

# INTRODUCTION AUX RÉSEAUX DE NEURONES

## Application données textuelles et aperçu de sujets avancés :

Autoencodeurs, Réseaux adversariaux, GAN, RNN, ...

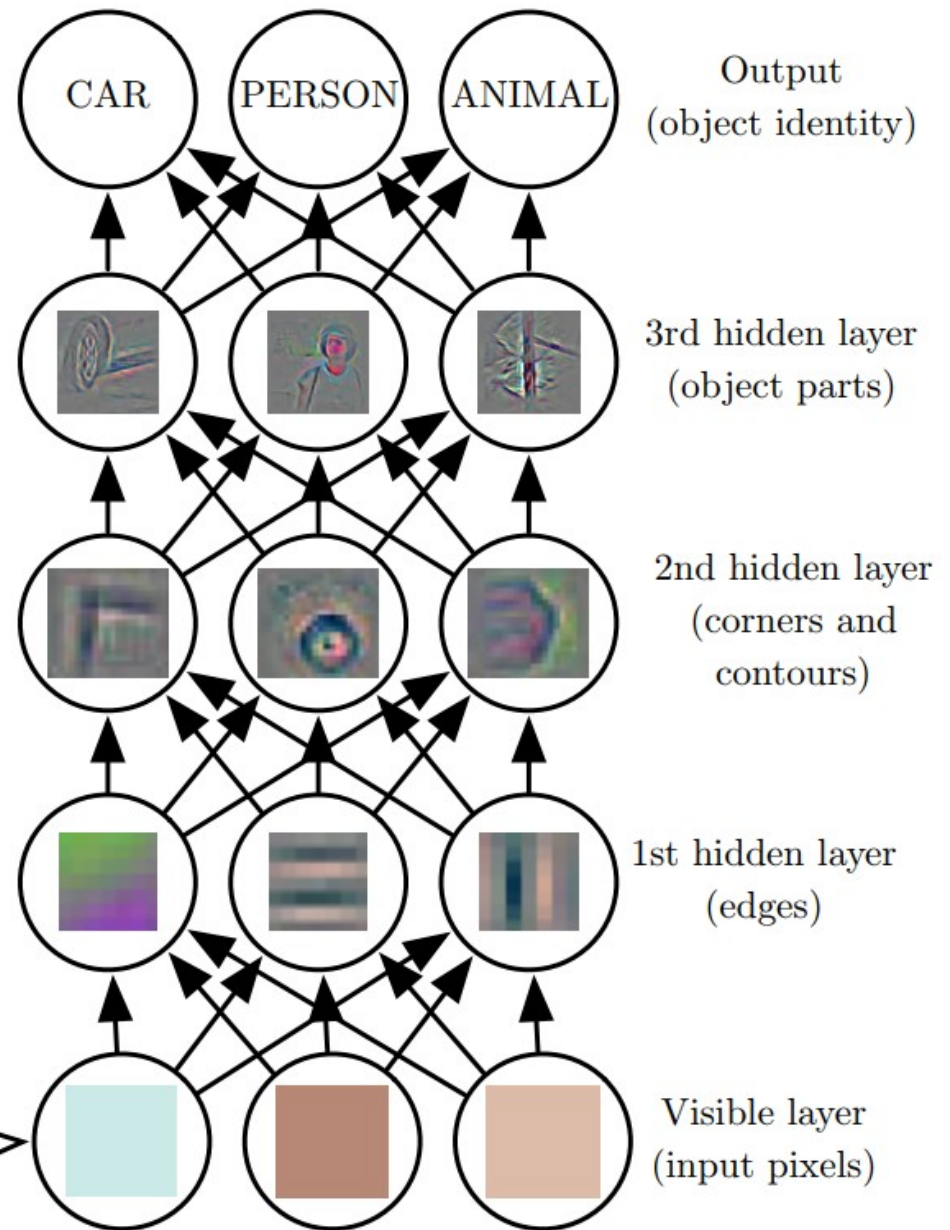
Pascal Germain\*, 2019

\* Merci spécial à [Philippe Giguère](#) pour m'avoir permis de réutiliser une partie de ces transparents.

# Transfert et apprentissage supervisé

# Hiérarchie de filtres

Va établir des liens entre des pixels de plus en plus éloignés



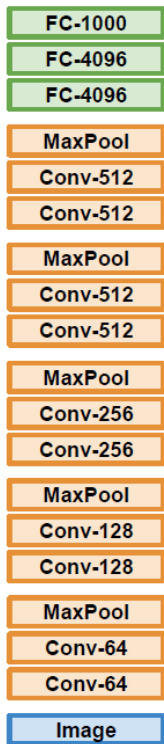
# « Finetuning »

Permet le transfert de l'apprentissage vers une tâche similaire

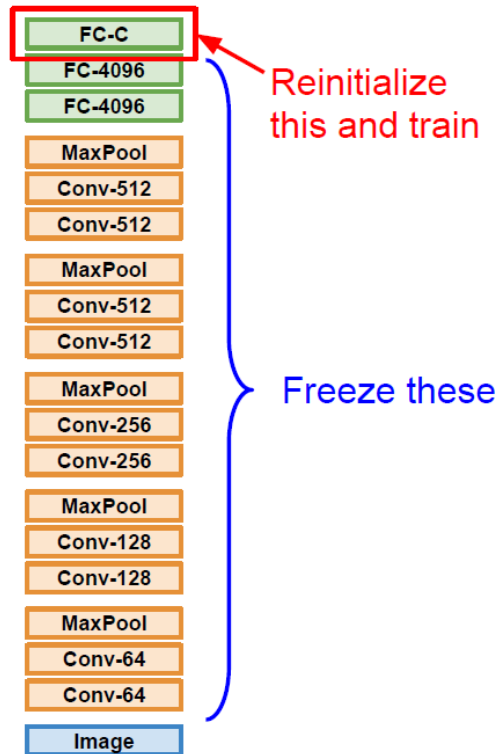
## Transfer Learning with CNNs

Donahue et al, "DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition", ICML 2014  
Razavian et al, "CNN Features Off-the-Shelf: An Astounding Baseline for Recognition", CVPR Workshops 2014

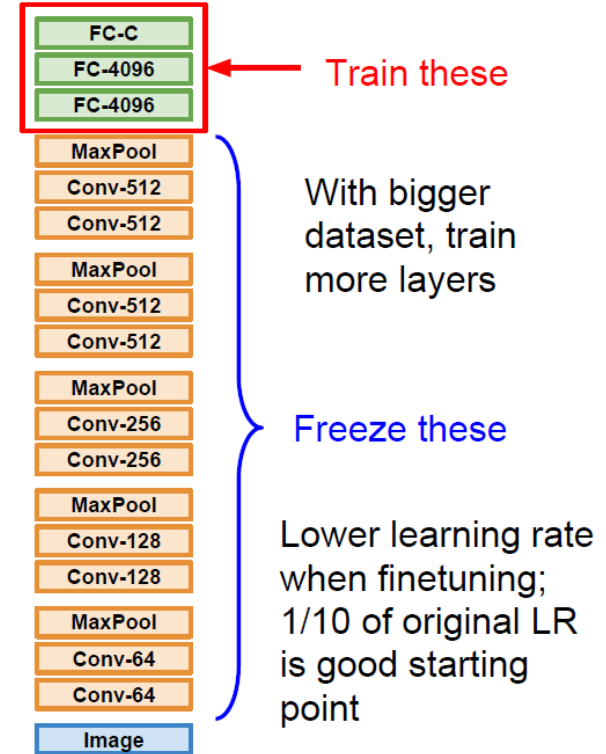
### 1. Train on Imagenet



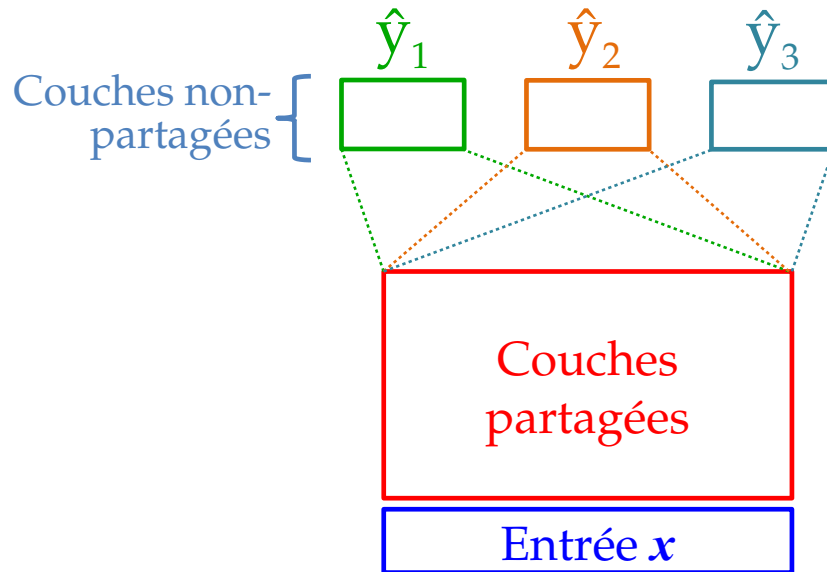
### 2. Small Dataset (C classes)



### 3. Bigger dataset



# Apprentissage multitâche



- Doit y avoir un certain lien entre les tâches

Word2vec

# Représentation d'un mot par un vecteur

- Soit un dictionnaire de  $N$  mots. Chaque mot peut être représenté par un vecteur «one-hot» de dimension  $N$  :
  - *Avion*  $\rightarrow (1, 0, 0, \dots, 0)$
  - *Ballon*  $\rightarrow (0, 1, 0, \dots, 0)$
  - ...
  - *Zèbre*  $\rightarrow (0, 0, 0, \dots, 1)$
- Dans cet espace, la distance Euclidienne entre deux mots-vecteurs est toujours la même :

$$\| \langle \text{«avion»} \rangle - \langle \text{«ballon»} \rangle \| = \| (1, -1, 0, \dots, 0) \| = \sqrt{2}$$

# Algorithme *Word2vec*

- Méthode d'apprentissage non-supervisée de représentations pour les mots
- Chaque mot est représenté par un vecteur dans un espace de valeurs réelles
- Appris sur de larges corpus de textes
- La représentation d'un mot dépend du contexte dans lequel il est utilisé.



# Contexte du voisinage

- Vous obtenez beaucoup d'information sur le sens d'un mot en regardant son voisinage dans une phrase

government debt problems turning into banking crises as has happened in  
saying that Europe needs unified banking regulation to replace the hodgepodge

↖ These words will represent *banking* ↗

- Chercher un encodage (embedding) qui permet de prédire un/des mots voisins

# Entraînement par prédiction

- L'apprentissage des vecteurs se fera via deux tâches
- **Tâche 1** : prédire le mot au centre d'un contexte de  $\pm T$

La nouvelle technologie ? permet les crypto-monnaies

← T = 3 →                      ← T = 3 →

- **Tâche 2** : prédire les mots voisins d'un mot central, pour un contexte de  $\pm T$

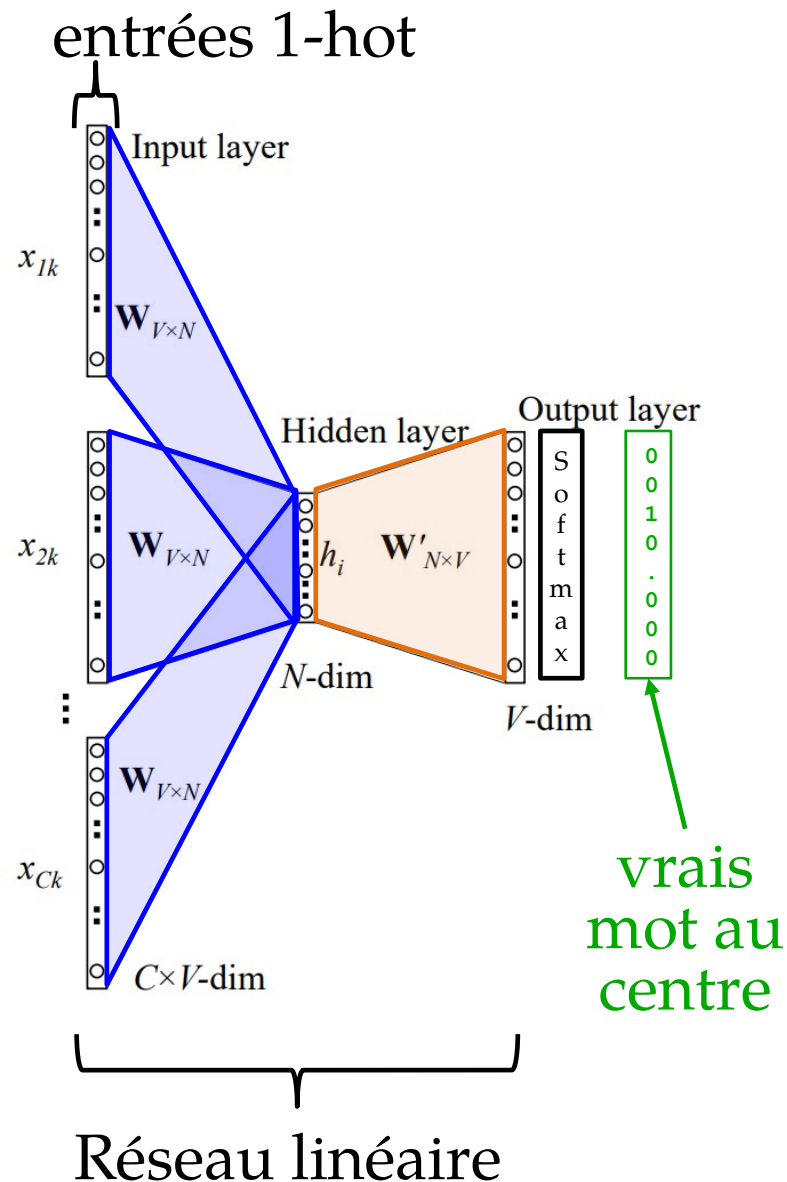
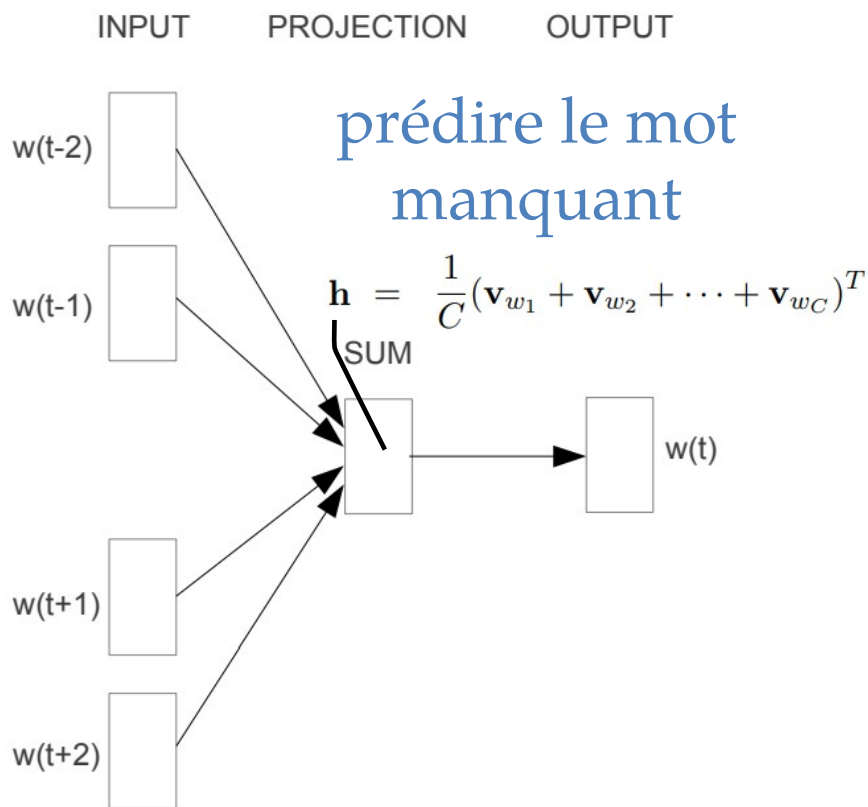
? ? ? blockchain ? ? ?

← T = 3 →                      ← T = 3 →

# Tâche 1 : prédiction d'un mot

## CBOW (continuous bag-of-words)

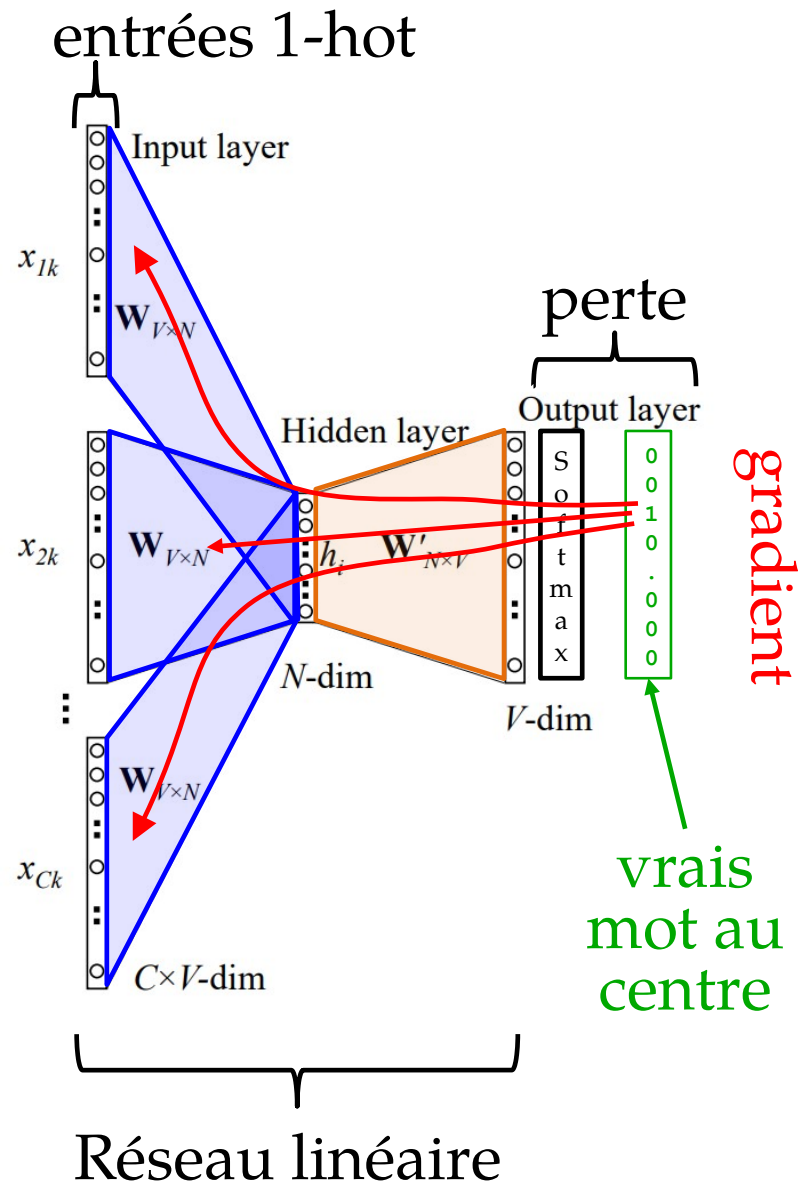
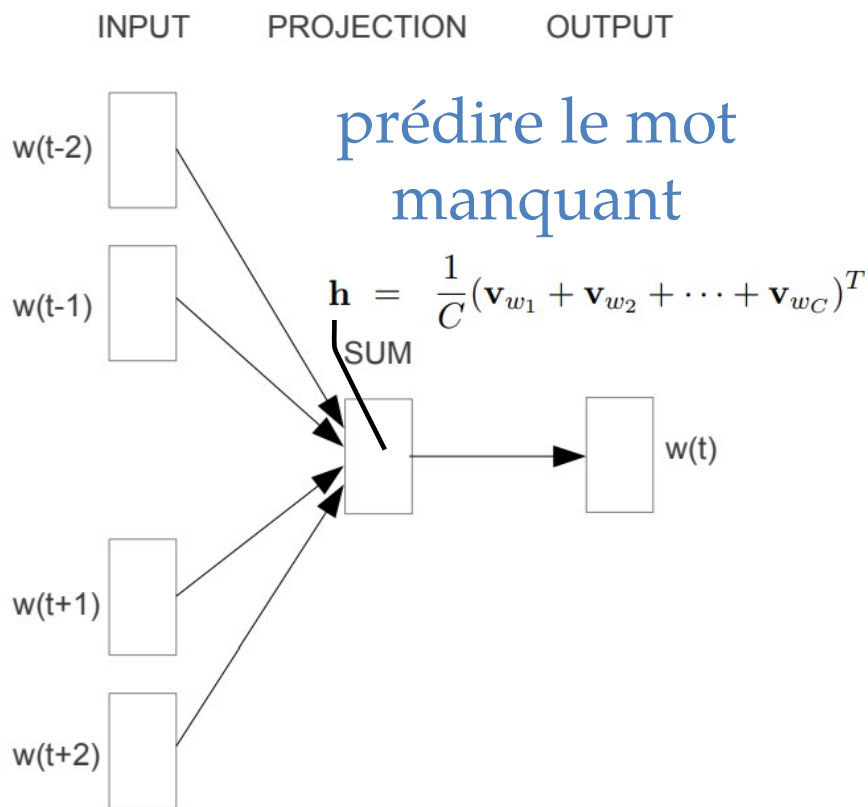
(ne tient pas compte de l'ordre des mots)



# Tâche 1 : prédiction d'un mot

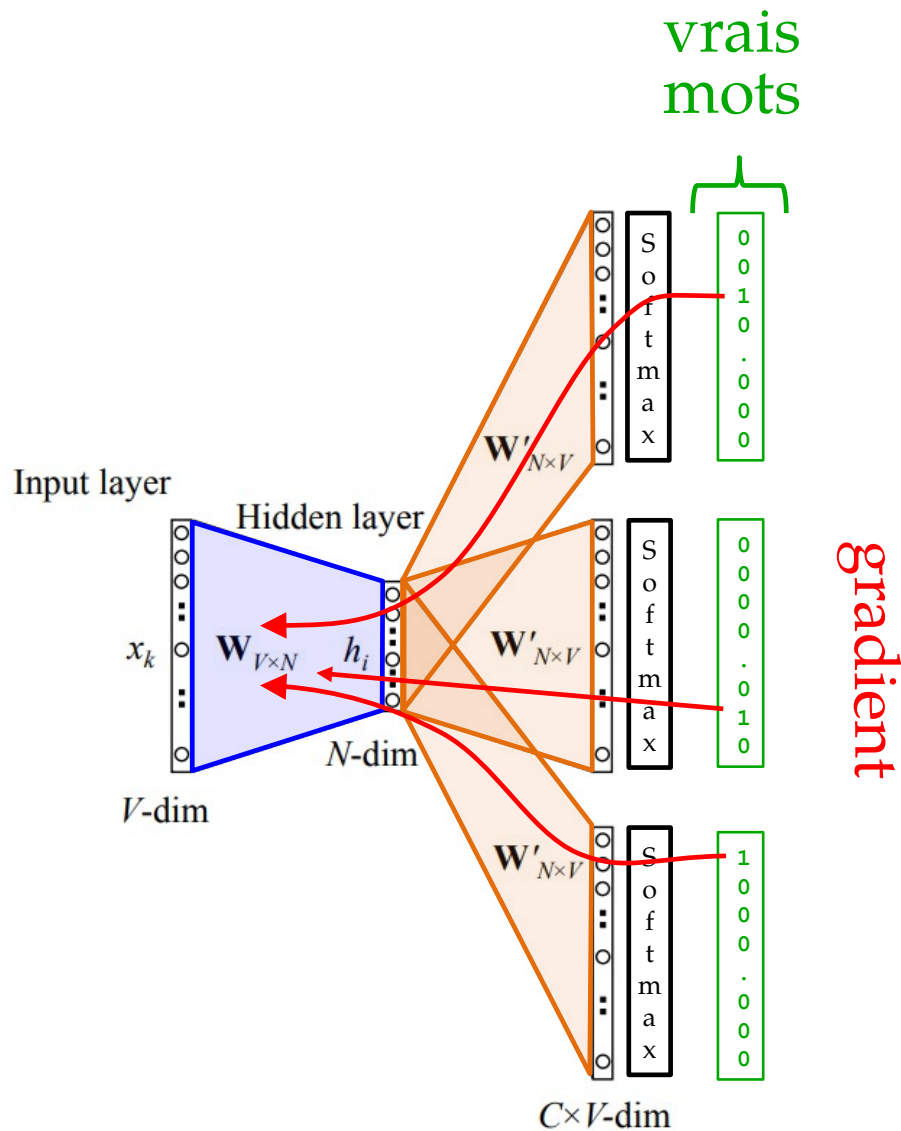
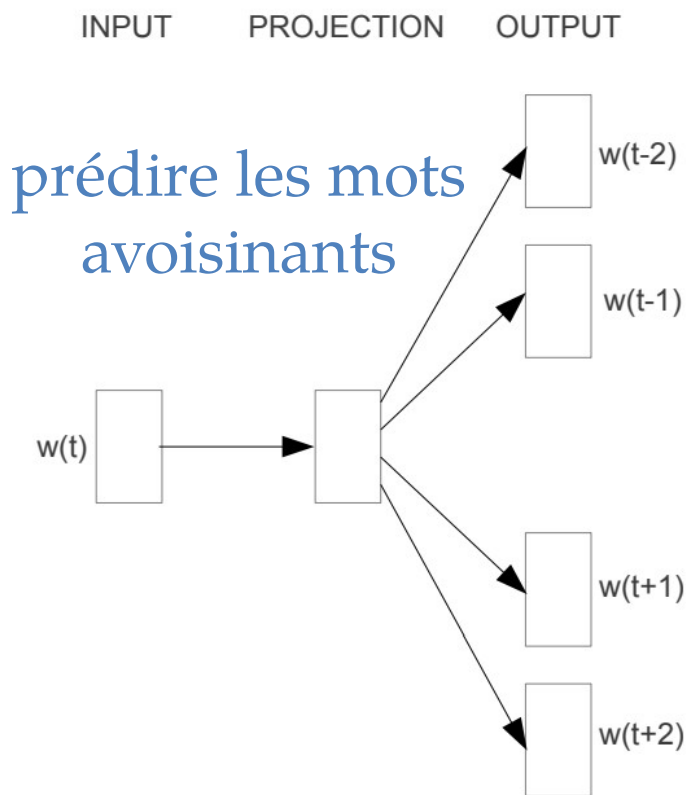
## CBOW (continuous bag-of-words)

(ne tient pas compte de l'ordre des mots)



# Tâche 2 : prédire mots voisins

## Architecture skip-gram



# Arithmétique sur «embeddings»?

- Quelle est la réponse à cette énigme :

France – Paris + Italy = ?

Rome!

# Algèbre sur ces mots-vecteurs

France - Paris + Italy = Rome

Relationship	Example 1	Example 2	Example 3
France - Paris	Italy: Rome	Japan: Tokyo	Florida: Tallahassee
big - bigger	small: larger	cold: colder	quick: quicker
Miami - Florida	Baltimore: Maryland	Dallas: Texas	Kona: Hawaii
Einstein - scientist	Messi: midfielder	Mozart: violinist	Picasso: painter
Sarkozy - France	Berlusconi: Italy	Merkel: Germany	Koizumi: Japan
copper - Cu	zinc: Zn	gold: Au	uranium: plutonium
Berlusconi - Silvio	Sarkozy: Nicolas	Putin: Medvedev	Obama: Barack
Microsoft - Windows	Google: Android	IBM: Linux	Apple: iPhone
Microsoft - Ballmer	Google: Yahoo	IBM: McNealy	Apple: Jobs
Japan - sushi	Germany: bratwurst	France: tapas	USA: pizza

# Attention aux biais !

Homme – Programmeur + Femme = Ménagère

Source : « Five Things That Scare Me About AI », Rachel Thomas



# Quelques liens

- Projet word2vec original, incluant les représentations obtenues par Google

<https://code.google.com/archive/p/word2vec/>

## **Pre-trained word and phrase vectors**

We are publishing pre-trained vectors trained on part of Google News dataset (about 100 billion words). The model contains 300-dimensional vectors for 3 million words and phrases. The phrases were obtained using a simple data-driven approach described in [2]. The archive is available here: [GoogleNews-vectors-negative300.bin.gz](https://code.google.com/archive/p/google-news-vectors/).

# Quelques liens

- Package Python «Gensim», incluant l'implémentation de word2vec (et plusieurs autres algorithmes pour données textuelles)  
<https://radimrehurek.com/gensim/models/word2vec.html>
- Tutoriel sur l'utilisation de «word embeddings» (incluant une discussion intéressante sur les biais sociaux que peuvent contenir les représentations apprises automatiquement)  
<https://github.com/fastai/word-embeddings-workshop>

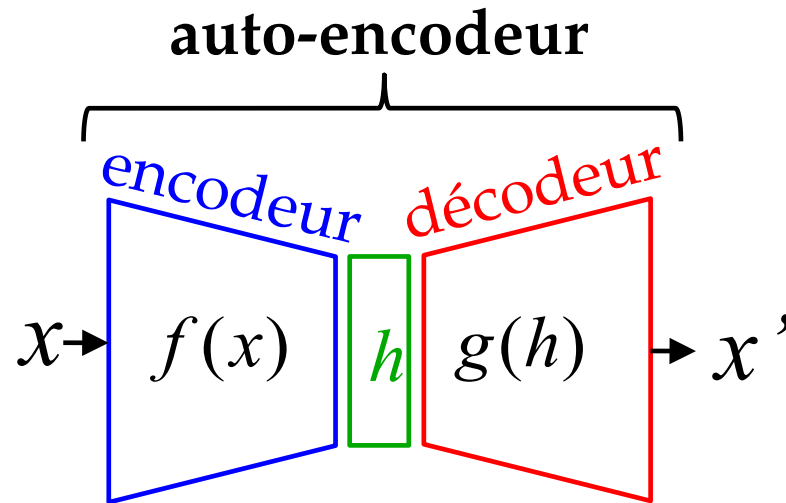
Auto-encodeur

# Perte de reconstruction

- **Supervisé** : Typiquement, la perte est basée sur l'erreur entre prédiction et la réponse désirée.

$$\text{Exemple : } L_{quad}(R(x), y) = (R(x) - y)^2$$

- **Non-supervisé** : Un **autoencodeur** minimise une perte basée sur la reconstruction de l'entrée  $x$

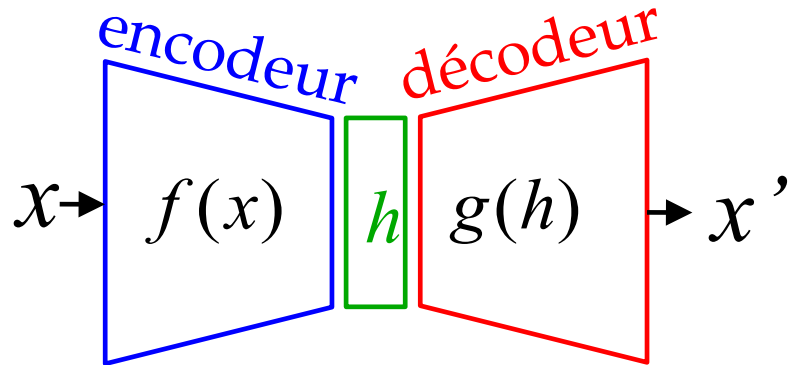


Pour éviter des solutions inintéressantes

$$\text{Perte : } L = |g(f(x)) - x|^2 + \text{régularisation}$$

# Taxonomie

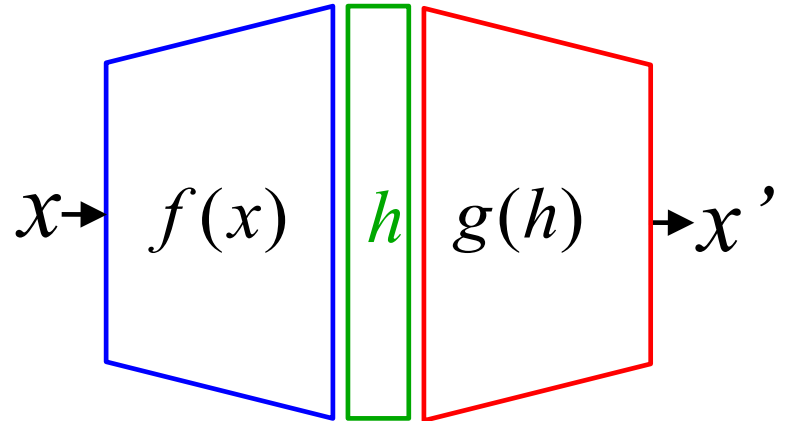
## «Undercomplete»



taille  $x >$  taille  $h$

- L'encodeur doit trouver une projection vers un espace de plus petite dimension
- si  $f, g$  sont linéaire:  
proche de l'ACP  
(exactement l'ACP avec  
 $f=U^T, g=U, U^T U=I$ )

## «Overcomplete»



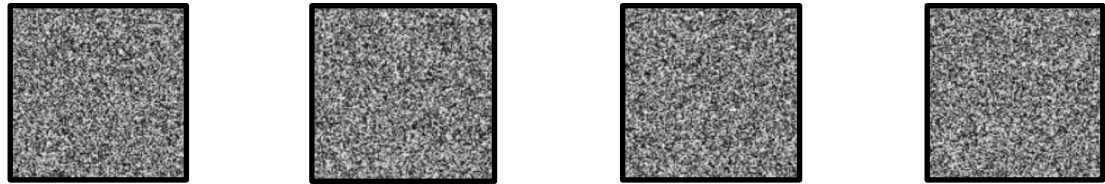
taille  $x <$  taille  $h$

- sera inutile sans régularisation :  
 $h$  peut contenir une copie de  $x$
- exemple de régularisation :  
 $x$  bruité

# Variété (*manifold*)

- La plupart des données réelles vont résider dans des sous-régions particulières de l'espace de  $x$

pixels tirés au  
hasard uniforme



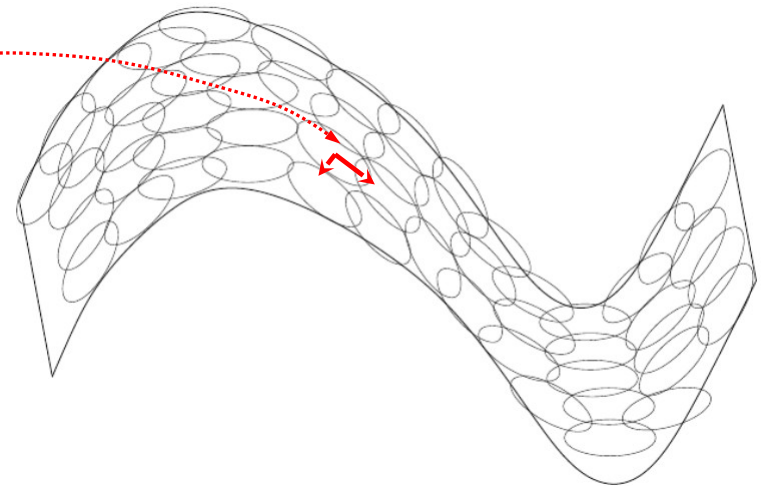
vs.



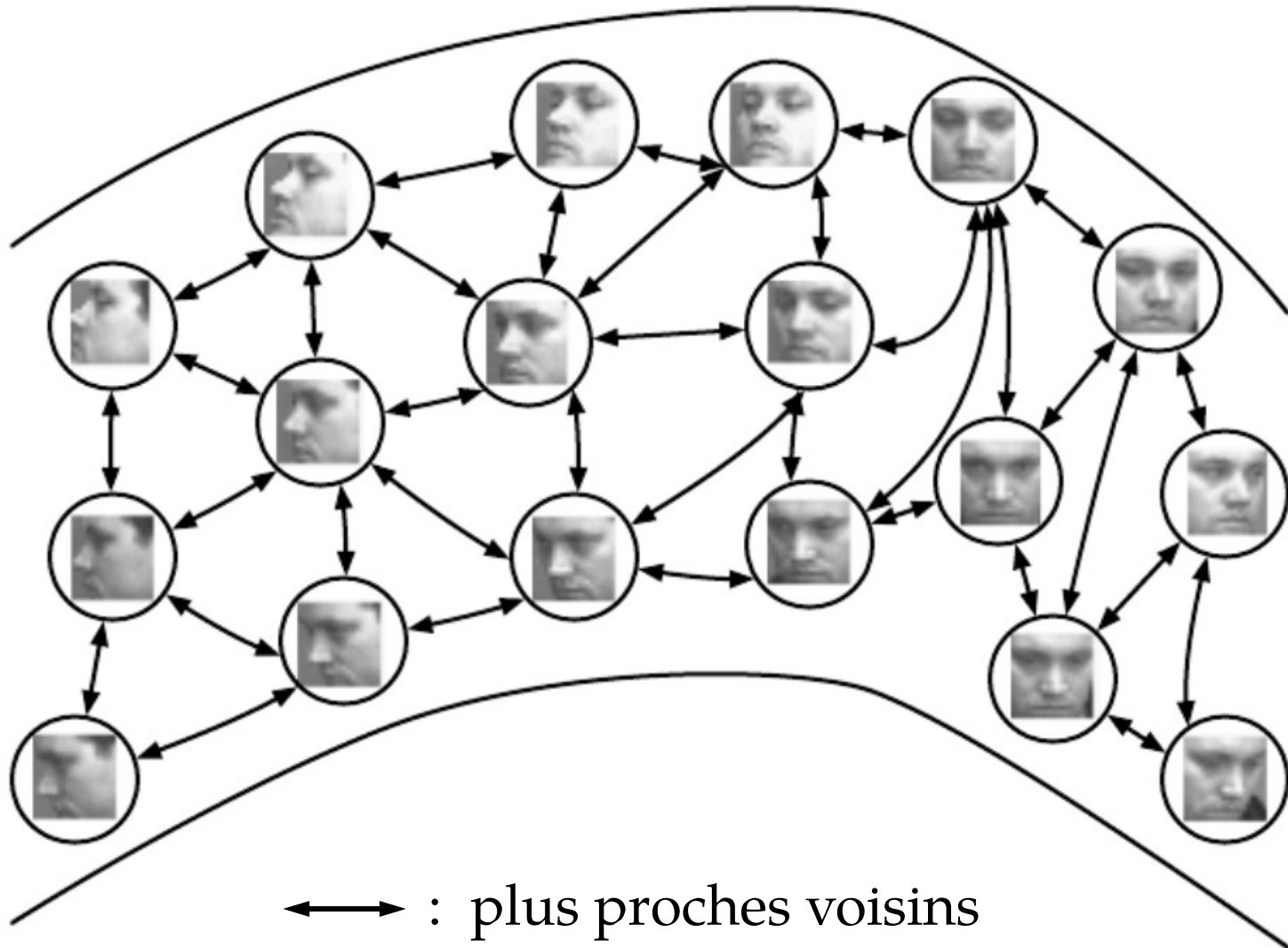
- Compression de  $x$  possible car réseau n'a pas à gérer les cas en dehors de la variété

# Variété (*manifold*)

- Idéalement, l'encodeur trouvera les variations pertinentes
  - apprendre la « surface » de la variété (**tangente**)
- Formuler l'architecture pour encourager un type de variété particulier



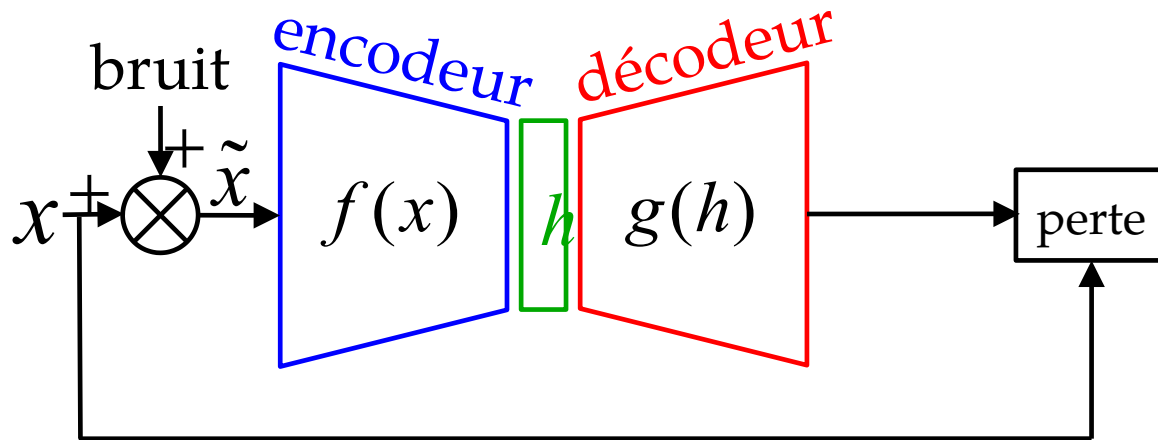
# Exemple de variété





# Autoencodeur «denoising»

- Ajoute du bruit aléatoire à l'entrée  $x$

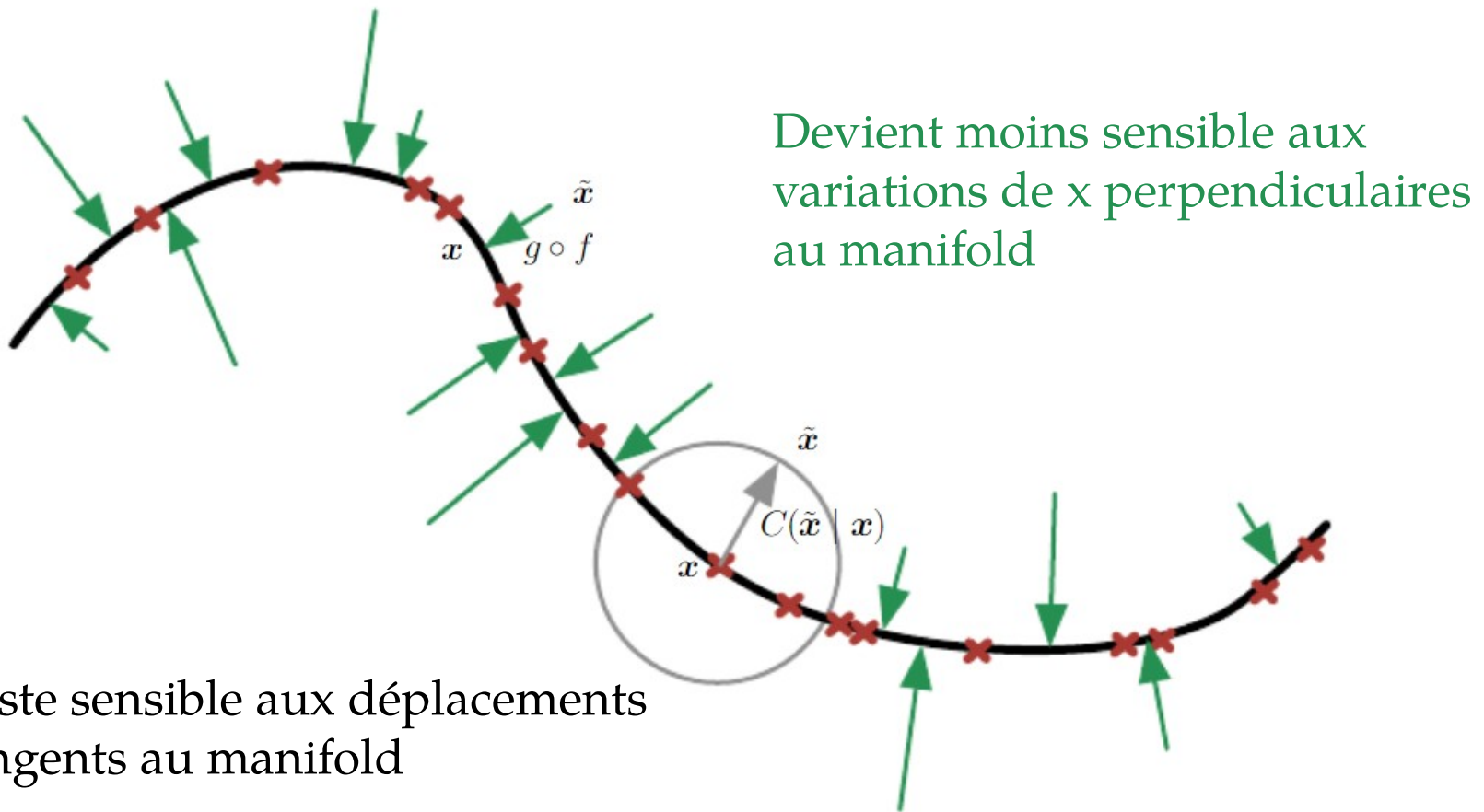


- Cherche quand même à reconstruire  $x$

$$L(x, g(f(\tilde{x})))$$

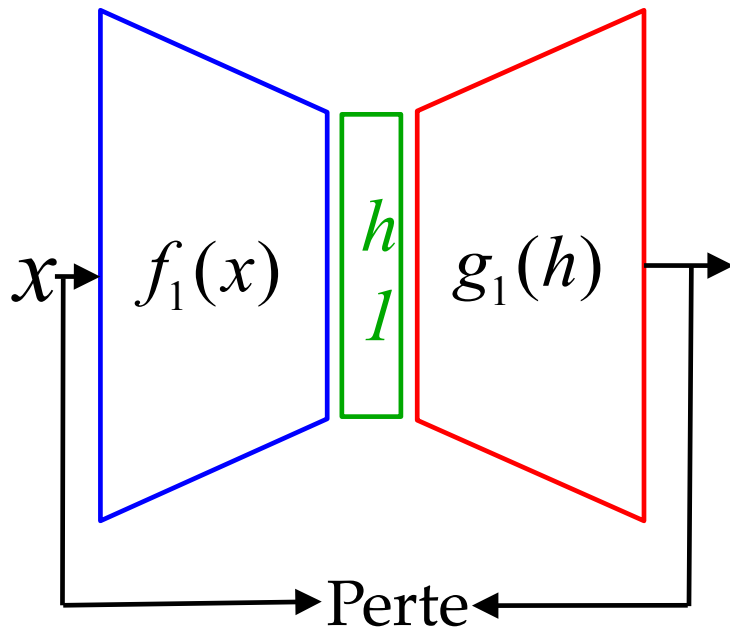
# Autoencodeur «denoising»

- Apprend à déplacer des entrées corrompues  $\tilde{x}$  vers le manifold



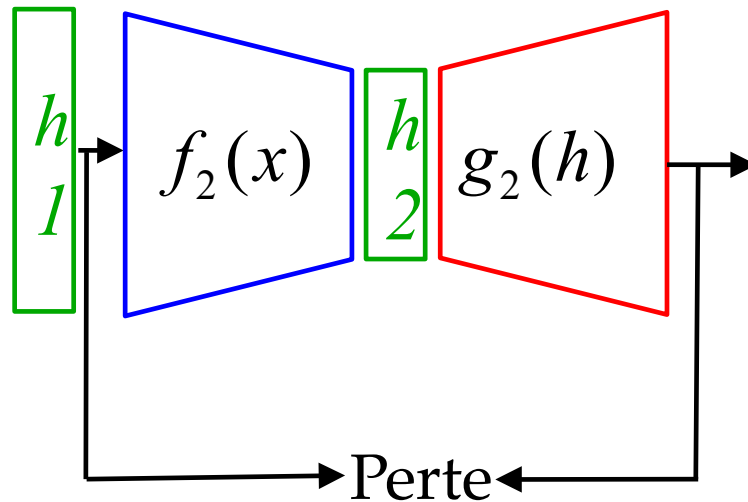
# Autoencodeurs profonds

- En cas de difficulté pour apprendre un autoencodeur profond, on procède par étape (couche par couche)

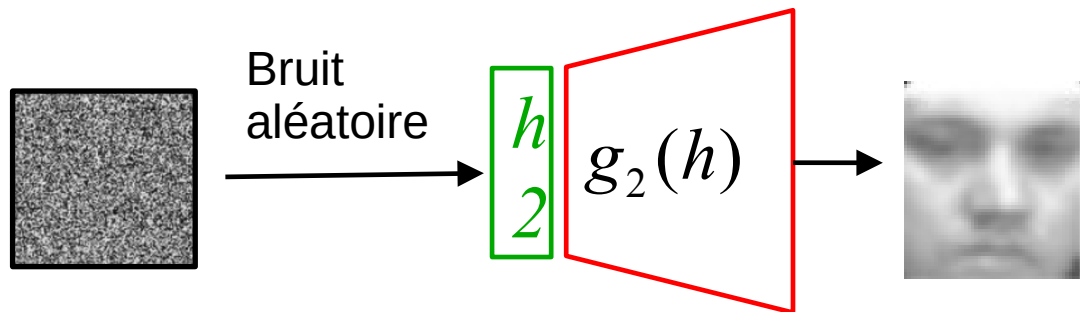


# Autoencoders profonds

- En cas de difficulté pour apprendre un autoencodeur profond, on procède par étape (couche par couche)

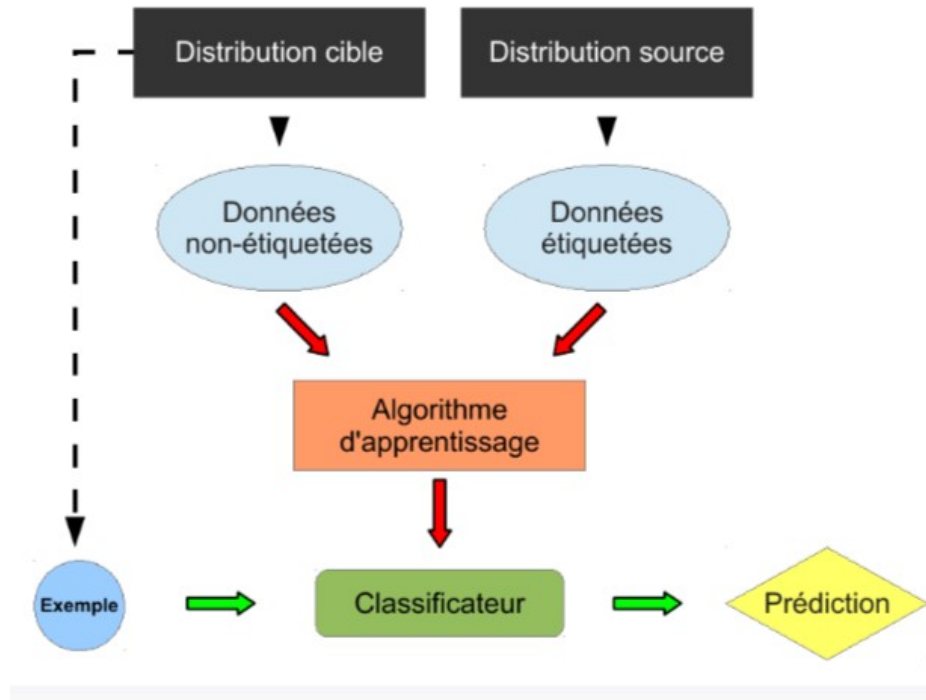


# Vers les réseaux génératifs

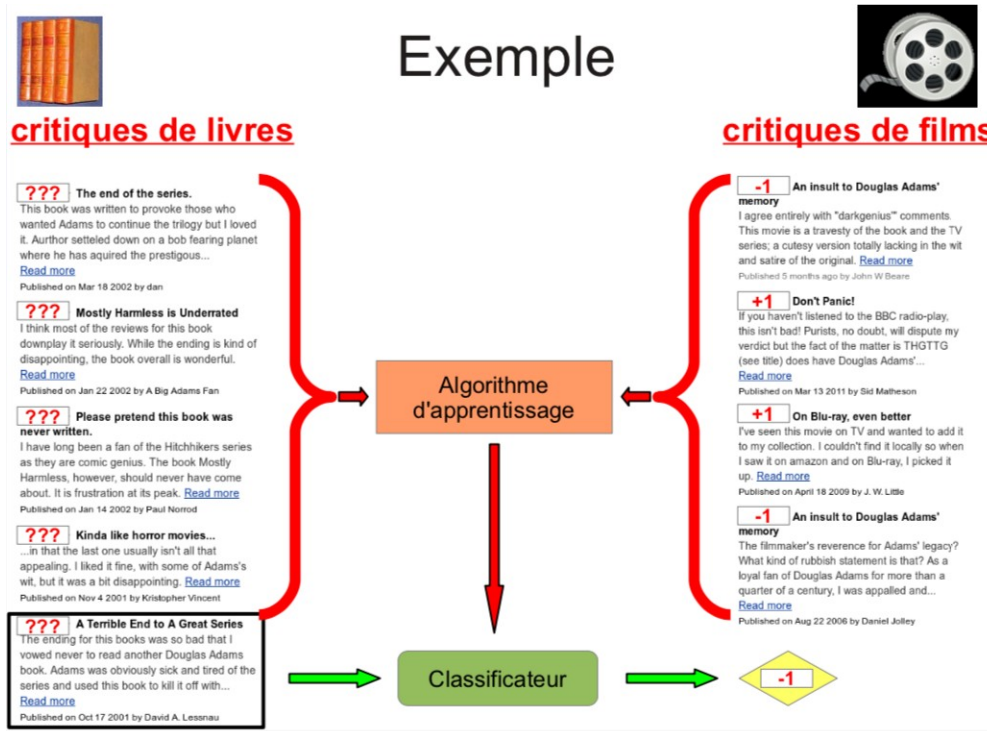


# Réseaux de neurones adversariaux

# Adaptation de domaine



# Adaptation de domaine





# « Domain Adversarial Neural Networks »

Idée : Favoriser l'émergence de représentations qui sont

- (i) discriminantes pour la tâche d'apprentissage principale sur le domaine source et
- (ii) non-discriminantes pour la détection du domaine.

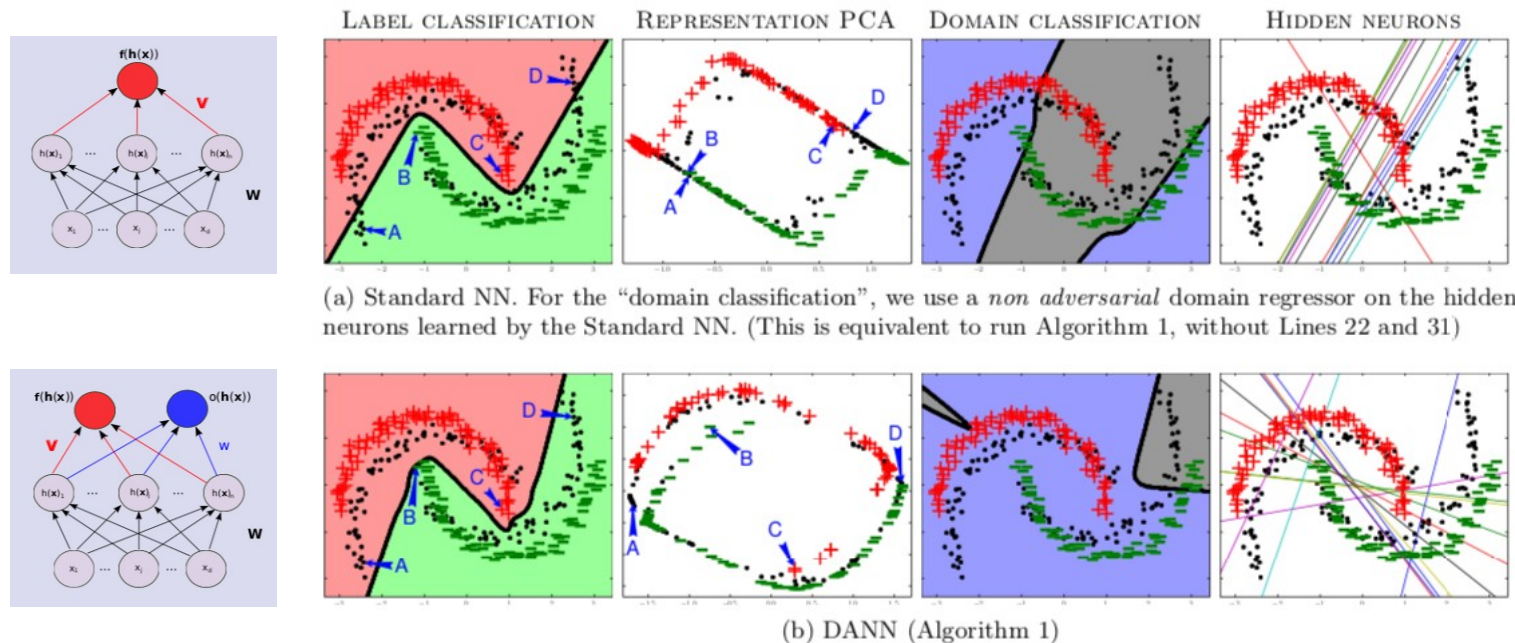


Figure 2: The *inter-twinning moons* toy problem. Examples from the source sample are represented as a “+” (label 1) and a “-” (label 0), while examples from the unlabeled target sample are represented as black dots. See text for the figure discussion.

# « Domain Adversarial Neural Networks »

$$\min_{\mathbf{W}, \mathbf{V}, \mathbf{b}, \mathbf{c}} \left[ \underbrace{\frac{1}{m} \sum_{i=1}^m -\log(f_{y_i^s}(\mathbf{x}_i^s))}_{\text{source loss}} + \lambda \max_{\mathbf{w}, \mathbf{d}} \left( \underbrace{\frac{1}{m} \sum_{i=1}^m \log(o(\mathbf{h}(\mathbf{x}_i^s))) + \frac{1}{m} \sum_{i=1}^m \log(1 - o(\mathbf{h}(\mathbf{x}_i^t)))}_{\text{adaptation regularizer}} \right) \right],$$

where  $\lambda > 0$  weights the domain adaptation regularization term.

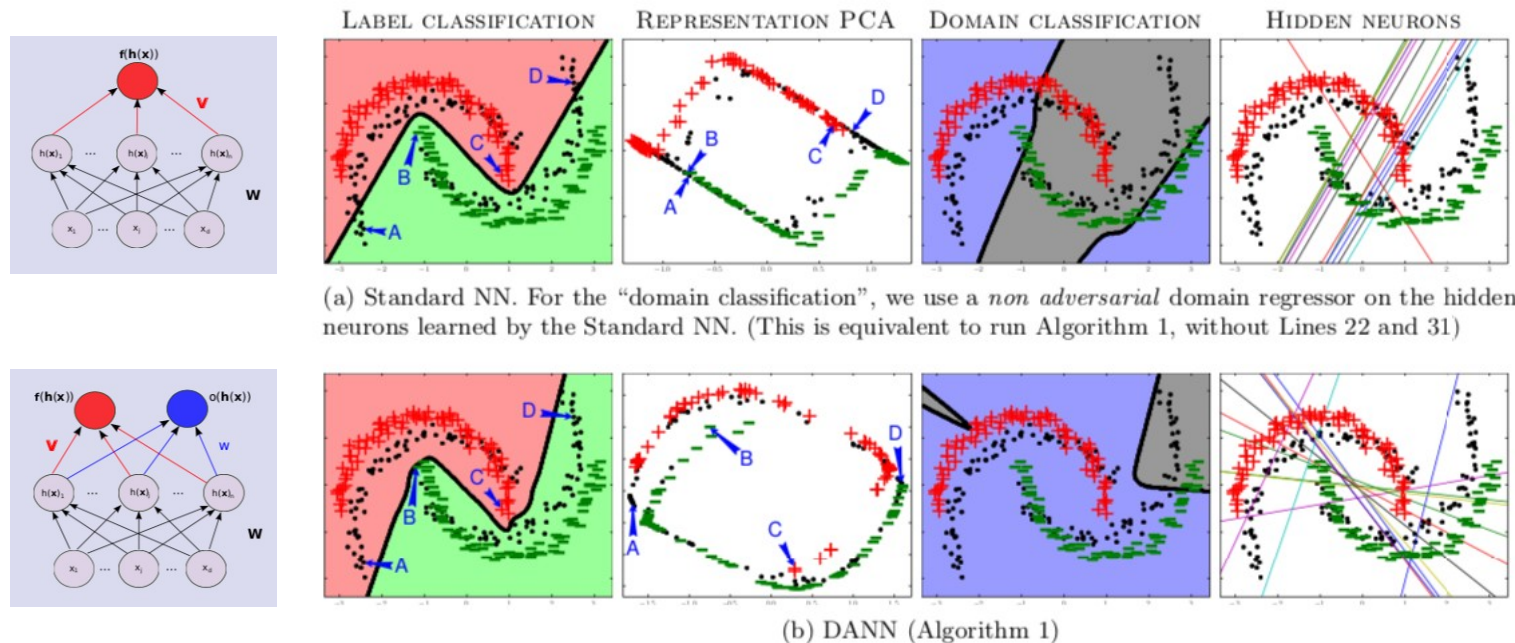
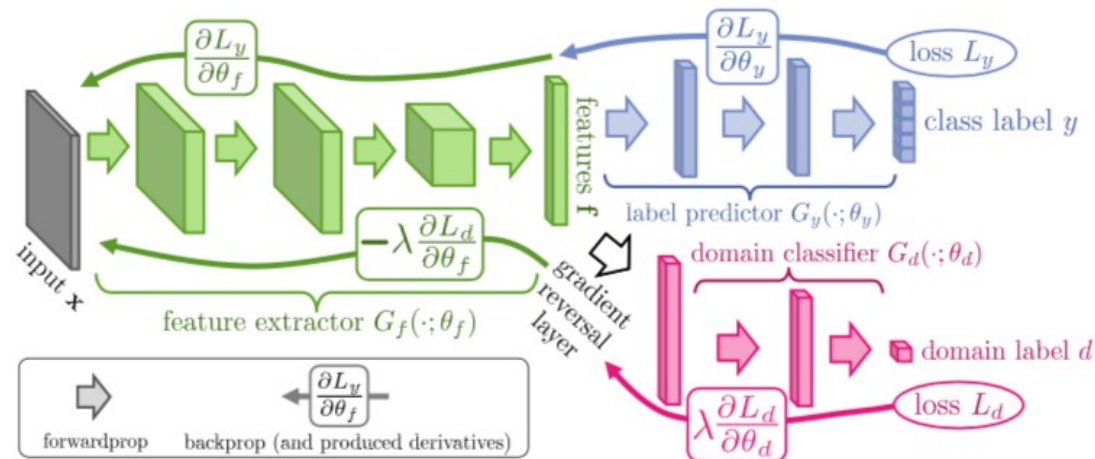



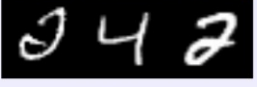
Figure 2: The *inter-twinning moons* toy problem. Examples from the source sample are represented as a “+” (label 1) and a “-” (label 0), while examples from the unlabeled target sample are represented as black dots. See text for the figure discussion.

# « Domain Adversarial Neural Networks »

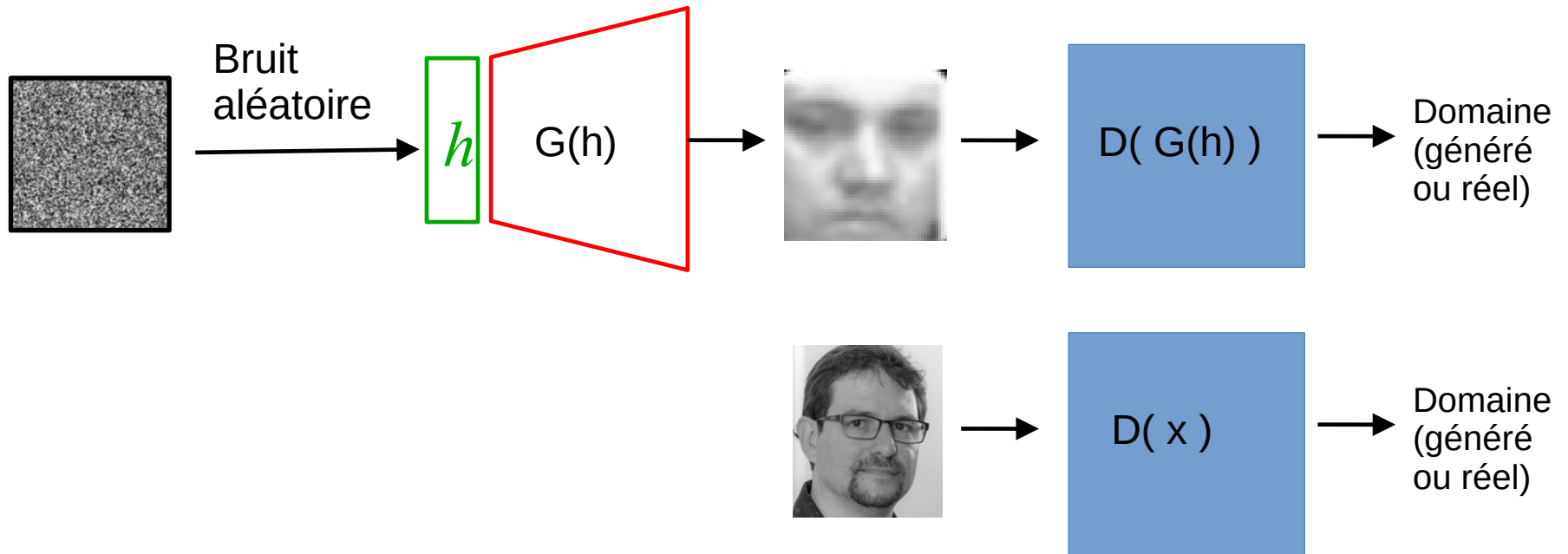
GANIN, USTINOVA, AJAKAN, GERMAIN, LAROCHELLE, LAVIOLETTE, MARCHAND AND LEMPITSKY



# « Domain Adversarial Neural Networks »

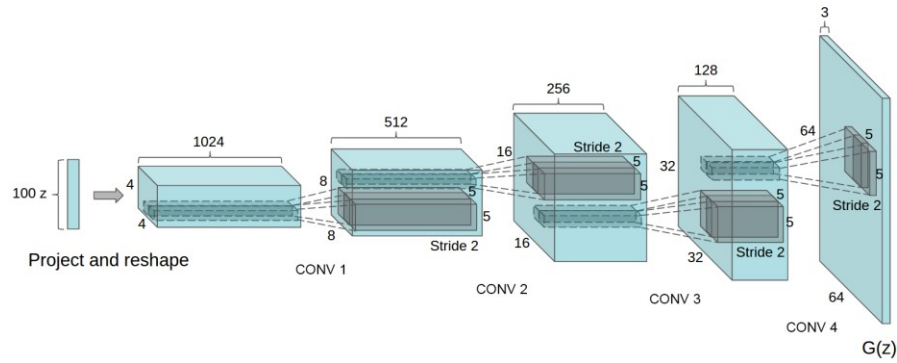
		MNIST	SYN NUMBERS	SVHN	SYN SIGNS
SOURCE					
TARGET					
		MNIST-M	SVHN	MNIST	GTSRB
METHOD	SOURCE	MNIST	SYN NUMBERS	SVHN	SYN SIGNS
	TARGET	MNIST-M	SVHN	MNIST	GTSRB
SOURCE ONLY		.5225	.8674	.5490	.7900
SA (Fernando et al., 2013)		.5690	.8644	.5932	.8165
DANN		<b>.7666</b>	<b>.9109</b>	<b>.7385</b>	<b>.8865</b>
TRAIN ON TARGET		.9596	.9220	.9942	.9980

# « Generative Adversarial Networks » (GAN)

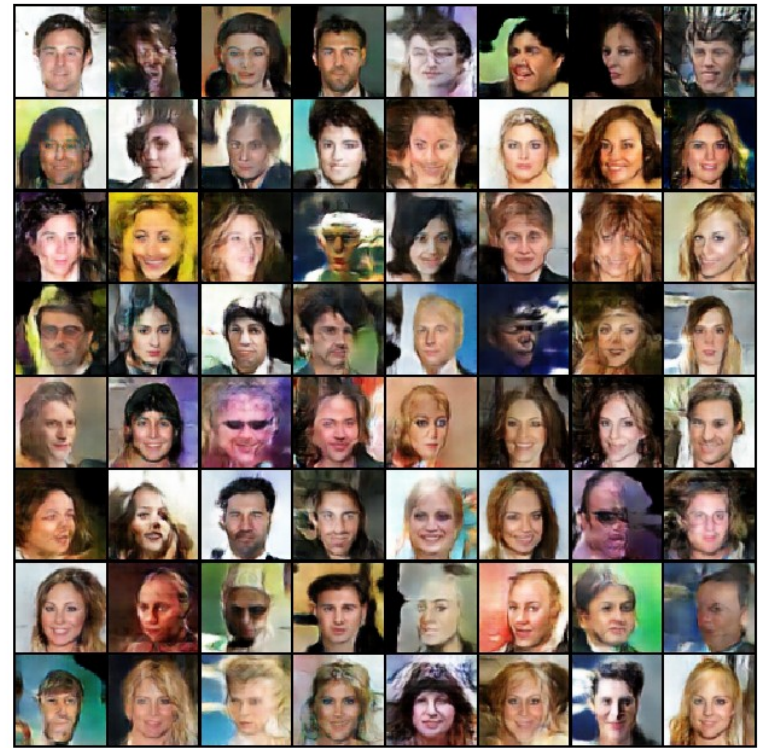


Idée : Minimiser  $L(F(x)) - L(G(h))$

# [https://pytorch.org/tutorials/beginner/dcgan\\_faces\\_tutorial.html](https://pytorch.org/tutorials/beginner/dcgan_faces_tutorial.html)



Training Images



# Réseaux de neurones récurrents

*RNN : Recurent Neural Networks*

# Pourquoi ?

- Traiter des données séquentielles

Image  $X$  vs.  $\{x^{(1)}, x^{(2)}, \dots, x^{(\tau)}\}$

– séries temporelles

- Souvent de longueur *variable*
- L'information pertinente n'est pas toujours située au même endroit

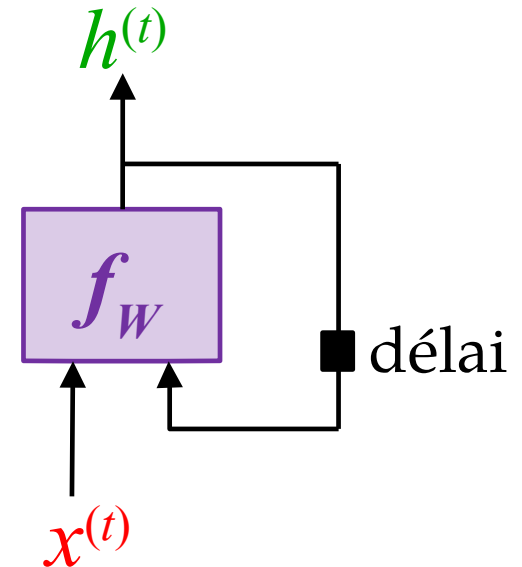
I went to Nepal in 2009

In 2009, I went to Nepal



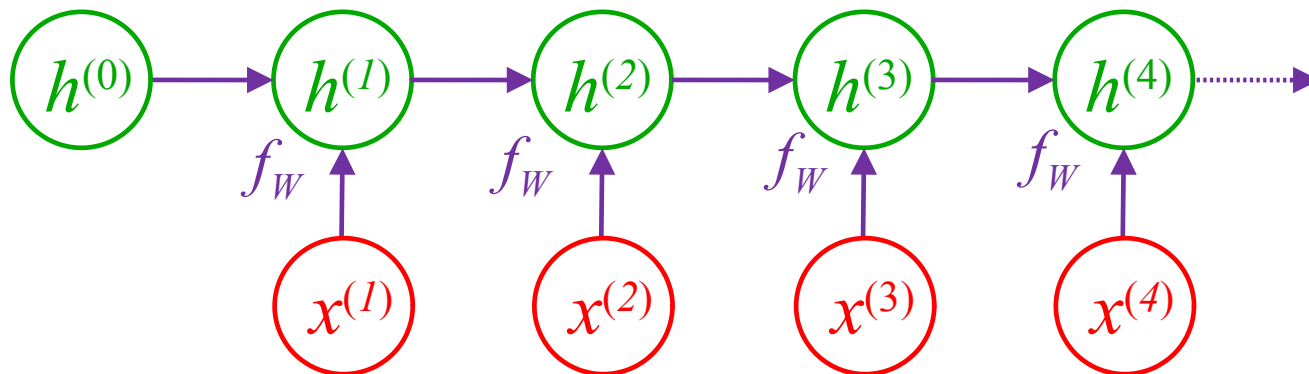
# Idée générale

$$h^{(t)} = f_W(h^{(t-1)}, x^{(t)})$$



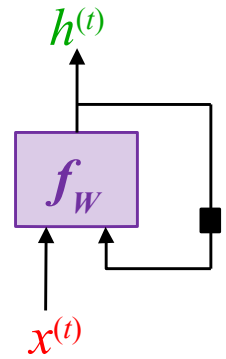
- Relation  $f_W$  stationnaire :

$W$  ne change pas selon  $t$



# Variable cachée $h$

- **Résumé sémantique** de la séquence
- En lien direct avec la tâche :
  - p. e. si on cherche des dates, des mots comme **vendredi** vont influencer  $h$  plus que **Lille**

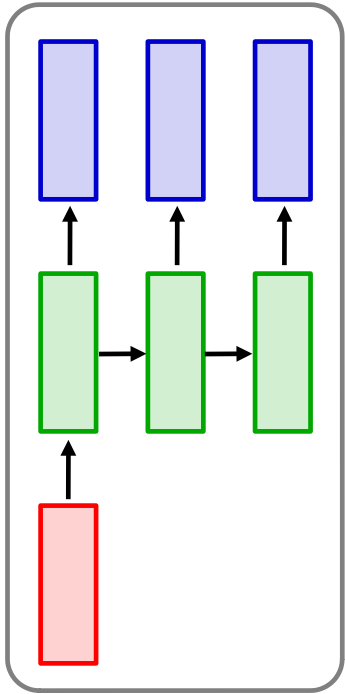


La rétropropagation des gradients fera le travail de trouver la fonction  $f_w$  favorisant cette représentation

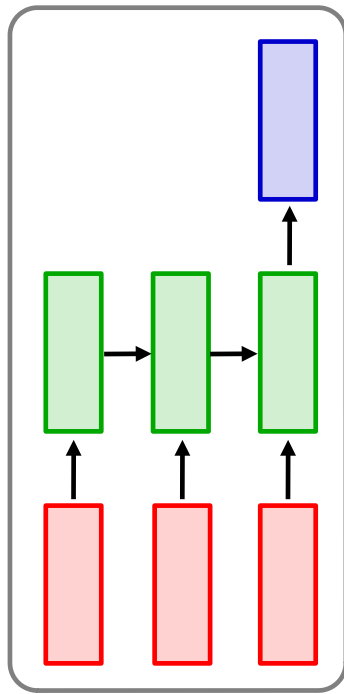
- La taille de  $h$  influencera la quantité d'information pouvant y être emmagasiné
  - pourra difficilement résumer *À la recherche du temps perdu* de M. Proust (4 215 pages)

# Topologies RNN

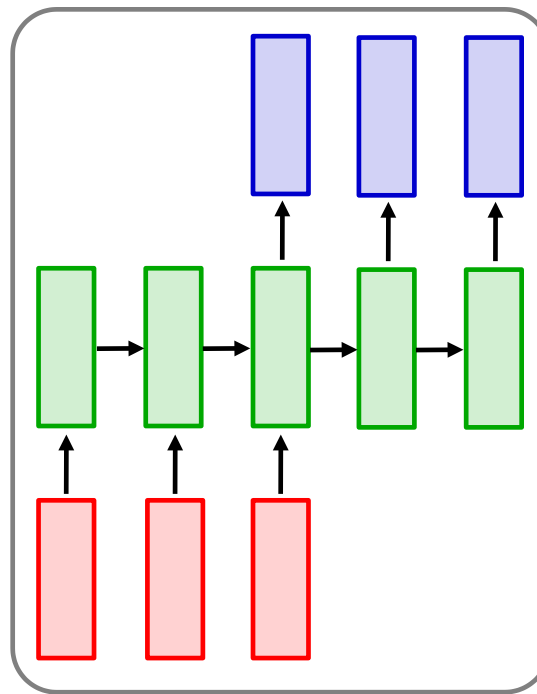
one to many



many to one



many to many



many to many

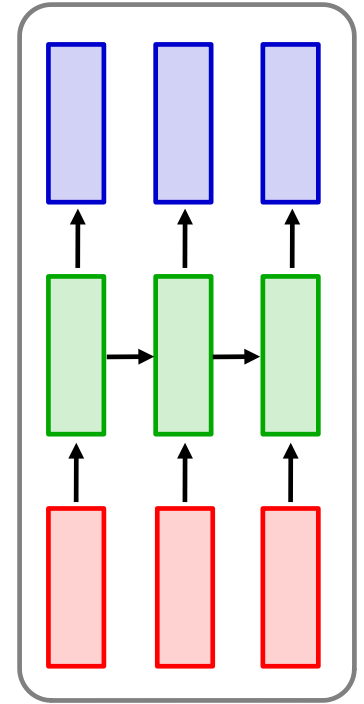


Image  
captioning

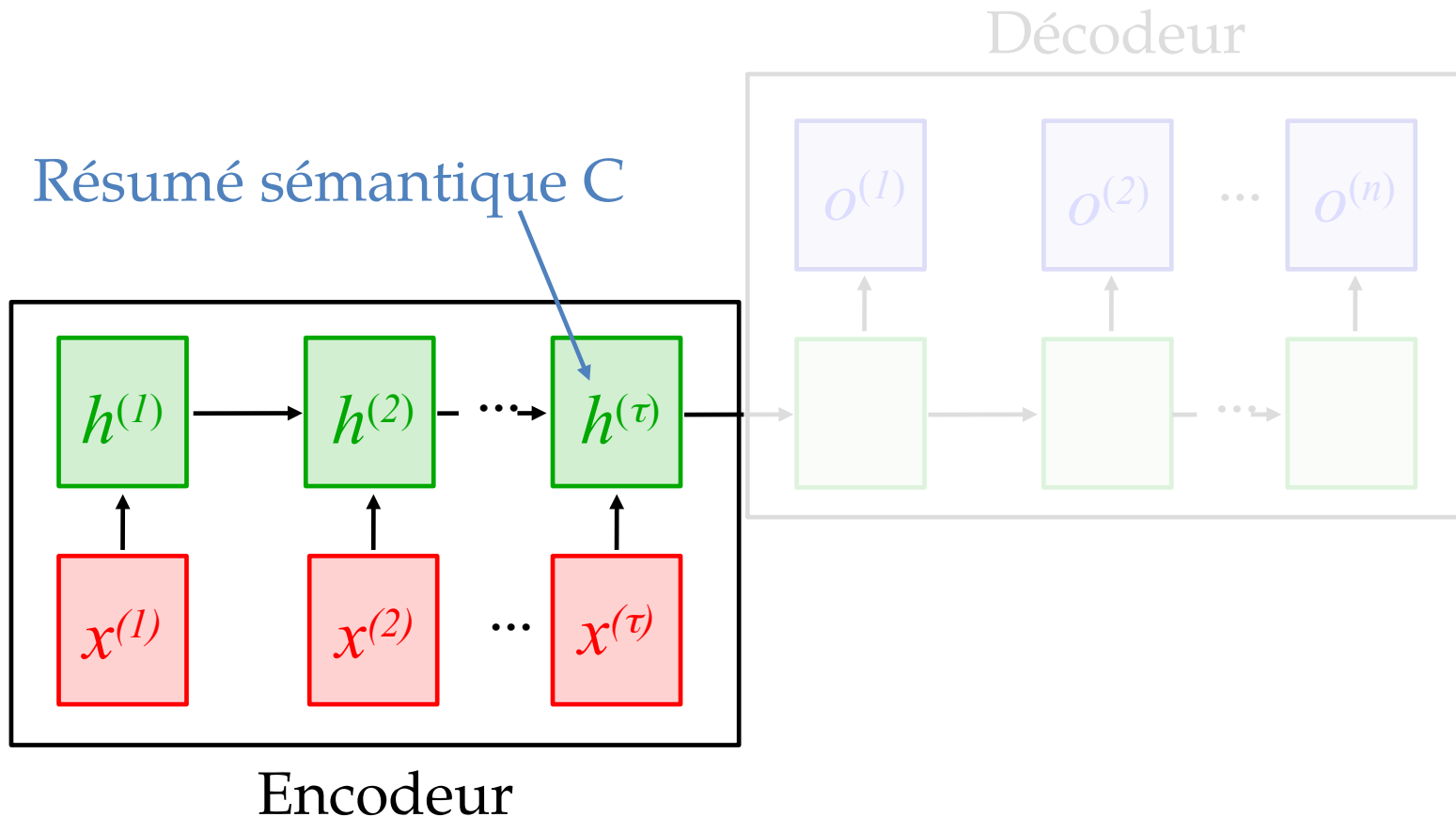
Classification  
de sentiment  
(texte)

Traduction,  
Réponse aux  
questions

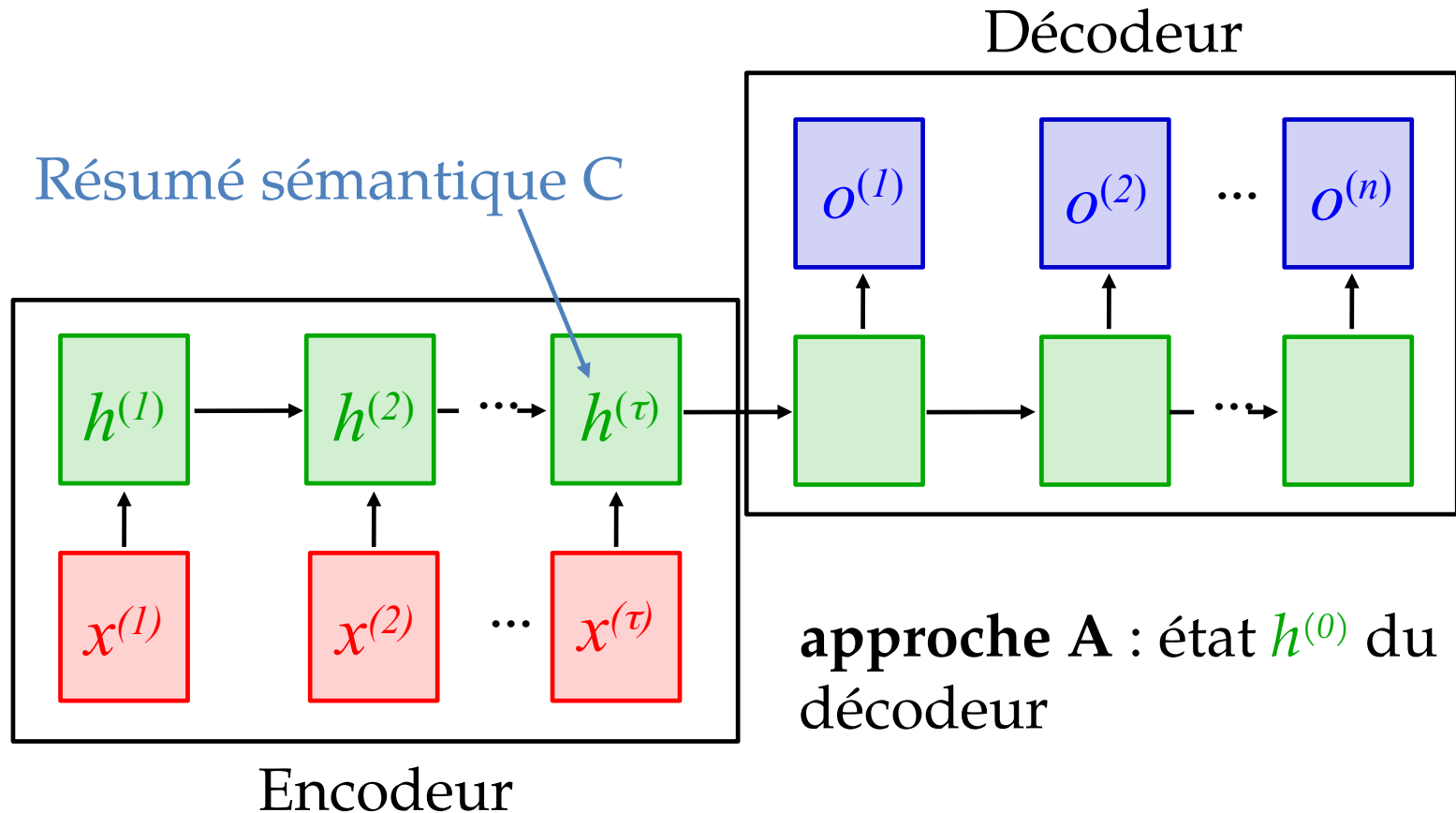
Classification  
de trames  
vidéos

# Sequence-to-sequence

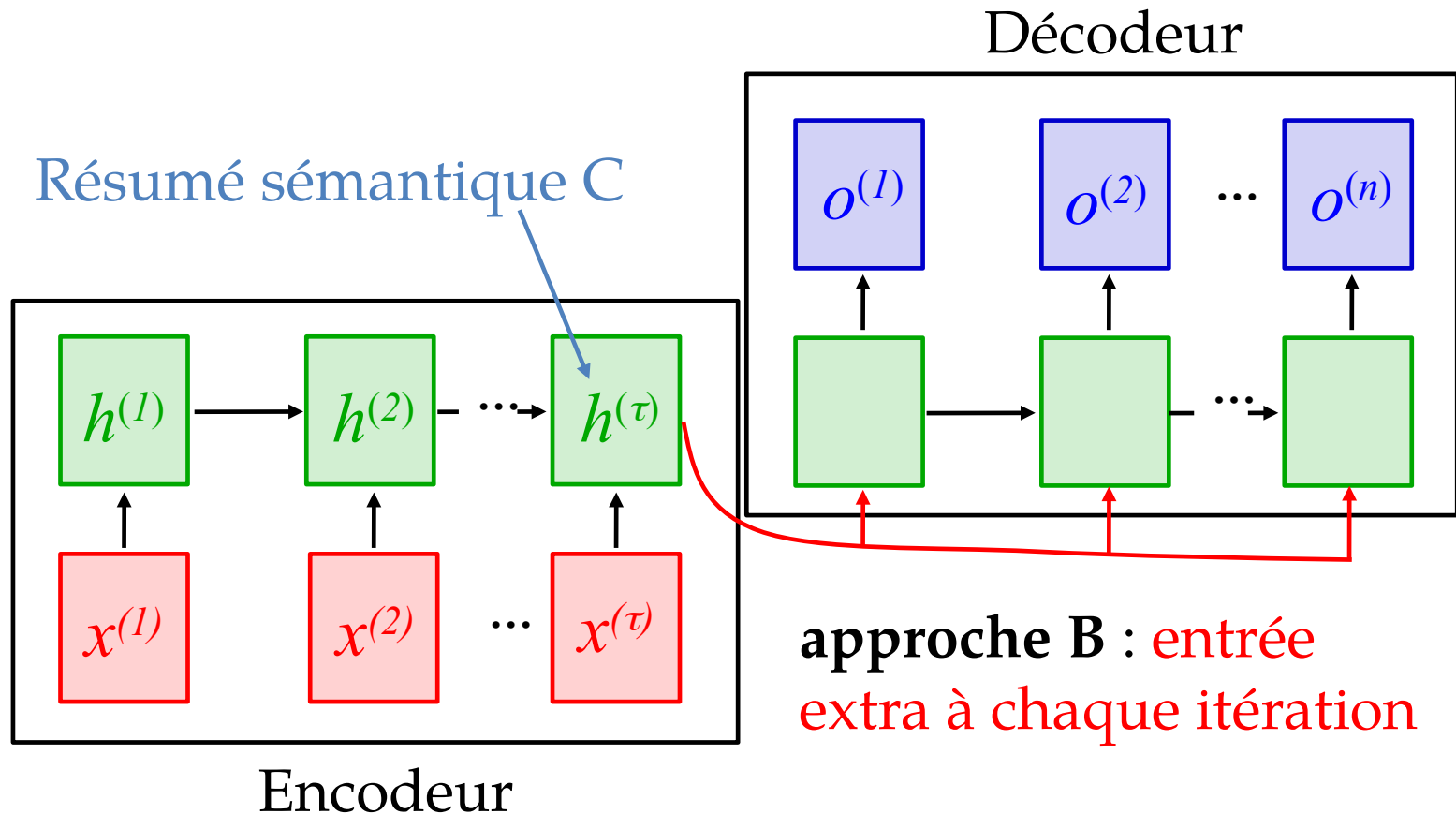
- Architecture *many to many* *contexte*
- Généré une séquence à partir d'un résumé **C**



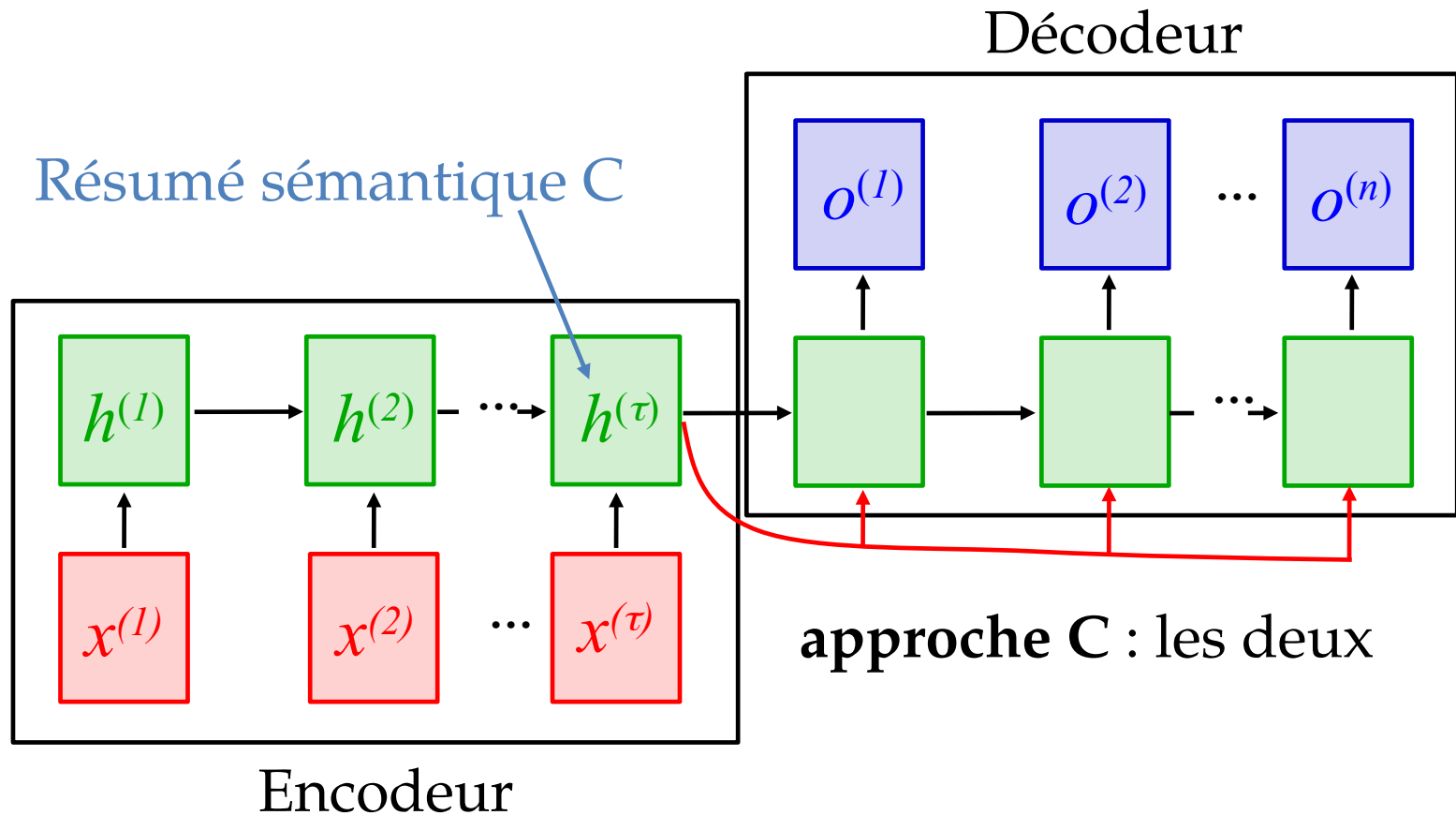
# Sequence-to-sequence



# Sequence-to-sequence



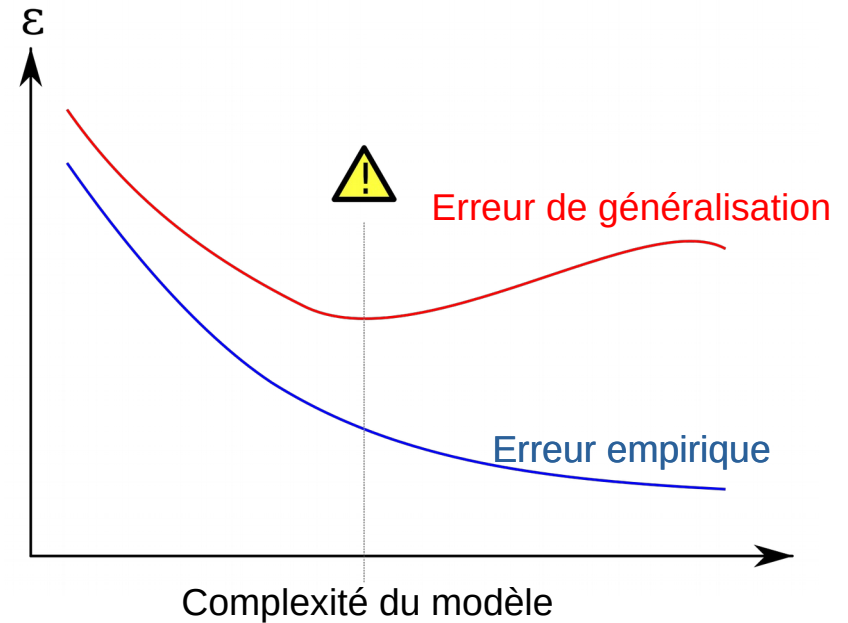
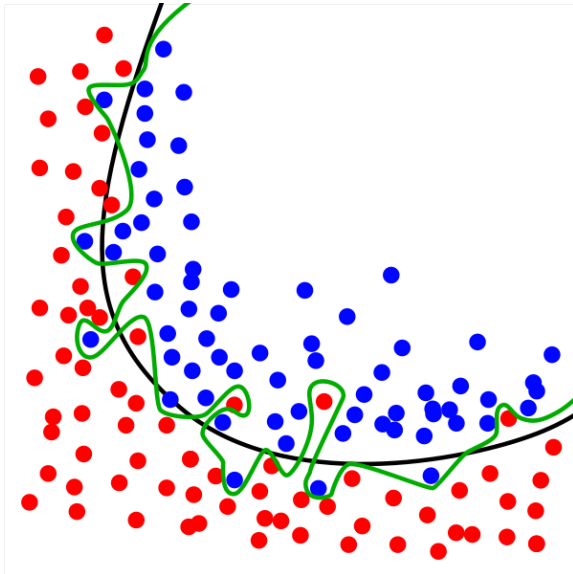
# Sequence-to-sequence



Retour sur les «mystères»  
des réseaux de neurones



# Approche classique: limiter la complexité du modèle pour éviter le sur-apprentissage



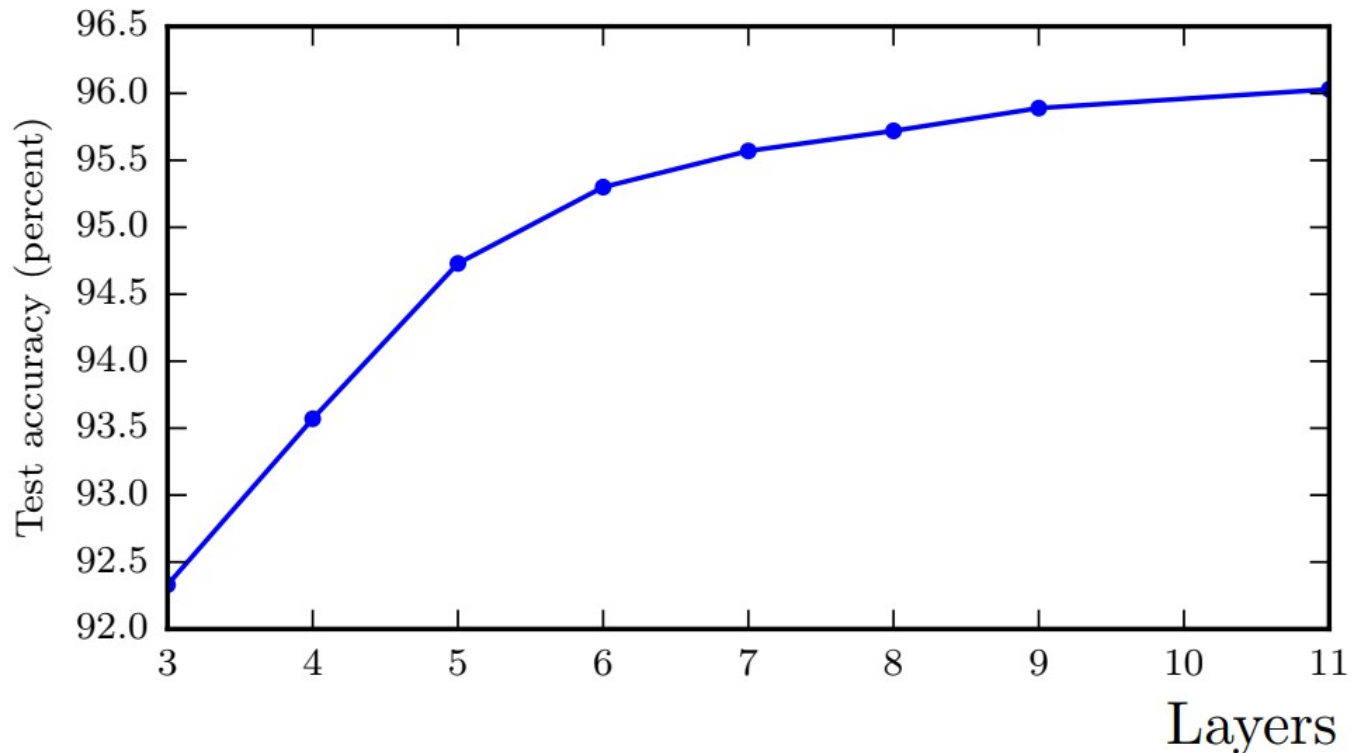
## Théorie statistique de l'apprentissage

$$\text{Erreur de généralisation} \leq \text{Erreur empirique} + \frac{1}{\sqrt{n}} \text{Mesure de complexité}$$

(avec grande probabilité)

Les réseaux de neurones tendent à mieux généraliser avec un grand nombre de couches cachées

- Street View Home Numbers SVHN



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

## UNDERSTANDING DEEP LEARNING REQUIRES RE-THINKING GENERALIZATION

**Chiyuan Zhang\***  
Massachusetts Institute of Technology  
chiyuan@mit.edu

**Samy Bengio**  
Google Brain  
bengio@google.com

**Moritz Hardt**  
Google Brain  
mrtz@google.com

**Benjamin Recht†**  
University of California, Berkeley  
brecht@berkeley.edu

**Oriol Vinyals**  
Google DeepMind  
vinyals@google.com

### ABSTRACT

Despite their massive size, successful deep artificial neural networks can exhibit a remarkably small difference between training and test performance. Conventional wisdom attributes small generalization error either to properties of the model family, or to the regularization techniques used during training.

Through extensive systematic experiments, we show how these traditional approaches fail to explain why large neural networks generalize well in practice. Specifically, our experiments establish that state-of-the-art convolutional networks for image classification trained with stochastic gradient methods easily fit a random labeling of the training data. This phenomenon is qualitatively unaffected by explicit regularization, and occurs even if we replace the true images by completely unstructured random noise. We corroborate these experimental findings with a theoretical construction showing that simple depth two neural networks already have perfect finite sample expressivity as soon as the number of parameters exceeds the number of data points as it usually does in practice.

We interpret our experimental findings by comparison with traditional models.

<https://openreview.net/forum?id=Sy8gdB9xx>

model	# params	random crop	weight decay	train accuracy	test accuracy
Alexnet	1,387,786	yes	yes	99.90	81.22
		yes	no	99.82	79.66
		no	yes	100.0	77.36
		no	no	100.0	76.07
		(fitting random labels)	no	no	99.82
MLP 3x512	1,735,178	no	yes	100.0	53.35
		no	no	100.0	52.39
		(fitting random labels)	no	no	100.0
MLP 1x512	1,209,866	no	yes	99.80	50.39
		no	no	100.0	50.51
		(fitting random labels)	no	no	99.34

# Reconciling modern machine learning practice and the bias-variance trade-off

Mikhail Belkin<sup>a</sup>, Daniel Hsu<sup>b</sup>, Siyuan Ma<sup>a</sup>, and Soumik Mandal<sup>a</sup>

<sup>a</sup>*The Ohio State University, Columbus, OH*

<sup>b</sup>*Columbia University, New York, NY*

September 12, 2019

## Abstract

Breakthroughs in machine learning are rapidly changing science and society, yet our fundamental understanding of this technology has lagged far behind. Indeed, one of the central tenets of the field, the bias-variance trade-off, appears to be at odds with the observed behavior of methods used in the modern machine learning practice. The bias-variance trade-off implies that a model should balance under-fitting and over-fitting: rich enough to express underlying structure in data, simple enough to avoid fitting spurious patterns. However, in the modern practice, very rich models such as neural networks are trained to exactly fit (i.e., interpolate) the data. Classically, such models would be considered over-fit, and yet they often obtain high accuracy on test data. This apparent contradiction has raised questions about the mathematical foundations of machine learning and their relevance to practitioners.

In this paper, we reconcile the classical understanding and the modern practice within a unified performance curve. This “double descent” curve subsumes the textbook U-shaped bias-variance trade-off curve by showing how increasing model capacity beyond the point of interpolation results in improved performance. We provide evidence for the existence and ubiquity of double descent for a wide spectrum of models and datasets, and we posit a mechanism for its emergence. This connection between the performance and the structure of machine learning models delineates the limits of classical analyses, and has implications for both the theory and practice of machine learning.

<https://arxiv.org/abs/1812.11118>

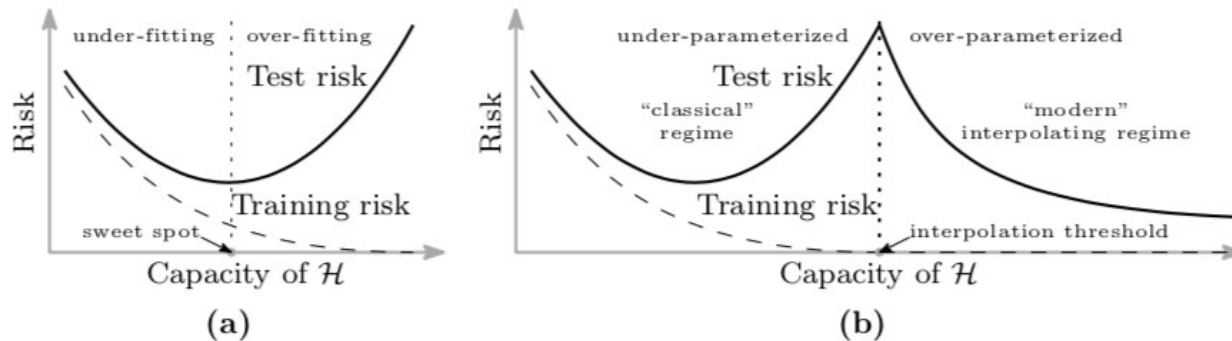


Figure 1: **Curves for training risk (dashed line) and test risk (solid line).** (a) The classical *U-shaped risk curve* arising from the bias-variance trade-off. (b) The *double descent risk curve*, which incorporates the U-shaped risk curve (i.e., the “classical” regime) together with the observed behavior from using high capacity function classes (i.e., the “modern” interpolating regime), separated by the interpolation threshold. The predictors to the right of the interpolation threshold have zero training risk.

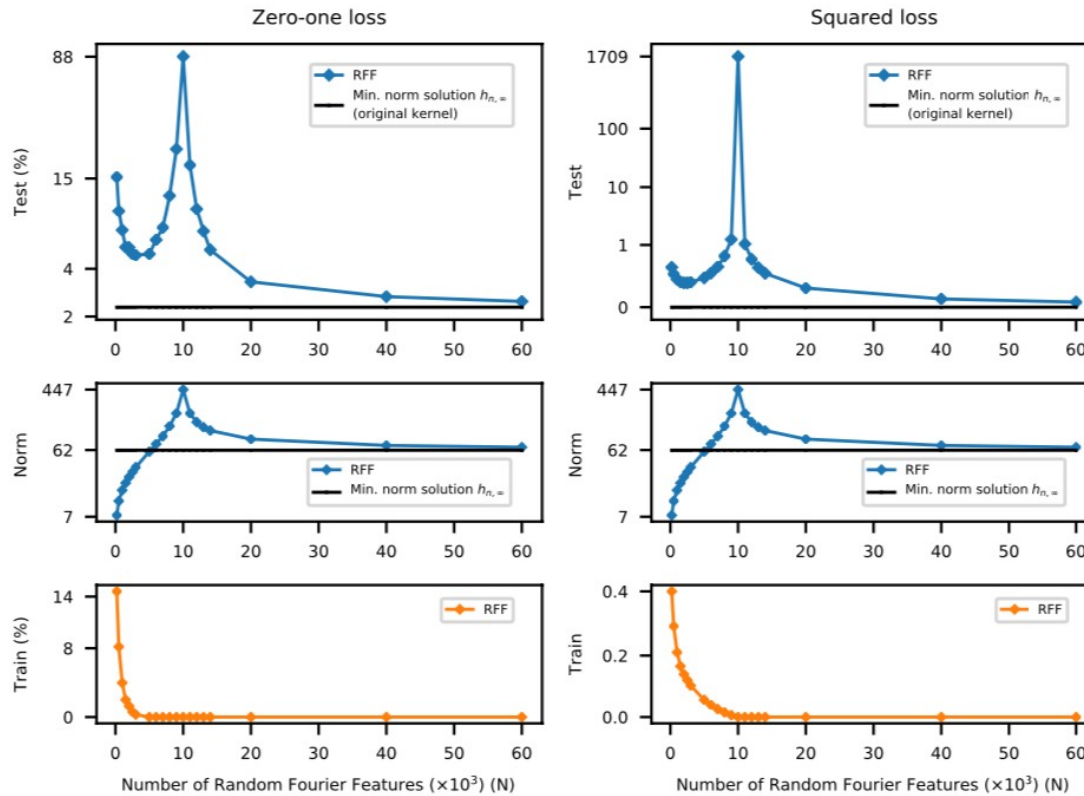


Figure 2: **Double descent risk curve for RFF model on MNIST.** Test risks (log scale), coefficient  $\ell_2$  norms (log scale), and training risks of the RFF model predictors  $h_{n,N}$  learned on a subset of MNIST ( $n = 10^4$ , 10 classes). The interpolation threshold is achieved at  $N = 10^4$ .

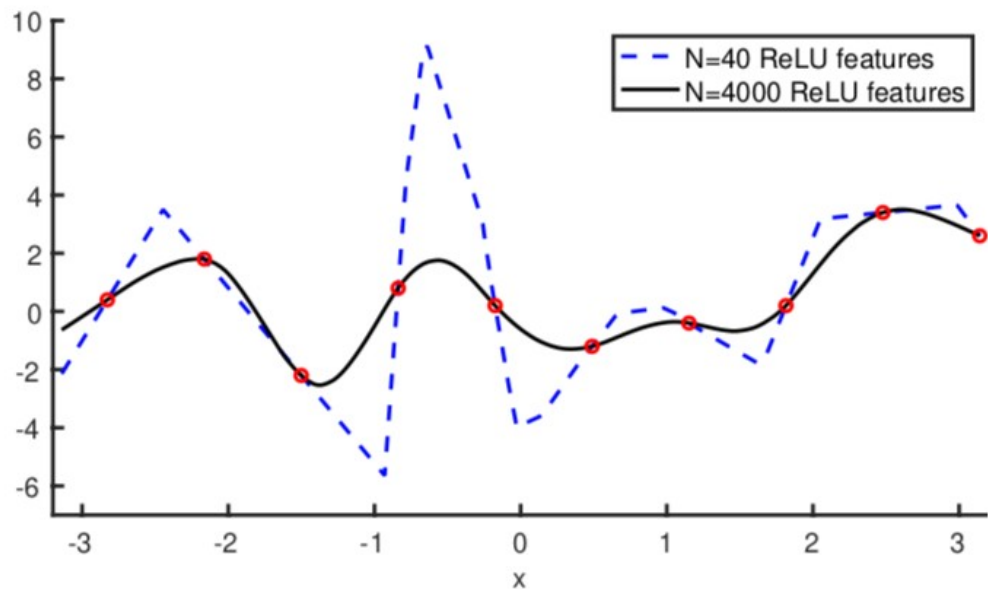


Figure 3: Plot of two univariate functions fitted to 10 data points using Random ReLU features  $\phi(x; (v_1, v_2)) := \max(v_1x + v_2, 0)$ . The data points are shown in red circles. The fitted function with  $N = 40$  Random ReLU features is the blue dashed line; the coefficient vector's norm (scaled by  $\sqrt{N}$ ) is  $\approx 695$ . The fitted function with  $N = 4000$  Random ReLU features is the black solid line; the coefficient vector's norm is  $\approx 159$ .