

# Introduction à l'apprentissage profond

Résultats fondamentaux et applications à l'imagerie médicale

Ayoub Ajarra

14 Mar 2024



**Ce n'est pas parce qu'il y a un problème qu'il s'agit d'un problème d'apprentissage**

Compte tenu d'une certaine quantité de **données**, comment comprendre le processus général de conception de solutions algorithmiques pour prendre des décisions **précises** avec une grande **confiance**.

## Modéliser des phénomènes complexes du monde réel

Langage naturel, images, ADN, écosystèmes, changement climatique ...

## Le cerveau humain

Une preuve que l'apprentissage profond peut être une solution ?

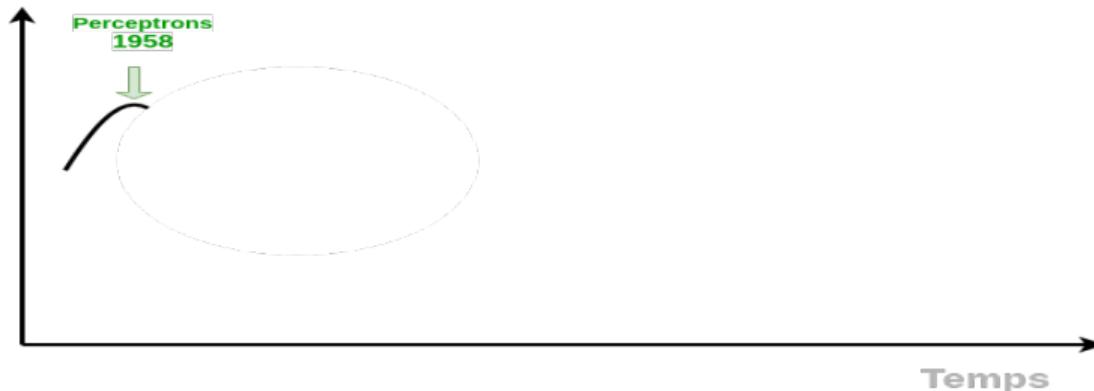
- Introduction
- Principes fondamentaux de l'apprentissage (automatique)
- Réseaux de neurones artificiels
- Surapprentissage
- Réseaux de neurones récurrent
- Réseaux de neurones convolutifs
- Applications à l'imagerie médicale
- **NB:** il ne s'agit pas d'une analyse théorique des algorithmes d'apprentissage profond, ni de leur aspect pratique.

# Introduction

*Aux origines de l'apprentissage profond*



Enthusiasme



## Perceptron

Incapacité d'un perceptron à une couche de résoudre des problèmes simples comme la fonction XOR.

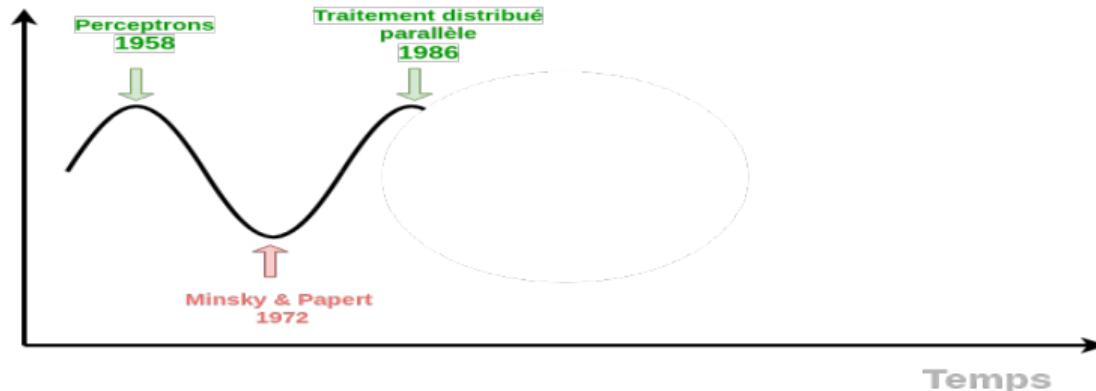
Enthusiasme



## Limites du perceptron

Incapacité d'un perceptron à une couche de résoudre des problèmes simples comme la fonction XOR.

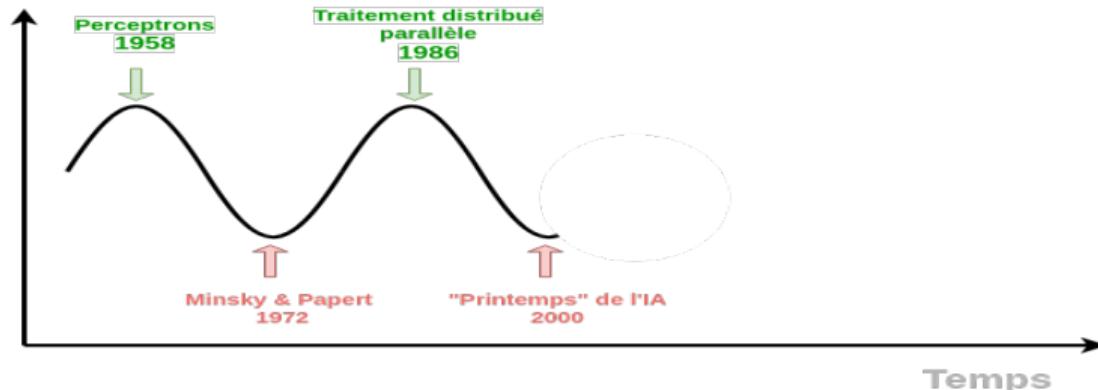
Enthusiasme



## Traitement distribué parallèle

Incapacité d'un perceptron à une couche de résoudre des problèmes simples comme la fonction XOR.

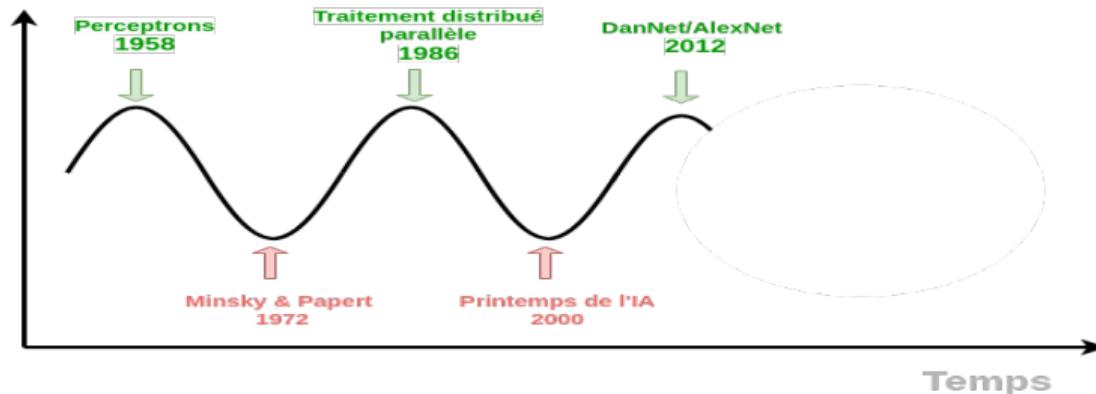
Enthusiasme



## Le printemps d'IA 2000

Incapacité d'un perceptron à une couche de résoudre des problèmes simples comme la fonction XOR.

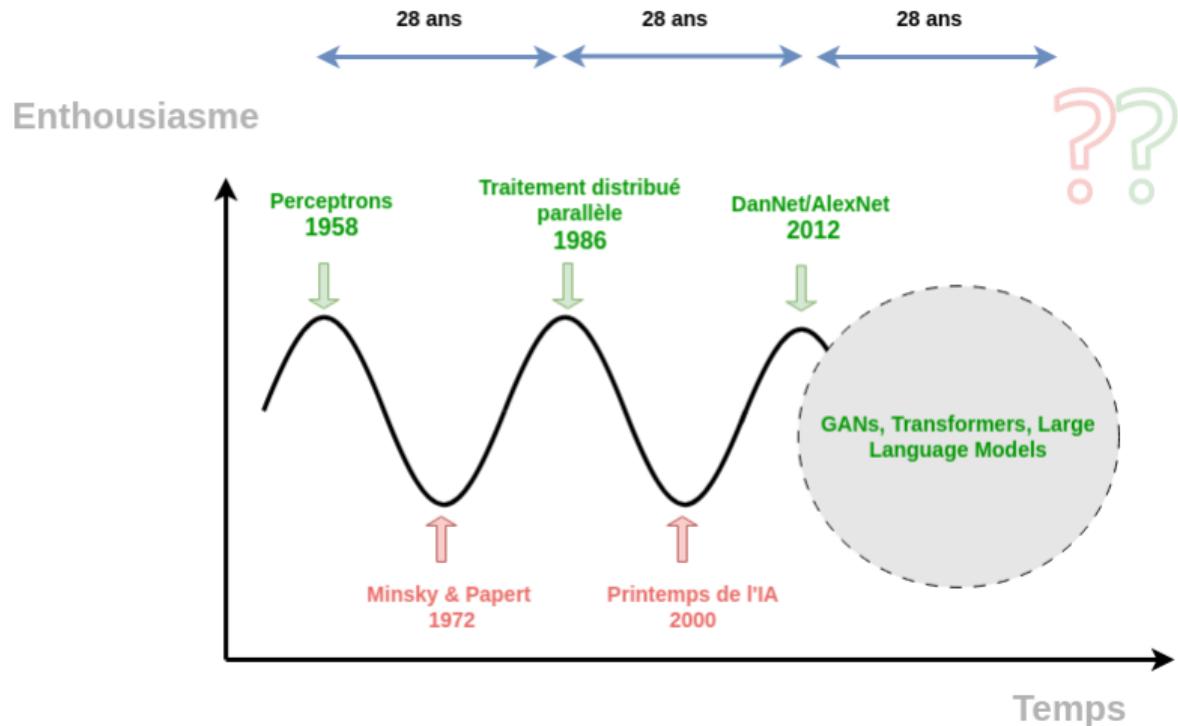
## Enthousiasme



## DanNet/AlexNet

Premiers modèles à l'état de l'art en reconnaissance d'image.

# L'apprentissage automatique: la révolution des machines pensantes

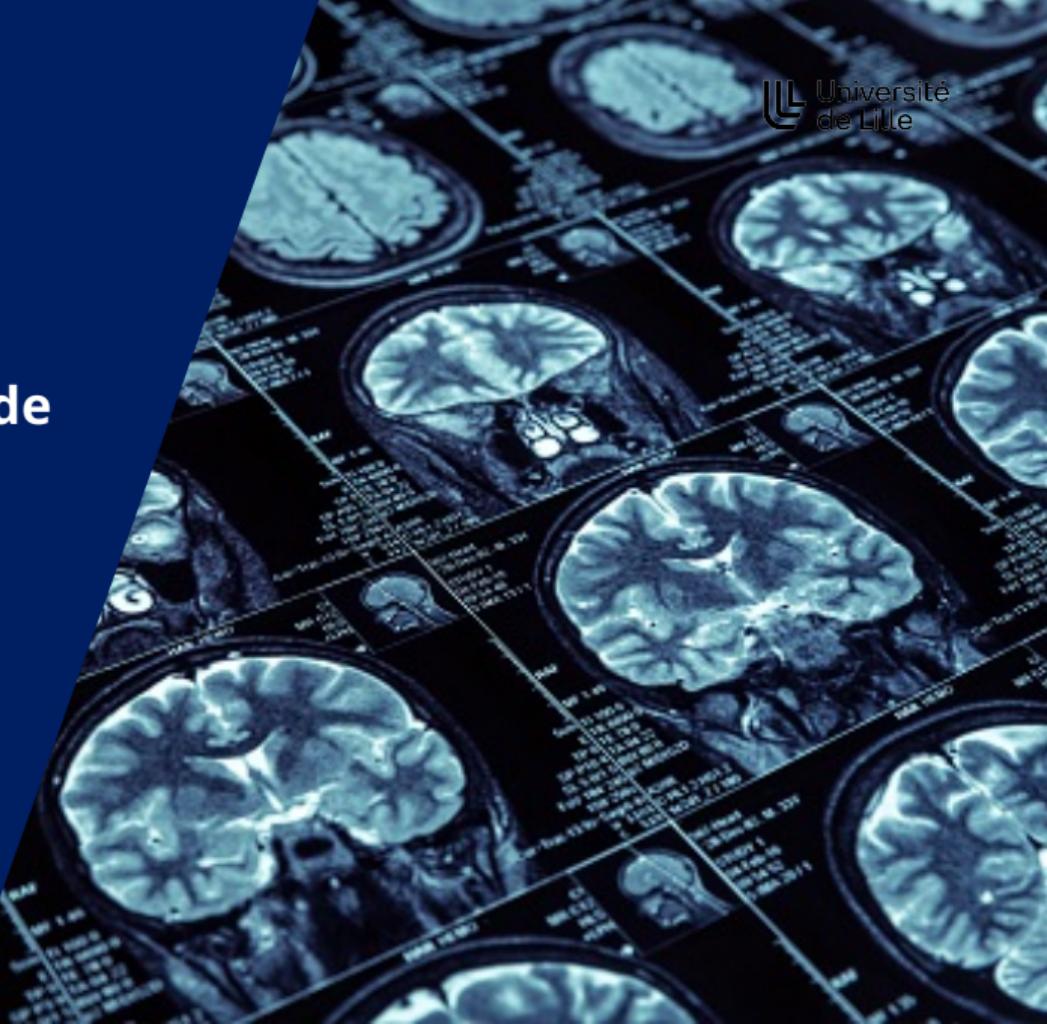


- "**Profond**" ≡ Plusieurs couches d'un réseau neuronal.
- L'apprentissage profond permet de réaliser des applications réelles dont les performances sont bien supérieures à celles des autres méthodes.
- Capacité à traiter des ensembles **larges** de données
- Pas besoin de "Feature Engineering", l'apprentissage profond le fait pour vous.
- Réduction de l'expertise nécessaire pour résoudre le problème.
- Calcul dans des manifolds de grandes dimension.
- Logiciels établis et robustes, faciles à mettre en œuvre et à utiliser tel que: PyTorch (Populaire), Keras, TensorFlow (moins populaire)

- Autograd (PyTorch, TensorFlow)
- Jeux de données de plus d'un milliard d'exemples
- Entraînement parallèle sur des milliers de GPU
- Architectures avec plus d'un milliard de paramètres
- Coûts d'entraînement de plusieurs millions de dollars
- Des résultats impressionnantes
- La taille massive n'est pas toujours nécessaire – ex. Stable Diffusion
- Communauté open source et réutilisation modulaire

# Principes fondamentaux de l'apprentissage

*Les types d'apprentissage*



## Données du problème

## Données du problème

- Domaine:  $\mathcal{X}$

## Données du problème

- Domaine:  $\mathcal{X}$
- Codomaine:  $\mathcal{Y}$

## Données du problème

- Domaine:  $\mathcal{X}$
- Codomaine:  $\mathcal{Y}$
- Concepts:  $\mathcal{C}$

## Données du problème

- Domaine:  $\mathcal{X}$
- Codomaine:  $\mathcal{Y}$
- Concepts:  $\mathcal{C}$

**Données du problème**

- Domaine:  $\mathcal{X}$
- Codomaine:  $\mathcal{Y}$
- Concepts:  $\mathcal{C}$

**Hypothèses**

### Données du problème

- Domaine:  $\mathcal{X}$
- Codomaine:  $\mathcal{Y}$
- Concepts:  $\mathcal{C}$

### Hypothèses

- $S^\infty \stackrel{iid}{\sim} \mathcal{D}$

### Données du problème

- Domaine:  $\mathcal{X}$
- Codomaine:  $\mathcal{Y}$
- Concepts:  $\mathcal{C}$

### Hypothèses

- $S^\infty \stackrel{iid}{\sim} \mathcal{D}$
- Classe d'hypothèses:  
 $\mathcal{H}$

### Données du problème

- Domaine:  $\mathcal{X}$
- Codomaine:  $\mathcal{Y}$
- Concepts:  $\mathcal{C}$

### Hypothèses

- $S^\infty \stackrel{iid}{\sim} \mathcal{D}$
- Classe d'hypothèses:  
 $\mathcal{H}$
- Séparabilité linéaire  
des données (en cas  
de classification) ...

### Données du problème

- Domaine:  $\mathcal{X}$
- Codomaine:  $\mathcal{Y}$
- Concepts:  $\mathcal{C}$

### Hypothèses

- $S^\infty \stackrel{iid}{\sim} \mathcal{D}$
- Classe d'hypothèses:  
 $\mathcal{H}$
- Séparabilité linéaire  
des données (en cas  
de classification) ...

### Données du problème

- Domaine:  $\mathcal{X}$
- Codomaine:  $\mathcal{Y}$
- Concepts:  $\mathcal{C}$

### Hypothèses

- $S^\infty \stackrel{iid}{\sim} \mathcal{D}$
- Classe d'hypothèses:  
 $\mathcal{H}$
- Séparabilité linéaire  
des données (en cas  
de classification) ...

### Données accessibles

**Données du problème**

- Domaine:  $\mathcal{X}$
- Codomaine:  $\mathcal{Y}$
- Concepts:  $\mathcal{C}$

**Hypothèses**

- $S^\infty \stackrel{iid}{\sim} \mathcal{D}$
- Classe d'hypothèses:  $\mathcal{H}$
- Séparabilité linéaire des données (en cas de classification) ...

**Données accessibles**

- Echantillon fini de points étiquetés.

### Données du problème

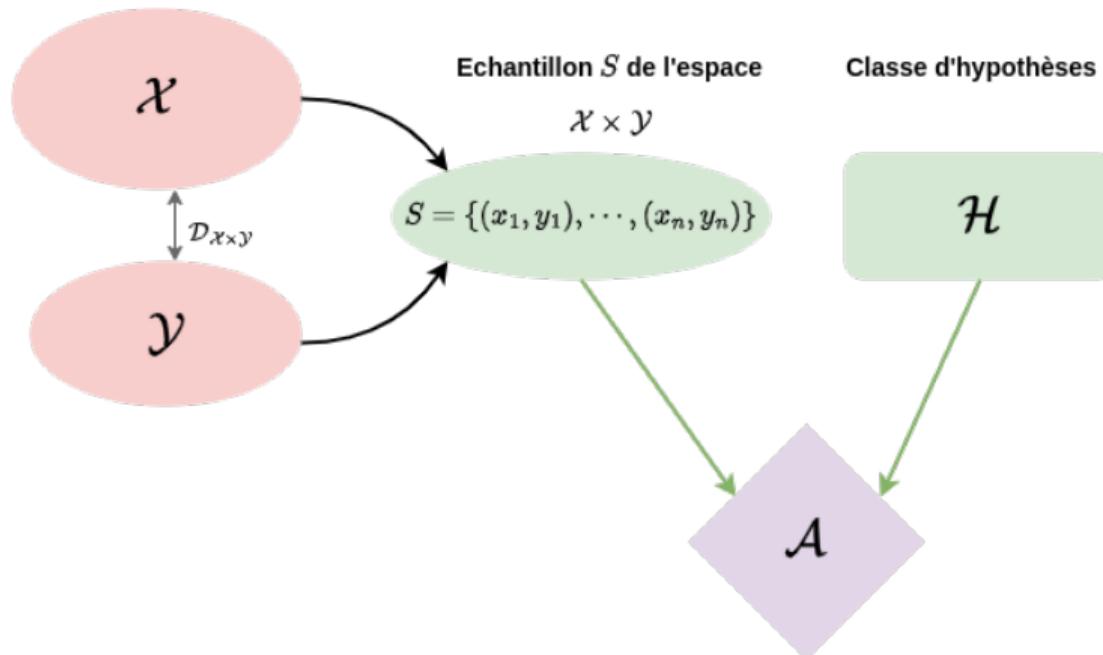
- Domaine:  $\mathcal{X}$
- Codomaine:  $\mathcal{Y}$
- Concepts:  $\mathcal{C}$

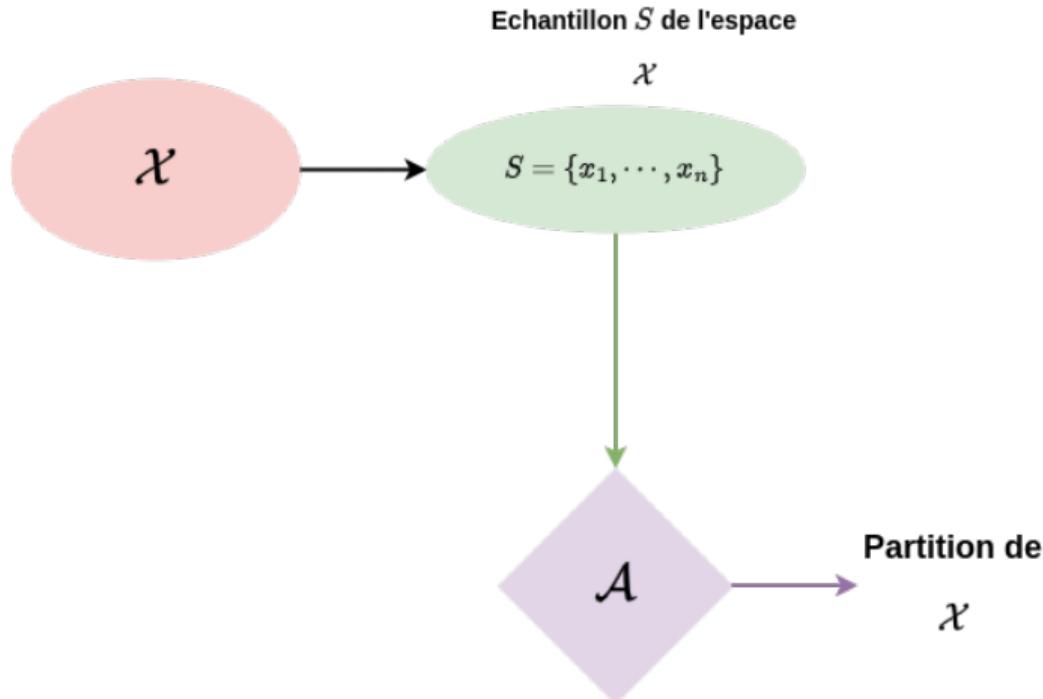
### Hypothèses

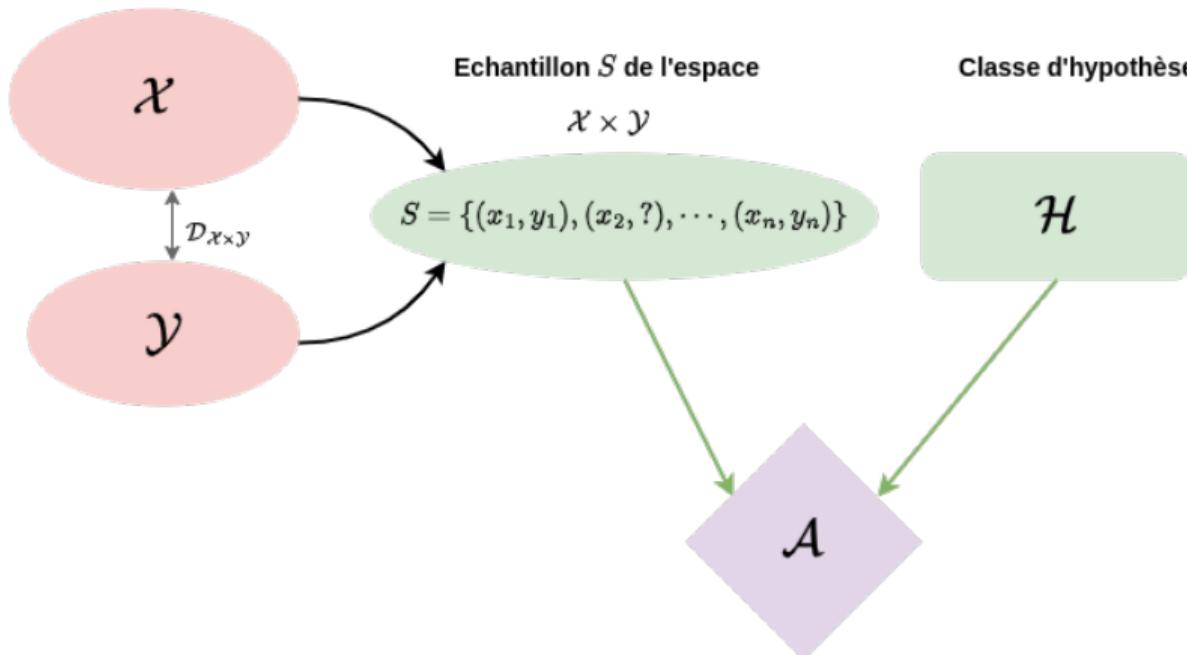
- $S^\infty \stackrel{iid}{\sim} \mathcal{D}$
- Classe d'hypothèses:  $\mathcal{H}$
- Séparabilité linéaire des données (en cas de classification) ...

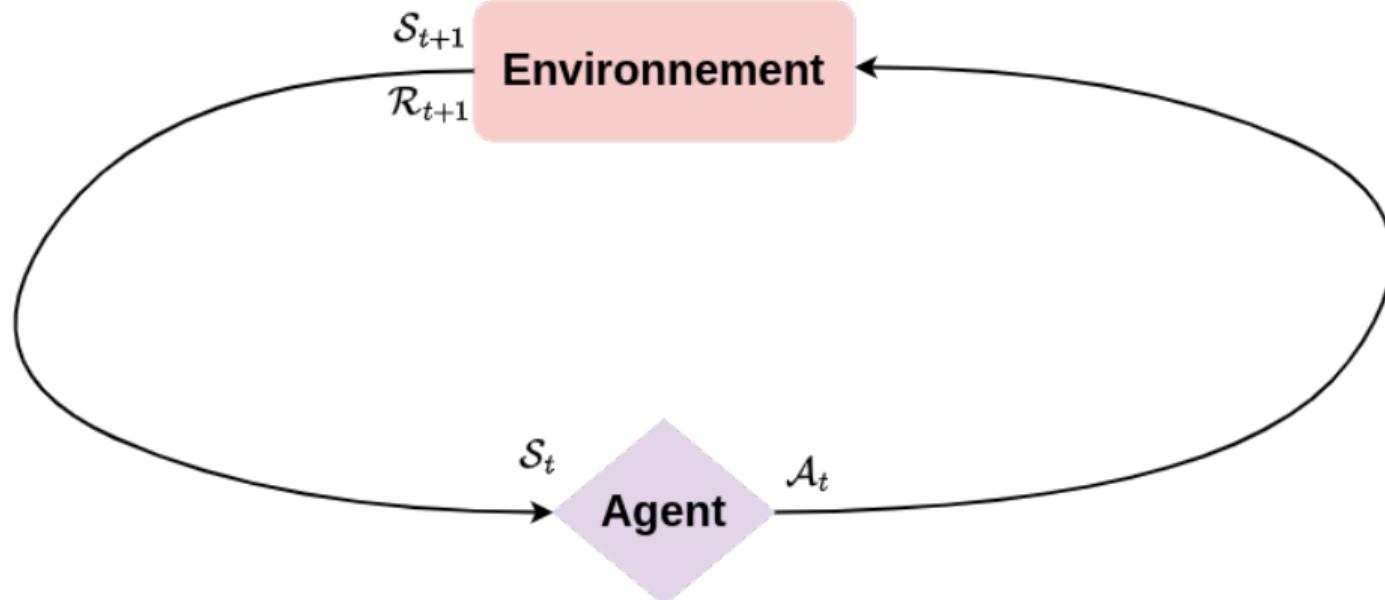
### Données accessibles

- Echantillon fini de points étiquetés.
- Mesure d'erreur.









# Réseaux de neurones artificiels

*Expressivité de la profondeur*

*Vs*

*Expressivité de la largeur*

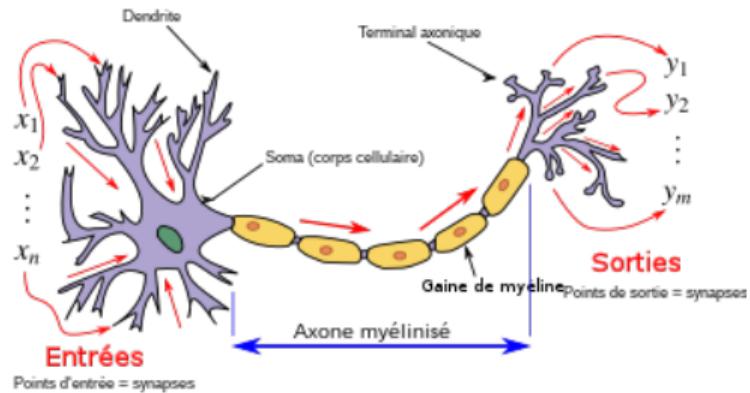
*Théorèmes d'approximation*

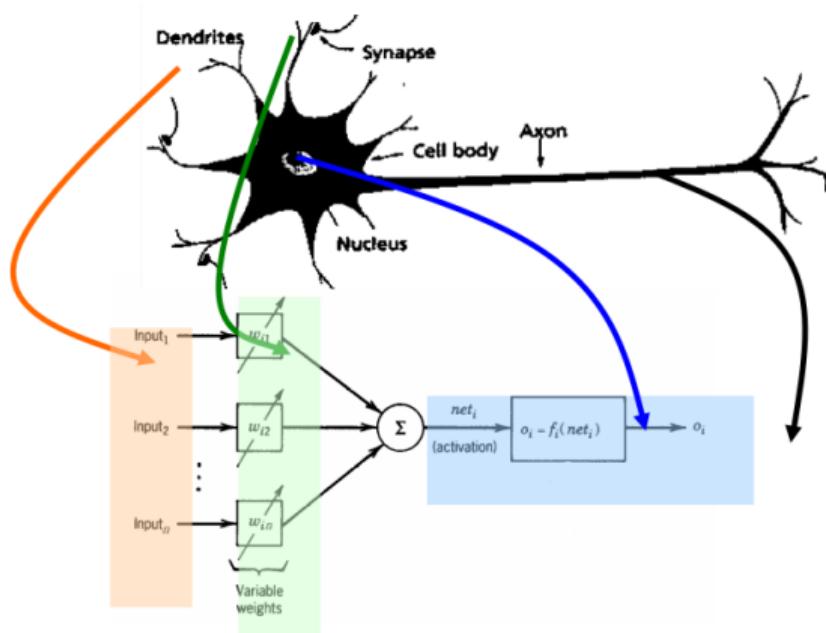


## Question fondamentale

Quel type de fonctions peut-on approximer avec des réseaux de neurones?

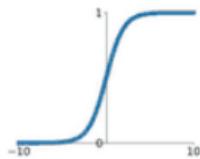
- Feed-forward (output unique)
- Recurrent (données séquentielles)



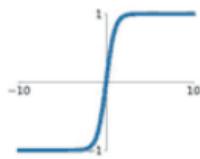


**Sigmoid**

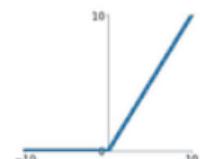
$$\sigma(x) = \frac{1}{1+e^{-x}}$$

**tanh**

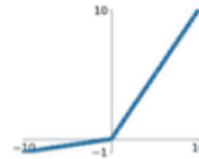
$$\tanh(x)$$

**ReLU**

$$\max(0, x)$$

**Leaky ReLU**

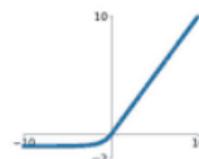
$$\max(0.1x, x)$$

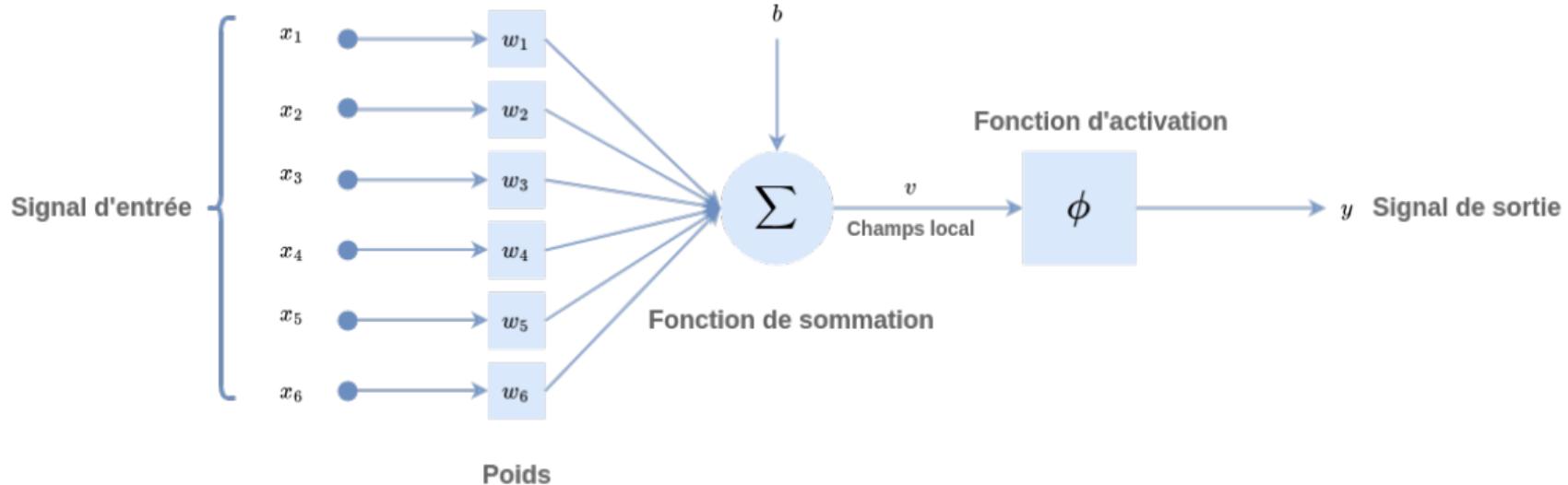
**Maxout**

$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

**ELU**

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$





## Avantages

- Ramène les valeurs entre 0 et 1.
- Bien étudié.

## Inconvénients

- Un neurone saturé a pour effet de d'annuler les gradients.
- Sortie d'une sigmoïde n'est pas centrée à zéro.
- l'exponentielle est très coûteux en termes de calcul.

## Avantages

- Ramène les valeurs entre -1 et 1.
- Sortie centrée à zéro.

## Inconvénients

Annulation du gradient lorsque la fonction saturée.

## Avantages

- Rapide.
- Aucune saturation.
- permet une convergence  $\sim 5 - 10$  plus rapide que sigmoid et tanh.

## Inconvénients

- Sortie non centrée à zéro.
- Pour les valeurs de  $x < 0$ , le gradient est nul.

## Avantages

- Aucune saturation.
- Très rapide.
- Le gradient ne s'annule pas.

## Inconvénients

0.01 est un hyperparamètre

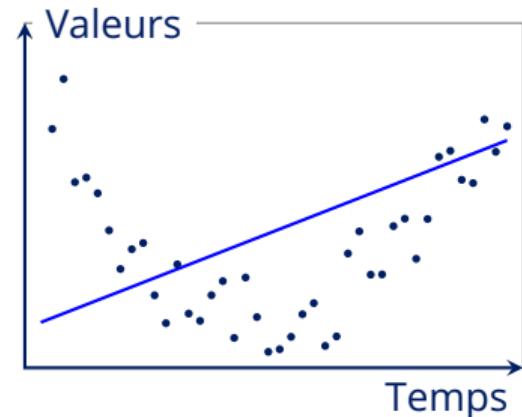
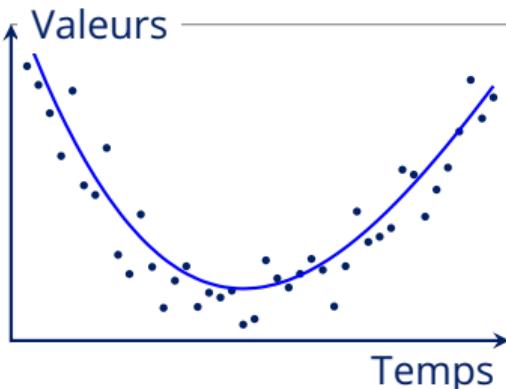
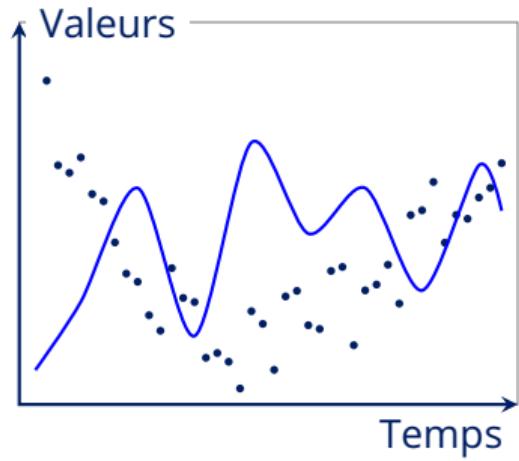
## Avantages

Mêmes avantages que Leaky ReLU.

## Inconvénients

l'hyperparamètre s'apprends durant la rétropropagation et est actualisé avec les autres hyperparamètres du problème.

- Par défaut, on utilise souvent la fonction d'activation ReLU.
- Les variations de la fonction ReLU sont encouragées.
- Sigmoid est uniquement utilisée à la sortie d'un réseau pour un problème de classification à deux classes.



## Theorème d'approximation universelle (informel)

Toute fonction continue  $f : [0, 1]^n \rightarrow [0, 1]$  peut être approximée arbitrairement par un réseau de neurones avec au moins une couche cachée et un nombre fini de poids.

## Exemple: Approximation des fonctions Lipschitz

Pour approximer une fonction  $L$ -Lipschitz  $f : \mathbb{R} \rightarrow \mathbb{R}$  dans l'interval  $[0, 1]$  avec une erreur d'au plus  $\epsilon$ , combien de bin a t-on besoin?

## Exemple: Approximation des fonctions Lipschitz

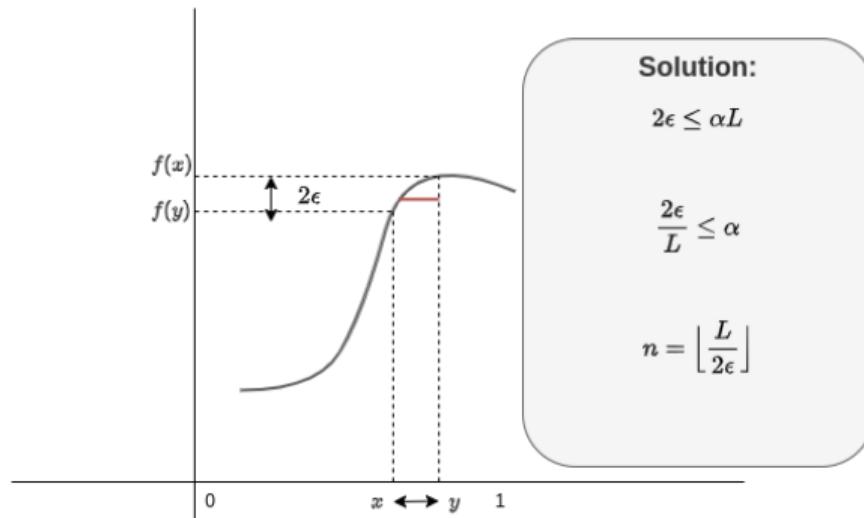
Une fonction  $f : \mathbb{R} \rightarrow \mathbb{R}$  est  $L$ -Lipschitz quand elle vérifie la propriété suivante:

$$\forall x, y \in \mathbb{R} : |f(x) - f(y)| < L|x - y|$$

# Exemple: Approximation des fonctions Lipschitz

Une fonction  $f : \mathbb{R} \rightarrow \mathbb{R}$  est  $L$ -Lipschitz quand elle vérifie la propriété suivante:

$$\forall x, y \in \mathbb{R} : |f(x) - f(y)| < L|x - y|$$



# Question: Approximation des fonctions Lipschitz dans un espace de grande dimension

Pour approximer une fonction  $L$ -Lipschitz  $f : \mathbb{R}^n \rightarrow \mathbb{R}$  dans l'intervalle  $[0, 1]^n$  avec une erreur d'au plus  $\epsilon$ , combien de bin a t-on besoin?

## Famille de fonctions Sigmoid

Une famille de fonctions Sigmoid est une famille de fonctions  $\sigma$  qui vérifient les propriétés suivantes:

- $\lim_{x \rightarrow +\infty} \sigma(x) = 1$
- $\lim_{x \rightarrow -\infty} \sigma(x) = 0$

## Approximation de Cybenko (informel)

Toute fonction continue peut être approximée par un MLP avec une seule couche intermédiaire et une fonction arbitraire de la famille Sigmoid.

## Approximation de Cybenko (formel)

Pour toute fonction de la famille Sigmoid  $\sigma$ ,

l'ensemble de fonctions  $\left\{ f : x \rightarrow \sum_{i=1}^N \alpha_i \sigma(w_i x + b_i) \right\}$  (MLP avec une couche intermédiaire) est dense dans  $\mathcal{C}([0, 1]^n)$

NB:  $N$  dépend de la complexité de la fonction à approximer et de l'erreur d'approximation.

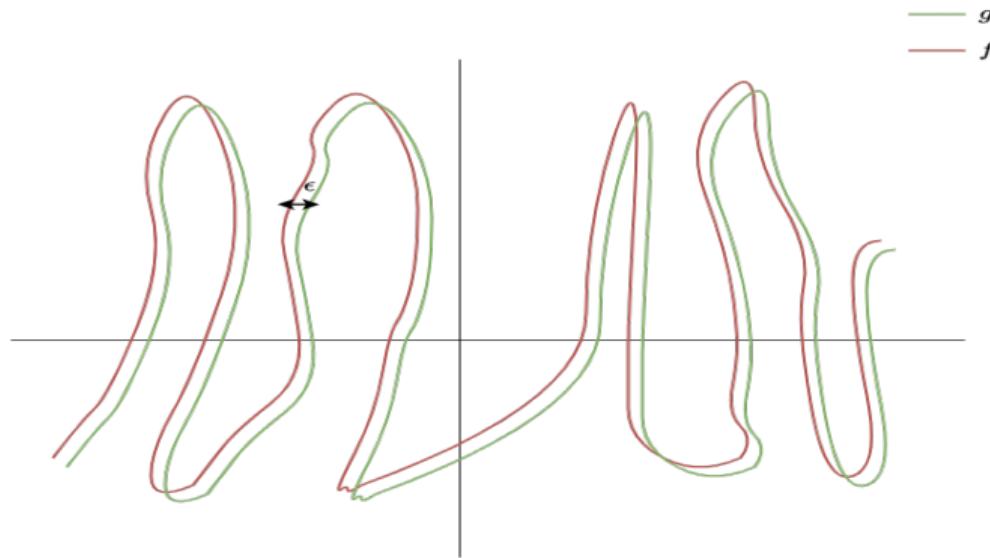


Illustration du théorème de Cybenko

## Theorème de Kolmogorov-Arnold (Informel)

Toute fonction continue à plusieurs variables peut être représentée comme une superposition de fonctions continues d'une seule variable.

## Theorème de Kolmogorov-Arnold (Formel)

Toute fonction continue  $f : [0, 1]^n \rightarrow \mathbb{R}$  peut être représentée de la façon suivante:

$$f(x_1, \dots, x_n) = \sum_{i=1}^{2n+1} \phi_i \left( \sum_{j=1}^n \psi_{ij}(x_j) \right)$$

### Theorème d'approximation universelle pour les réseaux ReLU avec une largeur borné

Pour tout  $\epsilon > 0$ , et toute fonction  $f : \mathbb{R} \rightarrow \mathbb{R}$  Lebesgue intégrable, il existe un réseau de neurones ReLU  $\mathfrak{N}$  (entièlement connecté) avec une largeur  $\leq n + 4$  tel que la fonction représentée par ce réseau de neurones  $F_{\mathfrak{N}}$  vérifie l'approximation:

$$\int_{\mathbb{R}^n} |f(x) - F_{\mathfrak{N}}(x)| dx < \epsilon$$

## L'importance de la profondeur dans l'apprentissage (Telgarsky 2015)

Soit  $m \geq 2^{\frac{k-3}{l-1}}$ ,  $k \in \mathbb{N}$ ,

Il existe un échantillon  $S = (x_i, y_i)_{i=1}^{2^k}$ , avec  $x_i \in [0, 1]$ ,  $y_i \in \{0, 1\}$  pour tout  $i \leq 2^k$  tel que:

$$\min_{f \in \mathfrak{N}(\sigma_R, m, l)} \mathcal{R}(f) = \frac{1}{6} \quad \& \quad \min_{f \in \mathfrak{N}(\sigma_R, 2, 2k)} \mathcal{R}(f) = 0$$

$\mathfrak{N}(\sigma_R, m, l)$  désigne l'ensemble de fonctions données par un feedforward réseau de neurones avec une fonction d'activation  $\sigma$ , à  $l$  couches avec au plus  $m$  nœuds.  
 $\sigma_R$  désigne la fonction d'activation ReLU.

## Comment apprendre via des réseaux de neurones

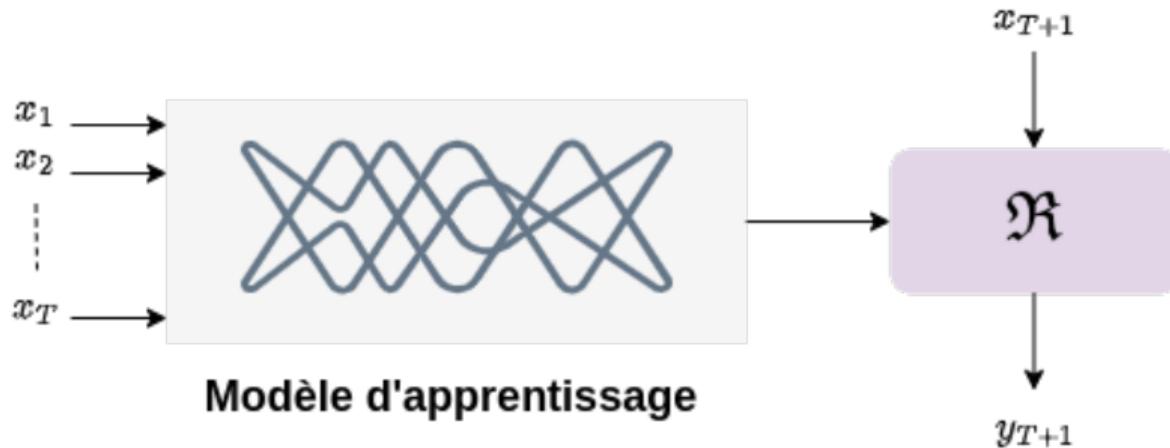
*Entrainement d'un réseau de  
neurones à multi-couches*



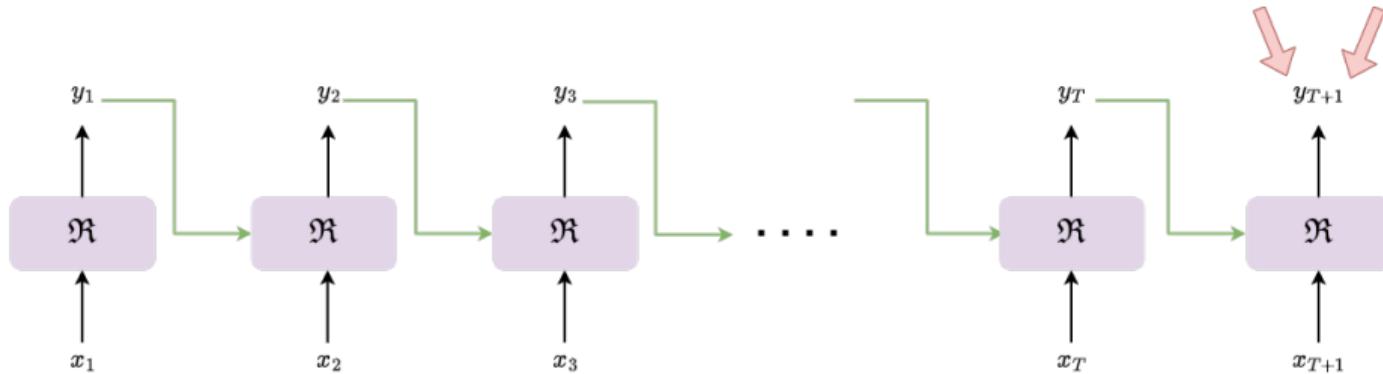
# Réseaux de neurones récurrents

*suivi temporel de l'information*

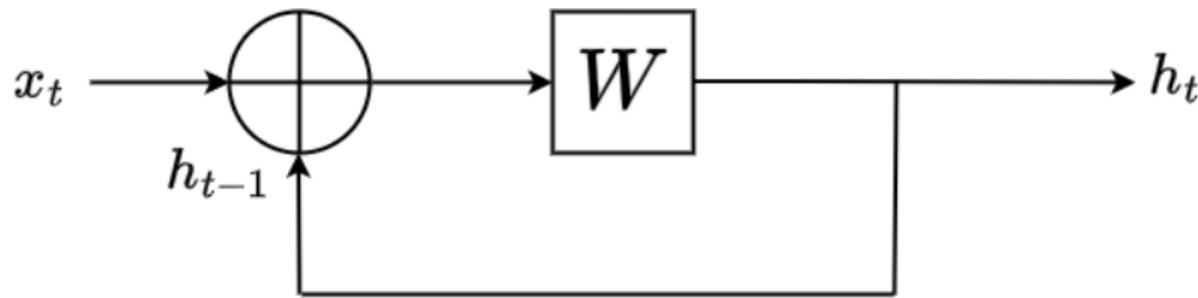




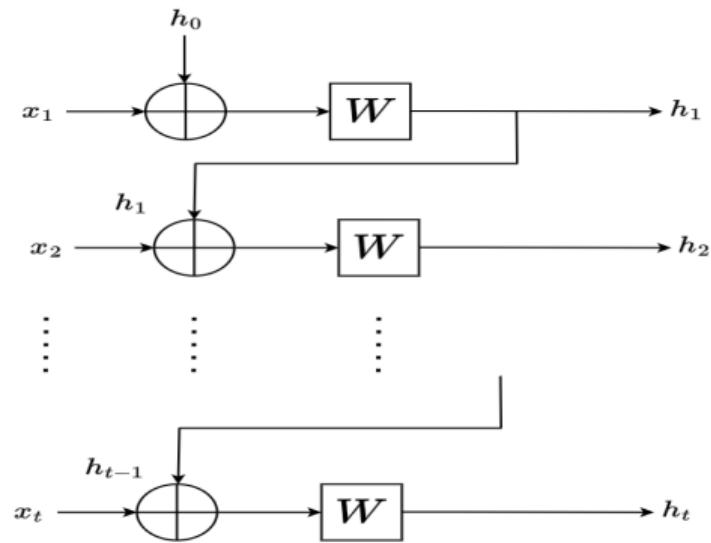
Structure générale de l'algorithme d'apprentissage dans le cas de données temporelles.



La nature séquentielle de l'algorithme d'apprentissage nécessite que chaque entrée à  $\mathfrak{R}$  dépendra de toutes les séquences temporelles qui précédent.



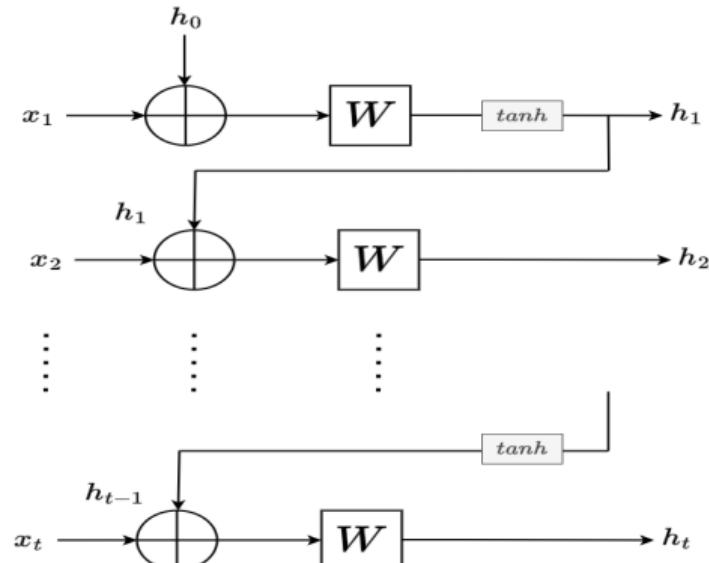
Block résiduel (Version sans rollout)



Block résiduel (Version rollout)

La construction est linéaire en fonction des états de chaque bloc et donc susceptible de mal représenter le flux d'information temporel (Transformation affine des données d'entrée).

La construction est linéaire en fonction des états de chaque bloc et donc susceptible de mal représenter le flux d'information temporel (Transformation affine des données d'entrée).



Structure finale.

# Expressivité de la largeur Vs Expressivité de la profondeur

Qu'en est t-il pour les réseaux récurrents?

Qu'en est t-il pour les réseaux récurrents?

## Définition (t-sawtooth fonction d'activation)

Pour tout  $t \in \mathbb{N}$ , une fonction  $f : \mathbb{R} \rightarrow \mathbb{R}$  est dite t-sawtooth quand elle est linéaire par morceaux avec  $t$  morceaux.

Example: La fonction ReLU est 2-sawtooth.

- Notation:  $\mathfrak{R}(\sigma, m, l, k)$  désigne  $k$  itérations d'un réseau de neurones à  $l$  couches d'au plus  $m$  noeuds.
- Conséquence:  $\mathfrak{R}(\sigma, m, l, k) \subseteq \mathfrak{N}(\sigma, m, lk)$

# Expressivité de la largeur Vs Expressivité de la profondeur

Qu'en est t-il pour les réseaux récurrents?

Qu'en est t-il pour les réseaux récurrents?

## L'importance de la profondeur dans l'apprentissage (Telgarsky 2015)

Soit  $k, t \in \mathbb{N}$ ,  $l$ : nombre de couches,  $m$ : nombre de noeuds (par couche)

Il existe un échantillon  $S = (x_i, y_i)_{i=1}^{2^k}$ , avec  $x_i \in [0, 1]$ ,  $y_i \in \{0, 1\}$  pour tout  $i \leq 2^k$  tel que:

$$\min_{f \in \mathfrak{N}(\sigma_R, 2, 2, k)} \mathcal{R}(f) = 0 \quad \& \quad \min_{f \in \mathfrak{N}(\sigma_t, m, l)} \mathcal{R}(f) \geq \frac{n - 4(tm)^l}{3n}$$

# Réseaux de neurones convolutifs

*Suivi spatial de l'information*



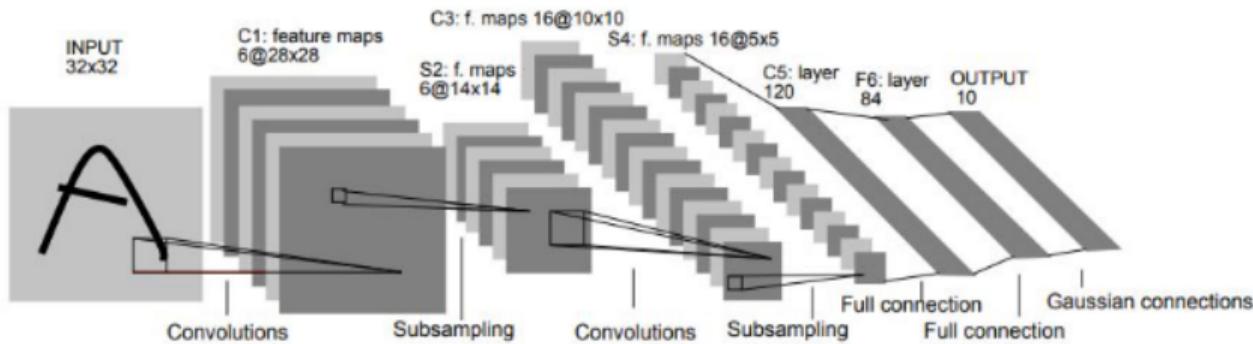
## Problème avec les MLPs

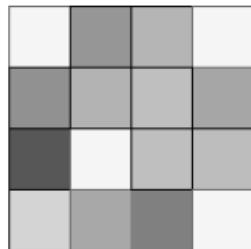
- Image  $28 \times 28$ , 8 bits  $\Rightarrow 800\,000$  paramètres
- Image 28M pixels ( $3 \times 8$  bits)  $\Rightarrow \approx 10^{11}$  paramètres  $\Rightarrow$  Explosion du nombre de paramètres .

## Solution: CNN

- Exploitent la structure locale des images
- Partage de poids  $\Rightarrow$  Réduction massive du nombre de paramètres
- Scalable aux grandes images

- Intuition:
  - Traiter des données de très haute dimension : $150 \times 150$  pixels = 22500 entrées.
  - Peut exploiter la topologie 2D des pixels
  - Peut intégrer l'invariance à certaines variations
- Connectivité locale: Chaque unité cachée n'est connectée qu'à une sous-région (patch) de l'image d'entrée.
- Partage des paramètres de poids: Partager la matrice des paramètres entre certaines unités.
- Convolution.
- Pooling/sous-échantillonnage des unités cachées

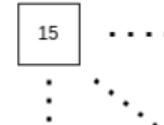


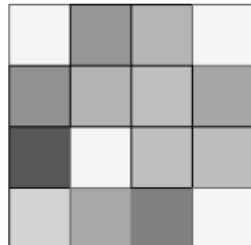


4	1	2	5
3	5	3	2
5	1	3	3
0	2	1	2

1	2	5
5	3	2
1	3	3

1	0	1
0	1	1
1	0	1

 $\times$  $\parallel$ 



4	1	2	5
3	5	3	2
5	1	3	3
0	2	1	2

1	2	5
5	3	2
1	3	3

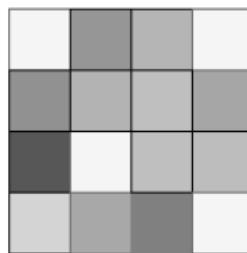
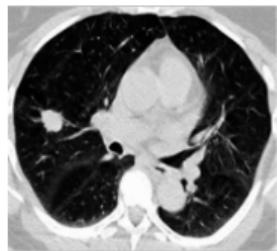
1	0	1
0	1	1
1	0	1

x

||

15	12
3	9

Reconnaitre des objets plus grands en empilant plusieurs petites convolutions permettant au réseau de capturer des structures de plus en plus larges tout en réduisant le nombre de paramètres.

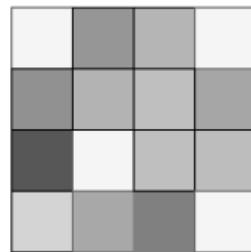


0	0	0	0	0	0
0	4	1	2	5	0
0	3	5	3	2	0
0	5	1	3	3	0
0	0	2	1	2	0
0	0	0	0	0	0

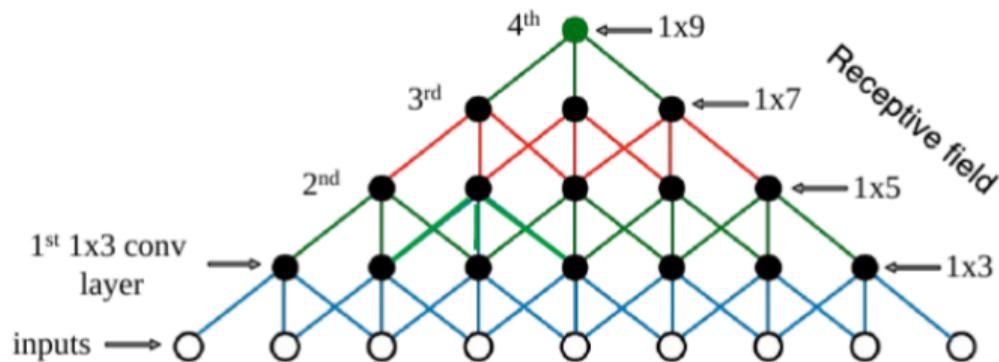
0	0	0
0	4	1
0	3	5

 $\times$ 

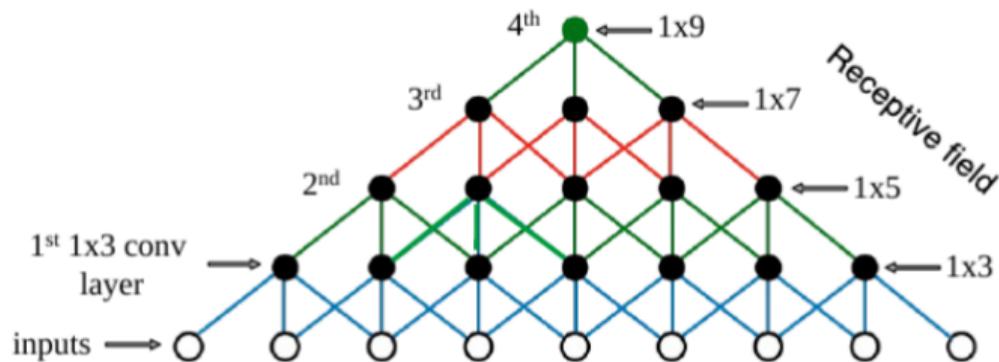
1	0	1
0	1	1
1	0	1



0	0	0	0	0	0	0
0	4	1	2	5	0	0
0	3	5	3	2	0	0
0	5	1	3	3	0	0
0	0	2	1	2	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

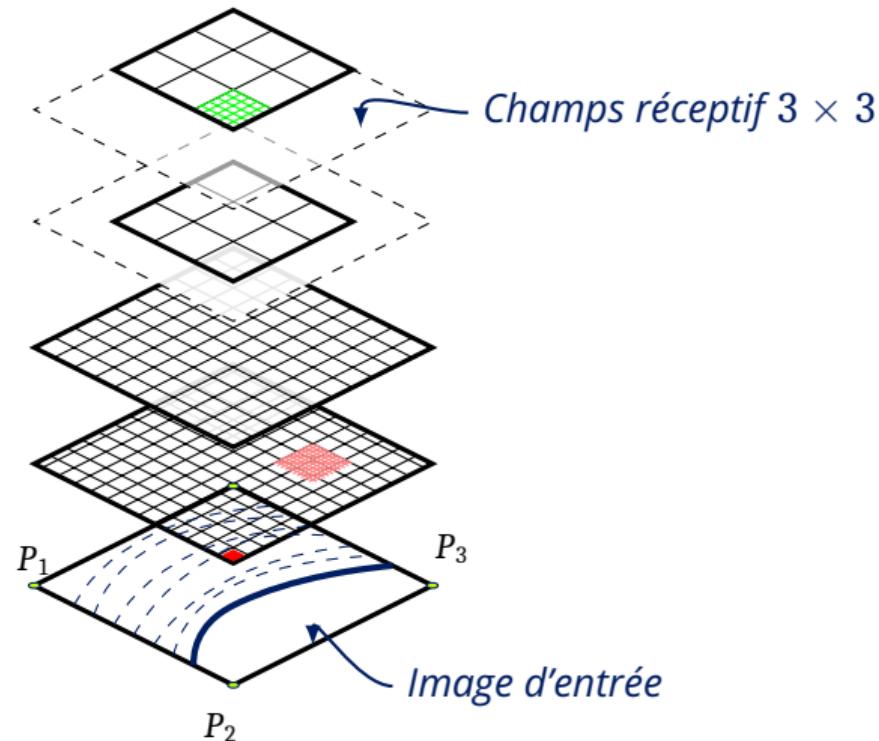


Combien de convolutions 3×3 devons-nous utiliser pour reconnaître un objet de 100x100 px ?



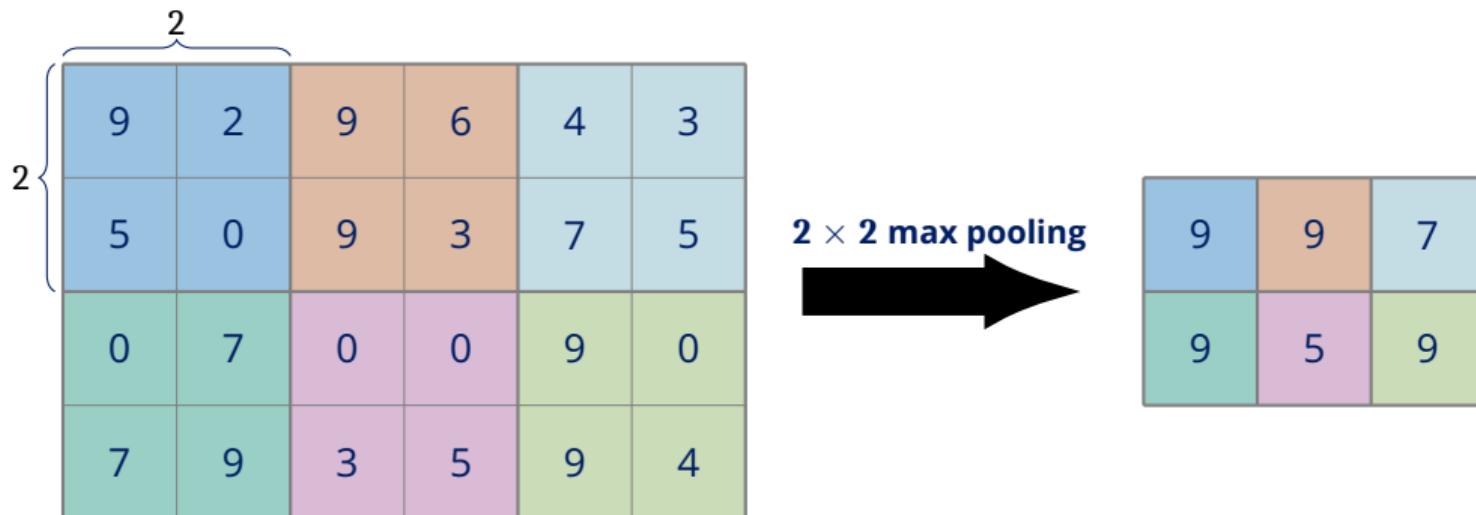
Combien de convolutions 3×3 devons-nous utiliser pour reconnaître un objet de 100x100 px ?

Environ 50... Il faut agrandir le champ réceptif plus rapidement !



## Motivation

- Réduire la taille des couches par un facteur
- Rendre le réseau de neurones moins sensible aux petits décalages.



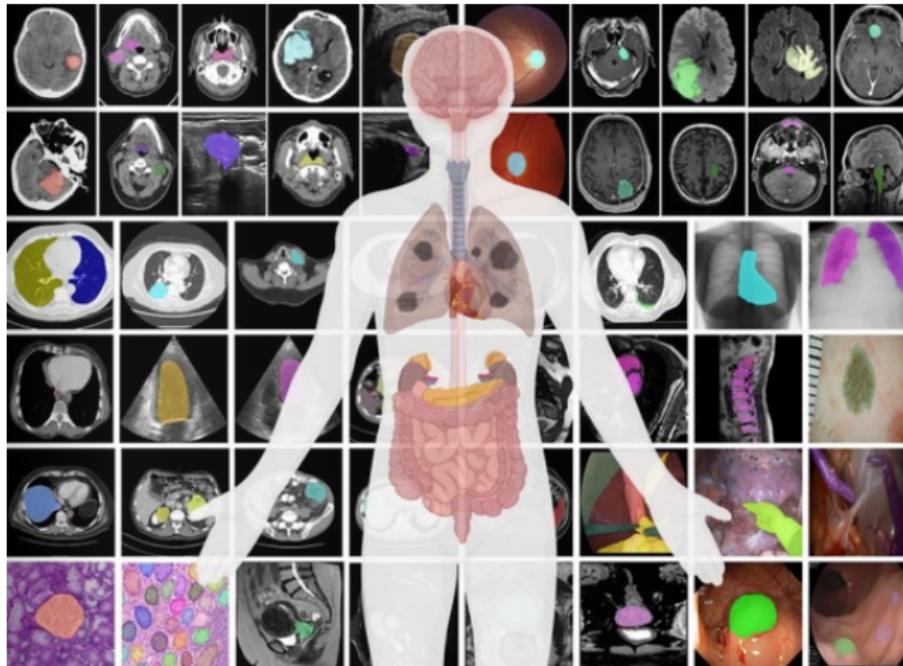
- Les CNN traitent des données de nature spatiale: les grilles à valeurs, par exemple les images. Les RNN est spécialisé pour une séquence de valeurs.
- Les CNN s'adaptent facilement aux images de grandes dimensions, les CNN peuvent traiter des images de taille variable Les RNN s'adaptent à des séquences plus longues qu'il ne serait pas pratique pour des réseaux sans adaptations basées sur les séquences, ils peuvent également traiter des séquences de longueur variable.
- Les RNN sont caractérisés par un état, les CNN n'ont pas d'état. (sauf pendant l'entraînement - la quasi-totalité de l'entraînement se fait sans état)
- Les RNN ont une mémoire, contrairement aux CNN.

## Applications dans la médecine

*L'intégration de l'information au service de la médecine: de la pré-diction à la segmentation*

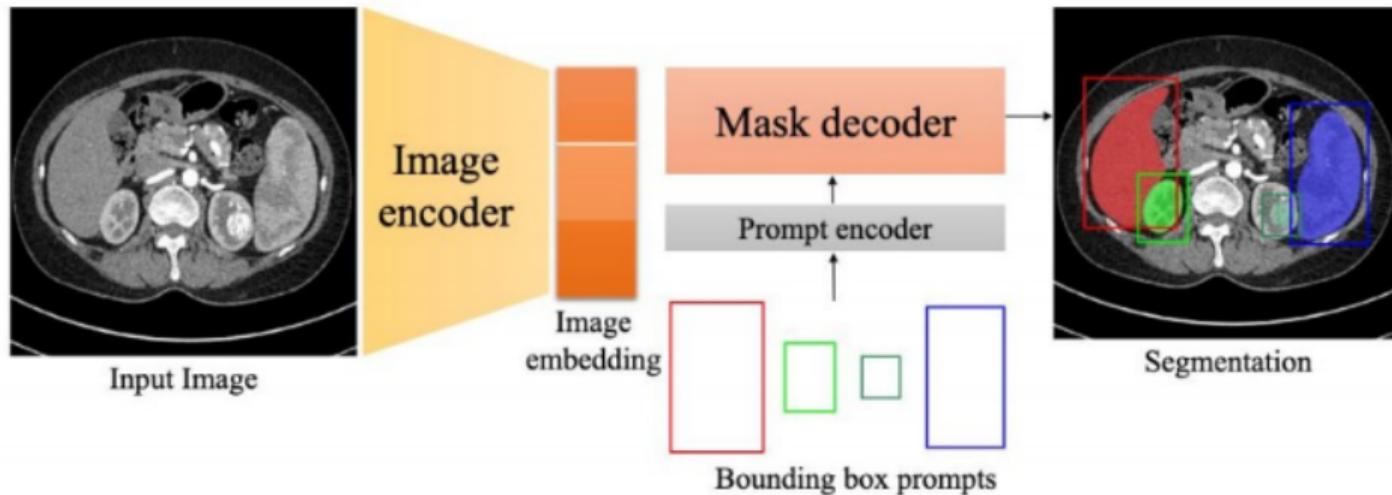


# Segmentation sémantique dans les images médicales

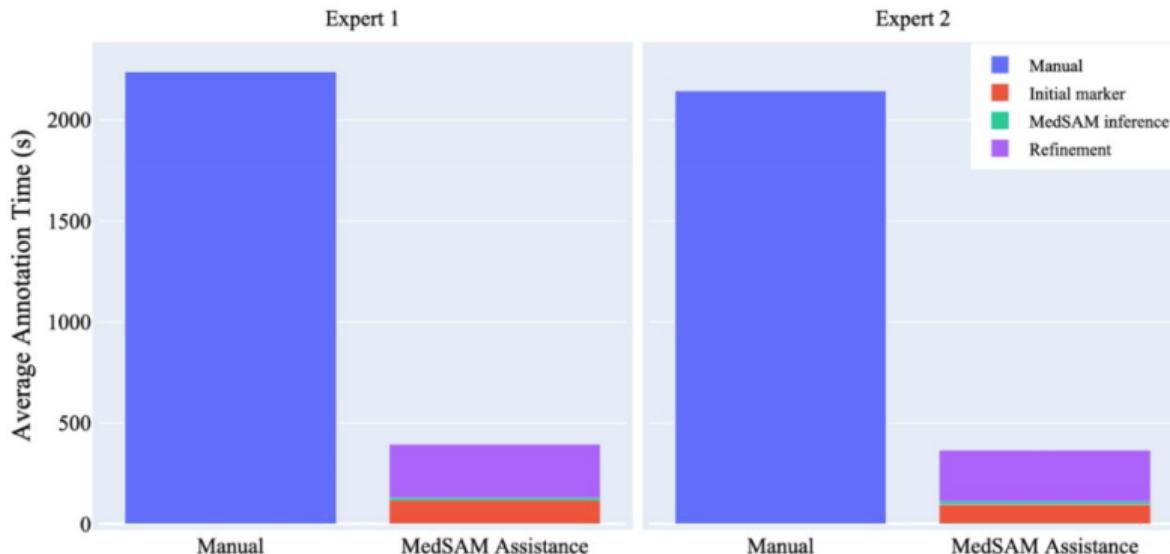


- Partie encoder: un "Vision transformer" pour extraire les caractéristiques de l'image et obtenir un embedding de l'image source, et un encodeur qui peut encoder l'information dans le box capturant l'information nécessaire qui permet l'extraction du embedding.
- Partie decoder: décodeur de masque basé sur un transformateur pour fusionner l'incorporation d'image et l'incorporation de l'embedding afin de générer les masques finaux.

# Segmentation sémantique dans les images médicales: MedSAM

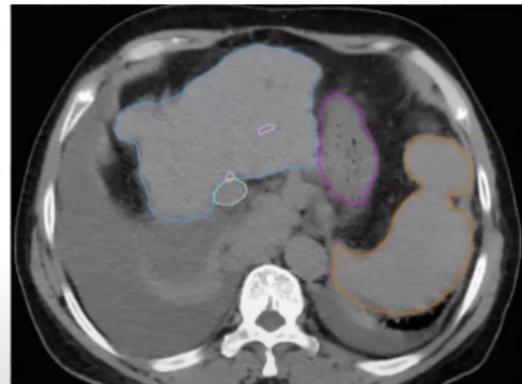
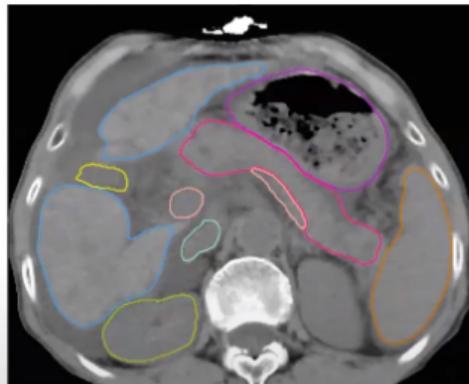


Un essai avec deux groupes de médecins a montré une augmentation de plus de 80 % de l'efficacité du flux de travail pour les médecins assistés par le modèle par rapport à ceux qui suivaient un flux de travail manuel standard.



## Pourquoi segmenter le foie ?

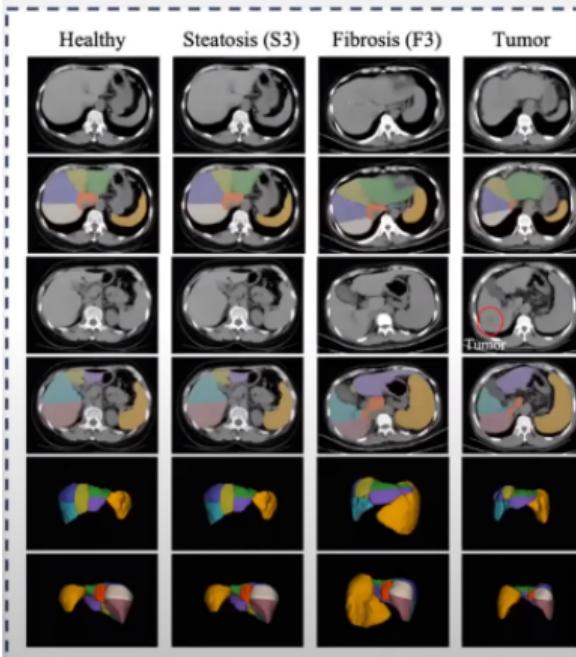
- Organe clé : Plus grand organe du corps, siège de nombreuses maladies chroniques et tumeurs.
- Rôle en imagerie médicale : Base de la tomographie assistée par ordinateur (CT).
- Apport du Deep Learning : Segmentation pour le dépistage, le diagnostic et le suivi des maladies hépatiques.



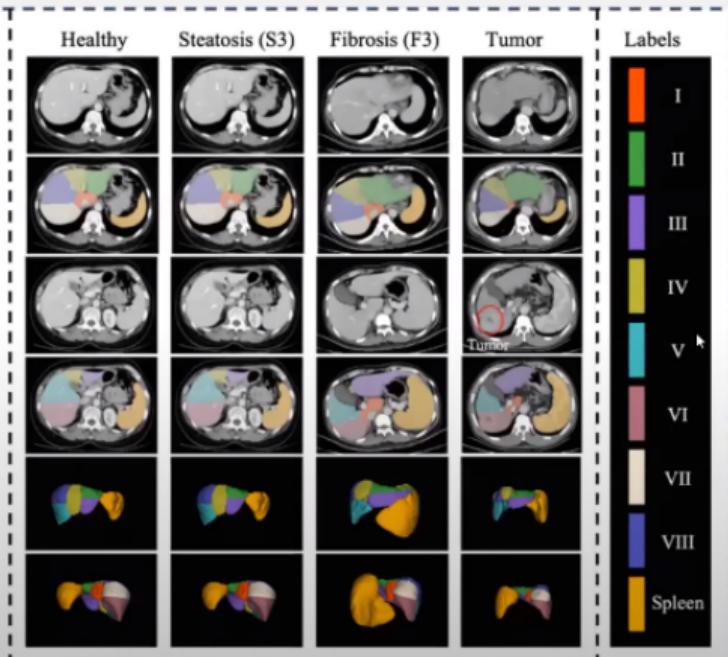
## Défis et segmentation par le deep learning

- **CT avec ou sans contraste** : Les algorithmes DL fonctionnent bien dans les deux cas.
- **Nombreuses maladies prises en compte** : Cirrhose, ascite, des cas post-opératoire...
- Partition en 8 segments de Couinaud : Essentiel pour la localisation des tumeurs et la planification chirurgicale.

(a) NC-CT

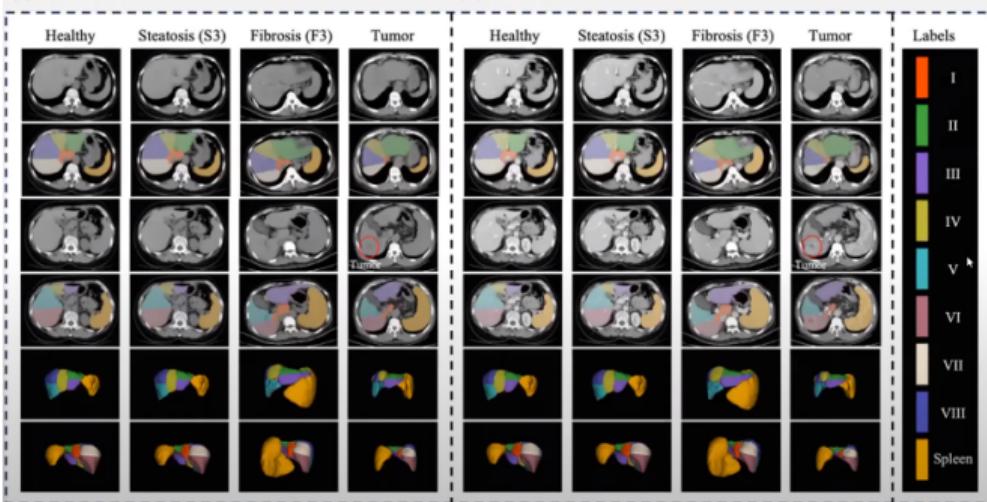


(b) CE-CT

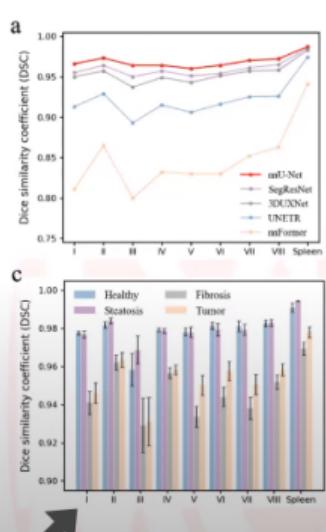


# Segmentation par des algorithmes d'apprentissage profond<sup>1</sup>

(a) NC-CT



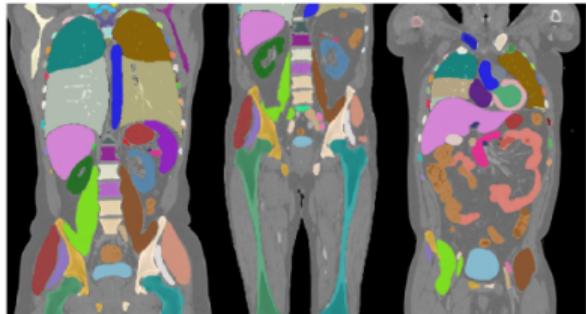
(b) CE-CT



Segmentation plus précise pour les patients sains ou atteints de stéatose.

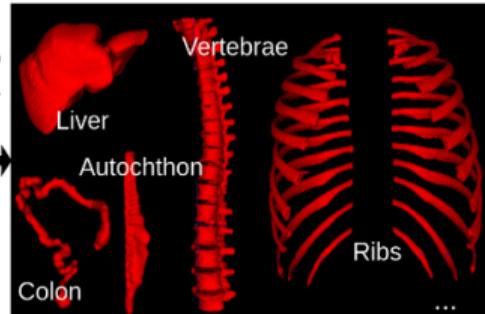
<sup>1</sup>BASSI, Pedro RAS et al. Touchstone benchmark: Are we on the right way for evaluating AI algorithms for medical segmentation?. Advances in Neural Information Processing Systems, 2024.

Segmentations from the TotalSegmentator dataset



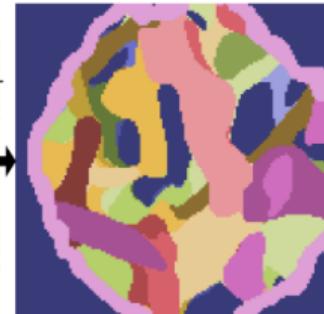
Sampling → Label

Randomly sampled 3D shape templates



Compose → Deform

3D shape ensemble

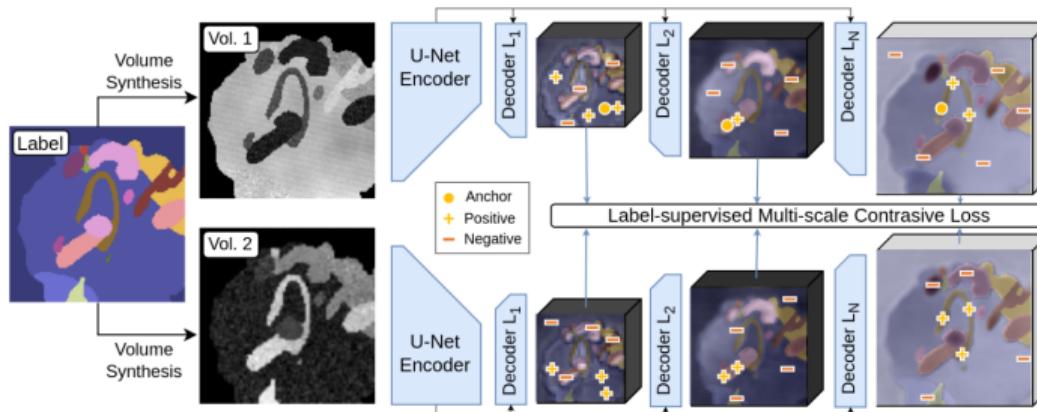


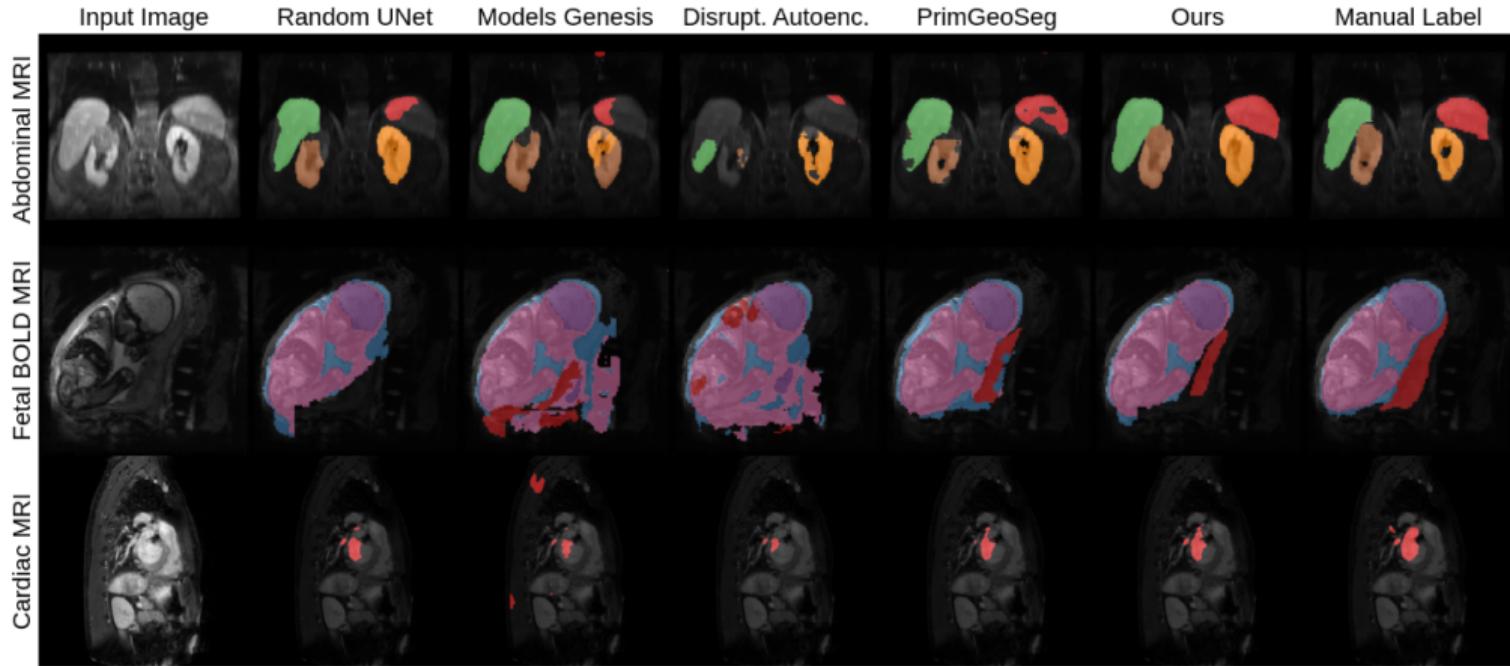
## Synthèse aléatoire pour un apprentissage robuste

- Générateur de données très variés pour anticiper les variations de domaine.
- Pré-entraînement d'un réseau 3D via un apprentissage par contraste, stable face aux changements d'imagerie

## Méthodologie

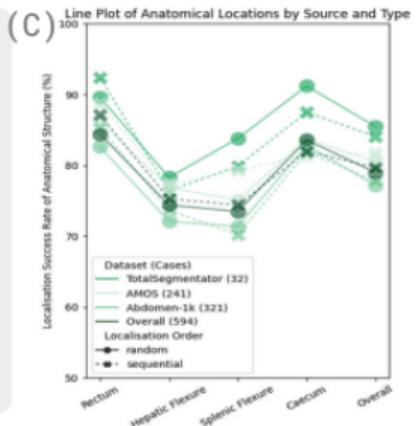
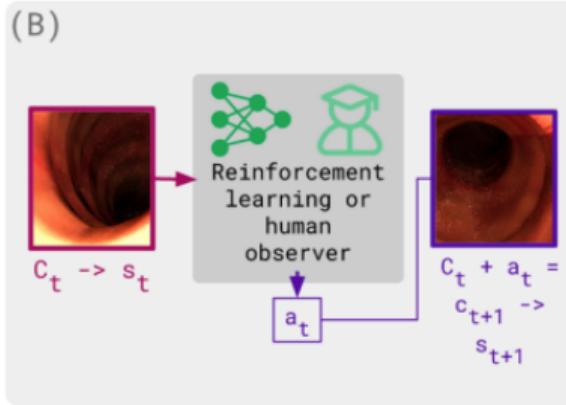
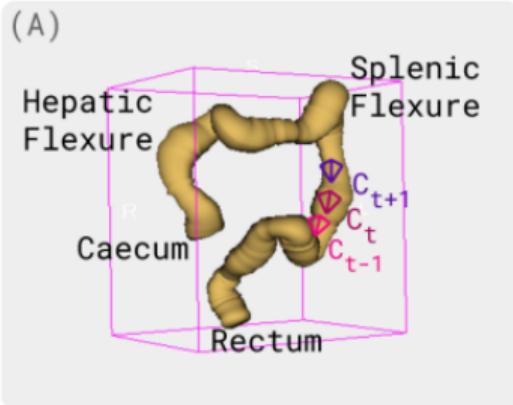
- Développement d'un moteur de données générant des échantillons très variés pour anticiper les variations de domaine.
- Pré-entraînement d'un réseau 3D via un apprentissage contrastif, stable face aux changements d'imagerie.





- Collecte des données réelles à l'aide de capteurs fixés sur des caméras afin d'entraîner les algorithmes est souvent très couteux.
- SARAMIS permet de fournir le premier ensemble d'environnements et de données qui peuvent être utilisés pour générer une multitude de données synthétiques.

# SARAMIS: Simulation pour la chirurgie assistée par robot pour un traitement mini-invasif



That's all folks!