

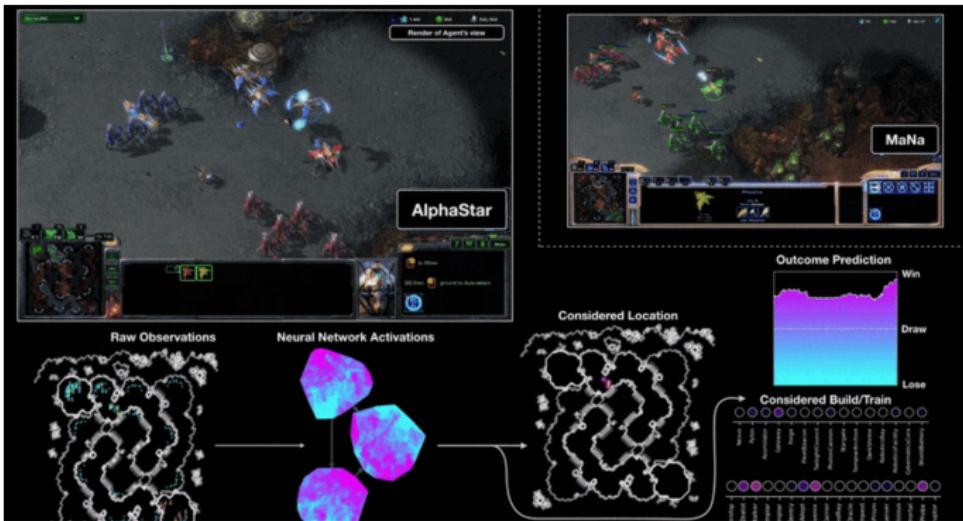
Cours de reconnaissance des formes

Introduction au deep learning - apprentissage profond

Adrien CHAN-HON-TONG  
ONERA

## Introduction

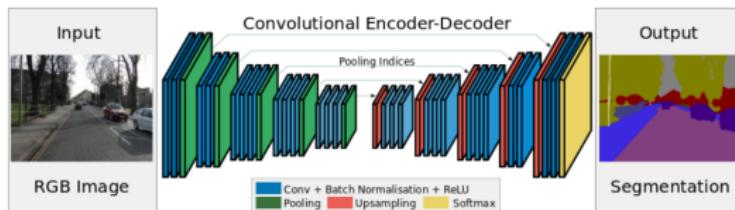
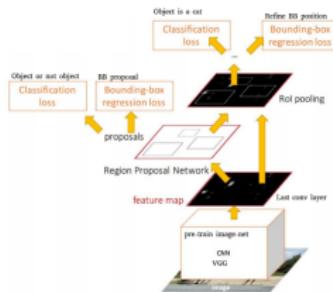
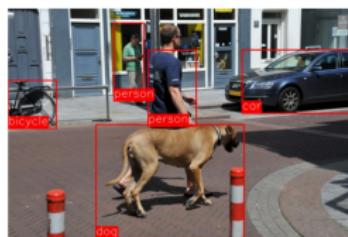
## Deep learning - apprentissage profond



Nous sommes probablement à l'aube d'une révolution industrielle :  
véhicule autonome, diagnostic médicaux, ...

# Introduction

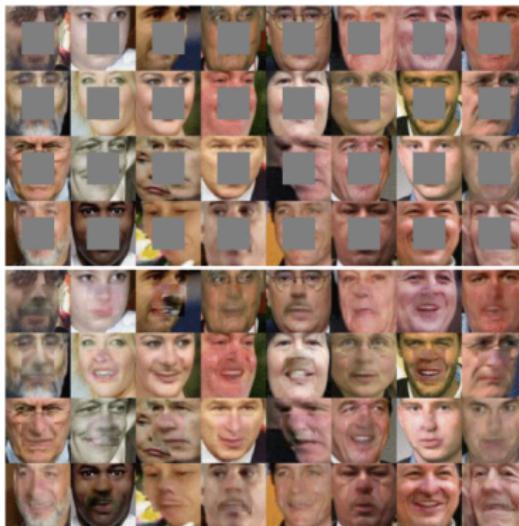
## Application aux voitures autonomes



Les performances de détection/segmentation ont fortement progressé grâce aux architectures d'apprentissage profond.

# Introduction

Et aussi



de la génération automatiques d'images (application aux jeux  
vidéos - au cinéma - à la production de fake)

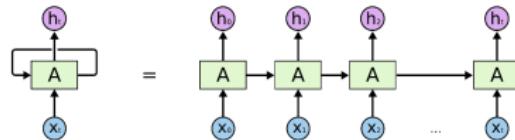
# Introduction

Et aussi



l'apprentissage de stratégie

Et aussi



le traitement de signaux temporelle (langage, son, finance...)

# Introduction

Une révolution industrielle ?

Monteriez vous dans un avion conduit par du deep learning ?

Accepteriez vous d'être opéré par un robot ?

# Plan

- ▶ Comment formaliser le problème étudié ?
- ▶ Comment fonctionnent ces réseaux de neurones ?
- ▶ Quelles différences avec les autres méthodes d'apprentissage ?

## La classification est le problème de base

On voudrait un système  $f$  qui prend une observation  $x \in \mathbb{R}^D$

En déduit la classe de cette observation  $f(x) \in \{-1, 1\}$

(généralement  $f$  est continue mais la décision est associée au signe de  $f(x)$ )

Exemple : à partir d'une image  $x \in \mathbb{R}^D$ , je voudrais savoir si c'est ou pas une image de chat  $f(x) \in \{-1, 1\}$ .

On peut sur un certain nombre d'exemples, vérifier si  $f(x)$  est bien égale à la valeur voulue  $y(x)$ .



chat

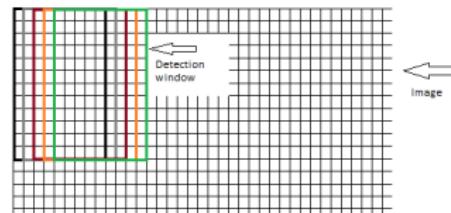
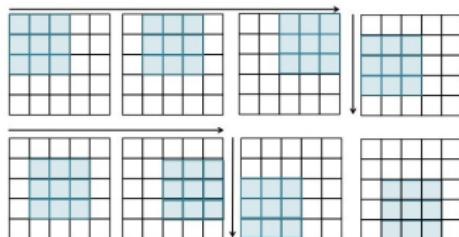


non chat

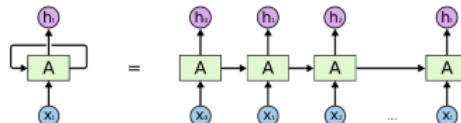


non chat

# La classification est le problème de base



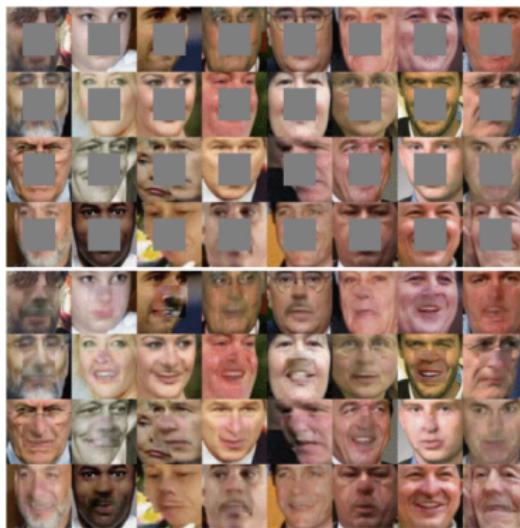
Si on sait classer, on sait détecter et segmenter



On peut dupliquer virtuellement le bloc de base, et, le problème n'est plus que de la classification

# La classification est le problème de base

1 réseau qui génère des images



1 réseau qui classe les images en vrai/fausse  
les 2 appris pour des objectifs opposées mais en forçant le  
discriminateur à aider le générateur

# Classification supervisée

- ▶ prend en entrée une **base d'apprentissage** : un ensemble d'observations étiquetées  $x_1, y(x_1), \dots, x_N, y(x_N)$  par la fonction désirée  $y$  (à valeur dans  $\{-1, 1\}$ ).
- ▶ produit un modèle  $f$  : une fonction qui, à  $x$ , associe une classe  $f(x) \in \{-1, 1\}$
- ▶ ce modèle est évalué sur une **base de test** disjointe de la base d'apprentissage  $\chi_1, y(\chi_1), \dots, \chi_N, y(\chi_N)$  en calculant :

$$\frac{1}{N} \sum_{n=1}^N |f(\chi_i) - y(\chi_i)| \approx \int_{\mathbb{R}^D} |f(x) - y(x)| P(x) dx = E_P[|f - y|]$$

( $E$  désigne l'espérance mathématique)

# Plan

- ▶ Comment formaliser le problème étudié ?  
⇒ **la classification**
- ▶ Comment fonctionnent ces réseaux de neurones ?
- ▶ Quelles différences avec les autres méthodes d'apprentissage ?

# Du neurone au réseau

## Le neurone

Dans les architectures de réseaux de neurones (profond ou pas), le neurone est un filtre linéaire couplé à une activation non linéaire par exemple  $\max(0, x)$  :

$$\text{neurone}_{w,b} : \begin{array}{ccc} \mathbb{R}^D & \rightarrow & \mathbb{R} \\ x & \rightarrow & \max(0, wx + b) \end{array}$$

$w \in \mathbb{R}^D$  et  $b \in \mathbb{R}$  sont les **poids** du neurones.

$\max(0, x)$  est appelé **relu** - parfois noté  $[x]_+$

**Attention, pas d'activation sur le tout dernier neurone !**

# Du neurone au réseau

## La couche de neurone

Une couche de  $K$  de neurones est une séquence de  $K$  neurones prenant la même entrée, et, dont les  $K$  sorties sont regroupées en 1 vecteur :

$$couche_{W,b} : \begin{array}{ccc} \mathbb{R}^D & \rightarrow & \mathbb{R}^K \\ x & \rightarrow & \begin{pmatrix} neurone_{W_1, b_1}(x) \\ \dots \\ neurone_{W_K, b_K}(x) \end{pmatrix} \end{array}$$

$W \in \mathbb{R}^{K \times D}$  et  $b \in \mathbb{R}^K$  sont les **poids** de chacun des  $K$  neurones.

# Du neurone au réseau

## Le réseau de neurone

Un réseau de neurones entièrement connectées (multi layer perceptron en anglais) de profondeur  $Q$  est un empilement de  $Q$  couche de neurones - la dernière est classiquement un seul neurone (sans activation) :

$$\text{reseau}_{\theta} : \begin{array}{ccc} \mathbb{R}^D & \rightarrow & \mathbb{R} \\ x & \rightarrow & C_{W_Q, b_Q}(C_{W_{Q-1}, b_{Q-1}}(\dots(C_{W_1, b_1}(x))\dots)) \end{array}$$

$\theta = (W_1, b_1, \dots, W_Q, b_Q)$  est un  $Q$  uplets de **poids** de couches de neurones - abrégé  $C$

# Du neurone au réseau

## le dernier neurone

L'objectif du réseau est de classer l'entrée (-1 ou 1).

Pour cela, la classe prédite est le signe de la valeur du dernier neurones (sans le relu). Si le dernier neurone a une valeur négative, on dira que le réseau choisit la classe -1, et inversement, si la valeur est positive on dira que le réseau prédit la classe 1.

Remarque :

$$\text{signe}(f(x)) = y(x) \Leftrightarrow y(x)\text{signe}(f(x)) = 1 \Leftrightarrow y(x)f(x) > 0$$

*On parle ici de classification binaire, sinon, on a généralement 1 neurone par classe exprimant le score de la classe.*

# Apprentissage profond

Quand le réseau a beaucoup de couche, on parle de réseau profond.

Ce terme est apparu en 2012 avec un réseau à 10 couches dit Alexnet (*imagenet classification with deep convolutional neural networks*).

Les réseaux de neurones existent depuis 1950 mais la puissance de calcul n'était pas suffisante pour envisager des réseaux de neurones profonds.

# Plan

- ▶ Comment formaliser le problème étudié ?  
⇒ **la classification**
- ▶ Comment fonctionnent ces réseaux de neurones ?  
⇒ **neurone / couche / réseau**  
⇒ seulement ?
- ▶ Quelles différences avec les autres méthodes d'apprentissage ?

# Classification par apprentissage profond - mais de quoi ?

Est ce qu'une image est  $x \in \mathbb{R}^D$  ?

1 image RGB de 480x340 = 1 vecteur dans  $[0, 255]^{163200}$

Différence image vecteur

- ▶ Le traitement d'images repose sur les opérations de **convolution** et **pooling** pour passer d'une image à un vecteur.
- ▶ le traitement se termine par une classification de vecteurs.

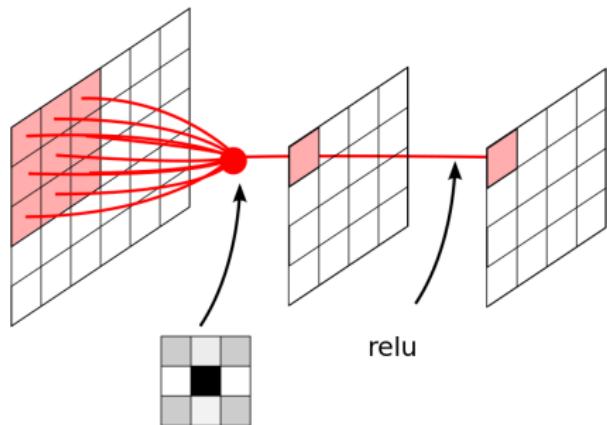
# Convolution

- ▶ l'entrée est  $x \in \mathbb{R}^{H \times W \times Ch}$
- ▶ les poids du neurones sont simplement  $w \in \mathbb{R}^{(2\delta_H+1) \times (2\delta_W+1) \times Ch}$  (et  $b$ )
- ▶ la sortie est une image  $\chi \in \mathbb{R}^{H \times W}$
- ▶ avec  $\chi_{h,w}$  égal (avant activation) à :

$$\sum_{dh,dw,ch \in [-\delta_H,\delta_H] \times [-\delta_W,\delta_W] \times [1,Ch]} x_{h+dh,w+dw,ch} \times w_{dh,dw,ch}$$

La convolution est bien un filtre linéaire mais un filtre linéaire local prenant en compte l'aspect spatial des images, et s'applique identiquement en chaque pixel. Elle permet d'extraire de l'image des informations locales (contrastes, texture, bords...) avec un nombre de poids réduit.

# Convolution



# Convolution

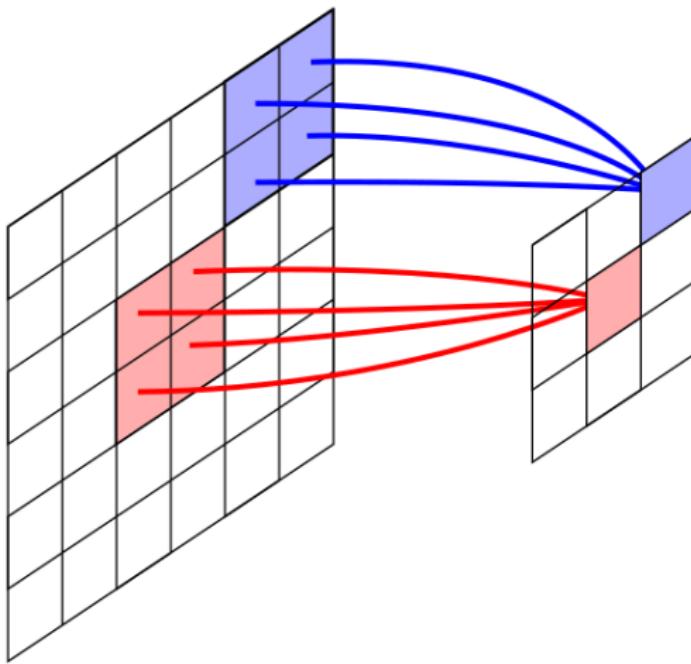
$$conv_{w,b}(x) = \left( b + \sum_{\substack{dh \in [-\delta_H, \delta_H] \\ dw \in [-\delta_W, \delta_W] \\ ch \in [1, C_h]}} x_{h+dh, w+dw, ch} \times w_{dh, dw, ch} \right)_+ \quad \begin{matrix} h \in [1, H] \\ w \in [1, W] \end{matrix}$$

# Pooling

- ▶ l'entrée est  $x \in \mathbb{R}^{H \times W \times Ch}$
- ▶ la sortie est une image  $y \in \mathbb{R}^{\frac{H}{2} \times \frac{W}{2} \times Ch}$
- ▶ avec  $\chi_{h,w,ch} = \max_{dh,dw \in [0,1]} x_{2h+dh, 2w+dw, ch}$

Le pooling n'est pas vraiment une couche de neurones c'est plutôt un autre type d'activation, mais, avec une dimension spatiale. Elle permet d'introduire de l'invariance aux petites déformations.

# Pooling



## Convolution, Pooling, Stride

Classiquement : la convolution produit une sortie qui a la même taille que l'entrée (au effet de bord près) alors que le pooling produit une sortie 2 fois plus petite.

Cependant, la bonne notion est la notion de **stride** : la convolution a classiquement une stride de 1 car on déplace la fenêtre de 1 case alors que le pooling a une stride de 2 on déplace la fenêtre de 2 cases.

**Rien n'interdit d'avoir des convolutions avec des strides supérieurs à 1 ou des pooling avec des strides de 1.**

# Convolution, Pooling, Stride, Padding

Dans la convolution, il y a un donc un effet de bord pour  $h < \delta_H \dots$ . Une solution à ce problème consiste à considérer au  $x = 0$  en dehors de l'emprise initiale. Dans ce cas, la sortie a exactement la même taille que l'entrée.

Informatiquement, cela revient à travailler sur une image où on a rajouter  $\delta_H$  lignes et  $\delta_W$  colonnes de part et d'autre.

au final : si l'entrée est  $c \times h \times w$  elle devient

$c' \times \frac{h+2*pad_h-kernel_h}{stride_h} + 1 \times \frac{w+2*pad_w-kernel_w}{stride_w} + 1$  en passant dans une convolution  $c' \times c \times kernel_h \times kernel_w$  avec un padding de  $pad_h, pad_w$  et un stride de  $stride_h, stride_w$ .

# Plan

- ▶ Comment formaliser le problème étudié ?  
⇒ **la classification**
- ▶ Comment fonctionnent ces réseaux de neurones ?  
⇒ **neurone / couche / réseau + convolution / pooling**
- ▶ Quelles différences avec les autres méthodes d'apprentissage ?

# Quelles différences avec les autres méthodes d'apprentissage ?

Aucune

Précisément, aucune en théorie - parce qu'il y en a d'innombrable en pratique - à commencer par les niveaux de performance qu'on arrive à atteindre !

# Plan

- ▶ Comment formaliser le problème étudié ?  
⇒ **la classification**
- ▶ Comment fonctionnent ces réseaux de neurones ?  
⇒ **neurone / couche / réseau + convolution / pooling**
- ▶ Quelles différences avec les autres méthodes d'apprentissage ?  
⇒ conceptuellement aucune

## Quelles différences avec les autres méthodes d'apprentissage ?

Les performances de ces algorithmes sont hautes, et, laissent envisager des applications critiques.

De nombreuses personnes refuseraient d'embarquer dans un avion sans pilote. Pourtant, les accidents d'avions n'ont pas attendu l'arrivée du deep learning !

## Petit rappel - apprentissage par ordinateur

On voudrait un système qui :

- ▶ prend une observation  $x \in \mathbb{R}^D$
- ▶ propose une classe pour cette observation  $f(x) \in \{-1, 1\}$
- ▶ la vraie classe est  $y(x)$  (mais on ne connaît pas  $y$  on peut l'estimer que sur un ensemble de points)



1



-1



-1

## Petit rappel - apprentissage par ordinateur

Mais, on ne sait **pas** définir formellement le lien entre  $x$  et  $y$ .

On utilise des **exemples** pour définir le concept sous jacent.

Sachant que cette *définition* restera nécessairement **ambigüe**.

# Apprentissage par ordinateur

## FAQ

- ▶ est ce qu'on peut être sur que  $f = y$  ?
- ▶ y a-t-il des garanties théoriques ?
- ▶ comment sont évalués ces algorithmes en pratique ?

# Apprentissage par ordinateur



1



-1



-1



?



?



?



?

# Apprentissage par ordinateur



1



-1



-1

Un détecteur de chat ?



1



-1



-1



-1

# Apprentissage par ordinateur



1



-1



-1

## Un détecteur d'animaux



1



1



-1



-1

# Apprentissage par ordinateur



1



-1



-1

Un détecteur de chat roux ?



-1



-1



-1



-1

# Apprentissage par ordinateur



1



-1



-1

Un détecteur de choses rassurantes ?



1



-1



1



-1

# Apprentissage par ordinateur

est ce qu'on peut être sur que  $f = y$  ?

- ▶ Si on peut être sur que  $f = y$ , c'est qu'on connaît  $y$ . Dans ce cas il était **inutile** d'utiliser de l'apprentissage !
- ▶ l'apprentissage est la **seule** méthode qui permet de traiter des problèmes non formalisées
- ▶ en contre partie, même si un oracle donne la solution  $y$ , il n'y a aucun moyen de tester que c'est bien la solution.
- ▶ a fortiori, on ne peut et on ne pourra jamais être sur que  $f = y$

# Apprentissage par ordinateur

## FAQ

- ▶ est ce qu'on peut être sur que  $f = y$ ?  
⇒ **non mais** c'était déjà comment ça avant
- ▶ y a-t-il des garanties théoriques ?
- ▶ comment sont évalués ces algorithmes en pratique ?

# Propriétés des réseaux de neurones

## Universalité

- ▶ Toute fonction continue sur une compact peut être approchée arbitrairement près par un réseau de neurones à 2 couches - mais le nombre de neurone sur la couche base est arbitrairement grand
- ▶ Les réseaux de neurones à  $L$  couches et  $n$  neurones (en tout) approchent les fonctions tempérées avec une erreur inférieure à  $\frac{\lambda}{n^{\alpha(L)}}$  avec  $\lambda$  une constante et  $\alpha$  une fonction (pour ceux que ça intéresse : <https://hal.inria.fr/hal-02117139/file/main.pdf>)

# Propriétés des réseaux de neurones

## dimension VC

- ▶ La dimension VC est une mesure la complexité d'un modèle : plus elle est basse, plus le modèle a *des chances* d'avoir le même taux d'erreurs entre deux bases de données ( $\Rightarrow$  cf cours sur le SVM - Stéphane Herbin)
- ▶ La dimension VC d'un réseau de neurones est équivalente à  $S \log(S)L$  avec  $S$  le nombre total de paramètre et  $L$  la profondeur (pour ceux que ça intéresse : <http://proceedings.mlr.press/v65/harvey17a.html>)

# Apprentissage par ordinateur

## FAQ

- ▶ est ce qu'on peut être sur que  $f = y$  ?  
⇒ **non mais** c'était déjà comment ça avant
- ▶ y a-t-il des garanties théoriques ?  
⇒ **oui mais** ce sont des garanties probabilistes : elles ne garantissent rien avec probabilité 1
- ▶ comment sont évalués ces algorithmes en pratique ?

# Classification supervisée

- ▶ prend en entrée une **base d'apprentissage** : un ensemble d'observations étiquetées  $x_1, y(x_1), \dots, x_N, y(x_N)$  par la fonction désirée  $y$  (à valeur dans  $\{-1, 1\}$ ).
- ▶ produit un modèle  $f$  : une fonction qui, à  $x$ , associe une classe  $f(x) \in \{-1, 1\}$
- ▶ ce modèle est évalué sur une **base de test** disjointe de la base d'apprentissage  $\chi_1, y(\chi_1), \dots, \chi_N, y(\chi_N)$  en calculant :

$$\frac{1}{N} \sum_{n=1}^N |f(\chi_i) - y(\chi_i)| \approx \int_{\mathbb{R}^D} |f(x) - y(x)| P(x) dx = E_P[|f - y|]$$

# Apprentissage par ordinateur

## FAQ

- ▶ est ce qu'on peut être sur que  $f = y$ ?  
⇒ **non mais** c'était déjà comment ça avant
- ▶ y a-t-il des garanties théoriques?  
⇒ **oui mais** ce sont des garanties probabilistes : elles ne garantissent rien avec probabilité 1
- ▶ comment sont évalués ces algorithmes en pratique?  
⇒ **empiriquement** comme à peu près tout!

# Conclusion

- ▶ Comment formaliser le problème étudié ?  
⇒ **la classification**
- ▶ Comment fonctionnent ces réseaux de neurones ?  
⇒ **neurone / couche / réseau + convolution / pooling**
- ▶ Quelles différences avec les autres méthodes d'apprentissage ?  
⇒ **aucune** sauf que les performances hautes de l'apprentissage profond permettent d'envisager des applications critiques.
- ▶ est ce qu'on peut être sur que  $f = y$  ?  
⇒ **non mais** c'était déjà comment ça avant
- ▶ y a-t-il des garanties théoriques ?  
⇒ **oui mais** ce sont des garanties probabilistes : elles ne garantissent rien avec probabilité 1
- ▶ comment sont évalués ces algorithmes en pratique ?  
⇒ **empiriquement** comme à peu près tout !

# Conclusion

Une révolution industrielle ?

Monteriez vous dans un avion conduit par du deep learning ?

Accepteriez vous d'être opéré par un robot ?

# Programme

## TP

- ▶ Exercices théoriques sur les réseaux de neurones
- ▶ Exercices machine autour de la convolution et du pooling

## La prochaine fois

On ouvre le capot : comment optimise-t-on les poids ?

## quelques liens

[towardsdatascience.com/multi-layer-neural-networks-with-sigmoid-function-deep-learning-for-rookies-2-bf464f09eb7f](https://towardsdatascience.com/multi-layer-neural-networks-with-sigmoid-function-deep-learning-for-rookies-2-bf464f09eb7f)

[towardsdatascience.com/deciding-optimal-filter-size-for-cnns-d6f7b56f9363](https://towardsdatascience.com/deciding-optimal-filter-size-for-cnns-d6f7b56f9363)

[cs231n.github.io/neural-networks-1/](https://cs231n.github.io/neural-networks-1/)