UNIVERSITÉ LAVAL

GLO-4030/7030
APPRENTISSAGE PAR RÉSEAUX DE NEURONES PROFONDS

Plan de cours

Introduction

Fonctions d'activation

Philippe Giguère

Plan de cours

Bienvenue à bord!

- 1ere édition
- L'équipe :

Mathieu Carpentier

Alexandre Gariépy

Laboratoires

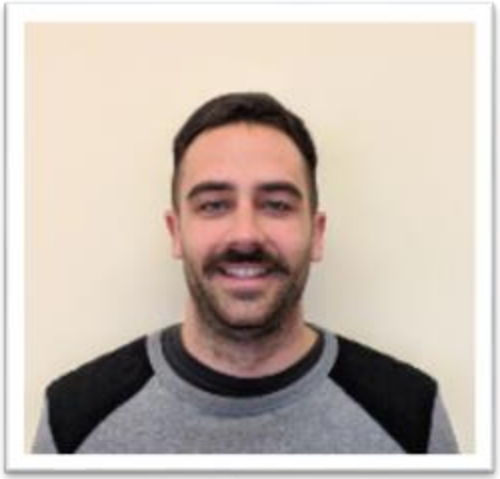
Ludovic Trottier

Semaine #3
Optimisation

Cumul de 5 ans+ de Deep Learning

Bienvenue à bord!

- 2e édition
- L'équipe :



Nicolas
Garneau

Ressources GPU

- Accès à des GPU via Jupyter notebook
  - NVIDIA K20
- https://jupyter.calculquebec.ca/
- Cœurs réservés pour les laboratoires : réservation `glo4030`
- Gros merci à
  - Florent Parent
  - Félix-Antoine Fortin

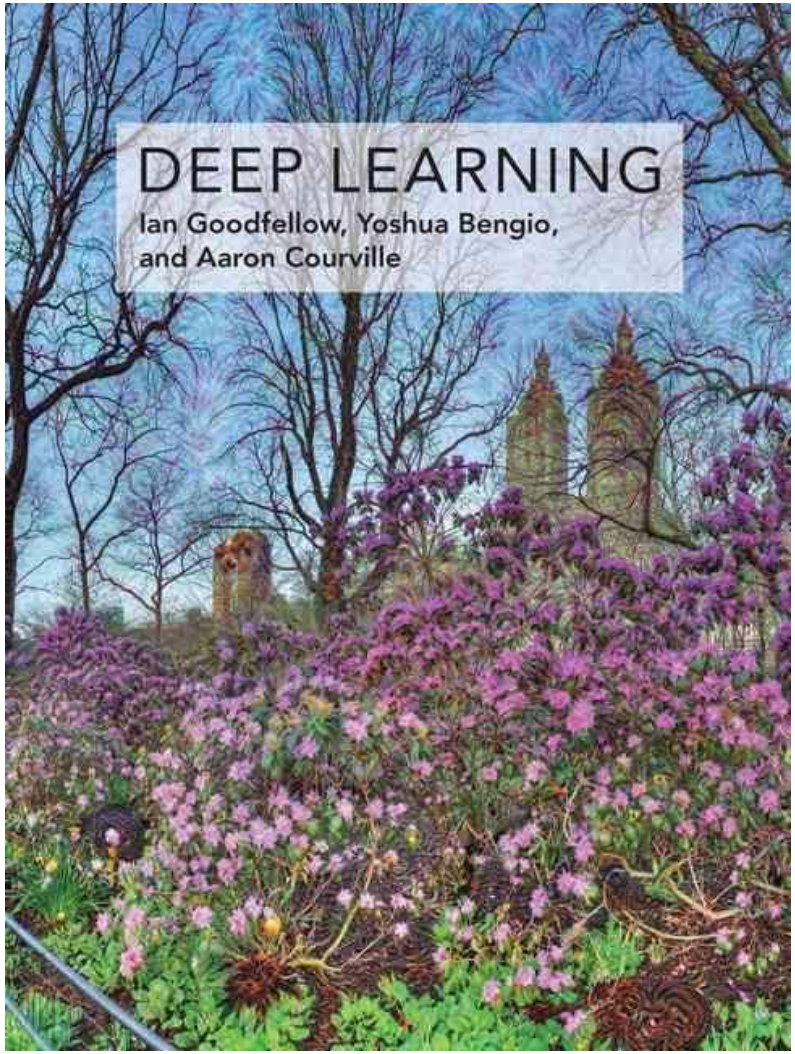
Ressources GPU

- Google Cloud Platform
  - Envoyez-moi un courriel pour obtenir un crédit
- Amazon Web Services
  - Crédits étudiants + Github education

Manuel (obligatoire)

- Bonne référence
- Version html gratuite

`http://www.deeplearningbook.org/`



Pré-requis

- Python (laboratoire, TP)
- Certaine connaissance du machine learning
- 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
    - 30+ 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 1ère moitié

- Connaissances de base (vertical)
  - Introduction, neurone, fondement apprentissage machine, 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 2ème 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
  - Et +

Présentations affiches des étudiants 2e et 3e cycle

Présentations affiches GLO-7030

- Sur votre travail de session
- Présentation style "poster session" en conférence
- Compte pour **12 %**

Librairie utilisée : PYTORCH

- 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

`http://pytorch.org/about/`

Examen

- Mi-Session
  - **35%** pour GLO-4030
  - **33%** pour GLO-7030

- Final
  - GLO-4030 seulement
  - Examen de 2 heures, **20 %**

- Pas de documents permis

Travaux pratiques

- 2 travaux
- Dans la première moitié du cours
- Total de **20 %**
- En python ou `PyTorch`

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 libraire de votre choix (TensorFlow, PyTorch, etc)
- Rapport sous format d'article scientifique

Sites web

- Site du cours :

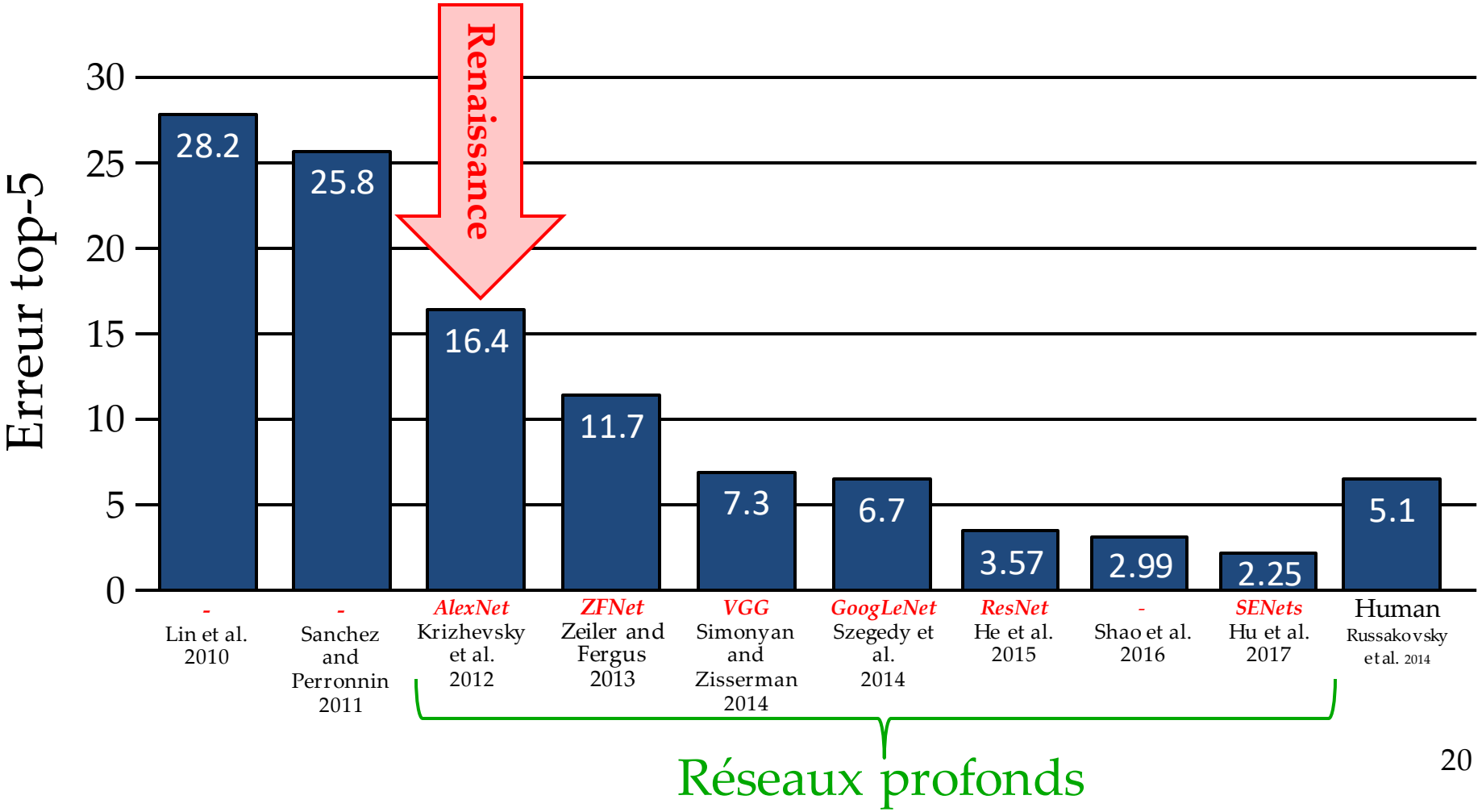
https://ulaval-damas.github.io/glo4030/

- Site pour le forum :
  - Forum MonPortail

Introduction

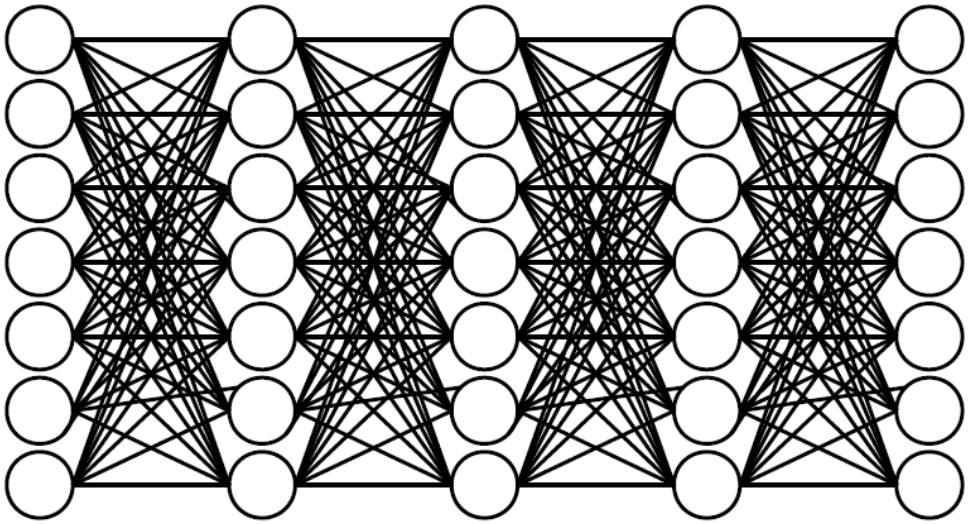
Large Scale Visual Recognition Challenge

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

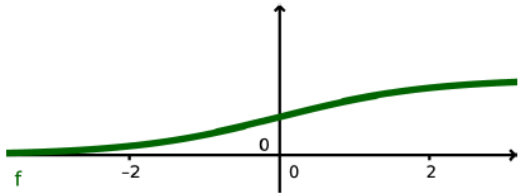


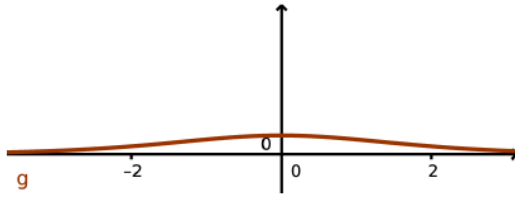
Causes de la renaissance #1
Nouvelles fonctions d'activations

pré-2006



Fonction logistique





Sa dérivé

∇_θ = 1e-15 ← ∇_θ = 1e-7 ← ∇_θ = 1e-3 ← ∇_θ = 0.1 ← ∇_θ = 10

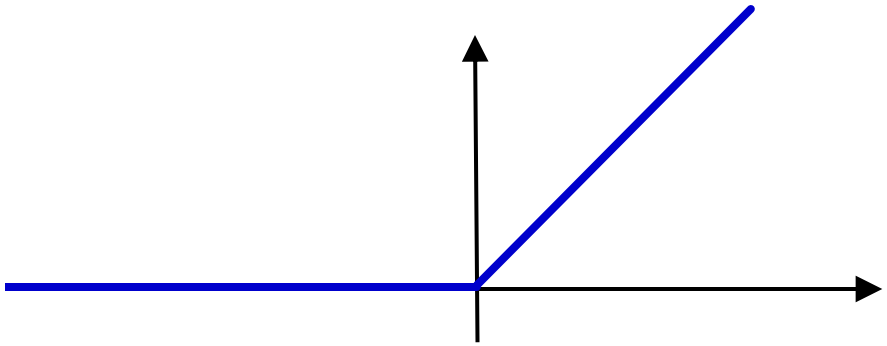
Vanishing gradient

Crédit : L. Trottier

Causes de la renaissance #1
Nouvelles fonctions d'activations

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

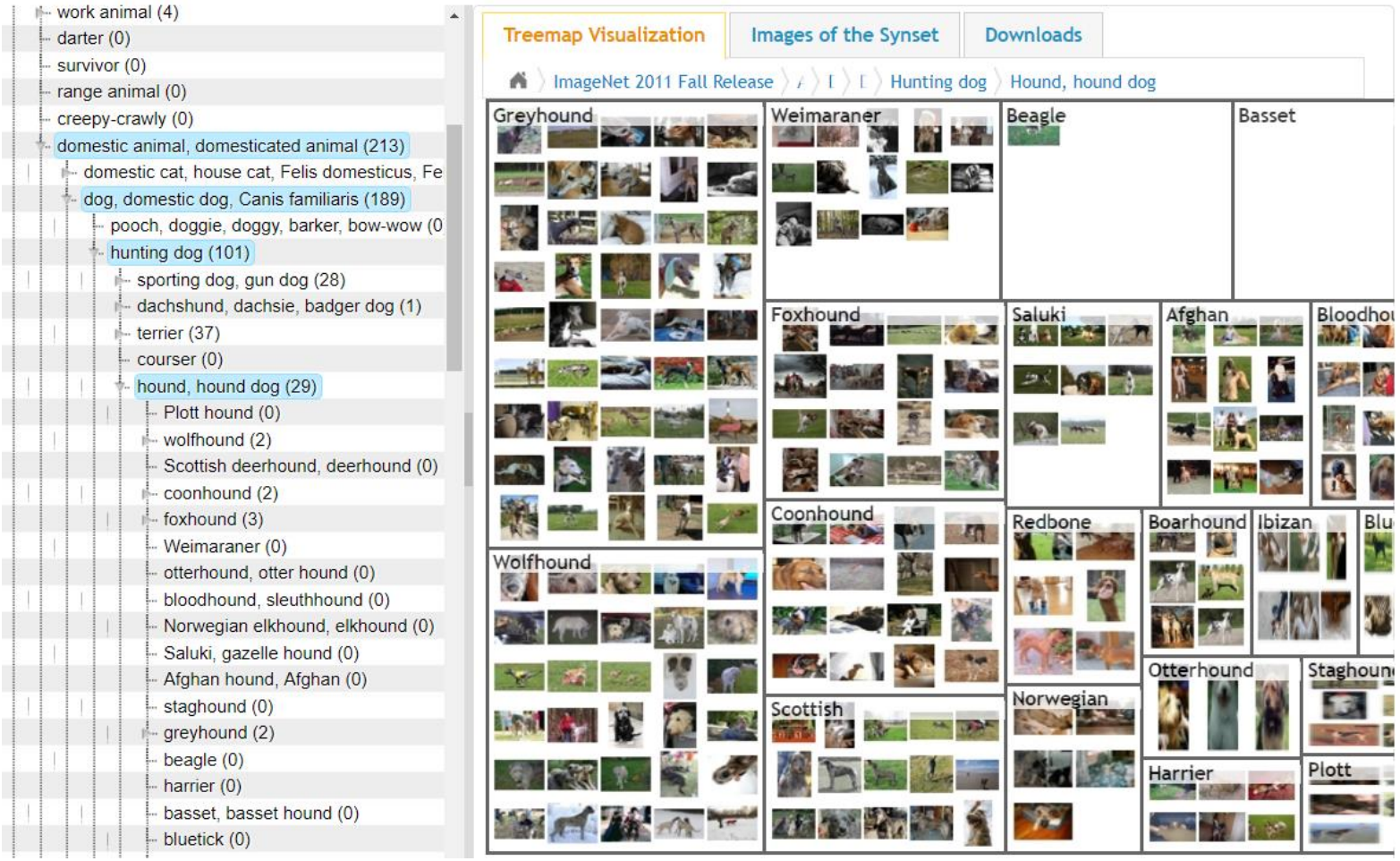
$$f(x) = \begin{cases} x & \text{si } x \geq 0 \\ 0 & \text{sinon} \end{cases}$$



- 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`
- 14 millions images, 22 000 catégories



Causes de la renaissance #3

Puissance de calcul via GPU

Août 2016

1,800 CAD

Novembre 2001

Rank	System	Cores	Rmax (GFlop/s)	Rpeak (GFlop/s)
1	ASCI White, SP Power3 375 MHz , IBM Lawrence Livermore National Laboratory United States	8,192	7,226.0	12,288.0
2	AlphaServer SC45, 1 GHz , HPE Pittsburgh Supercomputing Center United States	3,024	4,059.0	6,048.0
3	SP Power3 375 MHz 16 way , IBM DOE/SC/LBNL/NERSC United States	3,328	3,052.0	4,992.0
4	ASCI Red , Intel Sandia National Laboratories United States	9,632	2,379.0	3,207.0

12,000

Causes de la renaissance #3

Puissance de calcul via GPU

TOP 500 The List.

Juin 2005

Décembre 2017

2,995 USD

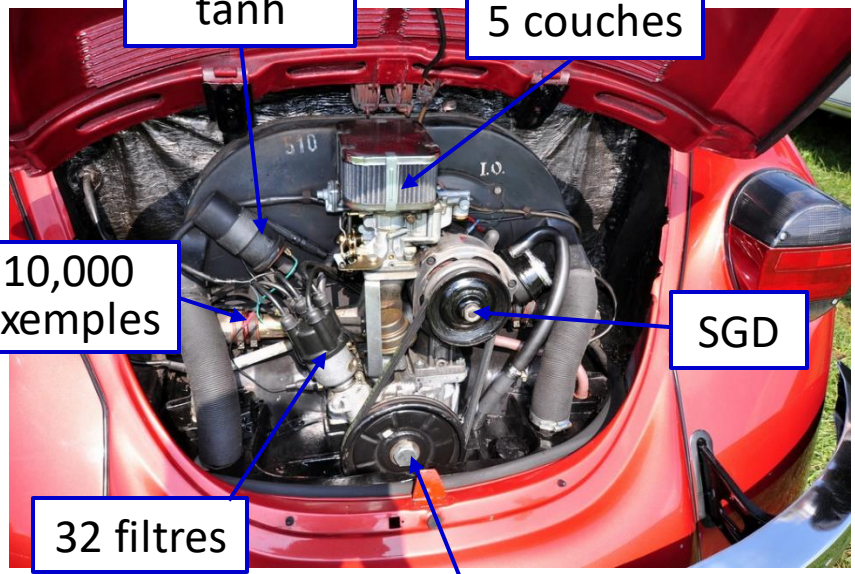
Rank	System	Cores	Rmax (TFlop/s)	Rpeak (TFlop/s)	Power (kW)
1	**BlueGene/L** - eServer Blue Gene Solution , IBM DOE/NNSA/LLNL United States	65,536	136.8	183.5	716
2	**BGW** - eServer Blue Gene Solution , IBM IBM Thomas J. Watson Research Center United States	40,960	91.3	114.7	448
3	**Columbia** - SGI 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

110

Continuation de la renaissance

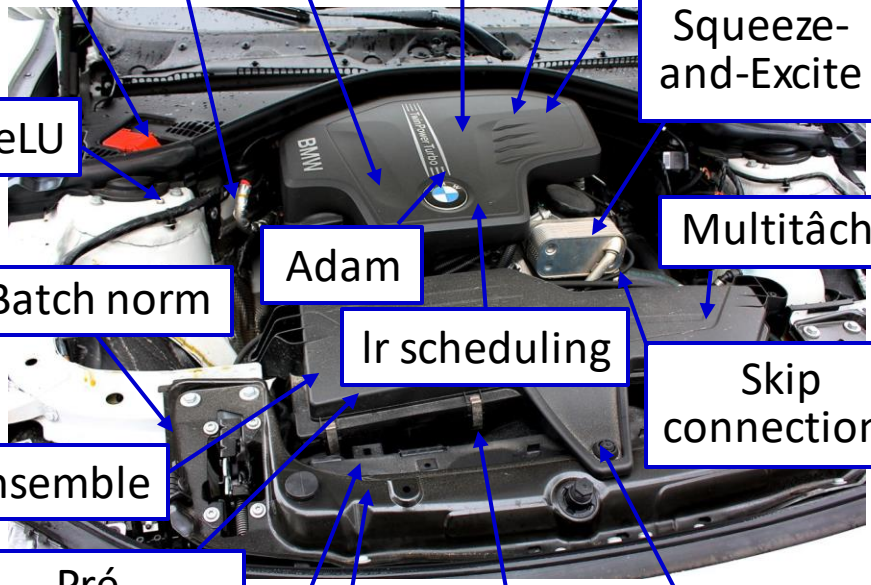
- Progrès très rapide avec arXiv.org
- Plus d'une dizaine de soumission par jour
- L'article est souvent périmé lorsque présenté en conférence

Évolution des réseaux



tanh
5 couches
10,000 exemples
SGD
32 filtres
Average pooling
ca. 1990

Crédit photo : Griffin's auto repair



Global average pooling
Convolutions à trous
Early stopping
1000 couches
Max pooling
Drop-out
Squeeze-and-Excite
ReLU
Multitâche
Adam
Batch norm
lr scheduling
Skip connections
Ensemble
Pré-entraînement ImageNet
Xavier Initialization
Data augmentation
geler des couches
3,000+ filtres
ca. 2017

Crédit photo : wildsau.ca

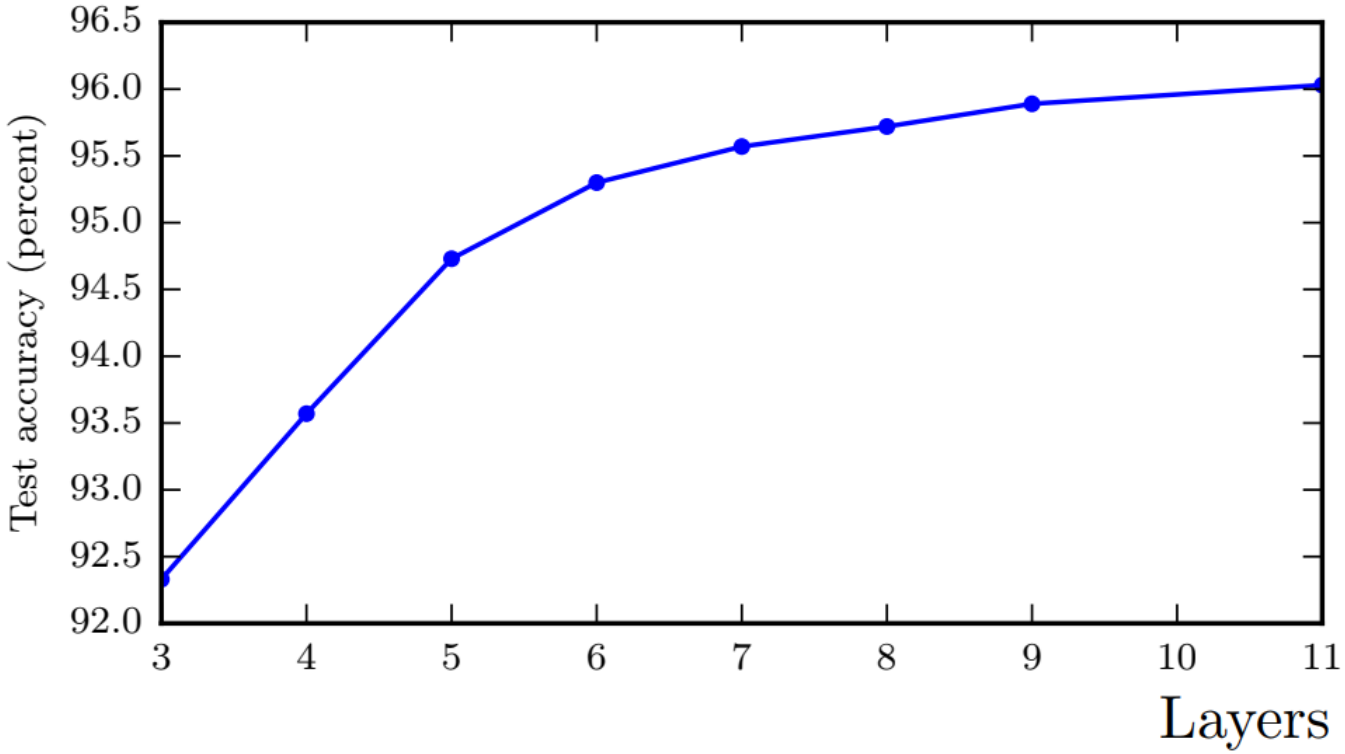
Pourquoi le Deep?

- 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

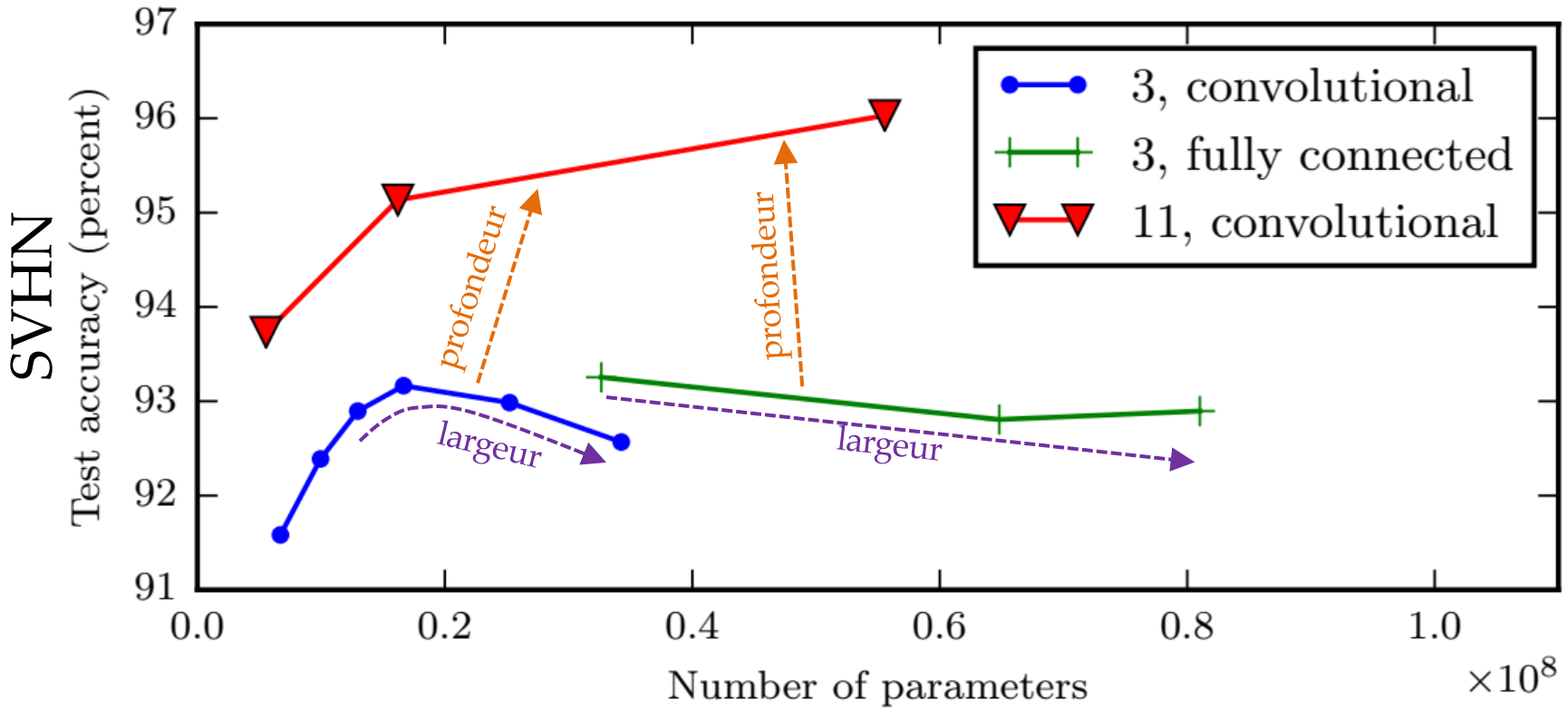
- Street View Home Numbers SVHN



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

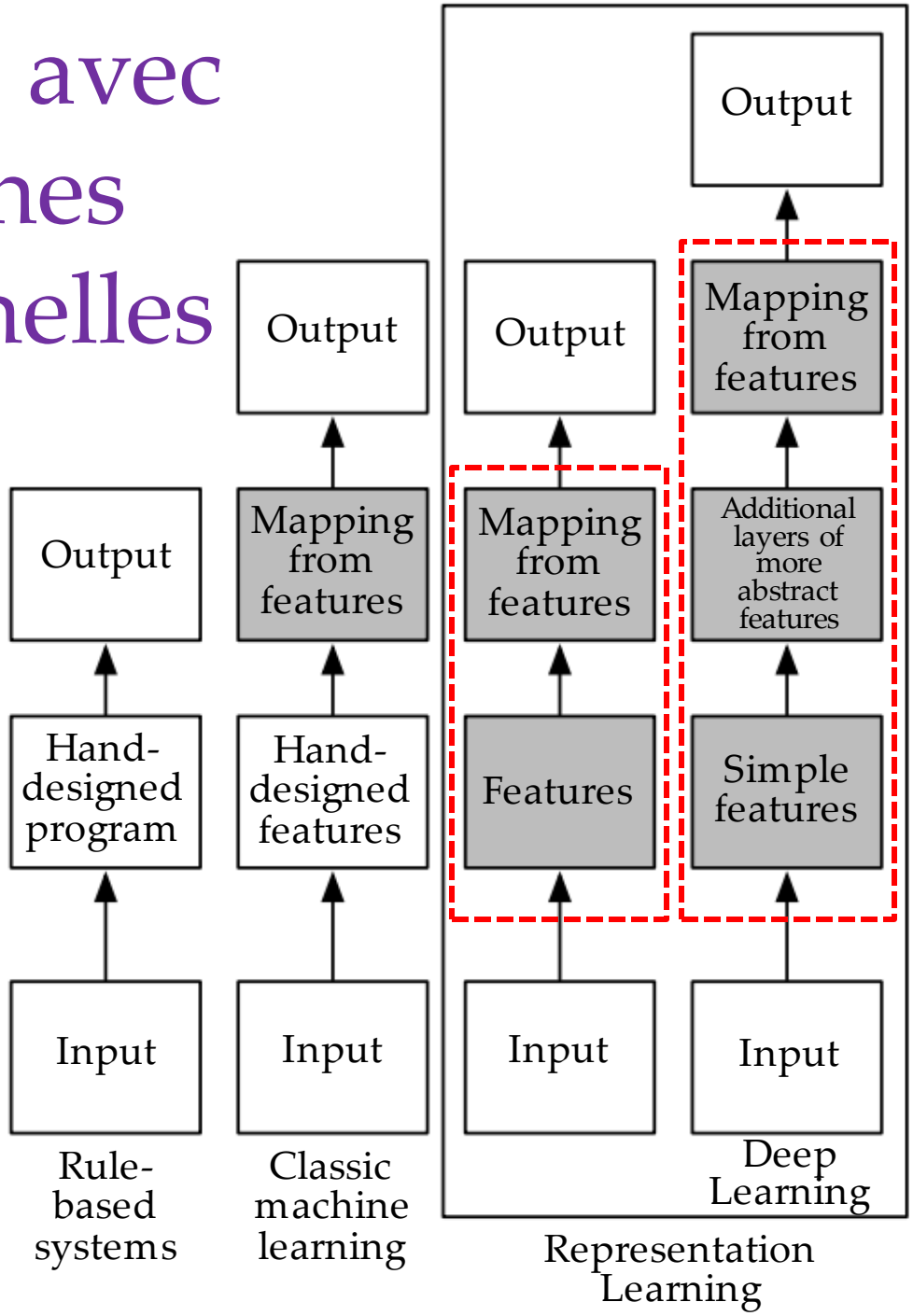
Meilleure généralisation

- Modèles larges ou peu profonds ont tendance à overfitter



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

Contraste avec approches traditionnelles



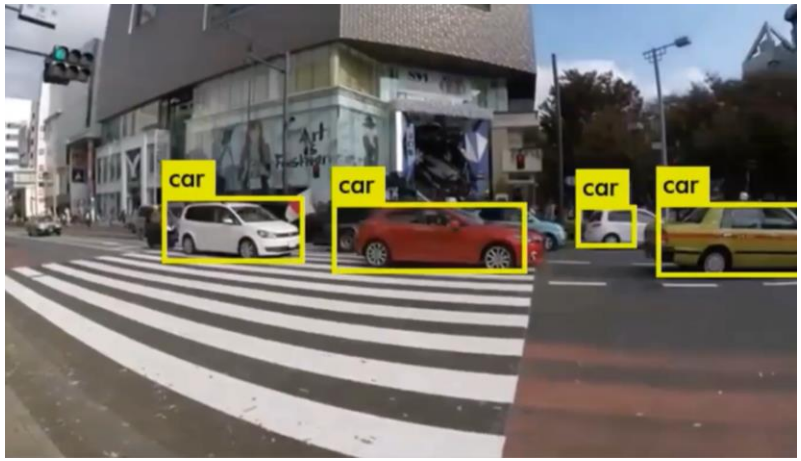
Appris conjointement

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'entrainement
  - 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 (et ML) : *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/

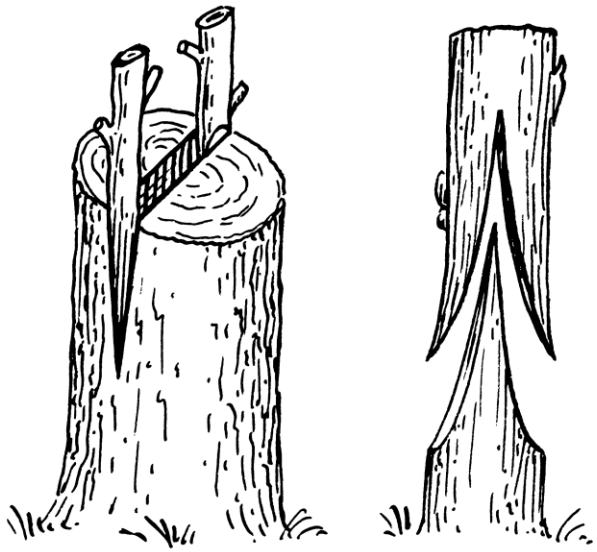
Transfert d'innovation

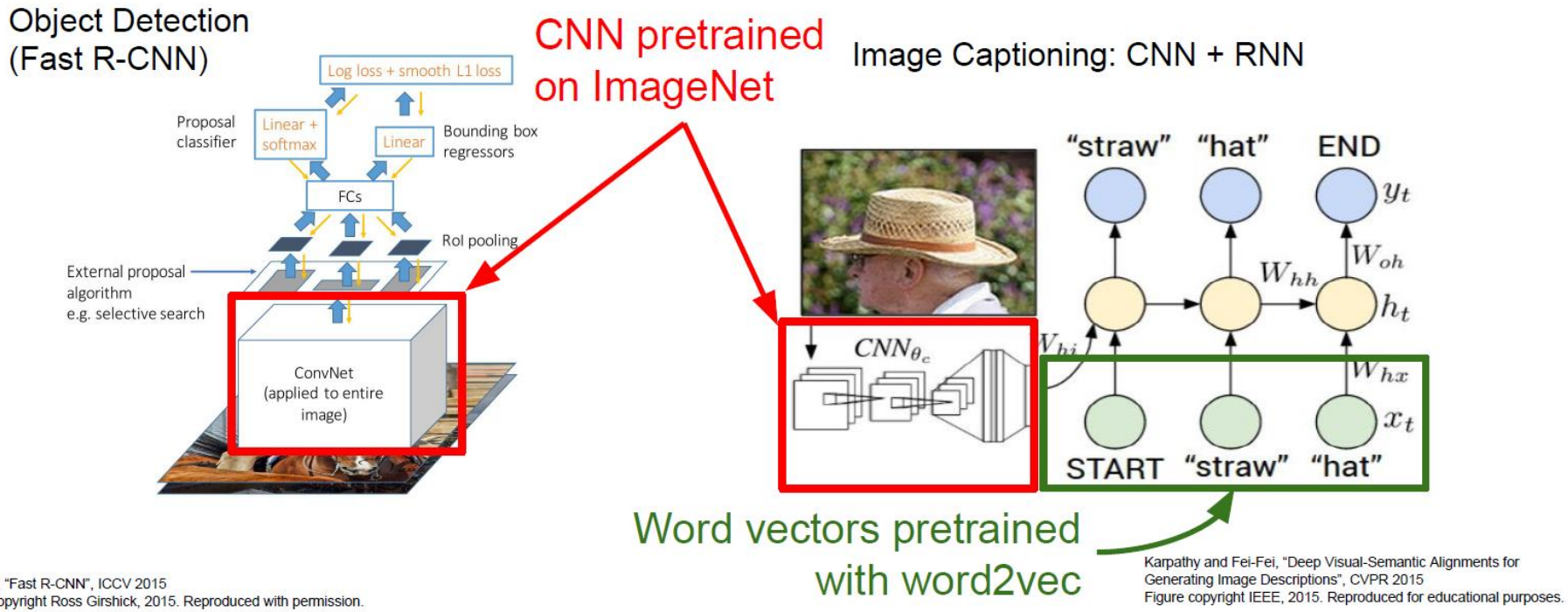
- Les innovations dans les architectures de réseaux faits pour une tâche X ont tendance à aussi aider pour de nombreuses autres tâches
- Synergies et emballements

Szegedy et al., Rethinking the Inception Architecture for Computer Vision, CPVR 2015.

Bouturage

- Combiner des réseaux pré-entraînés sur des tâches différentes



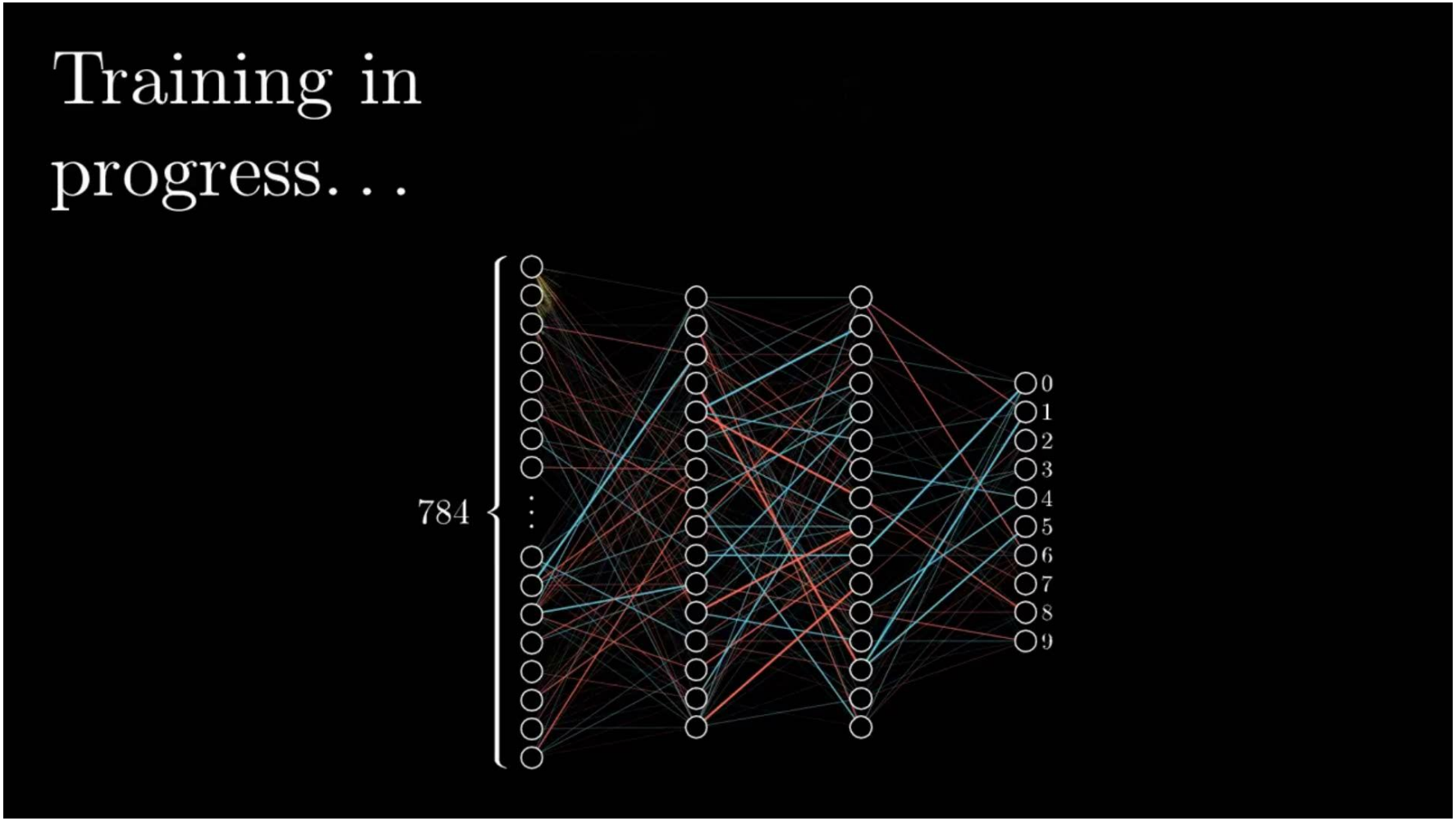


Girshick, "Fast R-CNN", ICCV 2015
Figure copyright Ross Girshick, 2015. Reproduced with permission.

Karpathy and Fei-Fei, "Deep Visual-Semantic Alignments for Generating Image Descriptions", CVPR 2015
Figure copyright IEEE, 2015. Reproduced for educational purposes.

Le gradient = sève

- Pas de gradient = pas d'apprentissage

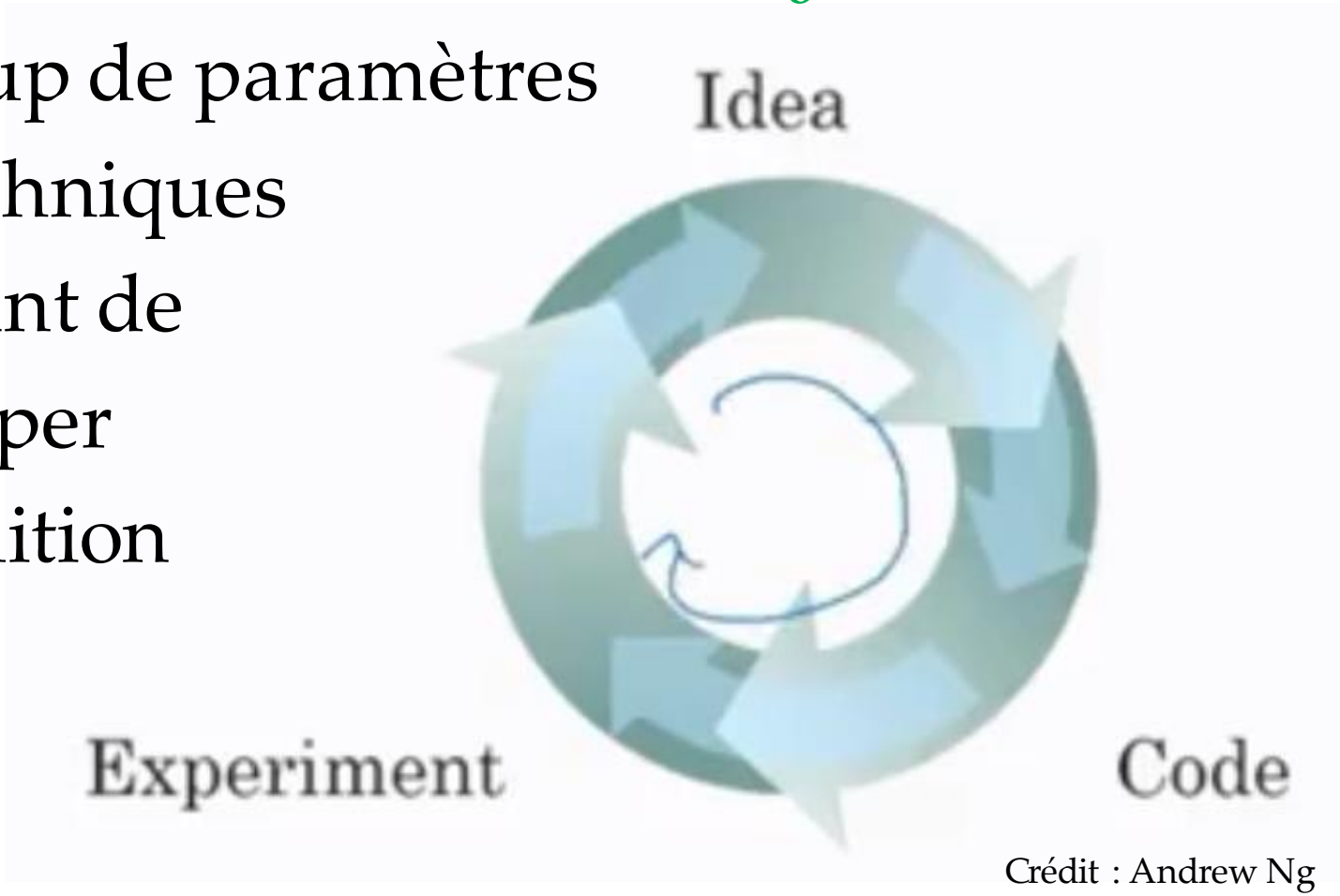


Extrait de https://youtu.be/IHZwWFHWa-w 3Blue1Brown

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

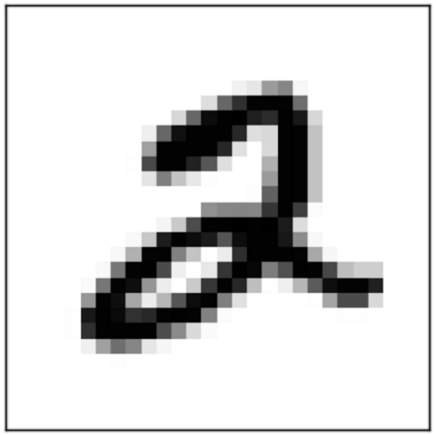
Deep Learning : appel à la prudence

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

Exemples d’applications

Reconnaissance de caractères

28 x 28

$f($  $;\theta)$ → vecteur de 10 scores, un par classe

MNIST

θ : paramètres de la fonction

(Labo 1)

Reconnaissance d'images

224 x 224 x 3

$f($  $;\theta)$ → vecteur de 1000 scores, un par classe

ImageNet

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

θ : paramètres de la fonction

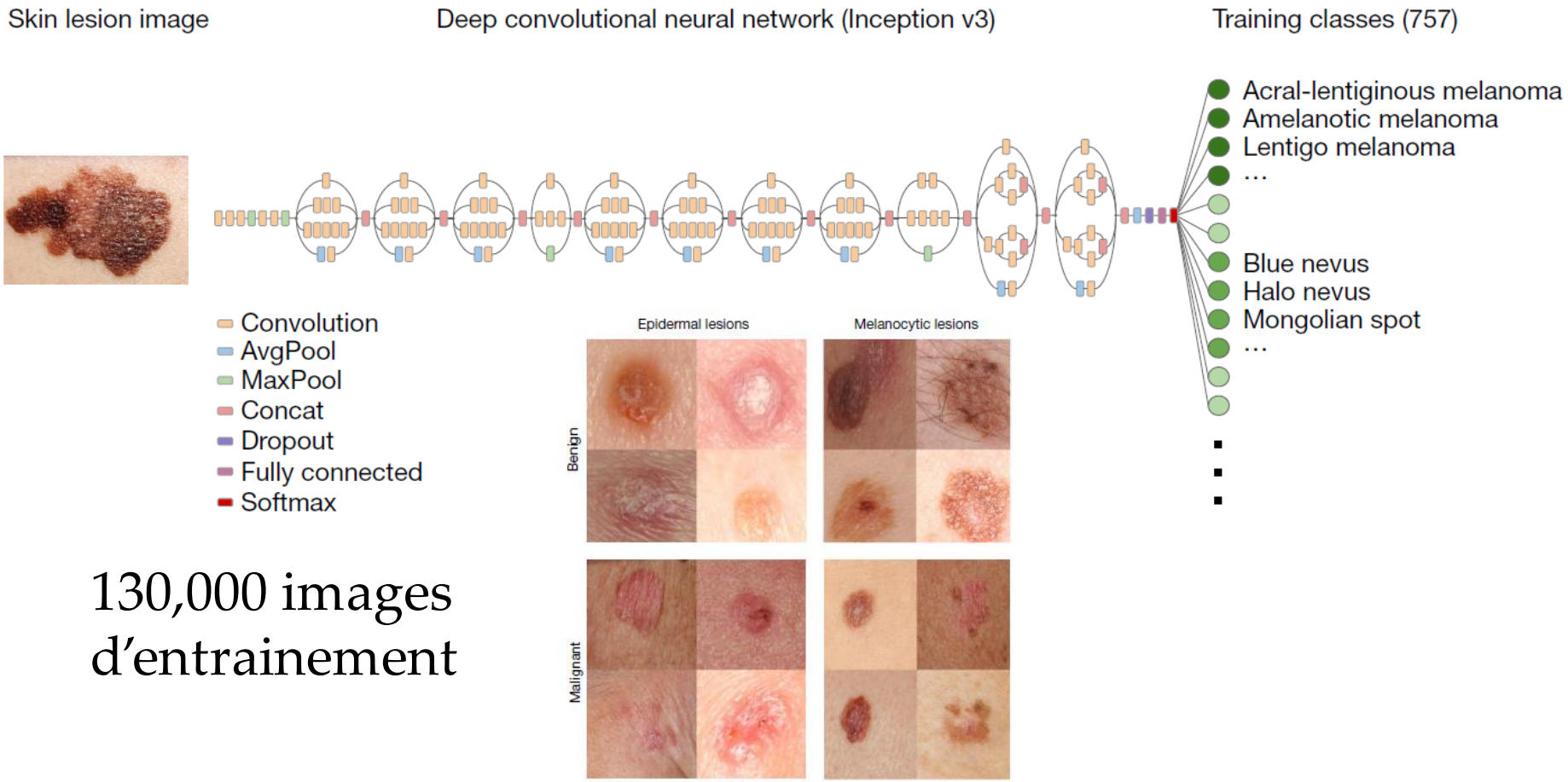
LETTER

Nature, Février 2017

doi:10.1038/nature21056

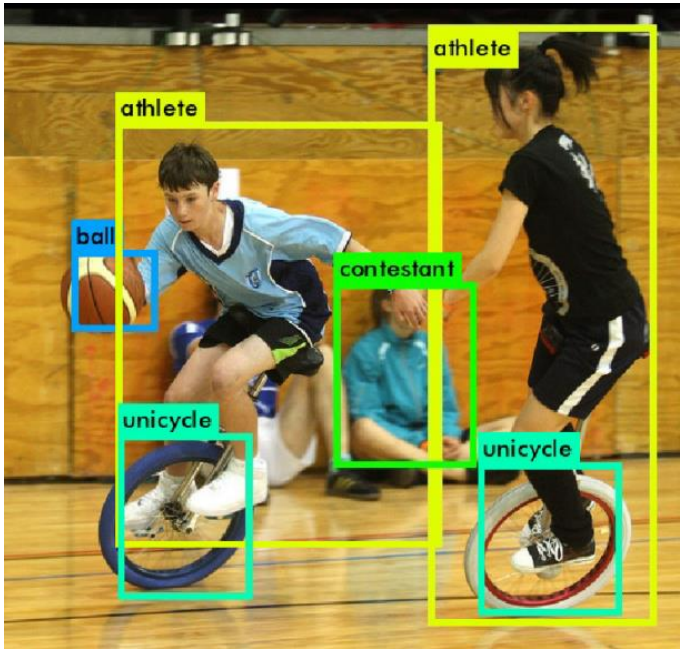
Dermatologist-level classification of skin cancer with deep neural networks

Andre Esteva[1]*, Brett Kuprel[1]*, Roberto A. Novoa[2,3], Justin Ko[2], Susan M. Swetter[2,4], Helen M. Blau[5] & Sebastian Thrun[6]



130,000 images d'entrainement

Détection d'objets

$f($  $;\theta) \rightarrow$ 

Redmon and Farhadi, YOLO9000: Better, Faster, Stronger, CVPR 2017.

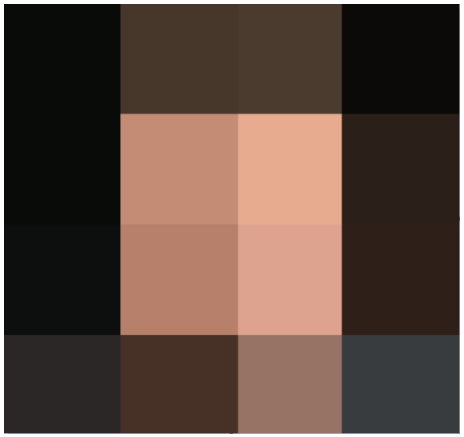
Description d'images

$f($  $;\theta)$ → Construction worker in orange safety vest is working on road

Karpathy and Fei-Fei, Deep Visual-Semantic Alignments for Generating Image Descriptions, CVPR 2015.

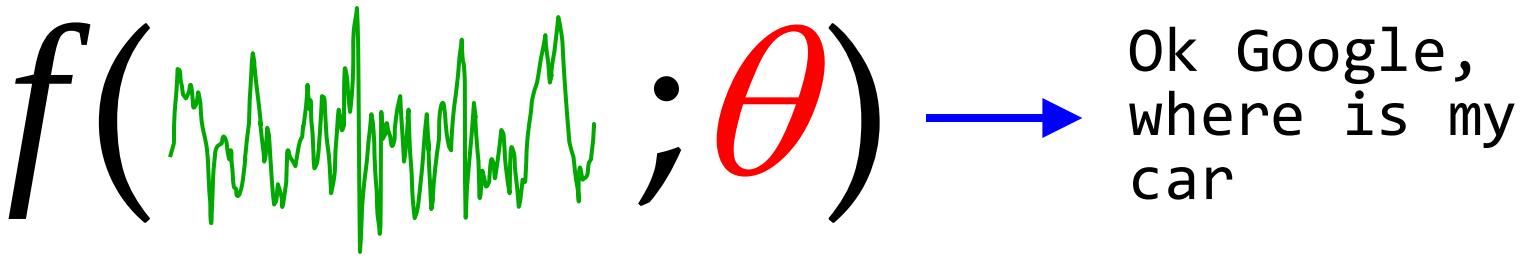
Qui est cet acteur?

Génération d'images

$f($  $;\theta) \rightarrow$ 

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

Reconnaissance voix



Génération de voix : *Tacotron 2*

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

Note : exemple d'entraînement : 

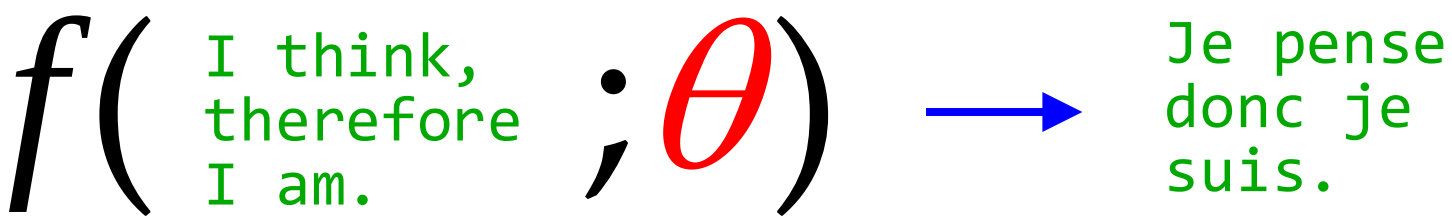
« George Washington was the first President of the United States »

A: 

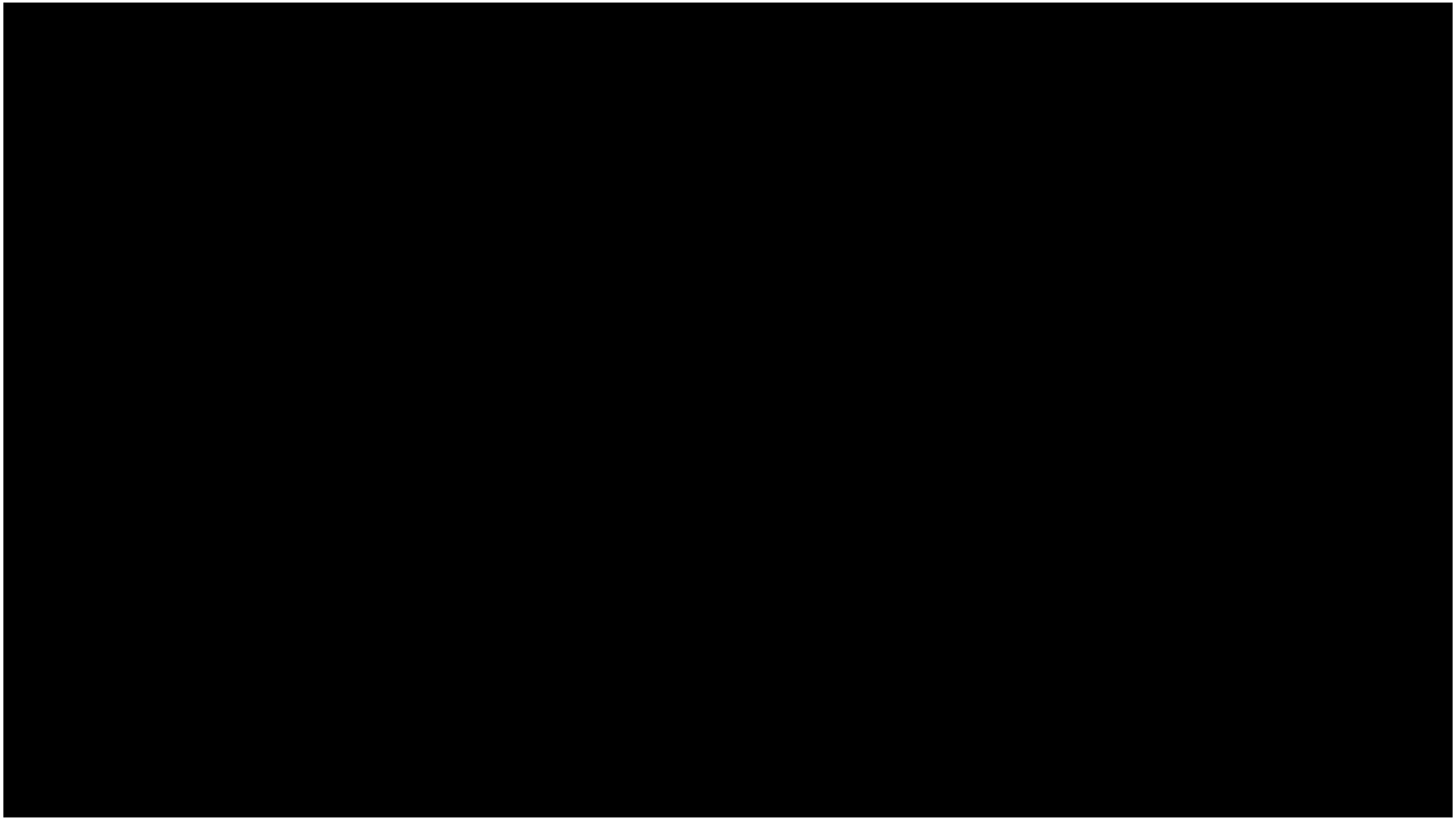
B: 

Natural TTS Synthesis by Conditioning WaveNet on Mel Spectrogram Predictions
Jonathan Shen, arXiv:1712.05884, dec 2017.

Traduction automatique



Apprentissage visuomoteur



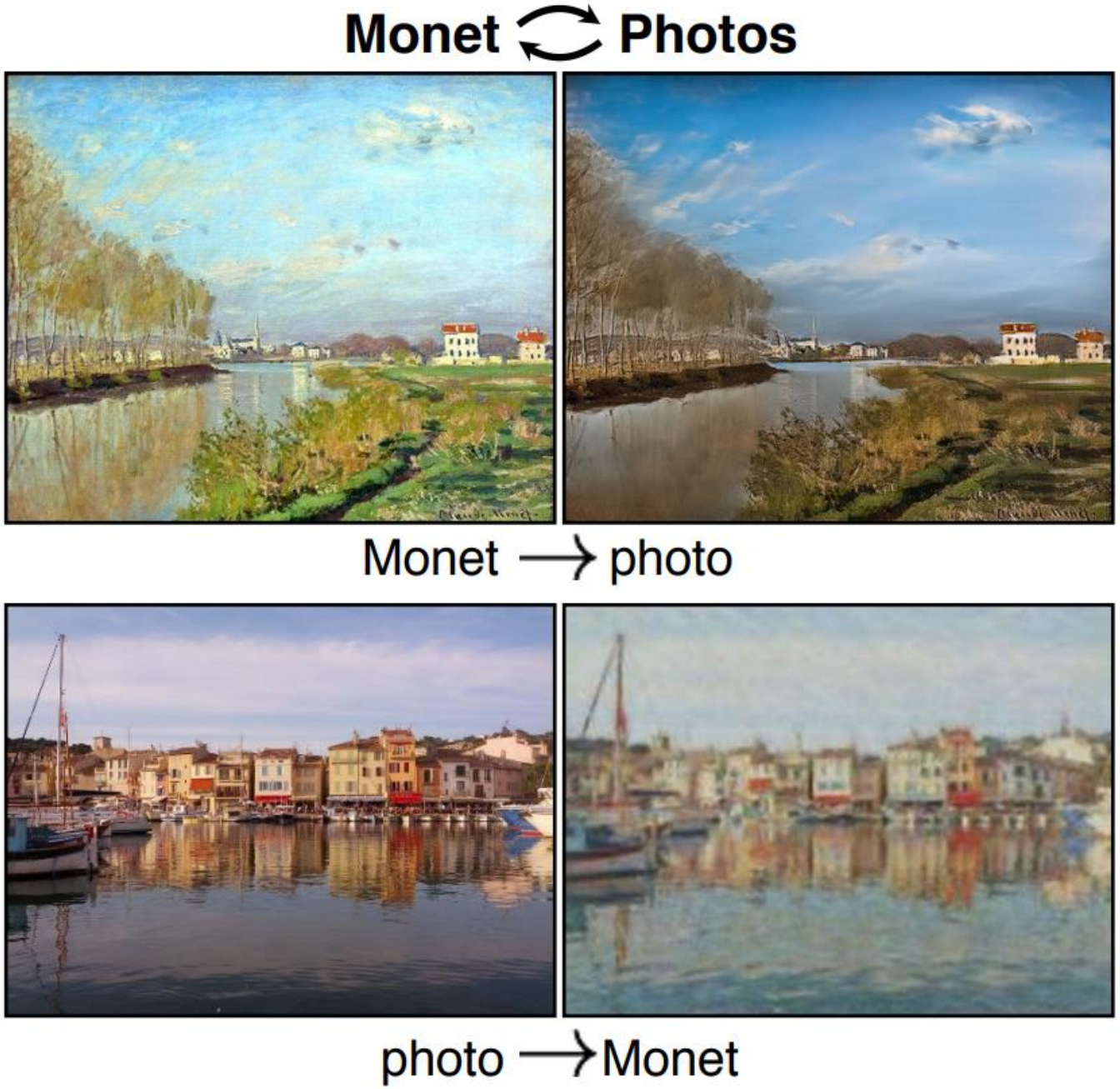
Transfert de style

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

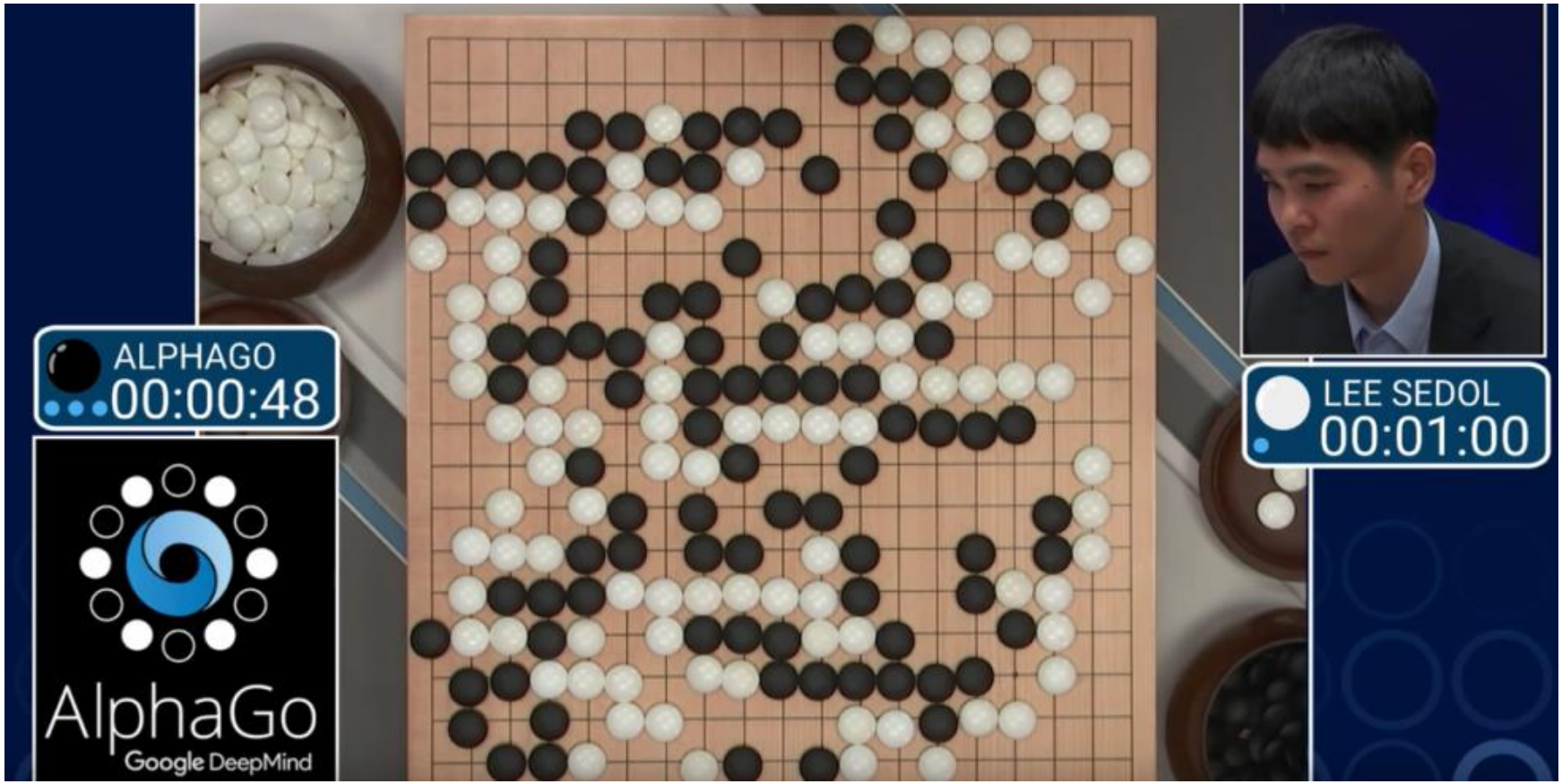
$F($ $;\theta) \rightarrow$ 

Zhu et al., Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks, CVPR 2017.

Transfert de style



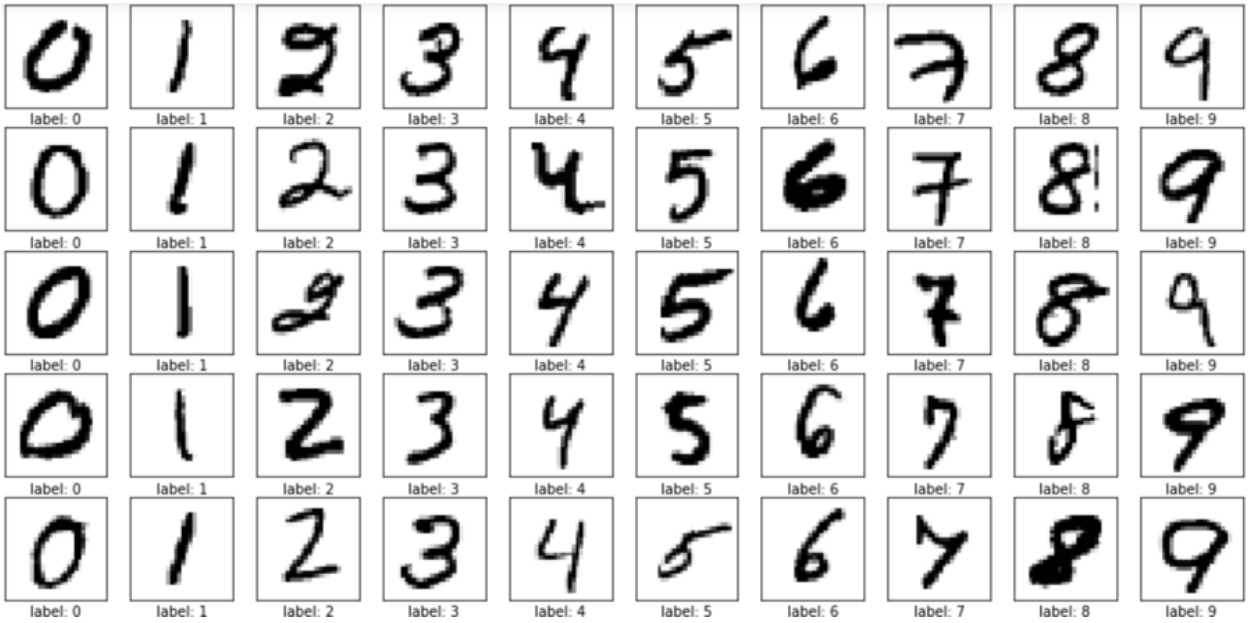
Jeu



Réseaux classiques peu profonds : classification d'images

MNIST

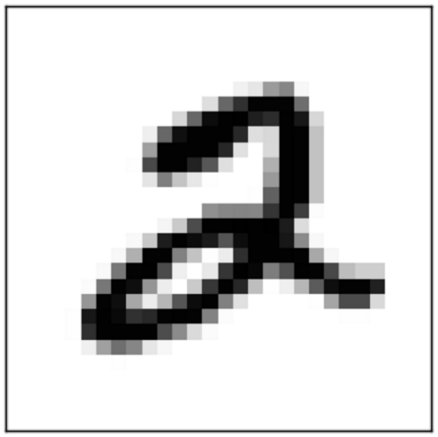
- 10 chiffres
- 70,000 images de 28 × 28 pixels



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

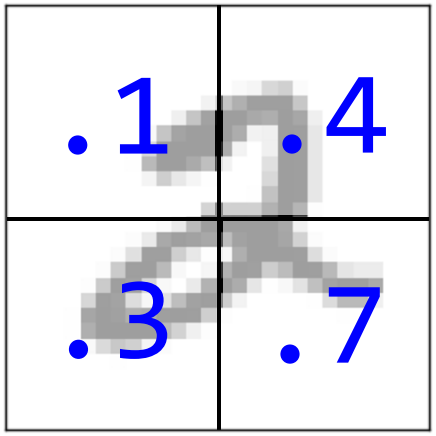
Classificateur linéaire sur MNIST

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

$f($  $; W, b) \rightarrow$ vecteur de 10 scores, un par classe

MNIST

Exemple simplifié



« Flatten »

$$f = Wx+b$$

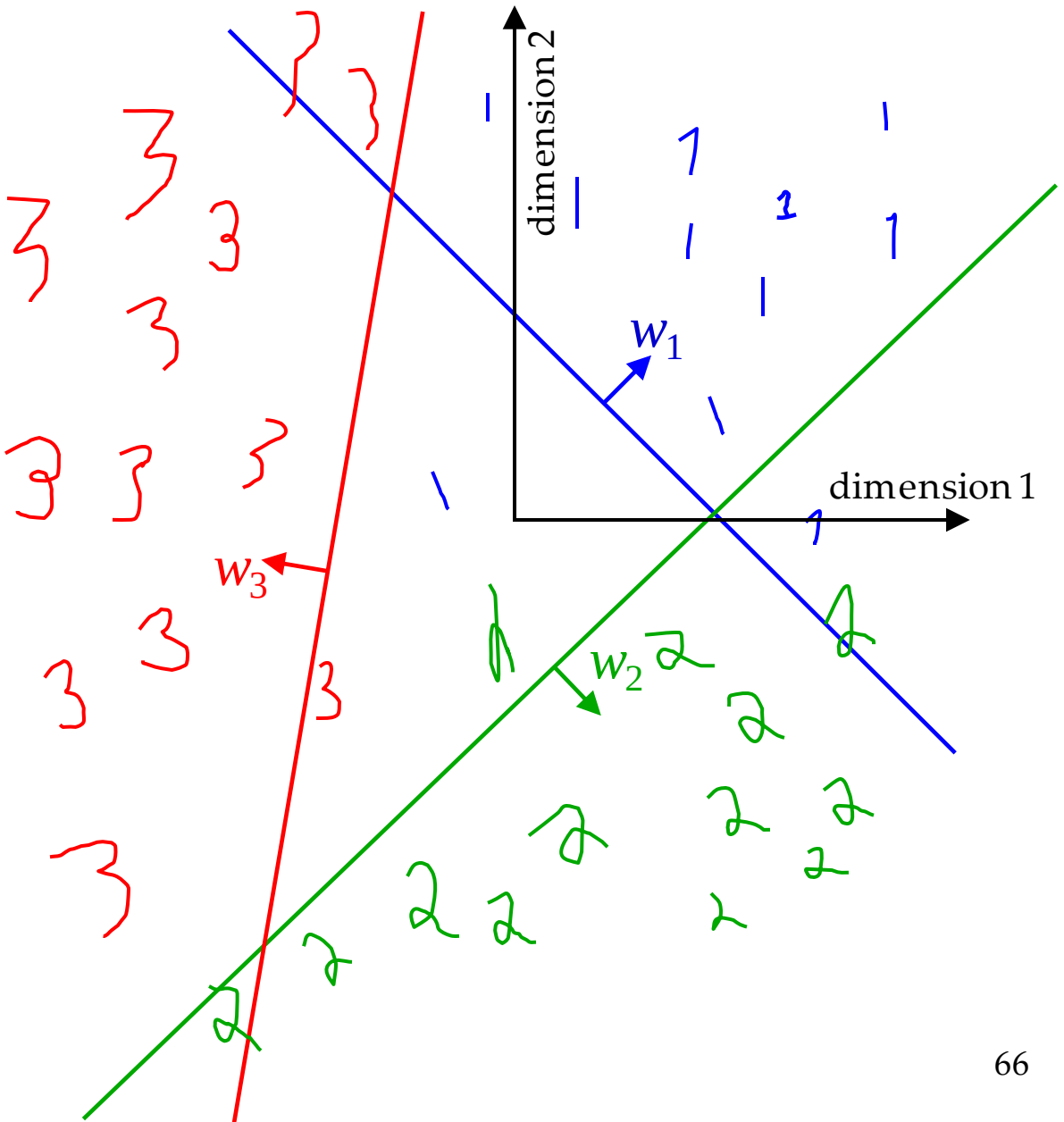
W				x		b		Score	
.7	-.3	.2	.3	.1		1		1.22	Chiffre 1
-.5	1.7	1.5	.4	.4	+	1.2	=	2.56	Chiffre 2
.7	1.1	-.4	-.1	.3		-.4		-.08	Chiffre 3
				.7					

adapté de cs231n

Interprétation classificateur linéaire

$$f = \boldsymbol{W}x+b$$

w_1	.7	-.3	.2	.3
w_2	-.5	1.7	1.5	.4
w_3	.7	1.1	-.4	-.1



Réseau 1 couche linéaire

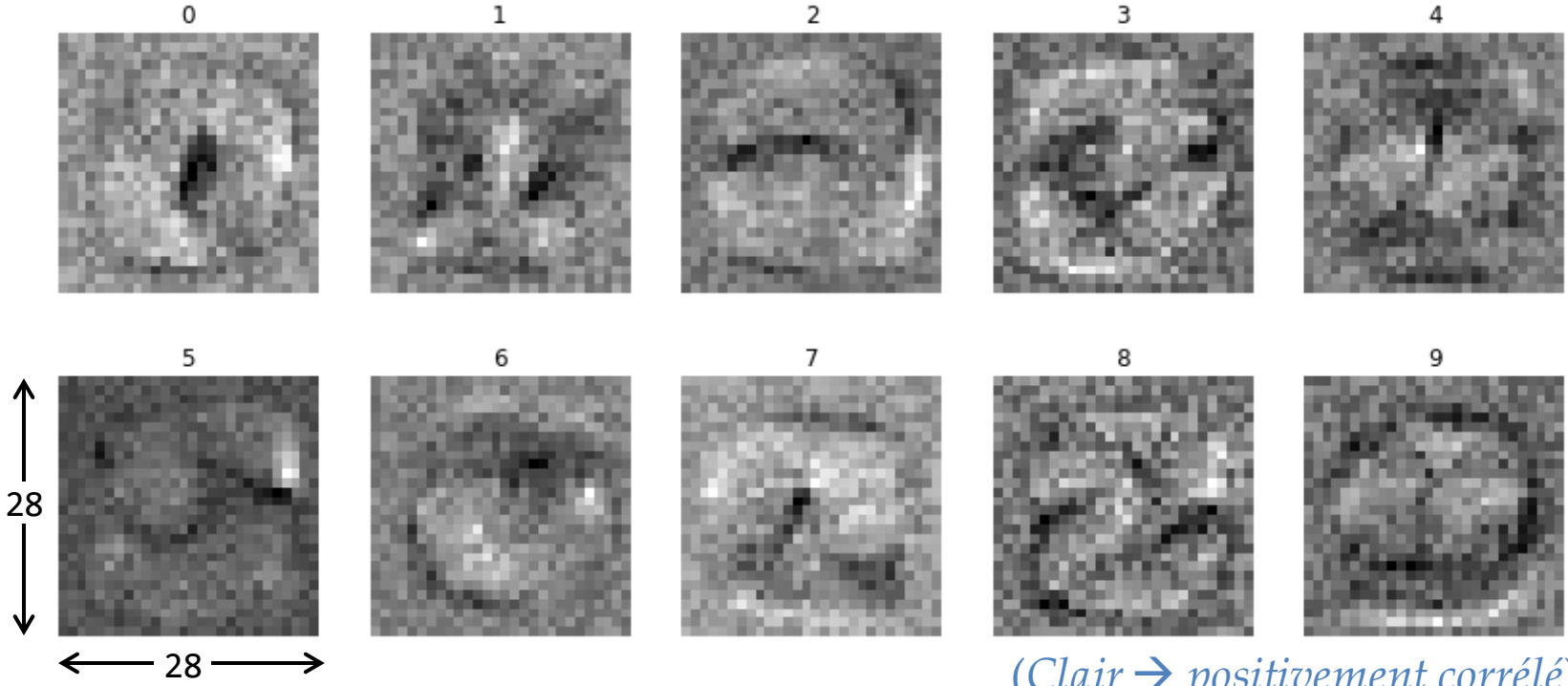
- Image 28x28 → 784x1
- Matrice ***W*** 10x784
- Biais *b* 10x1
- Initialisé au hasard
- Entrainement SGD sur perte *multiclass hinge loss*
- Train set : 60,000 exemples
- Test set : 10,000 exemples
- Résultat ~92% précision sur test

```
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
```

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

Poids *W* appris : patrons 28×28



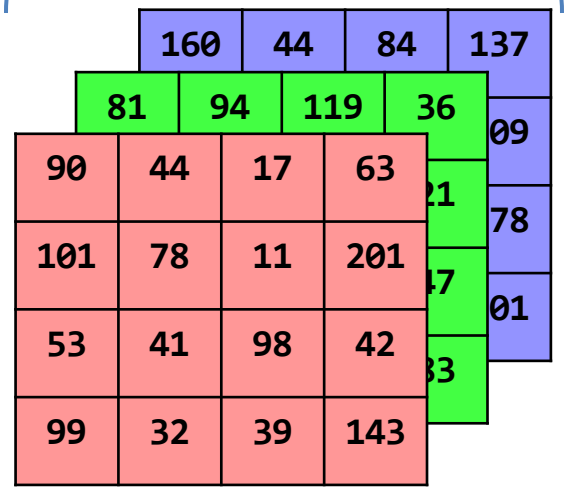
(*Clair → positivement corrélé*)

- On voit corrélation spatiale sur pixels voisins
- Réseau doit la découvrir : doit avoir + d'exemples d'entraînement
- Si applique permutation identique sur les pixels, même précision

Sur images couleur CIFAR-10

$32 \times 32 \times 3 = 3072$

Image RGB

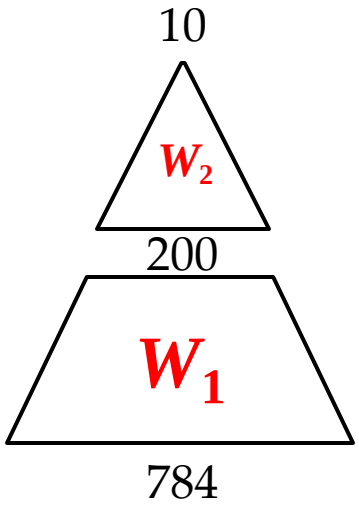


Exemple de W trouvé

(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 ~160,000
  - précédent ~8,000 paramètres
- Résultats? Encore ~92%!

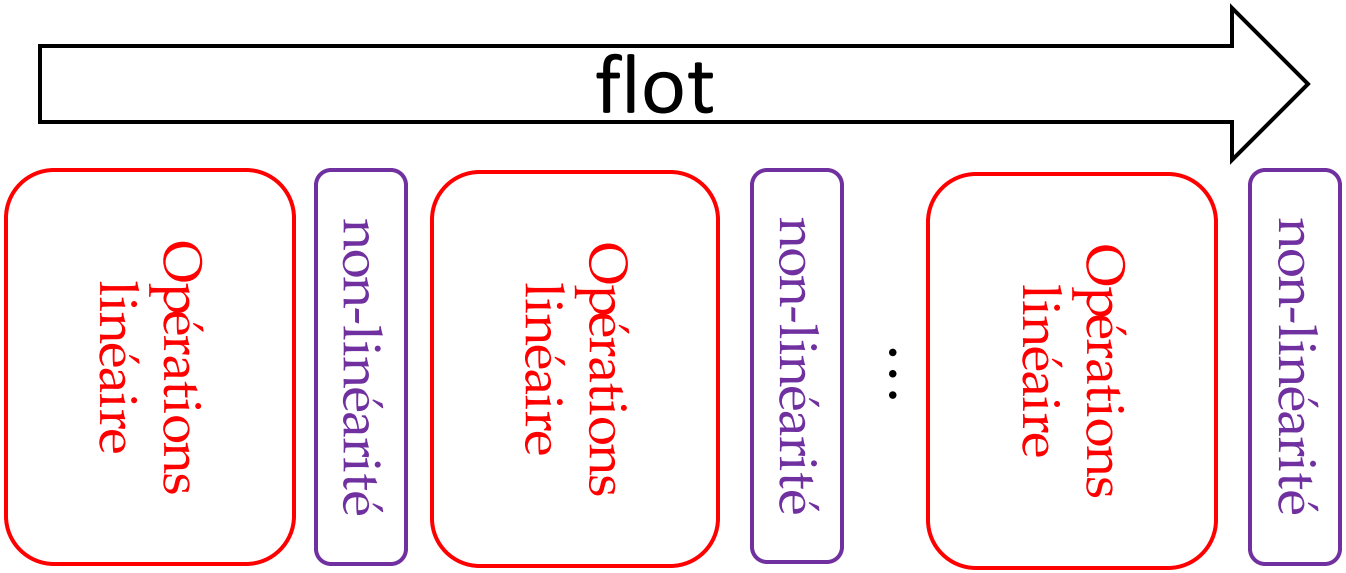


- Pas plus expressif que linéaire 1 couche, car se simplifie :

$$W_2(W_1x+b_1) + b_2 = Wx+b$$

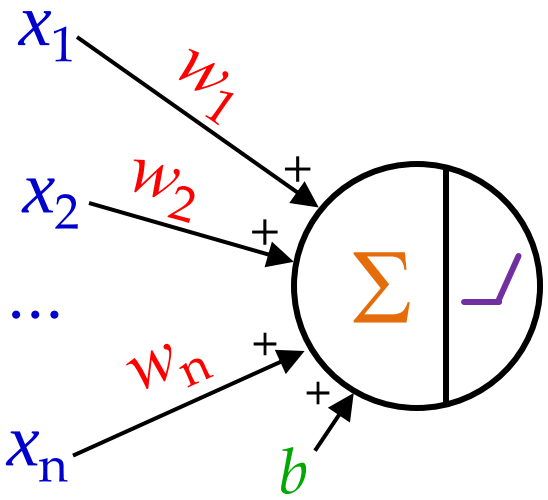
- Il faut ajouter non-linéarités pour augmenter la puissance d'expression

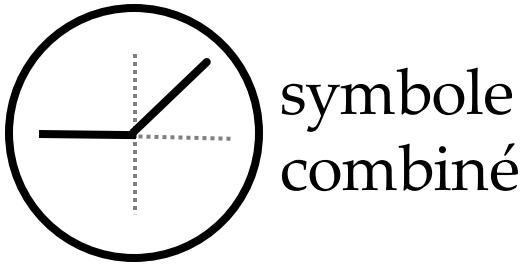
Ajout de non-linéarité



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

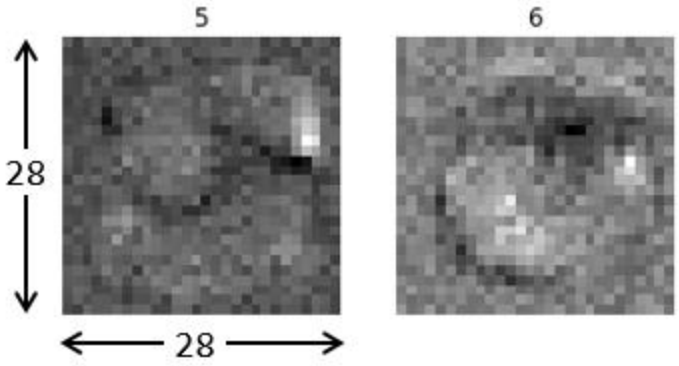
Activation : $h = g(z)$ ← non-linéaire

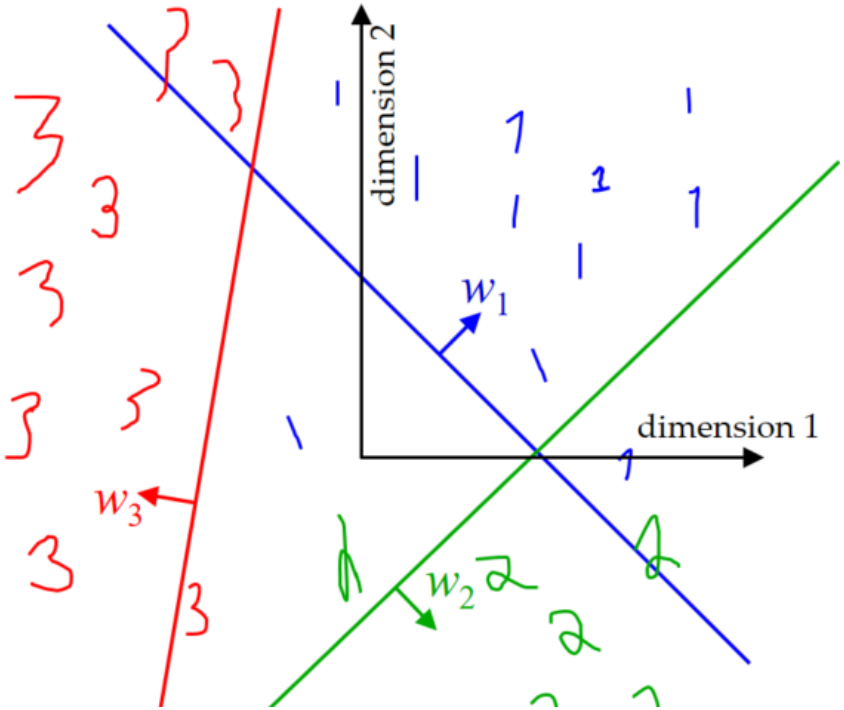




Recap de semaine 1

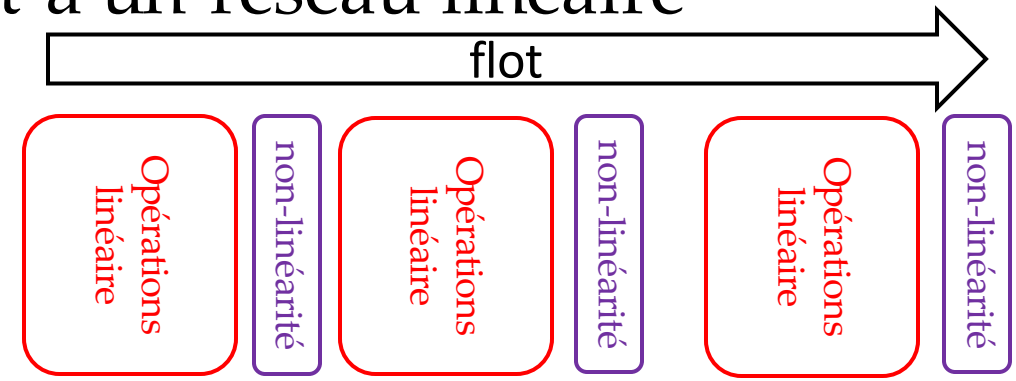
- Réseaux sont des fonctions, paramétrisés avec θ
- Toute la connaissance des données d'entraînement est résumé dans θ
- Réseau y = Wx+b est un séparateur linéaire
- W sont comme des patrons





Recap de semaine 1

- Doit intercaler des opérations non-linéaires, sinon se réduit à un réseau linéaire



- On a besoin d'un gradient qui « circule » bien pour l'apprentissage
- Apprentissage de représentation
- Renaissance : ReLU + Données + GPU

Exemple classique : XOR

Non-séparable linéairement

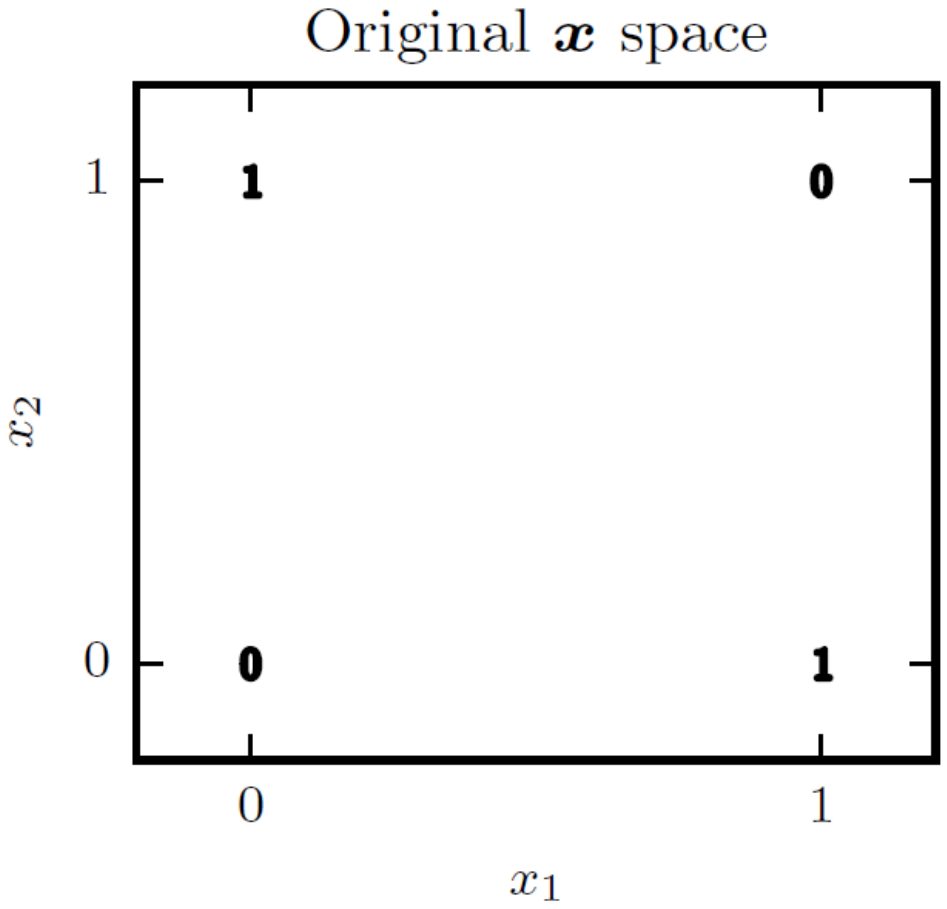
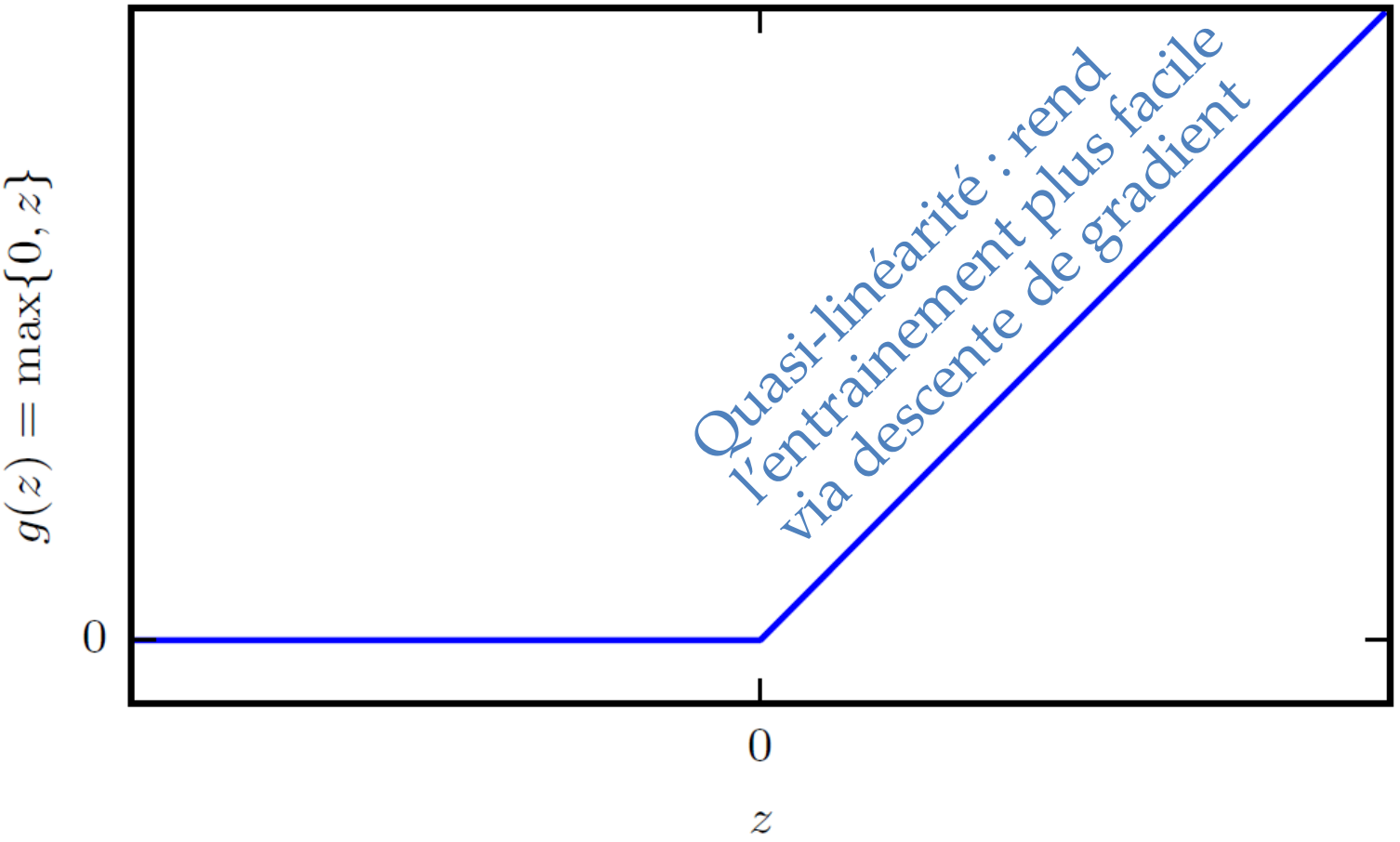


Image tirée de (I. Goodfellow) deeplearningbook.org

Activation non-linéaire ReLU



(La plus populaire en ce moment)

Image tirée de (I. Goodfellow) deeplearningbook.org

Solution XOR (format vectorisé)

4 exemples

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

$$W_1 = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}, \quad b_1 = \begin{bmatrix} 0 \\ -1 \end{bmatrix}$$

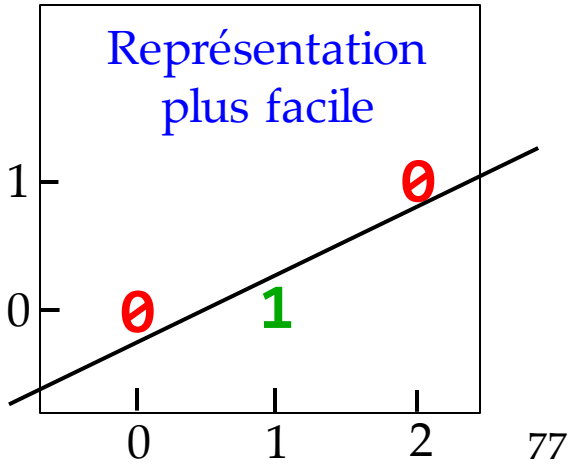
$$Y = [\ \mathbf{0} \quad \mathbf{1} \quad \mathbf{1} \quad \mathbf{0}\ ]$$

4 pré-activations

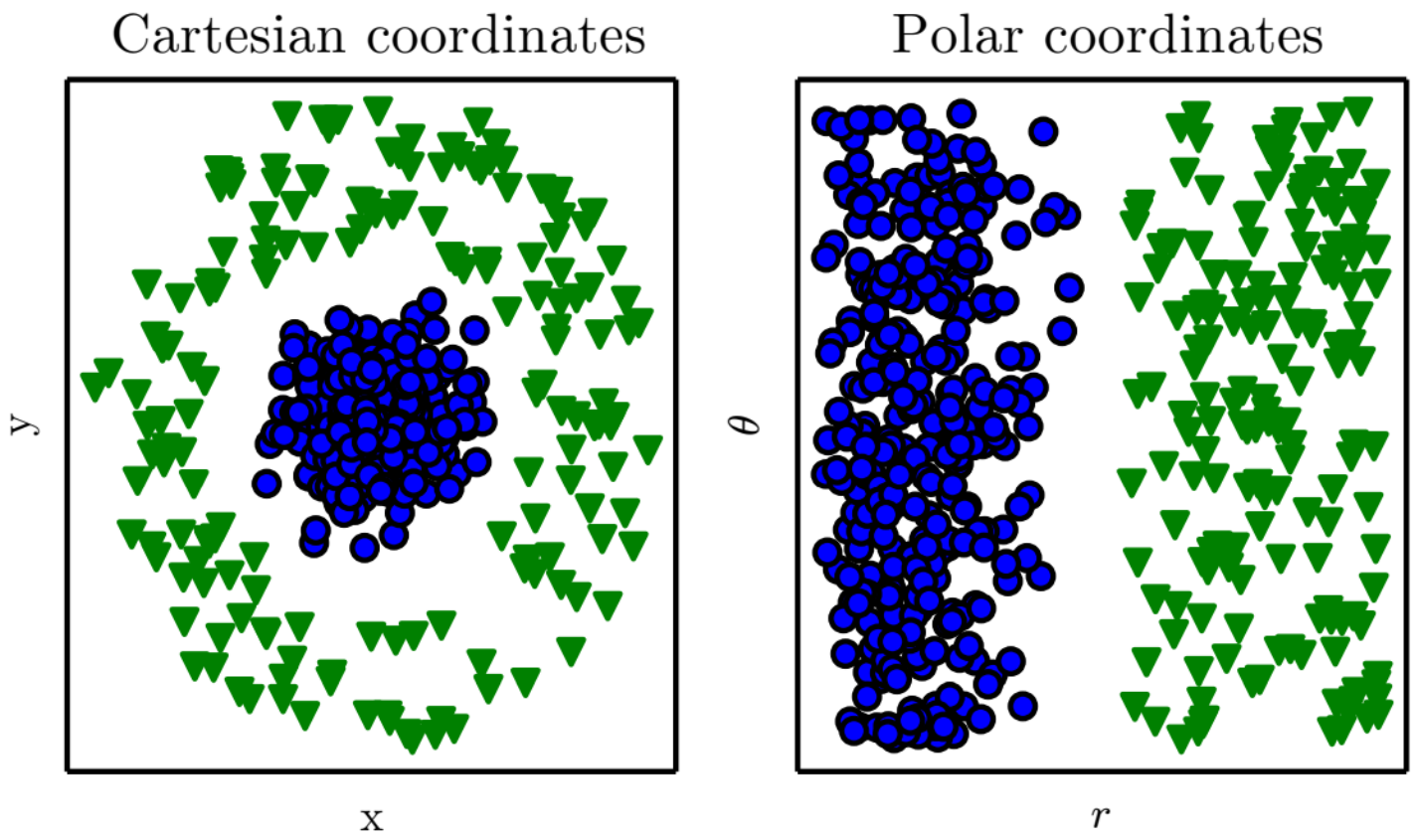
$$Z = \overbrace{\begin{bmatrix} z_1^{(1)} & z_1^{(2)} & z_1^{(3)} & z_1^{(4)} \end{bmatrix}} = W_1 X + b_1 = \begin{bmatrix} 0 & 1 & 1 & 2 \\ 0 & 1 & 1 & 2 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 & 0 \\ -1 & -1 & -1 & -1 \end{bmatrix}$$

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

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



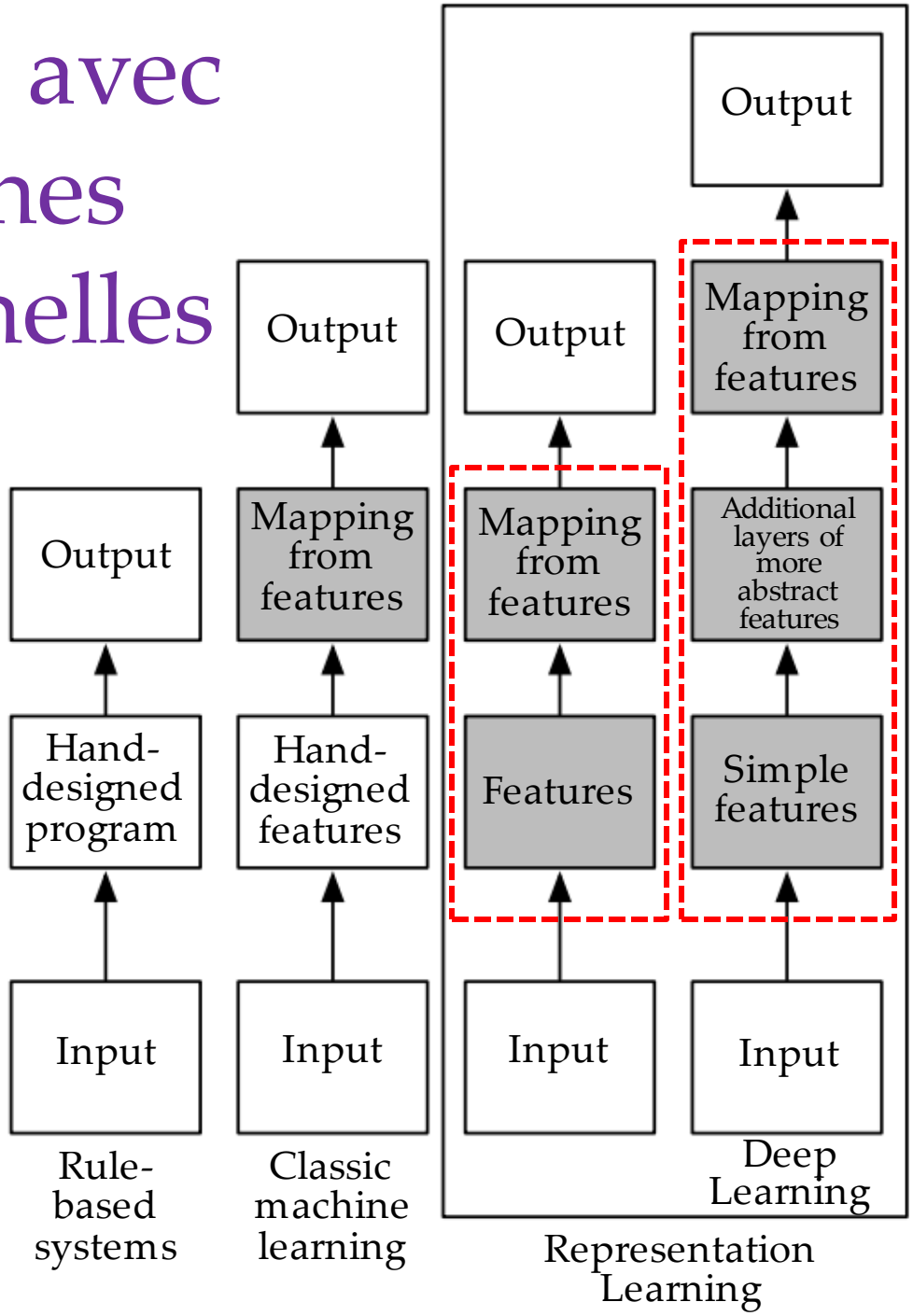
Importance de la représentation



L'un est séparable linéairement, l'autre non

Image tirée de (I. Goodfellow) deeplearningbook.org

Contraste avec approches traditionnelles



Appris conjointement

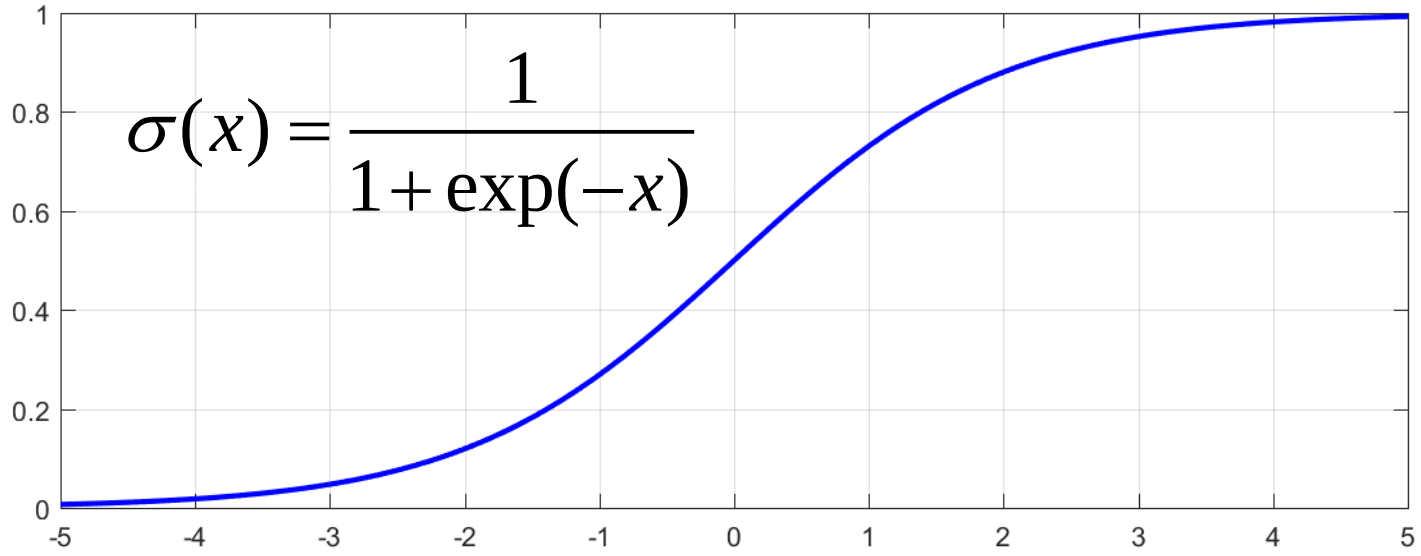
Fonctions d’activations

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 (probabilités, one-hot vector, angles, déplacements, etc.)
    - action particulière (*gating*)
  - temps de calcul

Fonction d'activation : sigmoide

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



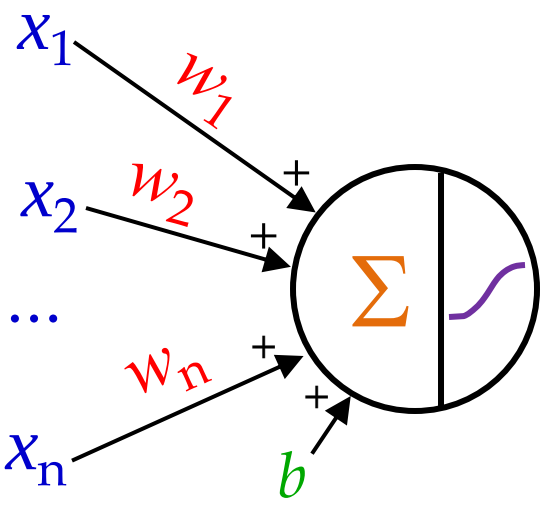
- Tombée en désuétude comme non-linéarité de base

Exemple utilisation sigmoïde

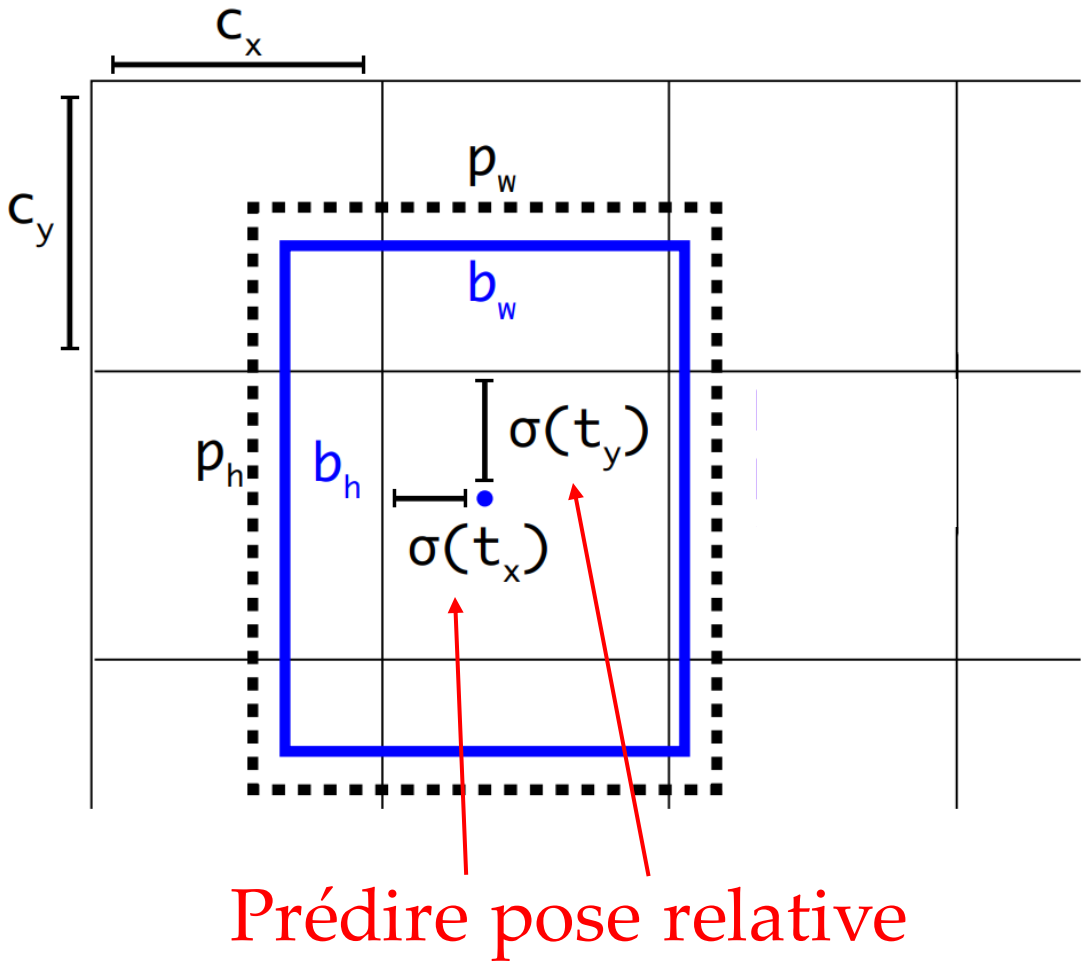
- Prédiction binaire (logistic regression)

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

$$sortie = \sigma\left(w^T x + b\right)$$



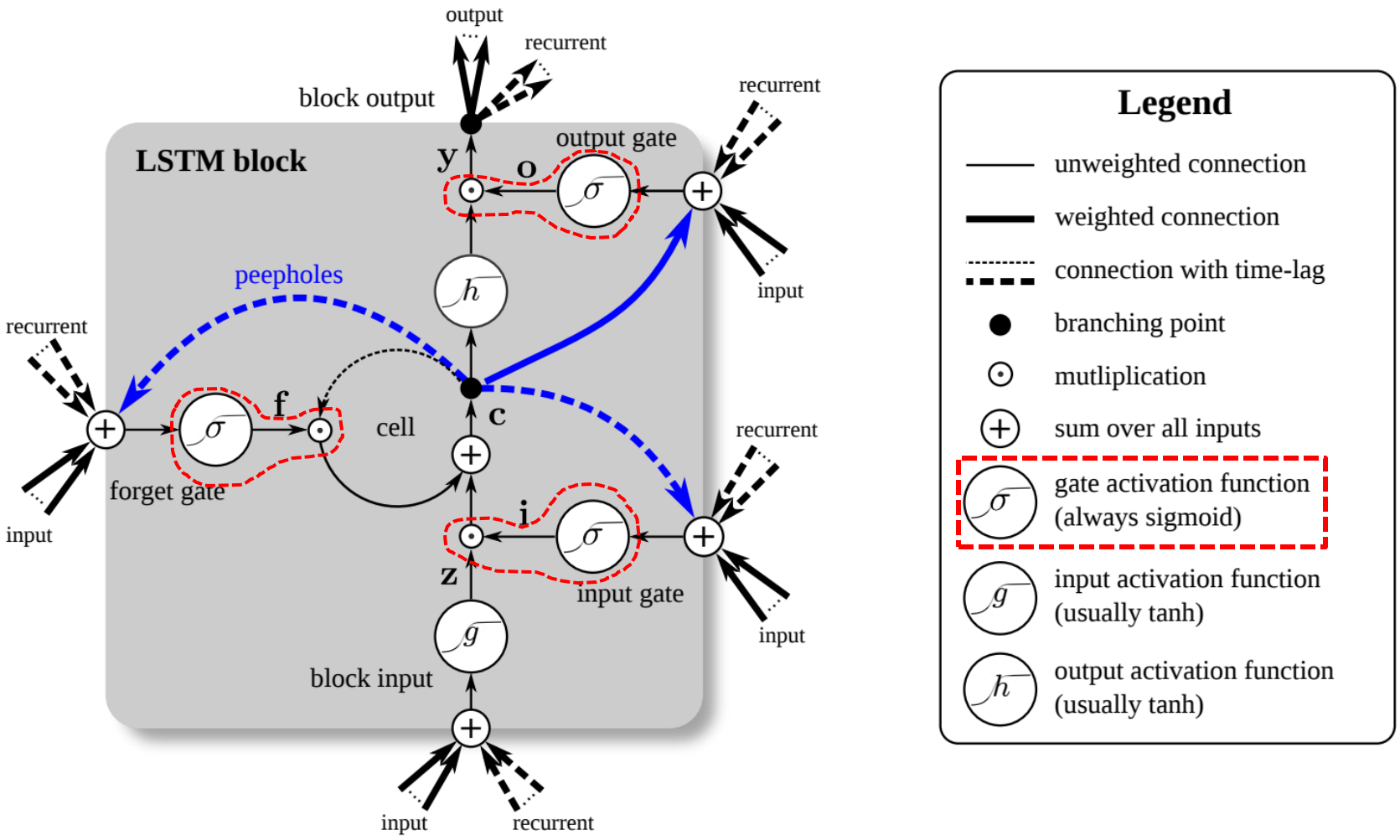
Exemples utilisation sigmoïde



Crédit image : Redmon et al., YOLO9000: Better, Faster, Stronger. CVPR 2017.

Exemple utilisation sigmoïde

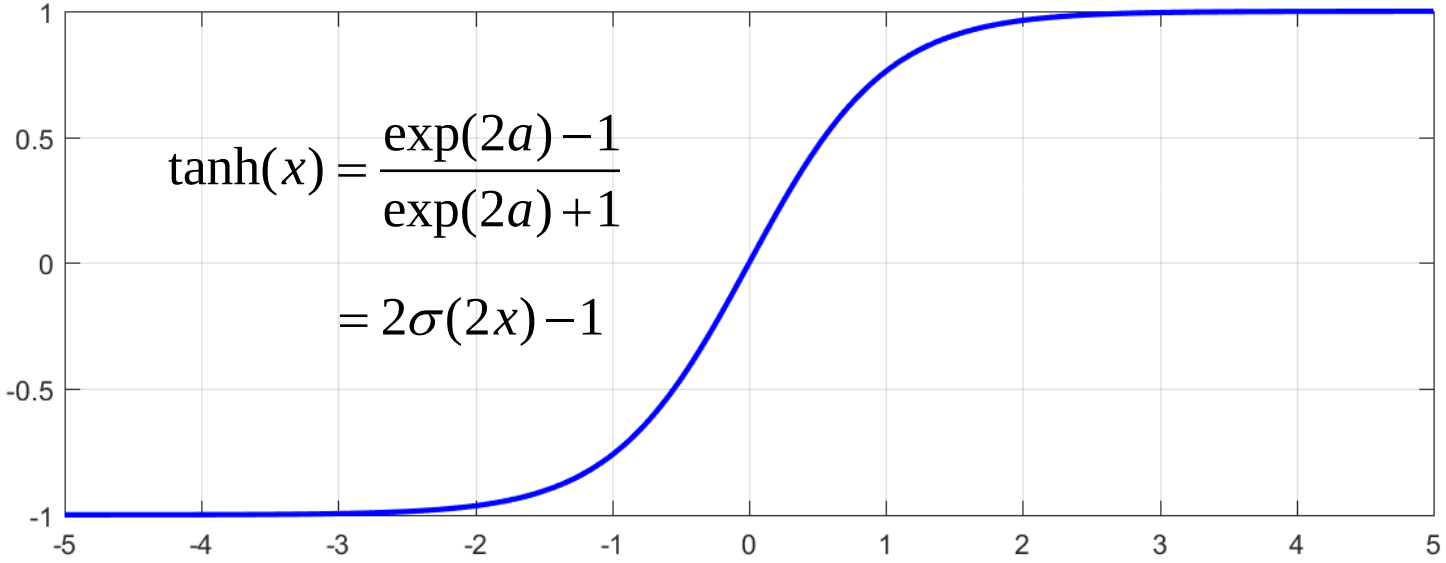
- *gating* dans Long short-term memory



Greff et al. LSTM : A Search Space Odyssey, T. on Neural Networks and Learning Systems, Oct 2017.

Fonction d'activation : tanh

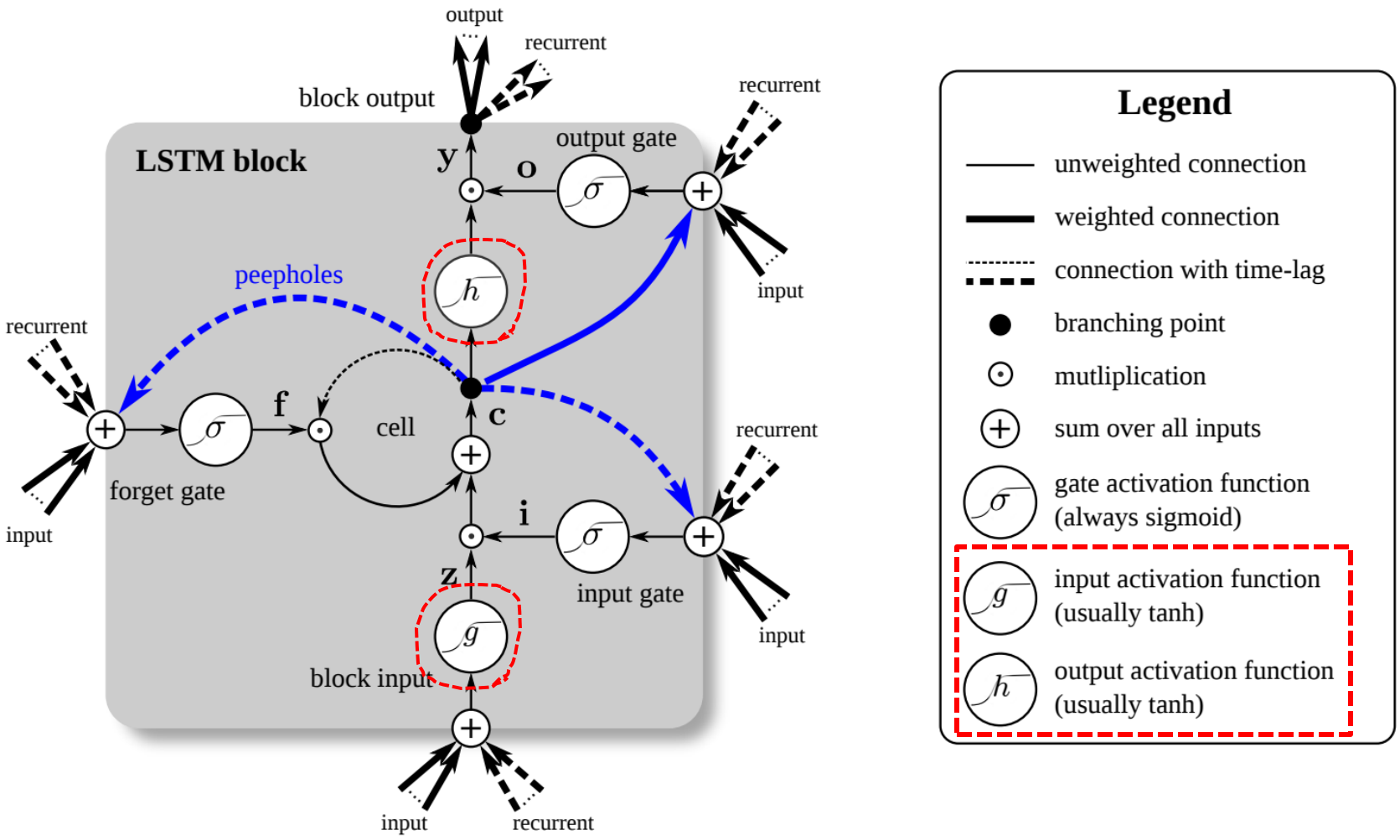
- Autre fonction historique
- Désire une sortie entre -1 et 1 *(squashing)*



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

Exemple utilisation tanh

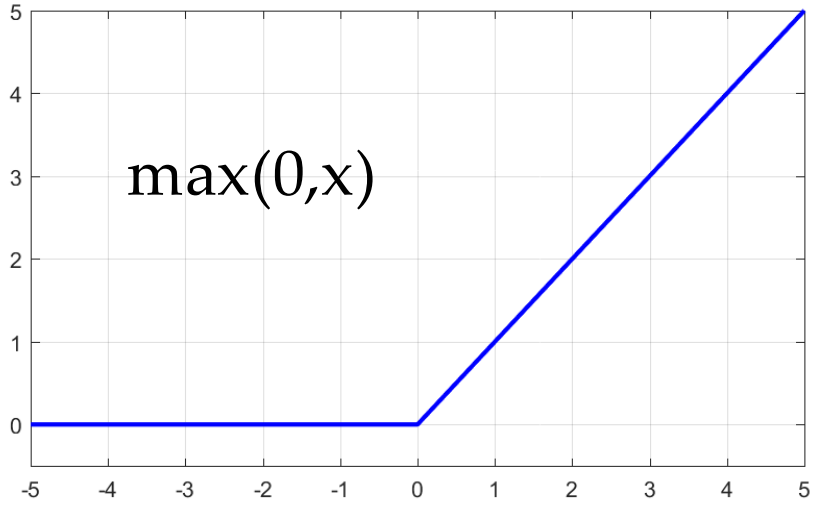
- LSTM : Long short-term memory



Greff et al. LSTM : A Search Space Odyssey, T. on Neural Networks and Learning Systems, Oct 2017.

ReLU

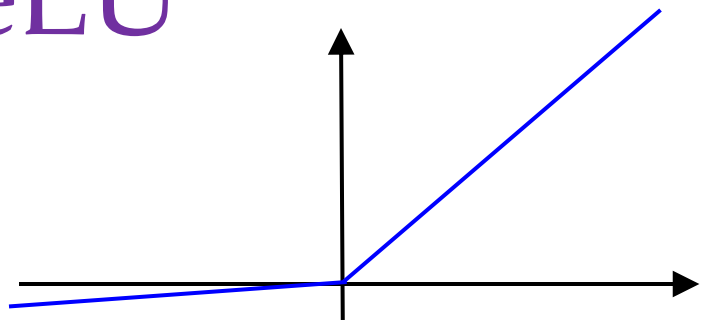
- La plus populaire comme non-linéarité
- Toujours non-négatif
  - moyenne sortie biaisée +
- Pas de limite supérieure
- Facile à calculer (**exp** est plus long)



- 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



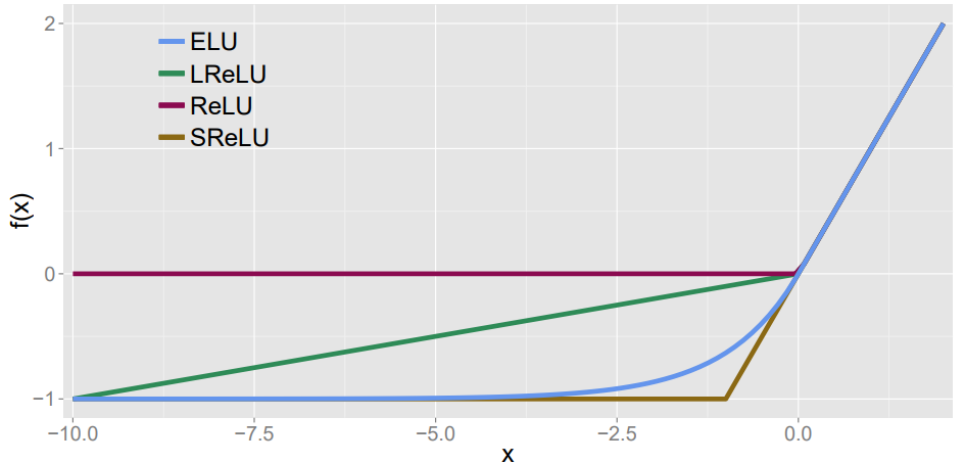
$$f(x) = \begin{cases} \alpha x & \text{si } x<0 \\ x & \text{autrement} \end{cases}$$

- Si un paramètre α (entraînable) par neurone/couche, on obtient la PReLU [1]
- Donne des distributions de sorties plus centrée à 0 que ReLU

[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.

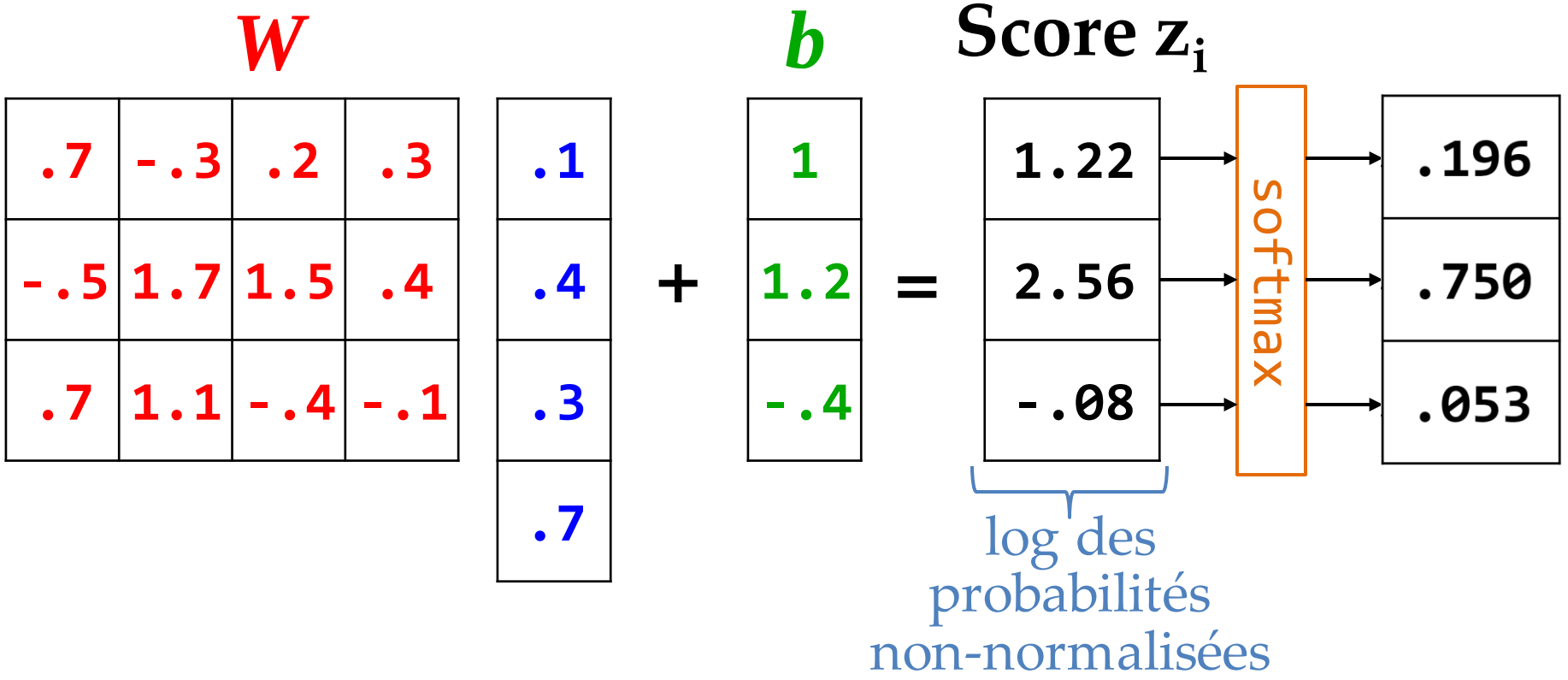
Softmax

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

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

- 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



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

Softmax

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

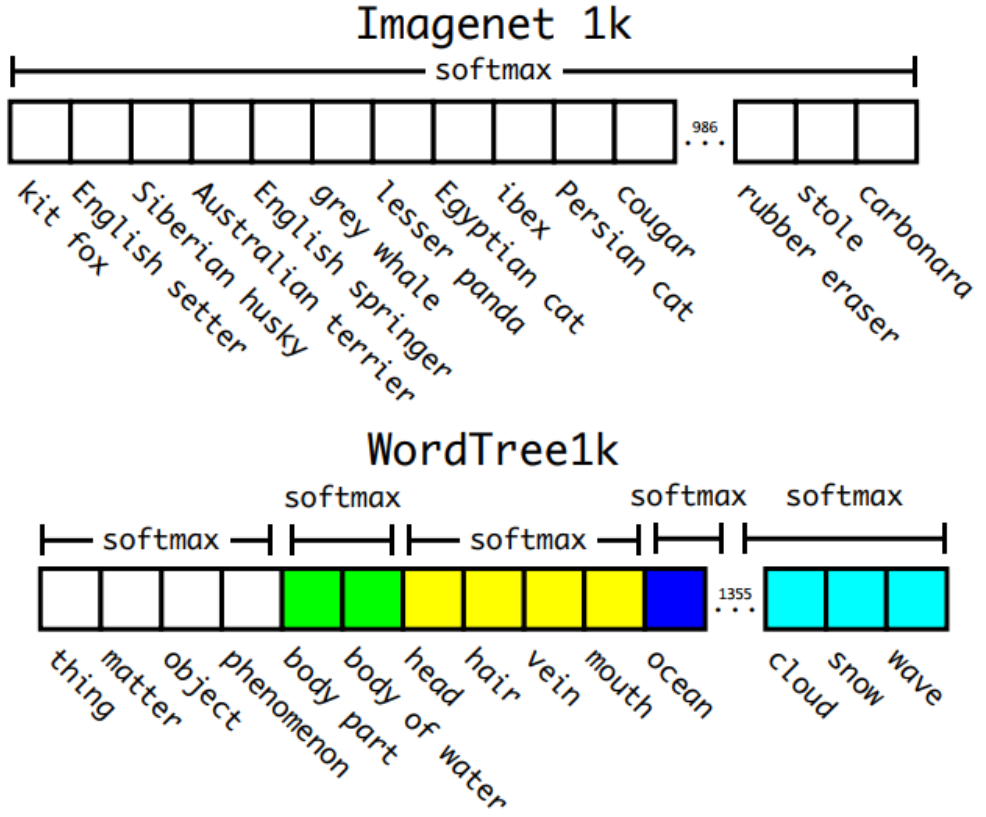


Figure 5: Prediction on ImageNet vs WordTree. Most ImageNet models use one large softmax to predict a probability distribution. Using WordTree we perform multiple softmax operations over co-hyponyms.

Crédit image : Redmon et al., YOLO9000: Better, Faster, Stronger. CVPR 2017.

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).