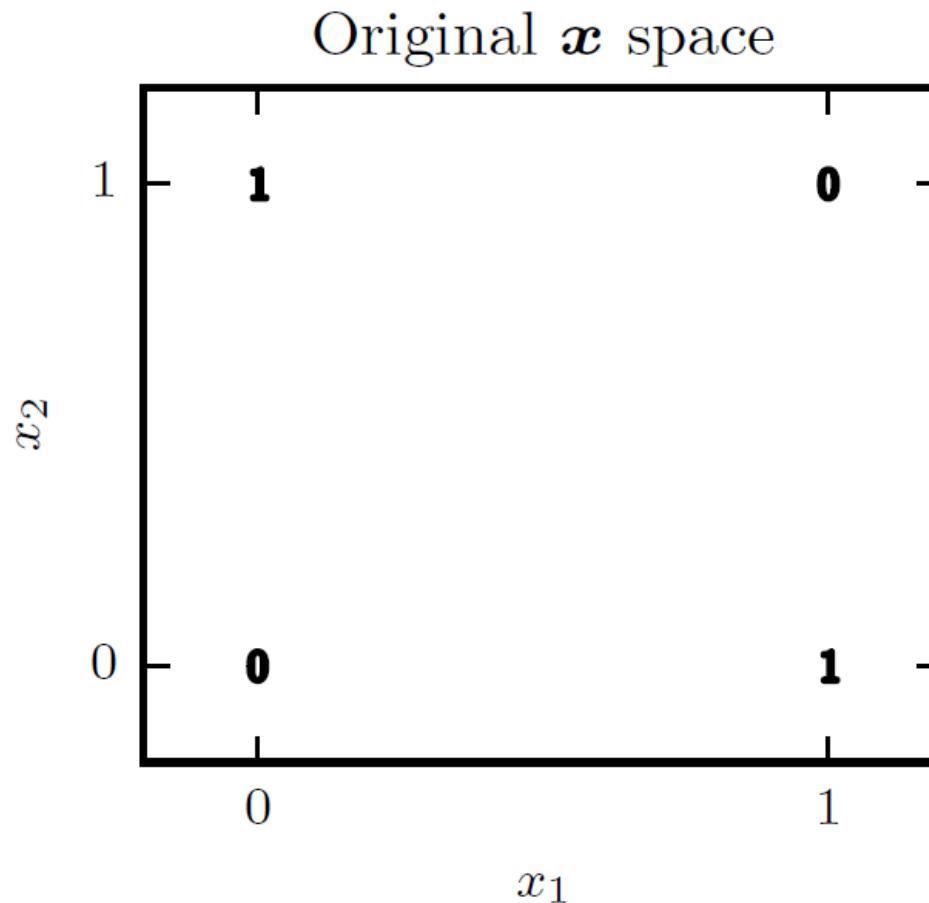
$$\text{Pré-activation : } z = b + \sum_i w_i x_i = b + w^T x$$

$$\text{Activation : } h = g(z) \leftarrow \text{non-linéaire}$$

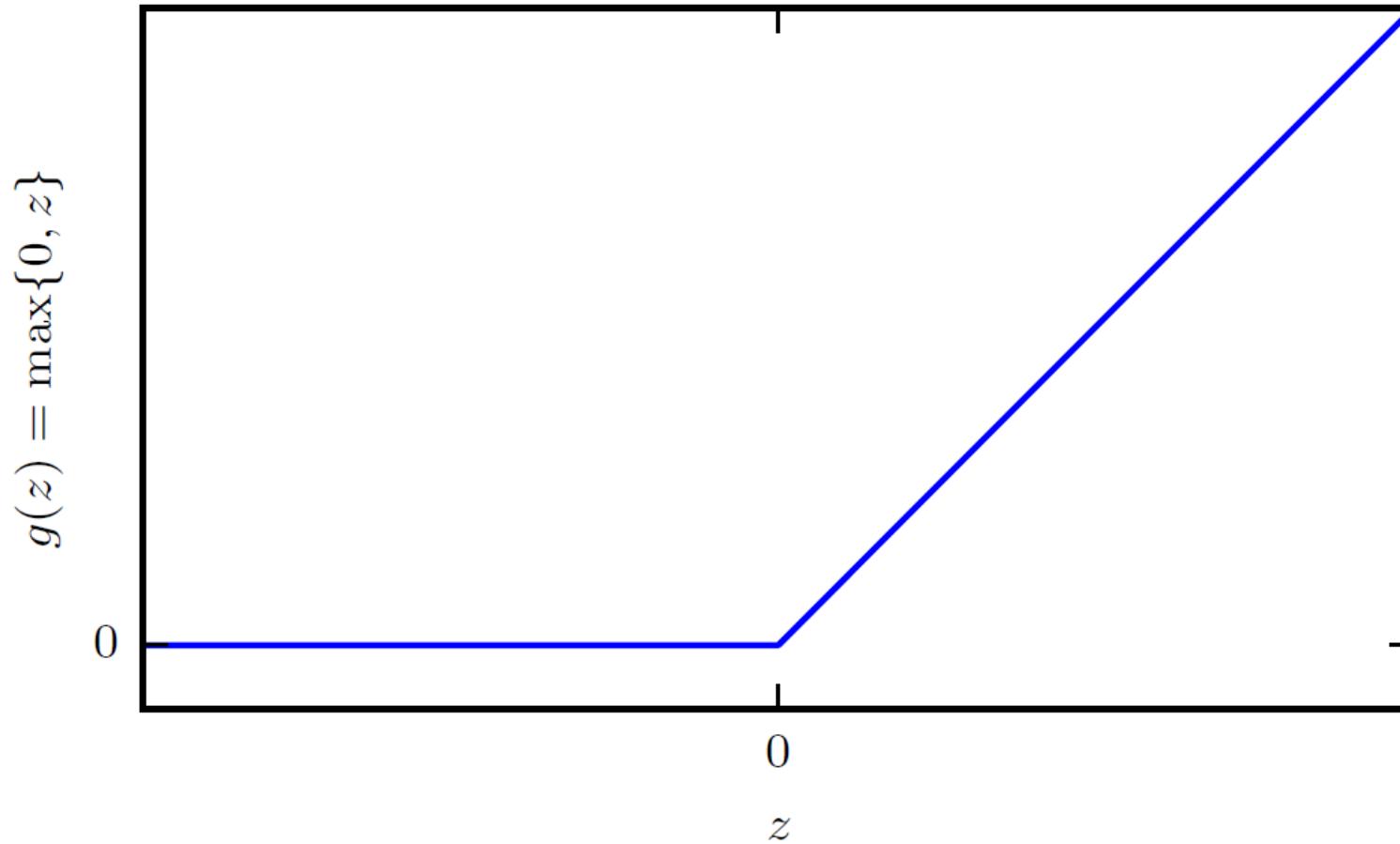


# Exemple classique : XOR

Non-séparable linéairement



# Activation non-linéaire ReLU



(La plus populaire en ce moment)

# Solution XOR (format vectorisé)

4 exemples

$$X = \begin{bmatrix} x^{(1)} & x^{(2)} & x^{(3)} & x^{(4)} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \end{bmatrix}$$

$$W_1 = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}, \quad b_1 = \begin{bmatrix} 0 \\ -1 \end{bmatrix} \quad Y = [ \textcolor{red}{0} \quad \textcolor{green}{1} \quad \textcolor{green}{1} \quad \textcolor{red}{0} ]$$

4 pré-activations

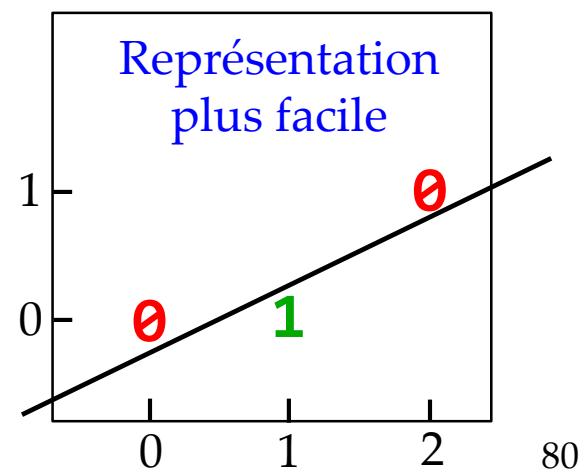
$$Z = \begin{bmatrix} z_1^{(1)} & z_1^{(2)} & z_1^{(3)} & z_1^{(4)} \end{bmatrix} = W_1 X + b_1 = \begin{bmatrix} 0 & 1 & 1 & 2 \\ 0 & 1 & 1 & 2 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 & 0 \\ -1 & -1 & -1 & -1 \end{bmatrix}$$

$$Z = \begin{bmatrix} 0 & 1 & 1 & 2 \\ -1 & 0 & 0 & 1 \end{bmatrix}$$

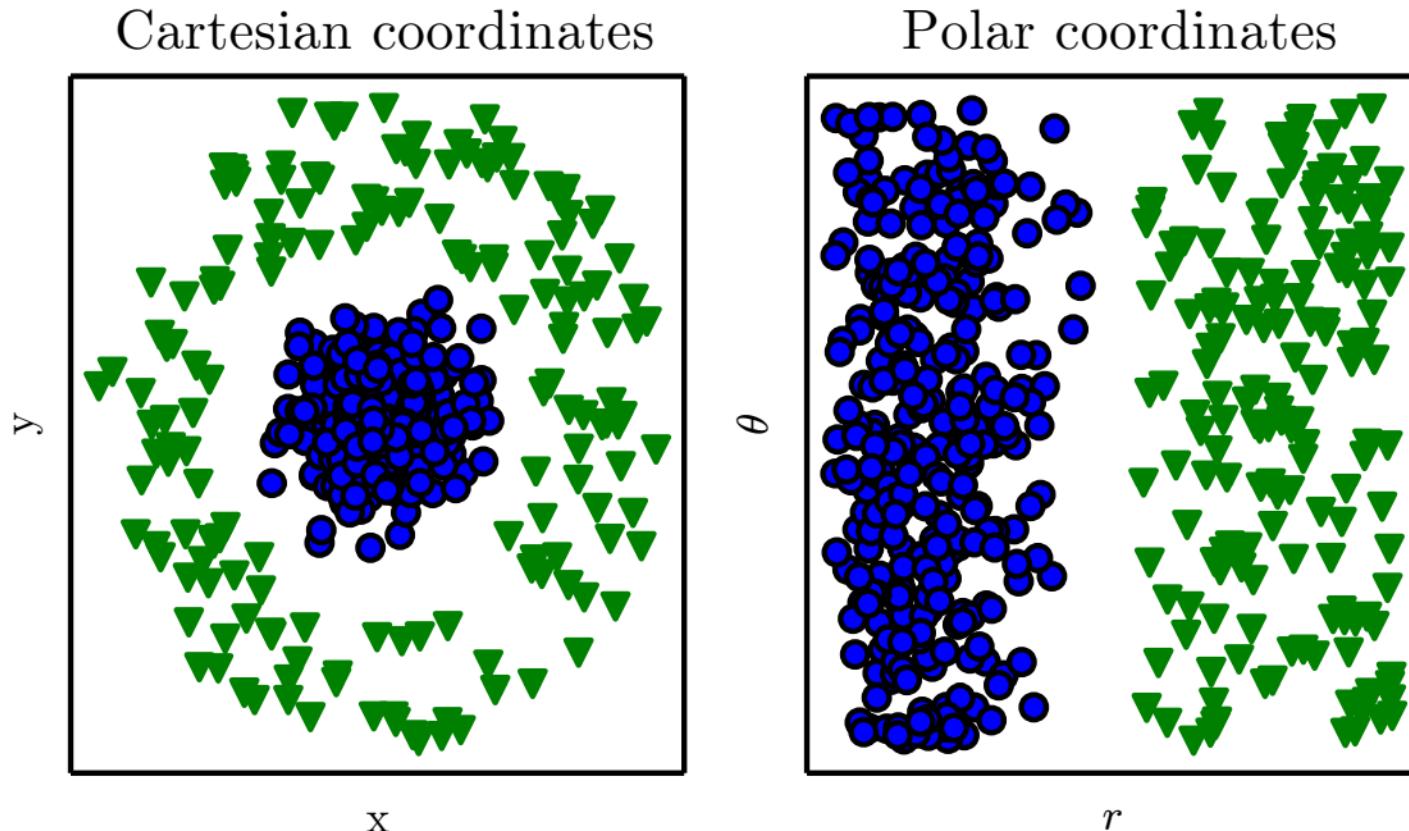
$$g(Z) = \max\left(0, \begin{bmatrix} 0 & 1 & 1 & 2 \\ -1 & 0 & 0 & 1 \end{bmatrix}\right) = \begin{bmatrix} 0 & 1 & 1 & 2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

ReLU

Représentation plus facile

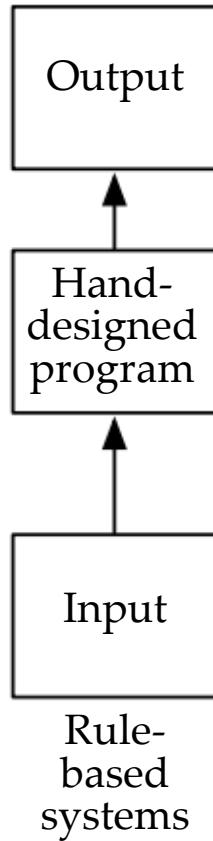


# Importance de la représentation

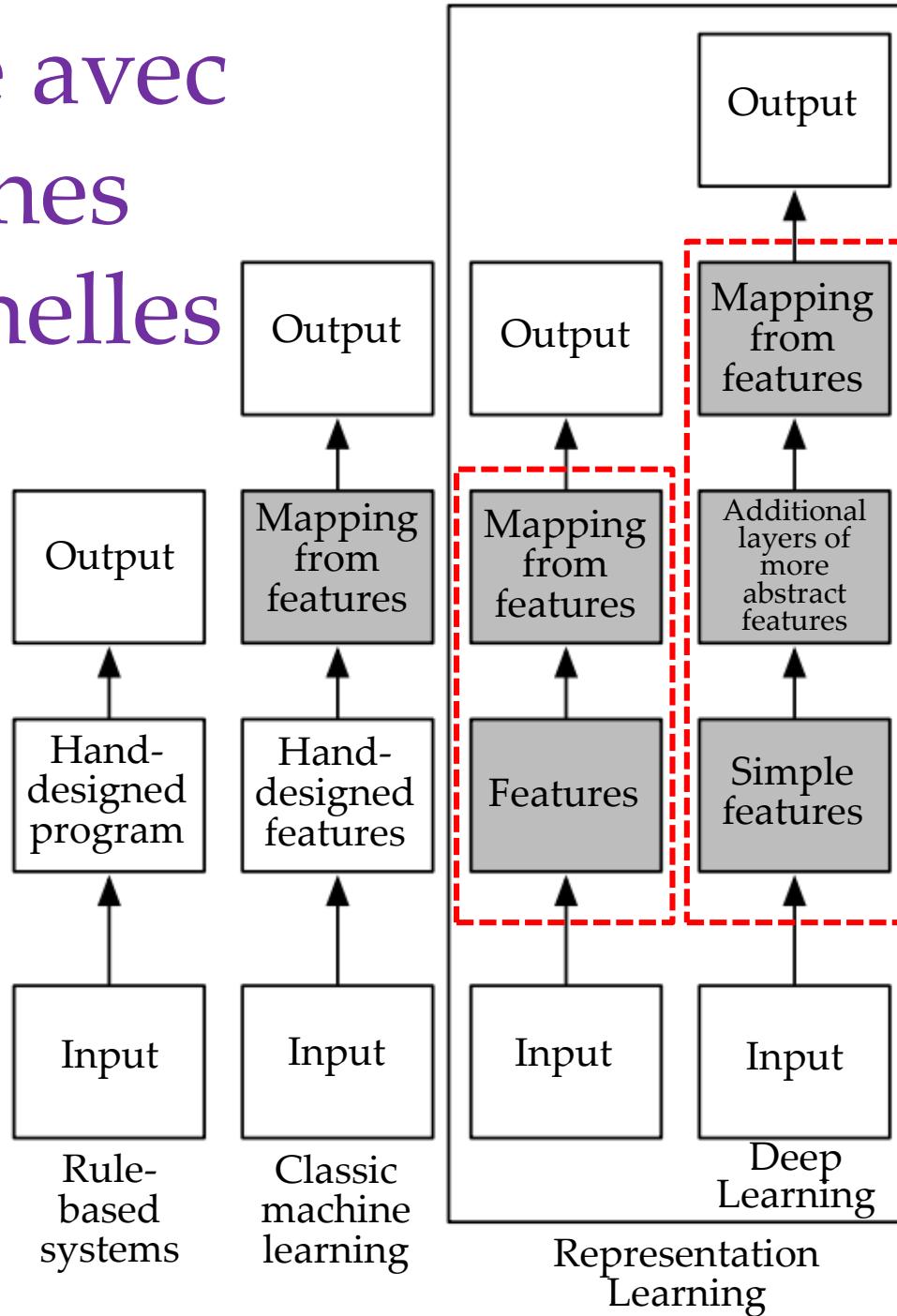


L'un est séparable linéairement, l'autre non

# Contraste avec approches traditionnelles



# Contraste avec approches traditionnelles



Appris  
conjointement

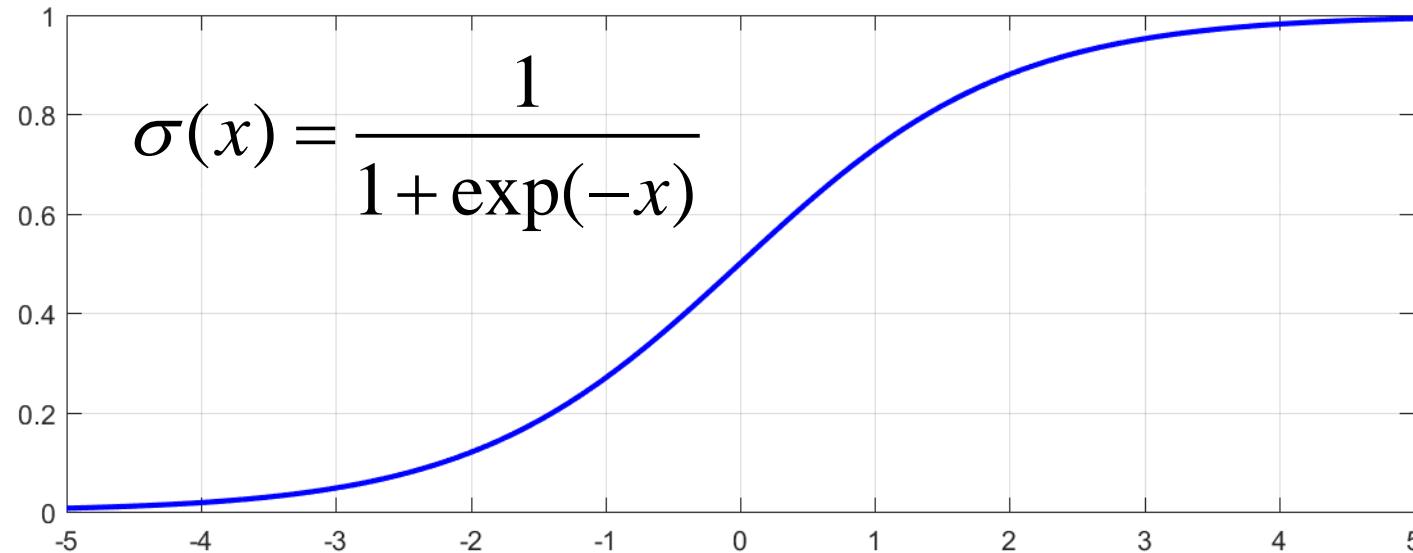
# Fonctions d'activations

# Rôles

- Apporte une non-linéarité dans le réseau
- Situé à l'interne, ou en sortie
- Considérations :
  - difficulté d'entraînement (son gradient)
  - comportement se rapproche :
    - de ce que l'on cherche à prédire en sortie (probabilités, one-hot vector, angles, déplacements, etc.)
    - action particulière (*gating*)
  - temps de calcul

# Fonction d'activation : sigmoïde

- Une des premières utilisées
- Si on désire une sortie entre 0 et 1 (*squashing*)



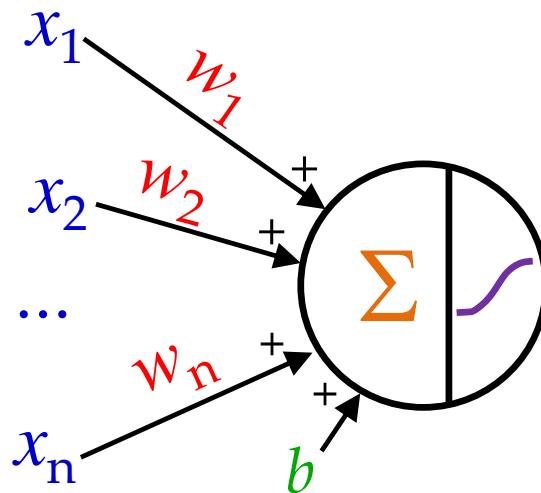
- Tombée en désuétude comme non-linéarité de base

# Exemple utilisation sigmoïde

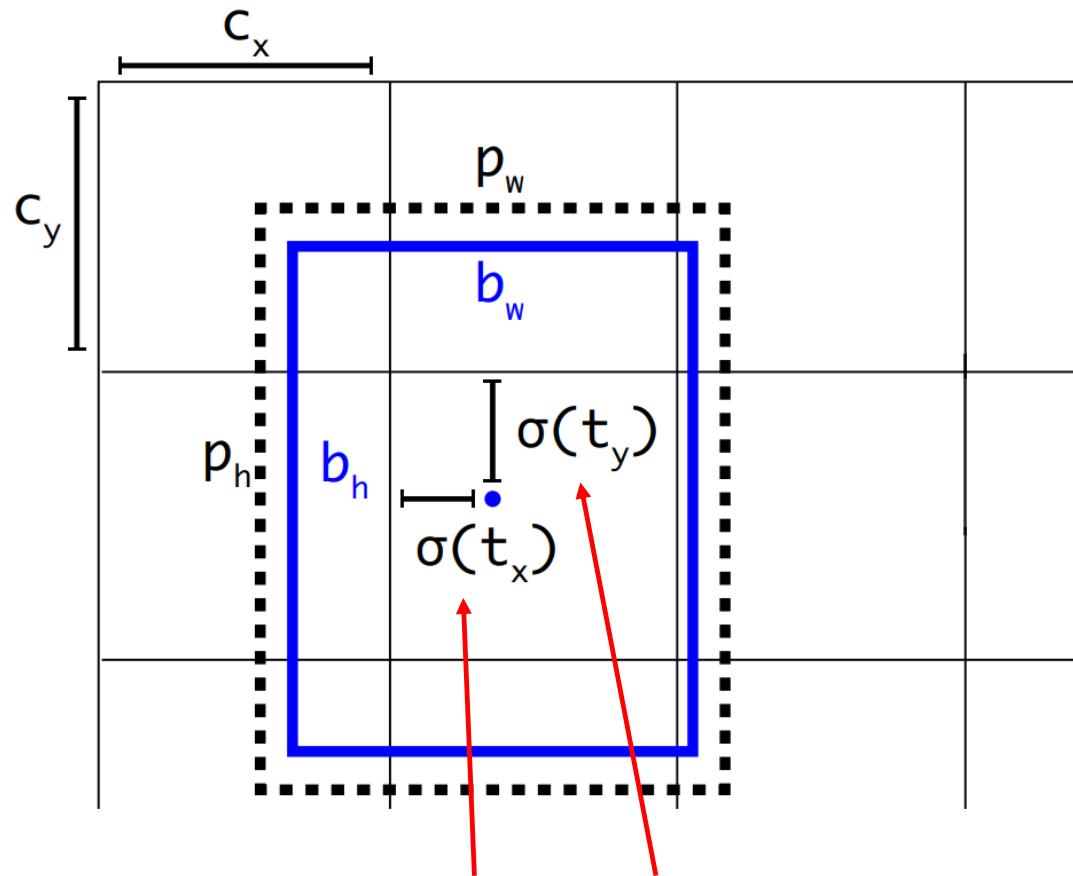
- Prédiction binaire (logistic regression)

$$P(y = 1 | x)$$

$$\text{sortie} = \sigma(w^T x + b)$$



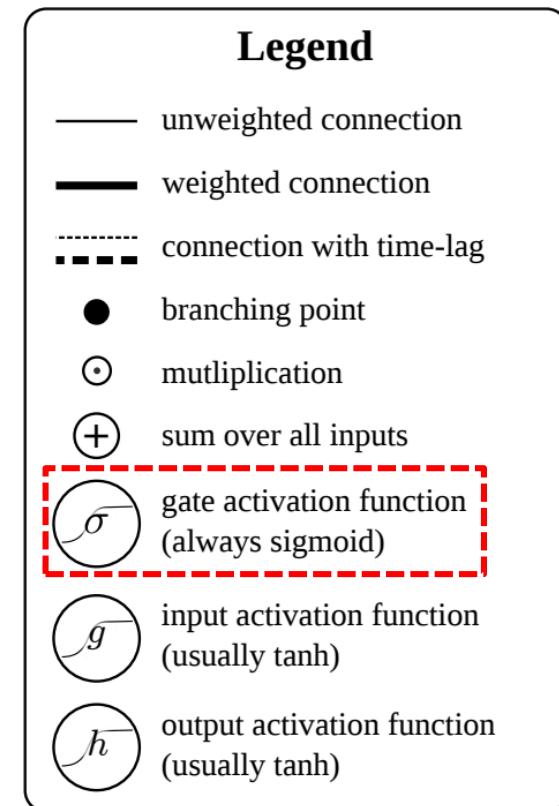
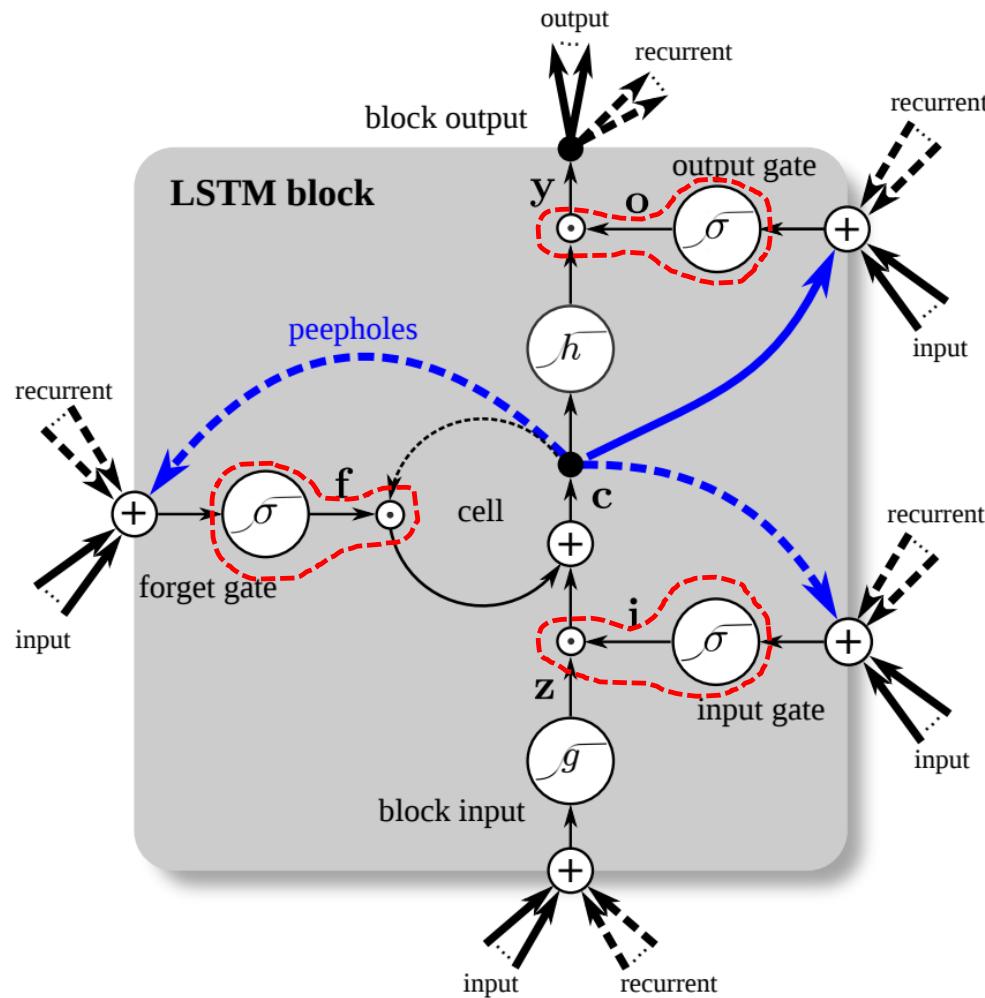
# Exemples utilisation sigmoïde



Prédire poses relatives

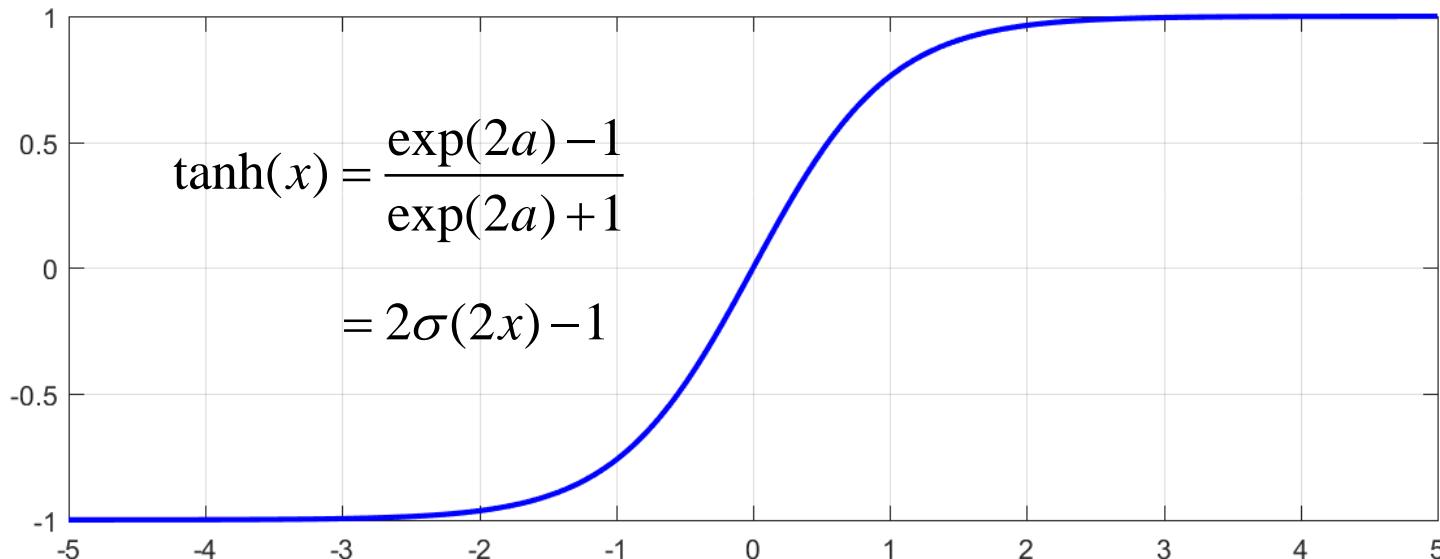
# Exemple utilisation sigmoïde

- *gating* dans Long short-term memory



# Fonction d'activation : tanh

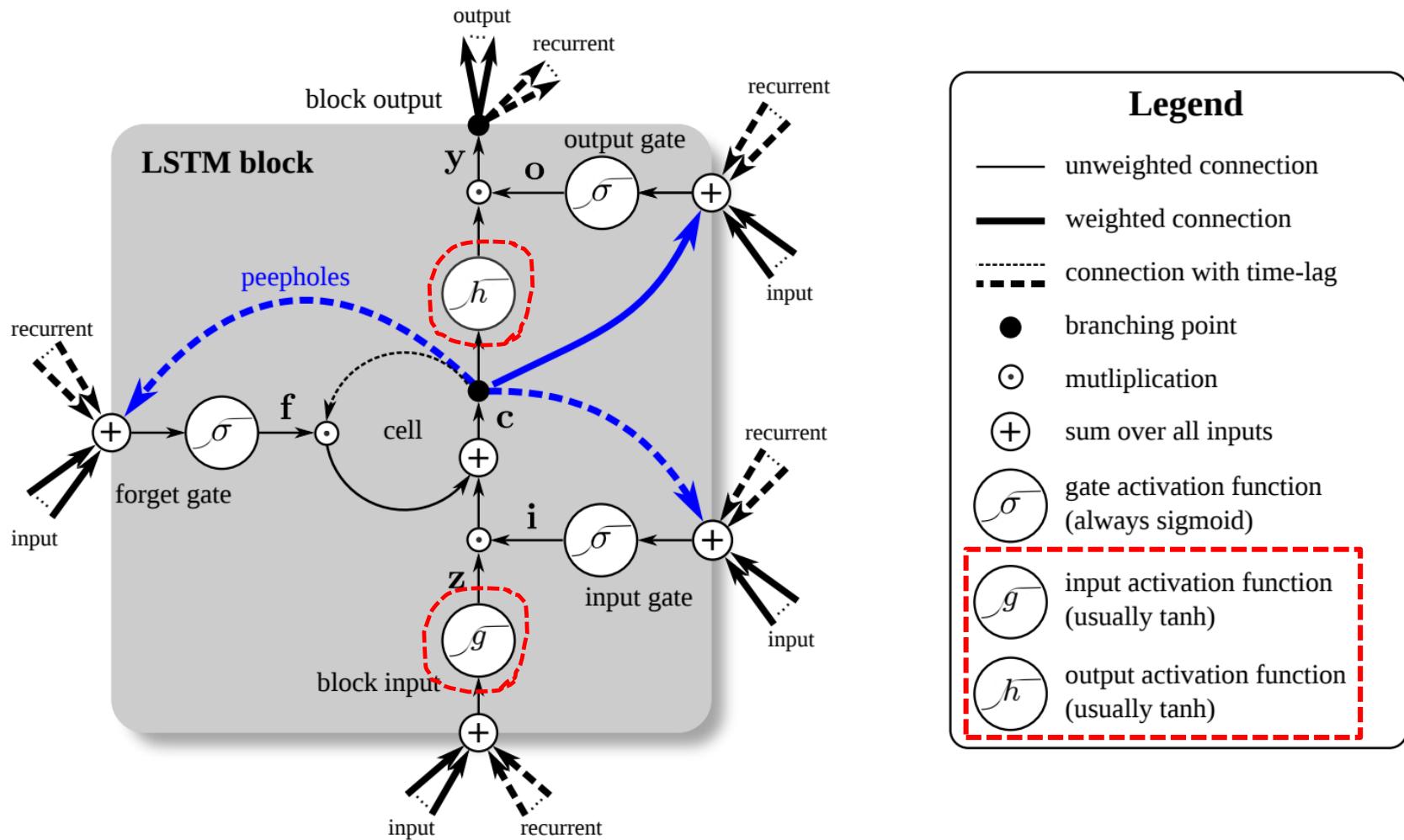
- Autre fonction historique
- Désire une sortie entre -1 et 1 (*squashing*)



- Donne une sortie centrée à 0 (préférable à 0.5 de la sigmoïde)

# Exemple utilisation tanh

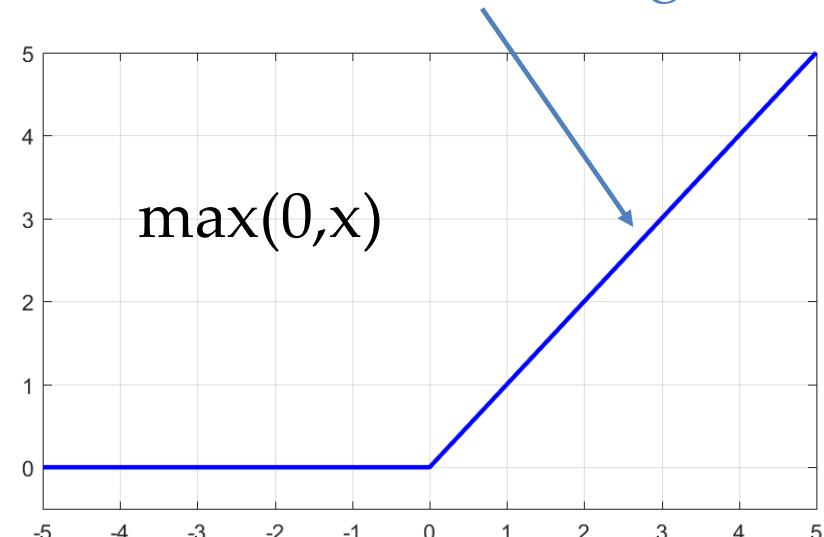
- LSTM : Long short-term memory



# ReLU

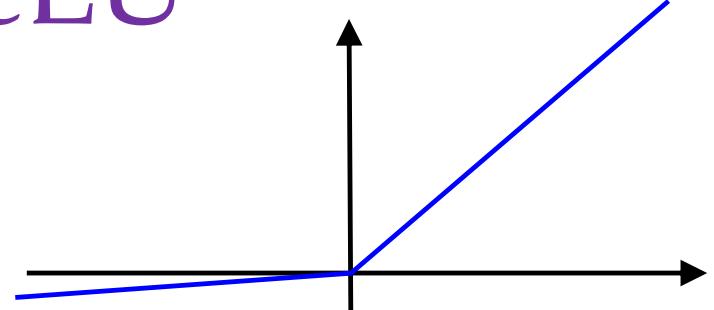
Quasi-linéarité : rend l'entraînement plus facile via descente de gradient

- La plus populaire comme non-linéarité
- Toujours non-négatif
  - moyenne sortie biaisée +
- Pas de limite supérieure
- Facile à calculer (**exp** est plus long)
- Accélère l'entraînement des réseaux (facteur 6 pour AlexNet).
- Résulte en des activations parcimonieuses (certains neurones sont à 0)
  - Parfois des neurones vont mourir, particulièrement si learning rate est trop grand ☹
- Rarement utilisé en sortie du réseau



# Leaky ReLU

- Gradient = 0 signifie impossibilité d'entraîner
- Pente très légère dans la partie négative : leaky ReLU
- Si un paramètre  $\alpha$  (entraînable) par neurone/couche, on obtient la PReLU [1]
- Donne des distributions de sorties plus centrées à 0 que ReLU

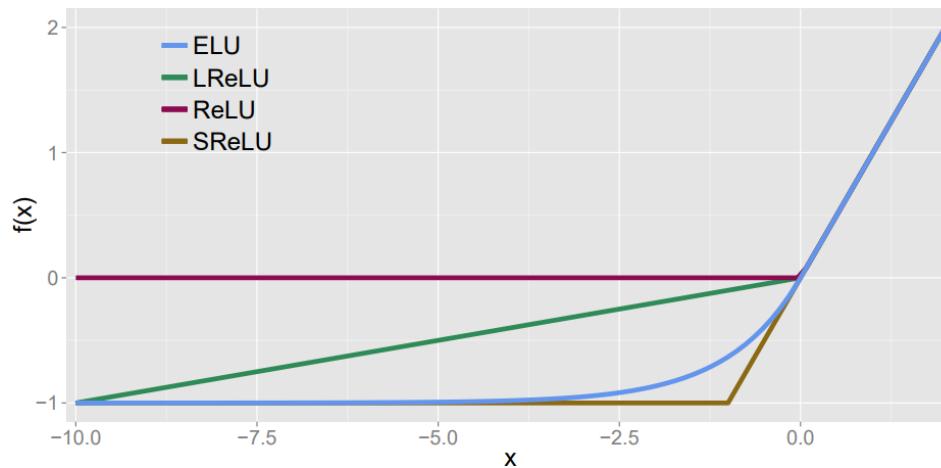


$$f(x) = \begin{cases} \alpha x & \text{si } x < 0 \\ x & \text{autrement} \end{cases}$$

[1] He et al. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification, ICCV 2015.

# Autres activations

- Maxout [1] :  $\max(w_1^T x + b_1, w_2^T x + b_2)$
- ELU [2]



- Parametric ELU [3]

[1] Goodfellow et al., Maxout Network, ICML 2013.

[2] Clevert et al., Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs), ICLR 2016.

[3] Trottier et al., Parametric exponential linear unit for deep convolutional neural networks, ICMLA 2017.

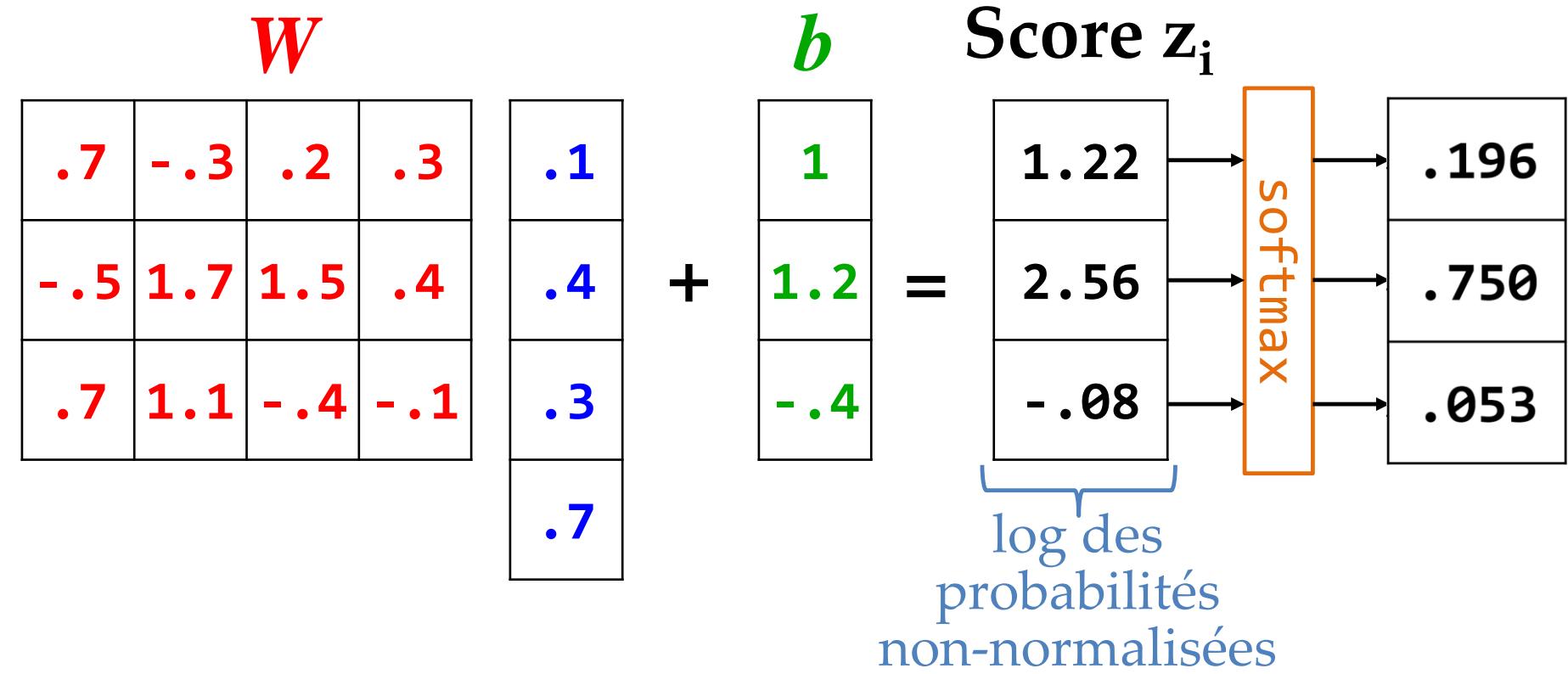
# Softmax

- Utilisé en sortie, prédiction multi-classe
- Version continue, *douce*, de  $\max([\dots])$

$$\hat{y}_i = \frac{\exp(z_i)}{\sum_{j \in \text{groupe}} \exp(z_j)}$$

- Va dépendre de l'ensemble des sorties du groupe
- Sortie somme à 1, chaque sortie entre 0 et 1 :
  - distribution de probabilité multinouilli
- Manière d'indiquer au réseau de chercher l'appartenance exclusive à une seule classe

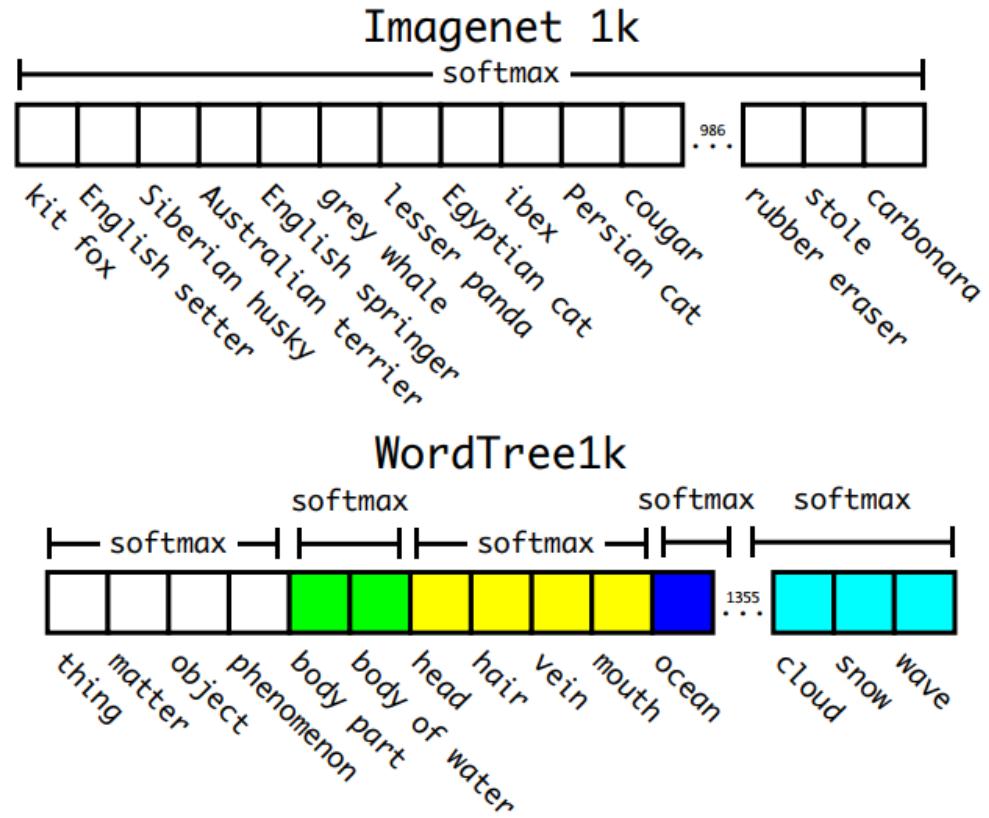
# Softmax



$$P(\text{classe} | x) = \frac{\exp(z_i)}{\sum_{j \in \text{groupe}} \exp(z_j)}$$

# Softmax

- Peut avoir plus d'un softmax en sortie
- Combiner des jeux de données avec différentes granularités de classes
  - chien vs labrador



**Figure 5: Prediction on ImageNet vs WordTree.** Most ImageNet models use one large softmax to predict a probability distribution. Using WordTree we perform multiple softmax operations over co-hyponyms.

# Softmax avec température $T$

$$\hat{y}_i = \frac{\exp(z_i / T)}{\sum_j \exp(z_j / T)}$$

- Si  $T$  est élevé, les sorties seront plus égalitaires. Si  $T$  faible, winner-takes-all
- Utilisé dans :
  - LSTM pour varier la confiance/répartition des sorties
  - distillation des réseaux [1]

[1] Hinton et al. Distilling the knowledge in a neural network. arXiv preprint arXiv:1503.02531 (2015).