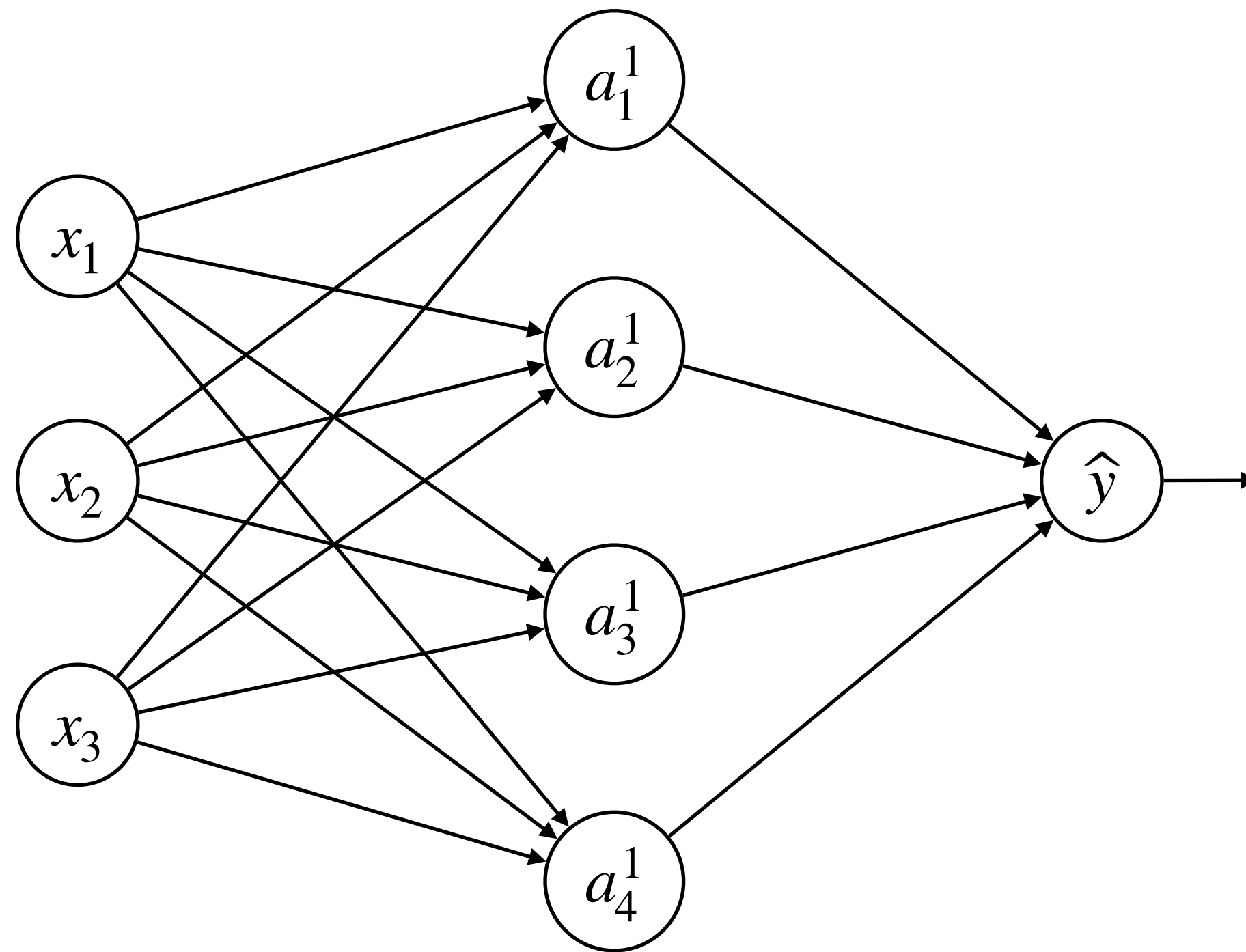


Deep Learning

Martin Vielvoye - 2021

Shallow Networks

Calcul Matriciel

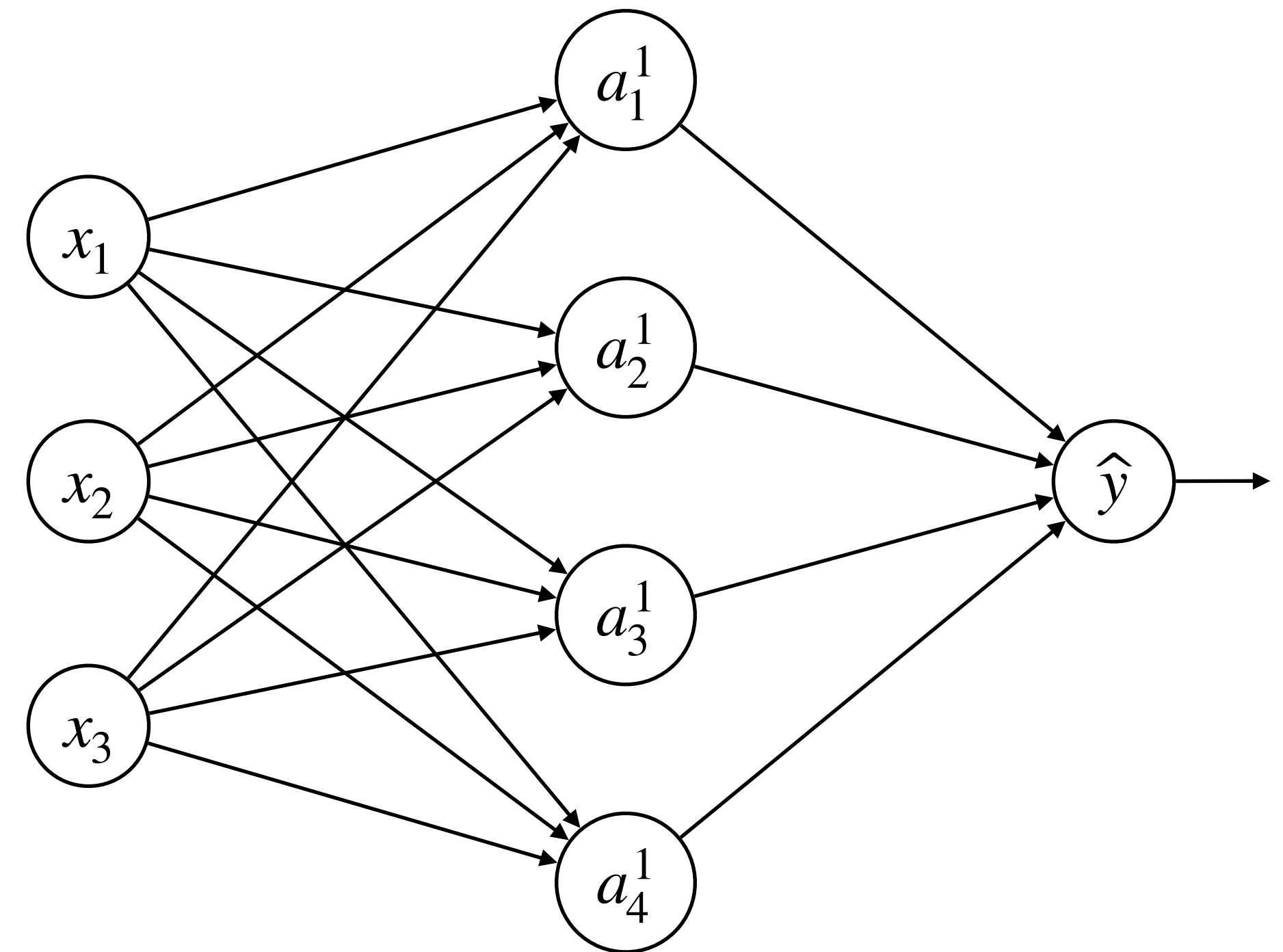


Notation : $a_i^{[l]}$ \leftarrow # de couches
 \leftarrow # de noeuds dans la couche

$$Z^{[1]} = W^{[1]}X + b^{[1]} = \begin{bmatrix} -w_1^{[1]} - \\ -w_2^{[1]} - \\ \vdots \end{bmatrix}^T * \begin{bmatrix} x_1 \\ x_2 \\ \vdots \end{bmatrix} + \begin{bmatrix} b_1^{[1]} \\ b_2^{[1]} \\ \vdots \end{bmatrix}$$

Réseaux superficiel

- Les réseaux superficiels ont longtemps été présents avant les réseaux profonds (*DLL*). Le temps de calcul étant plus long et la donnée plus limitée, ils représentaient la limite de l'apprentissage artificiel et du modèle neuronale.
- Même si la robustesse des modèles peut laisser à désirer, la rapidité d'amélioration des algorithmes créés une confiance grandissante dans cette discipline.



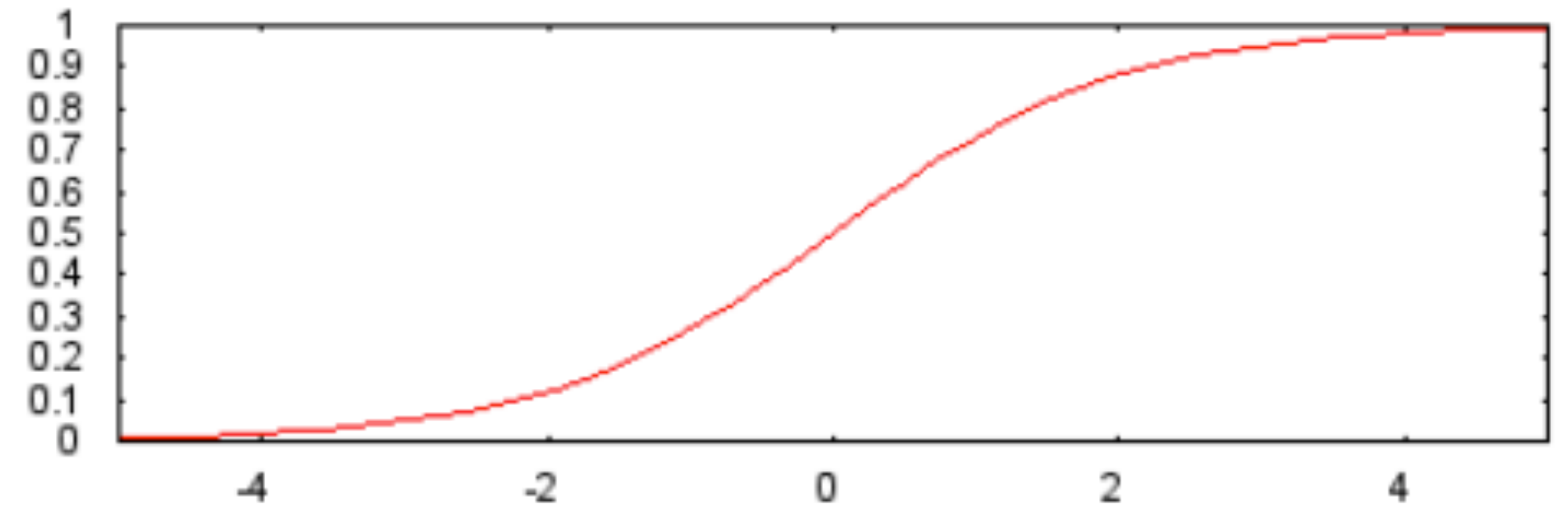
Fonctions d'activation

Fonction *logistique*

$$g(z) = \frac{1}{1 + e^{-z}}$$

- Fonction très populaire en classification binaire.

Dérivée : $g'(z) = g(z)(1 - g(z))$

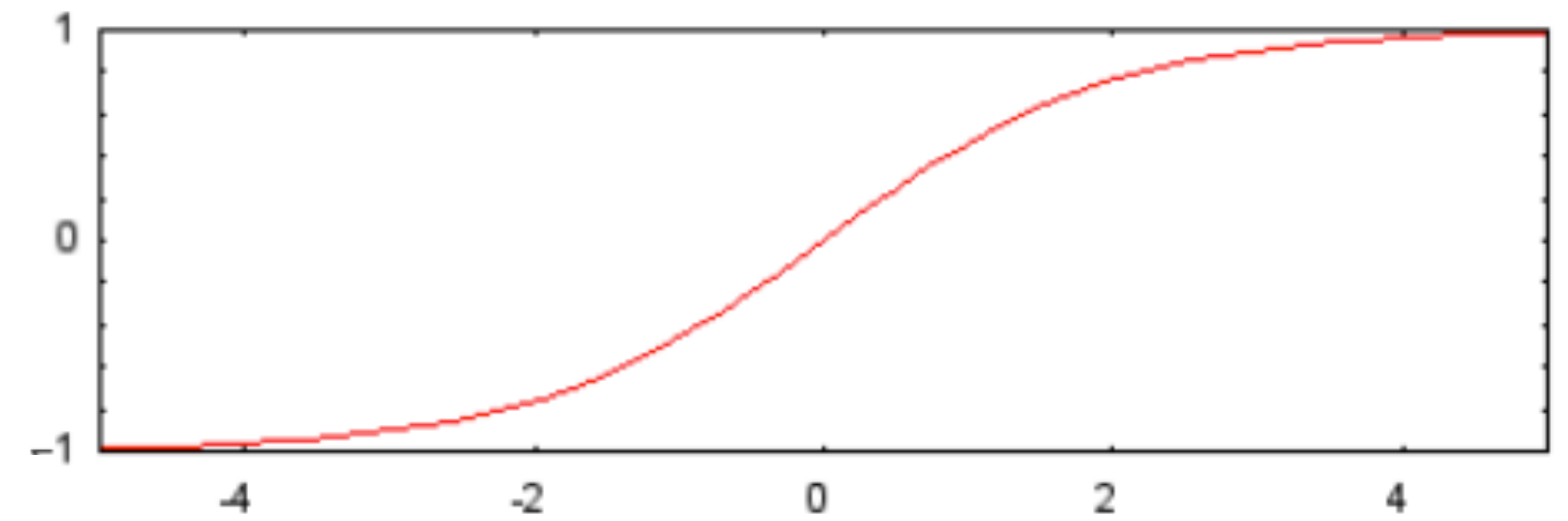


Fonctions d'activation

Fonction tanh

$$g(z) = \tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$$

- La fonction *tanh* peut être utilisé pour les couches internes et comme une fonction logistique pour la couche d'output.



Dérivée : $g'(z) = 1 - (g(z))^2$

Fonctions d'activation

Fonction *Softmax* (σ)

$$\sigma(\vec{\mathbf{z}})_j = \frac{e^{x_j}}{\sum_{k=1}^K e^{z_k}} \text{ for } j = 1, \dots, K$$

Exemple : Classification de 6 races de chiens a partir de photos.

On doit pouvoir prédire une race sur les 6 a l'output du réseau !

- La fonction *Softmax* est une généralisation de la fonction logistique qui s'applique à un vecteur.
- Il peut être utilisé pour prédire des outputs de classes multiples (et pas simplement une prédiction *binaire* $[0; 1]$).
- La somme des éléments du vecteur résultant est **1** et chaque éléments peut être comparé à un pourcentage.

$$\vec{\mathbf{z}} = [1; 3; 2.5; 5; 4; 2]$$

$$\sigma(\vec{\mathbf{z}})_j = [0.01; 0.08; 0.04; 0.60; 0.22; 0.03]$$



La prédiction la plus forte est sur l'index 3, donc la race 4 est prédite !

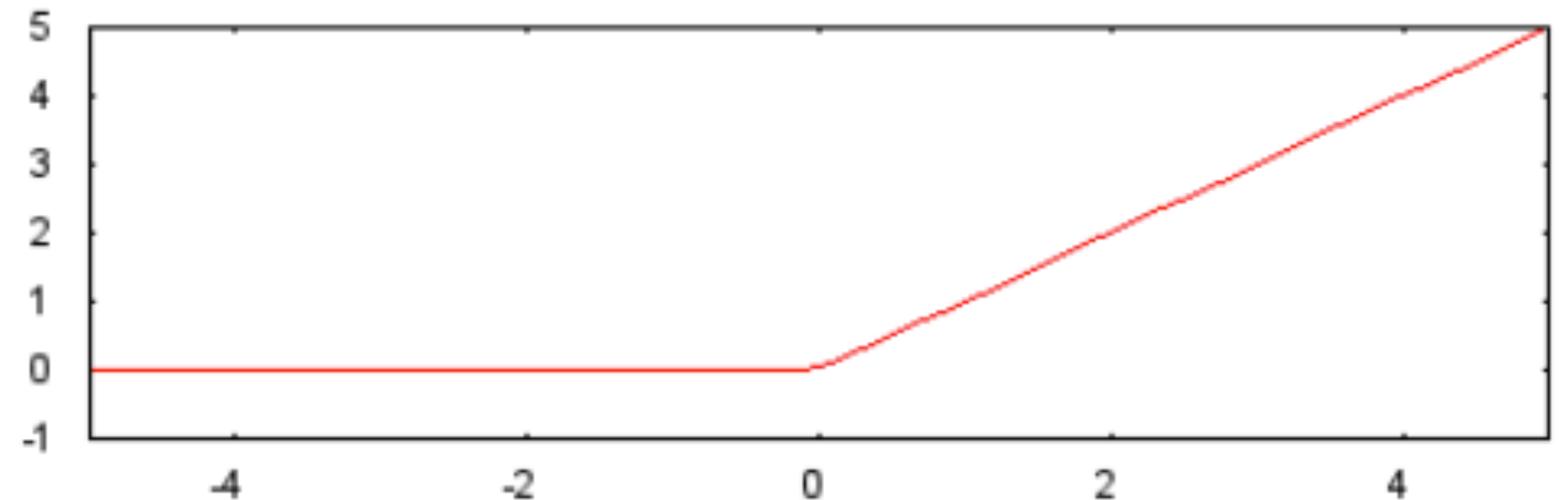
Fonctions d'activation

Fonction *ReLU*

$$g(z) = \max(0, z)$$

- Fonction d'activation très populaire.
- Pratique pour les valeurs z très grandes ou très petites.
- Dérivée facile à calculer.

$$\text{Dérivée : } g'(z) = \begin{cases} 0 & z < 0 \\ 1 & z \geq 0 \end{cases}$$

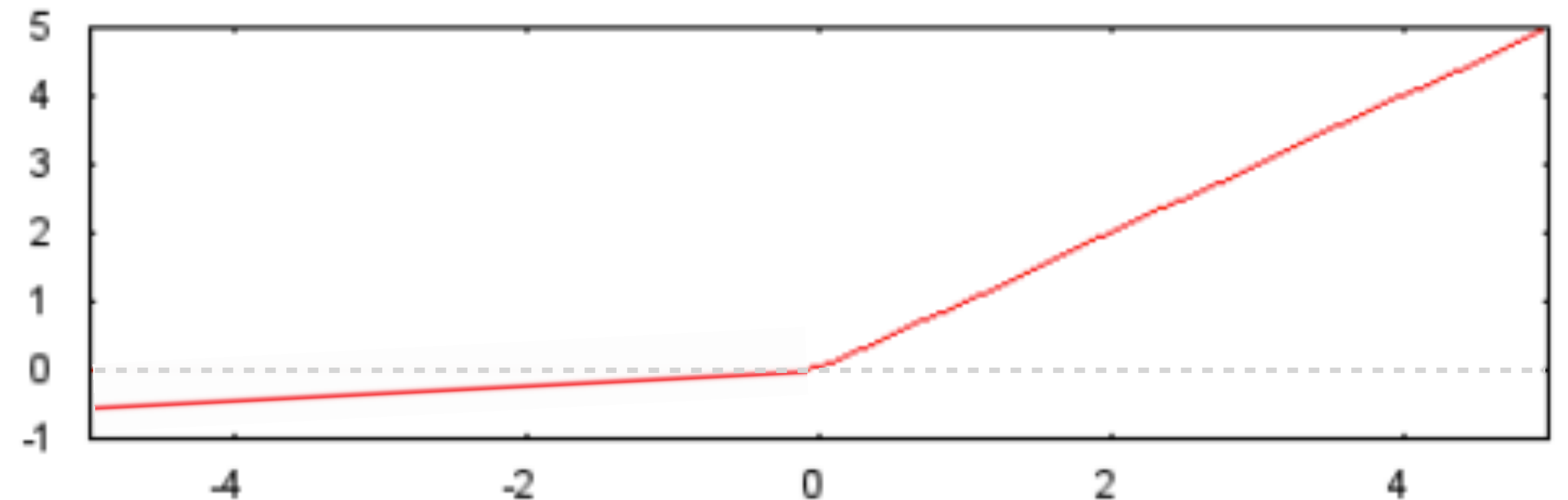


Fonctions d'activation

Fonction *Leaky ReLU*

$$g(z) = \max(0.01z, z)$$

- Peut servir dans des cas pratiques avec beaucoup de valeurs négatives dans les inputs.
- Le gradient 0 du ReLU peut ralentir fortement l'apprentissage.

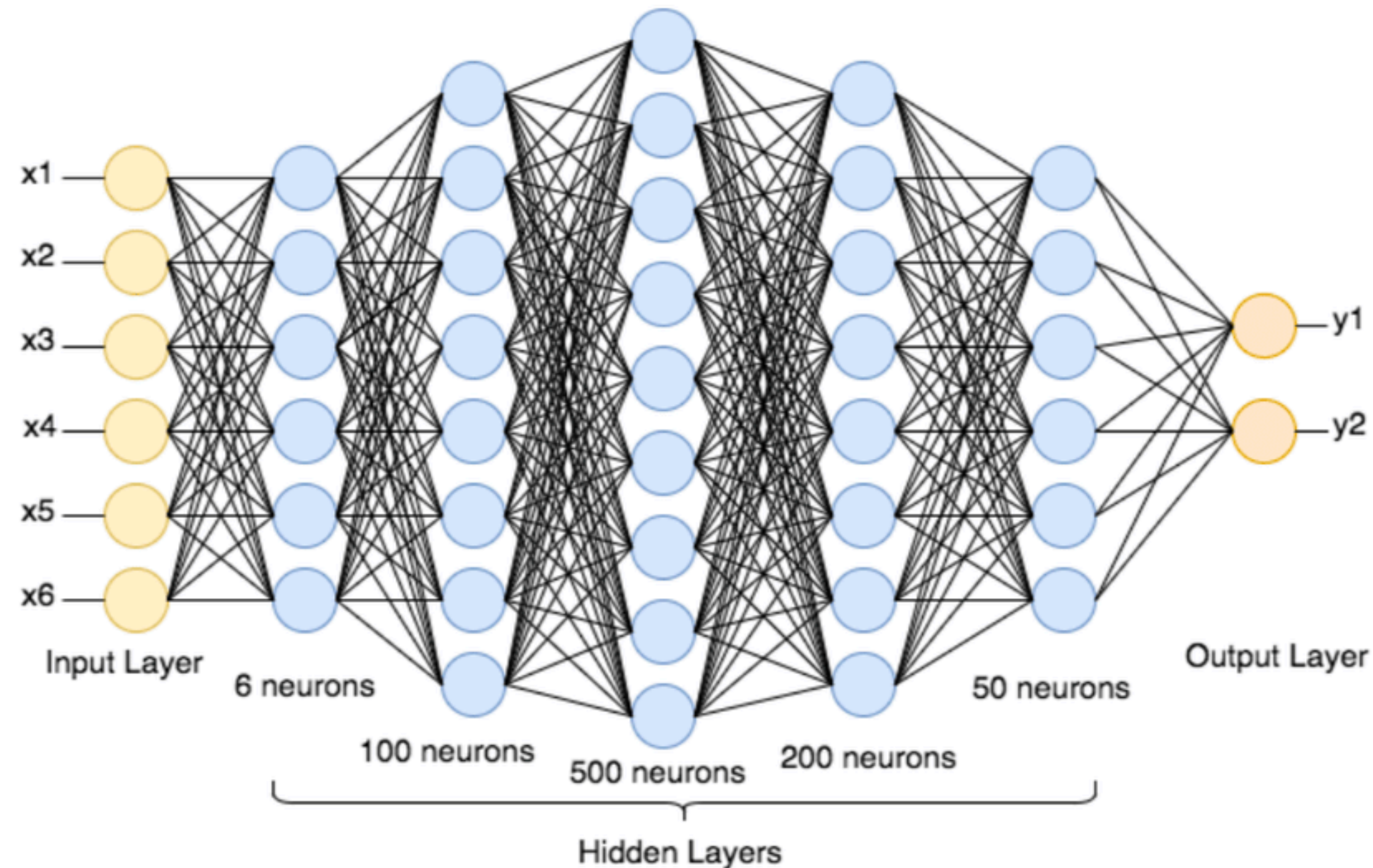


$$\text{Dérivée : } g'(z) = \begin{cases} -0.01 & z < 0 \\ 1 & z \geq 0 \end{cases}$$

Deep Learning

I. Introduction

Le Deep Learning c'est comme les oignons



Ils ont des couches !

Notation

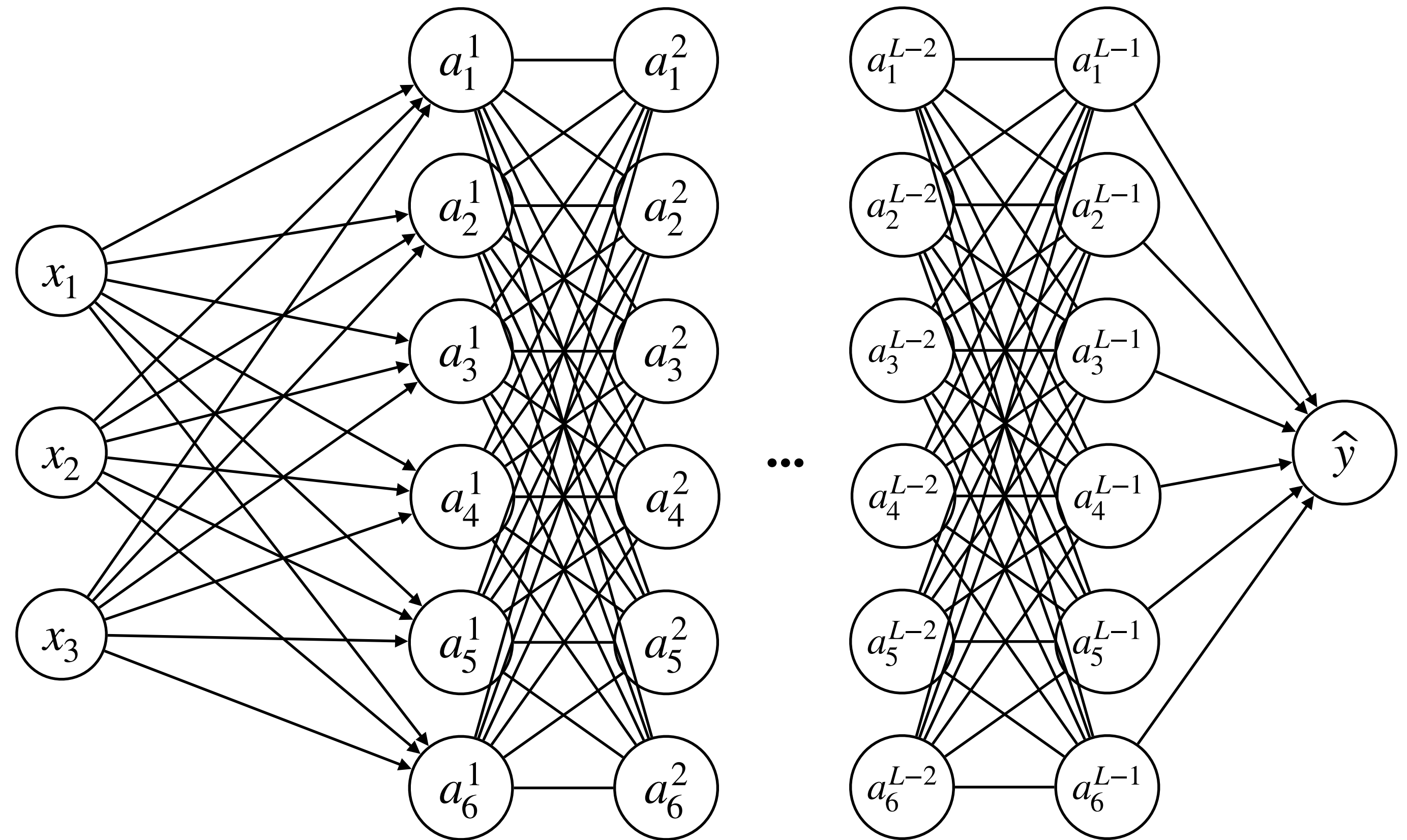
L : # de couches

$n^{[l]}$: Les noeuds dans la couche l

n_i : i -ème noeuds dans les couches

$n_i^{[l]}$: i -ème noeuds dans la couche l

n^x : Couche d'input



Paramètres et complexité

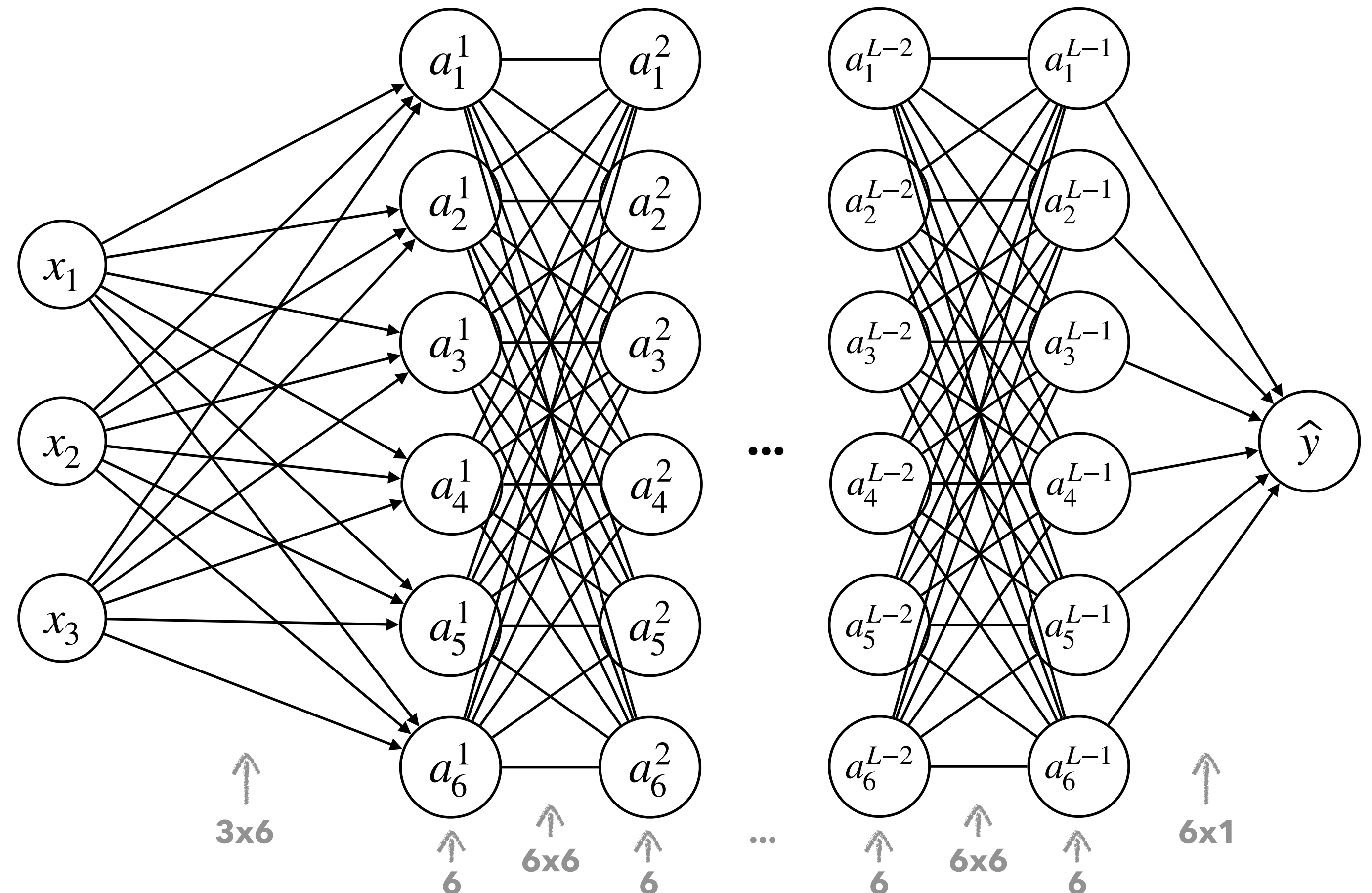
- Nombres de paramètres

= Nombres de connexion entre chaque couche + le nombre de biais dans chaque couche

$$= \sum_{l=2}^L n^{l-1} * n^l + n^{[l]}$$

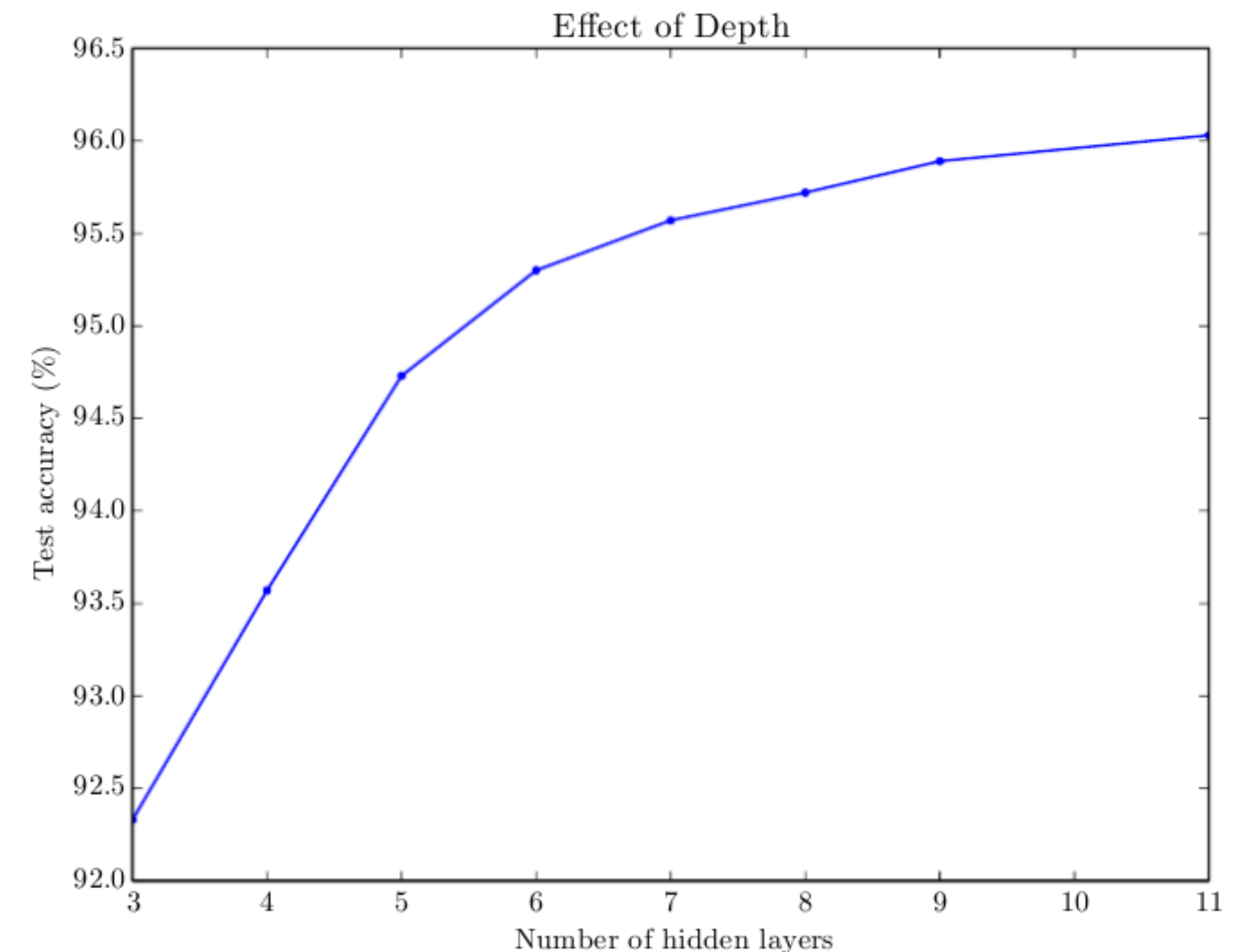
- Plus la complexité d'un modèle est grand, plus il peut apprendre des hypothèses complexes.

- Avoir un grand nombre de noeuds par couches permet d'augmenter fortement le nombre d'inputs.



+ de paramètres === meilleur modèle ?

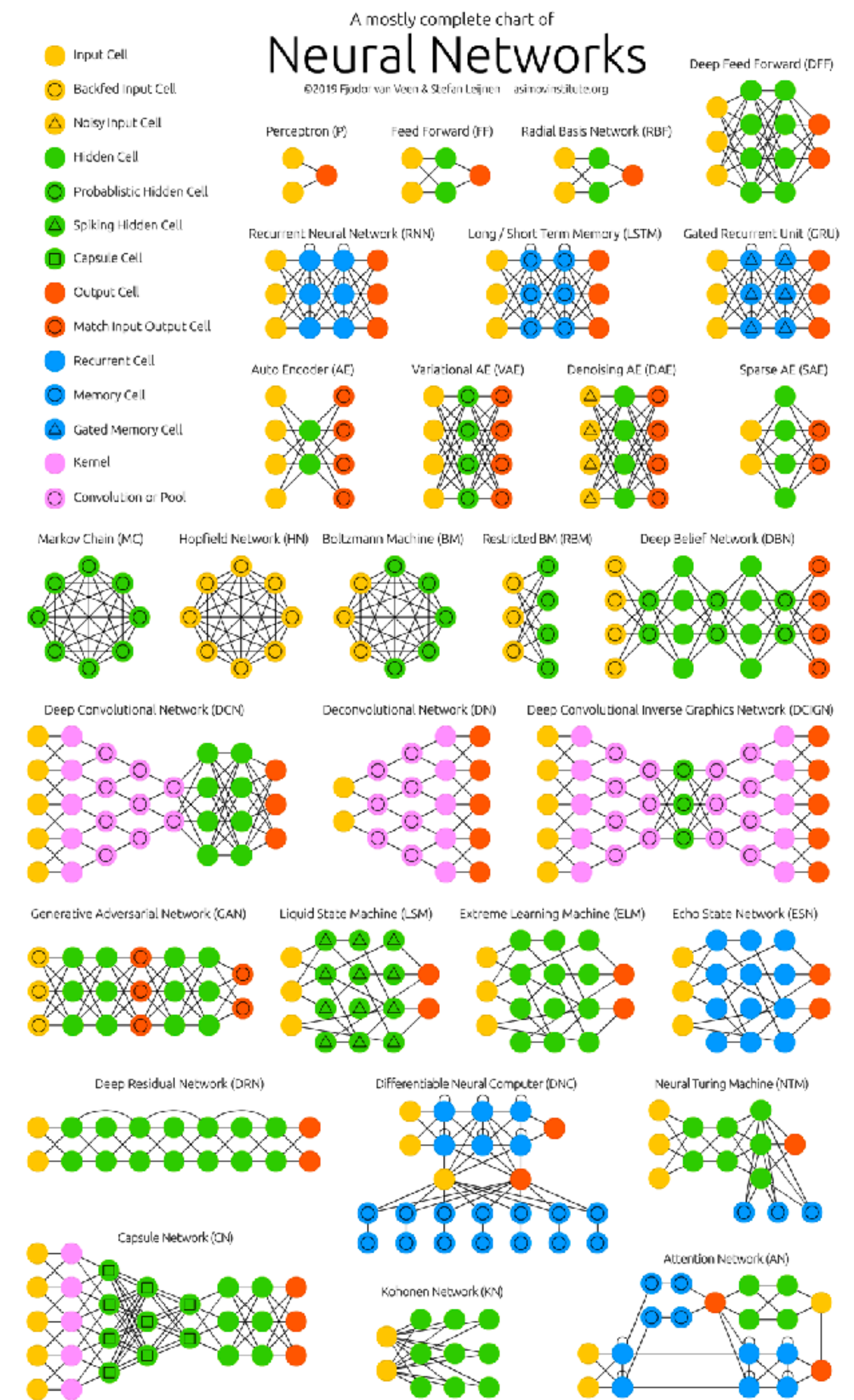
- Même si augmenter le nombre de paramètres a des avantages, il y a aussi des limites :
 - Entraînements plus longs.
 - Nécessité grandissante du nombre de données d'entraînement.
 - Une difficulté grandissante à entraîner les couches primaires.
 - Une gain de performance proche de zero (et des fois négatif) au bout d'un certains nombres de couches.



The Neural Network Zoo

- Depuis la montée en popularité du machine learning, la famille des Réseaux Neuraux a fortement grandi (même si beaucoup ont pas mal d'années d'existences).
- Beaucoup de ces familles de réseaux sont présents et/ou inspirées par les réseaux neuronaux du cerveau.
- Une description de chaque famille est présent dans le lien.

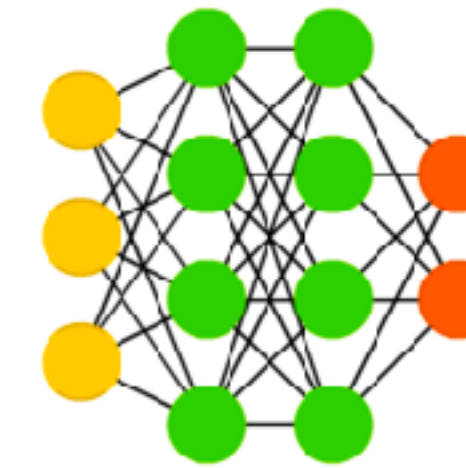
<https://www.asimovinstitute.org/neural-network-zoo/>



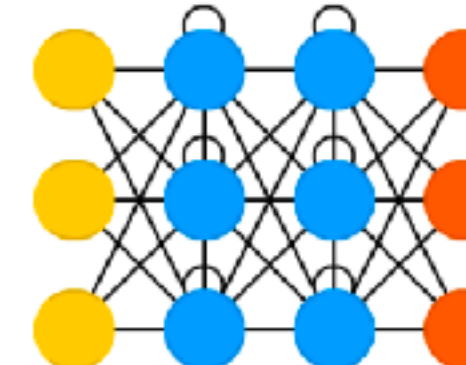
Déclinaison en Deep Learning

- Le **MLP** (Multi-Layered Perceptron) est un **DFF**.
Le signal va de l'avant à l'arrière, depuis l'input vers l'output.
- Les **RNNs** possèdent des états permettant aux inputs précédents d'influencer les inputs futur. Très utile pour des données où l'ordre est important. (ex : une phrase, des jeux, ...)
- Les **DCNs** possèdent des *kernels* permettant de regrouper des groupes de noeuds de la couche précédentes. Très utile pour les inputs possédant énormément de features. (ex: les images)

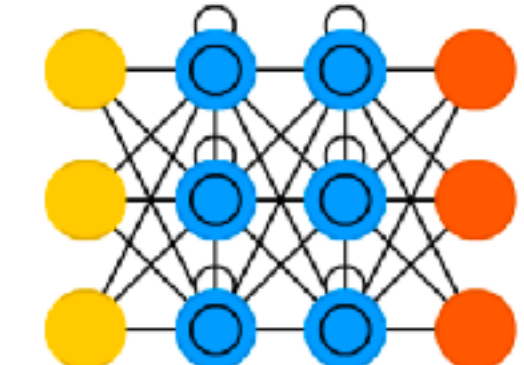
Deep Feed Forward (DFF)



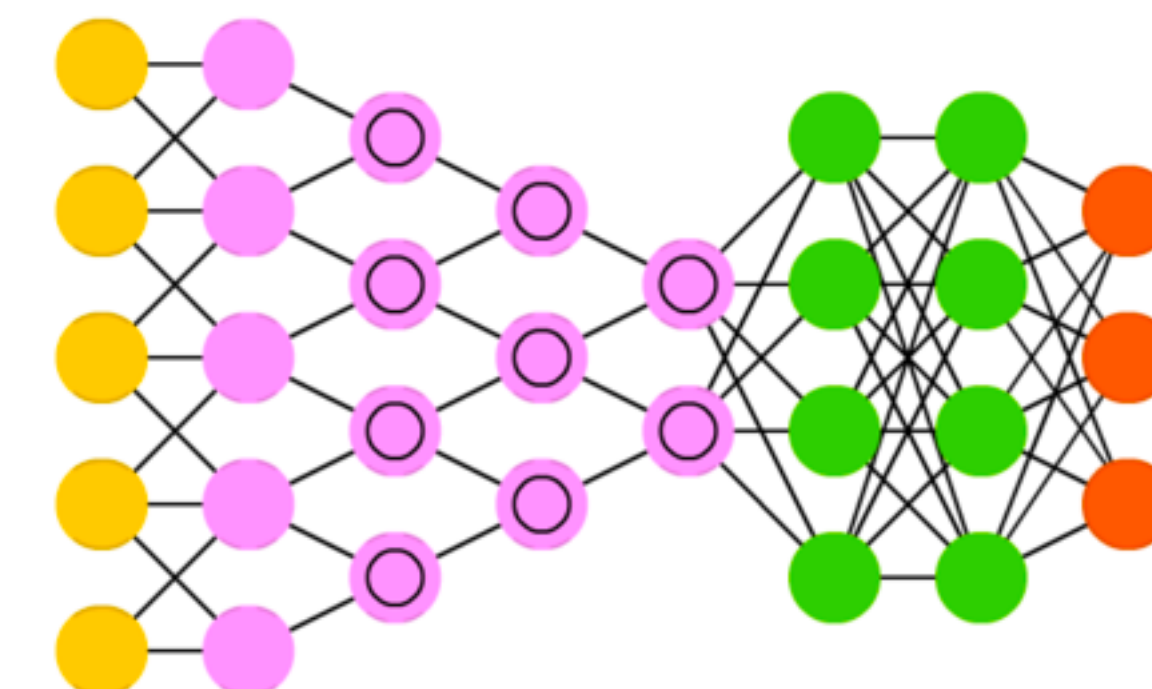
Recurrent Neural Network (RNN)



Long / Short Term Memory (LSTM)



Deep Convolutional Network (DCN)



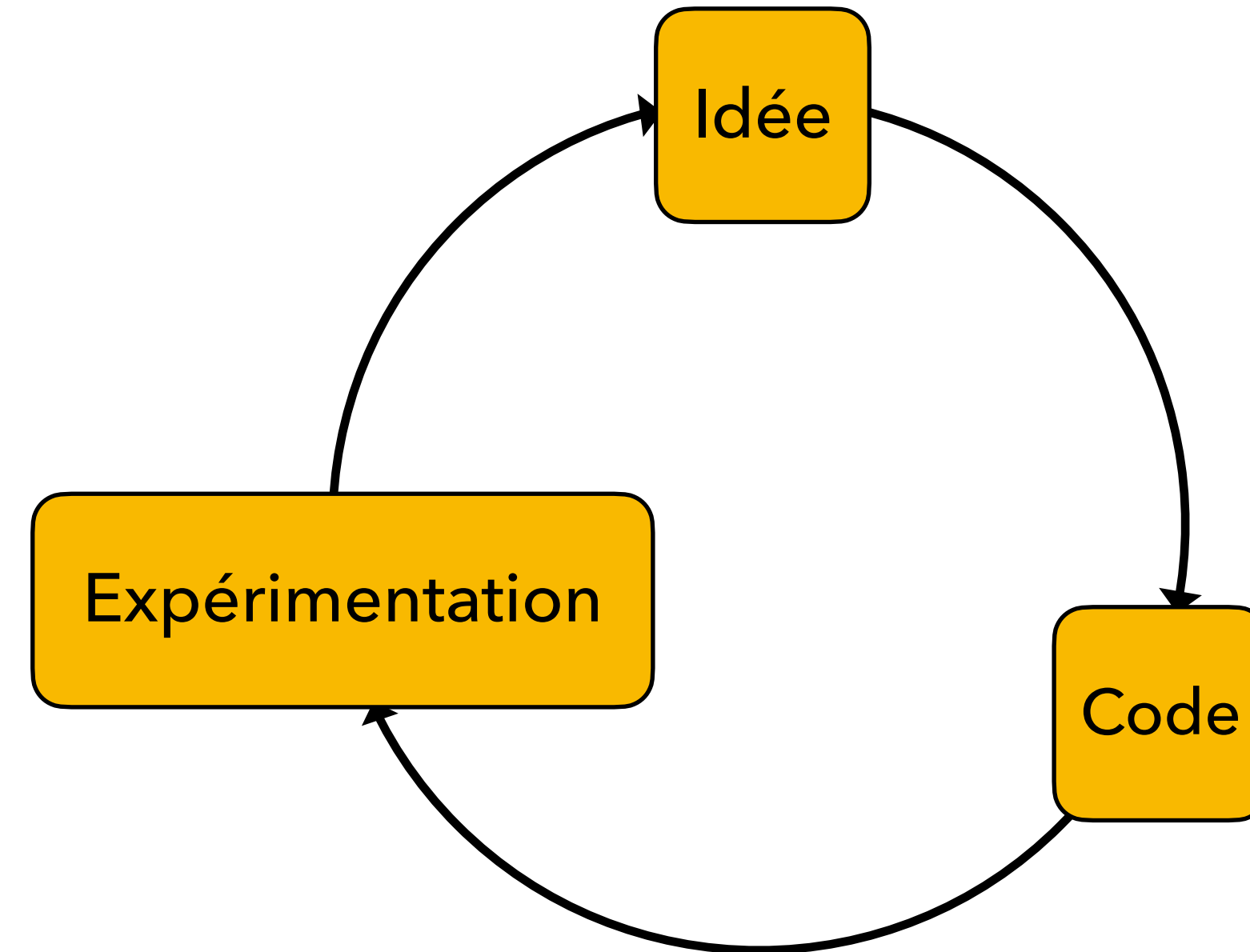
Deep Learning

II. Construction & Optimisation

Différentes familles d'algorithmes, un nombre de couches, un nombre de noeuds par couches, divers types de noeuds, plusieurs fonctions d'activations, un ratio d'apprentissage, régularisation, un taux de dropout, initialisation des poids, des lettres grecques partout, bulbizarre, carapuce ou salamèche,

Comment choisir?

Le Machine Learning pratique est un procédé **fortement** itératif.



L'itération de ce cycle est ce qui nous permet de calibrer nos hyperparamètres.

Hyperparamètres et Structure

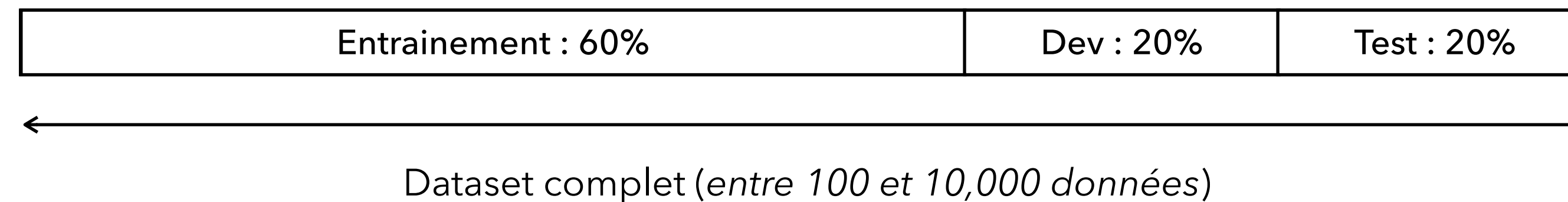
Hyperparamètres :

- L'ensemble de paramètres déterminants l'architecture du réseaux de neurones *avant* le début de l'entraînement.
- La forte quantités d'hyperparamètres dans les réseaux ne rend pas évident la direction à prendre pour **construire son réseaux face à une hypothèse**.
- Les hyperparamètres ne **sont pas simple à voir/déterminer** au début d'un projet.
- Il existe plusieurs méthodes pour adapter la structure de son modèle, mais les possibilités sont tellement grandes que **l'intuition, l'expérience et la maîtrise** de cette science reste les meilleures guides en optimisation.

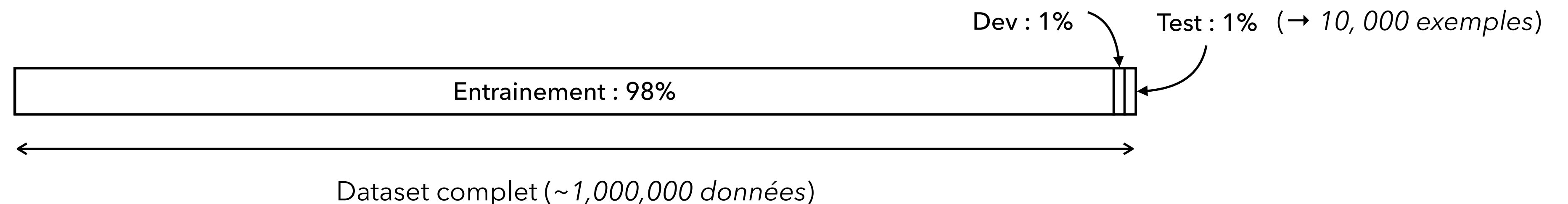
Construction & Optimisation

I. Datasets : Comment bien diviser un dataset?

- Il y a quelques années :



- A l'ère de la BigData, le facteur le plus important dans la division d'un dataset est que le jeu de test doit être assez grand pour représenter la diversité de la data et l'utilisation finale de l'hypothèse.



Peter Norvig (2015) - Research Director at Google

**"We don't have better algorithms.
We just have more data."**

Peter Norvig (2015) - Research Director at Google

**"We don't have better algorithms.
We just have more data."**

**Most of the times.
Less true in 2021.**

Construction & Optimisation

II. Familles d'algorithmes : Spécialisations en ML

- Selon la structure de la donnée et l'hypothèse, certains algorithmes de ML peuvent être plus ou moins adaptés.
- Même si des grandes familles se sont spécialisées dans certains type de données, il est important d'explorer plusieurs structures, algorithmes, innovations, etc pour la résolution d'un problème.

Data / Hypothèse	Familles de Machine Learning efficace
Classification multiclassés, régression,...	Regression lineaire, Multi Layered Perceptron (MLP), Softmax
Analyse de phrase, traduction, ...(NLP)	Residual NN, Recurrent NN (RNN), Long/Short Term Memory NN,
Reconnaissance d'images	Convolution Neural Networks (CNN), deep CNN
Détection de fraudes, d'erreurs, ...	Random Forest, Decision Tree (Adaboost, XGBoost)
Generation, style transfert	Generative Adversarial Networks (GAN), Attention Networks (AN)

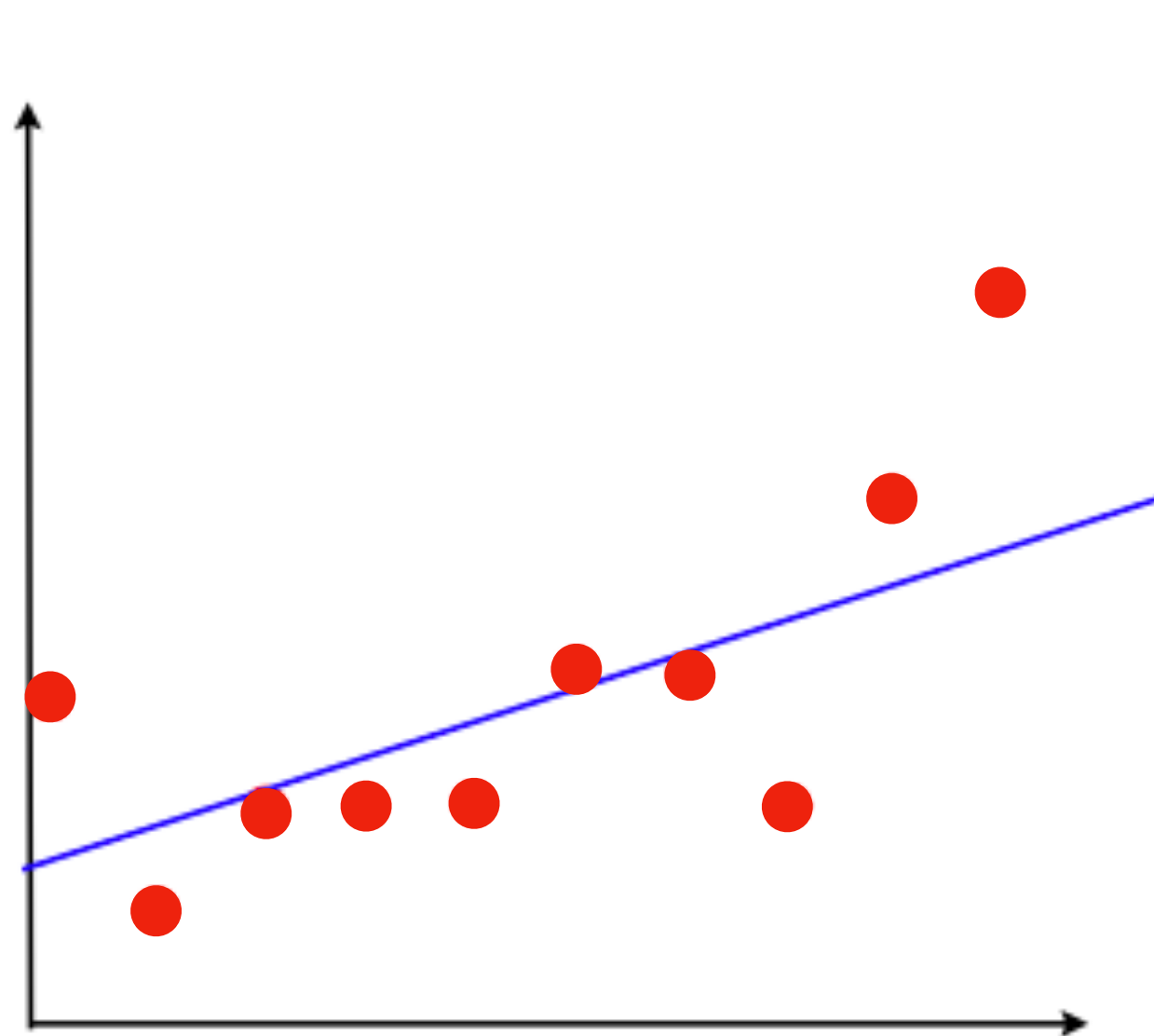
Construction & Optimisation

III. Biais et Variance : Des mesures de généralisation

Recap

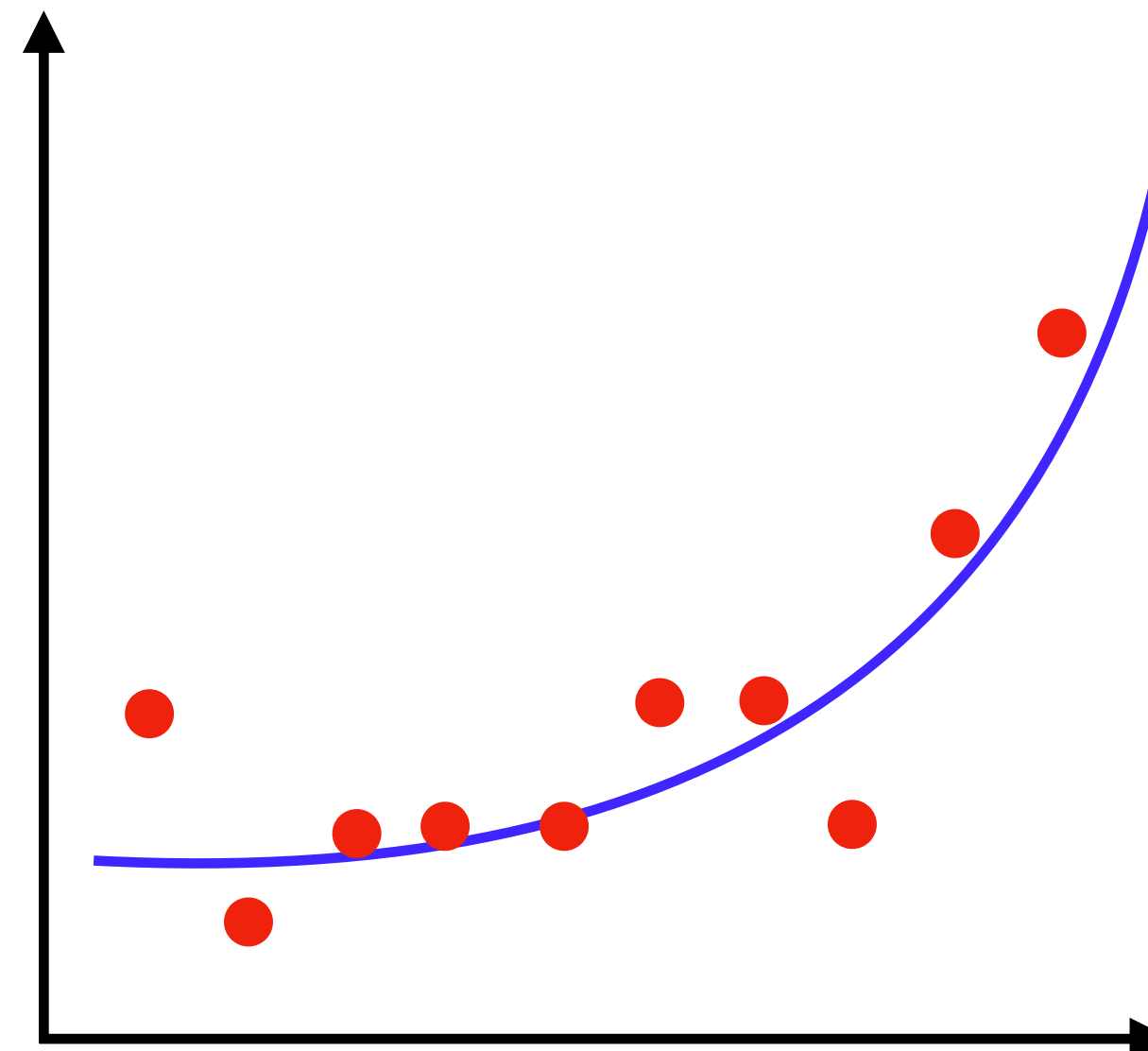
1. Datasets
2. Famille d'algorithme
3. Biais et Variance
 1. Des mesures de généralisation
 2. Un arbre de décisions
 3. Se comparer aux humains
4. Choisir parmi différentes structures
5. Et beaucoup d'autres

- Recap :

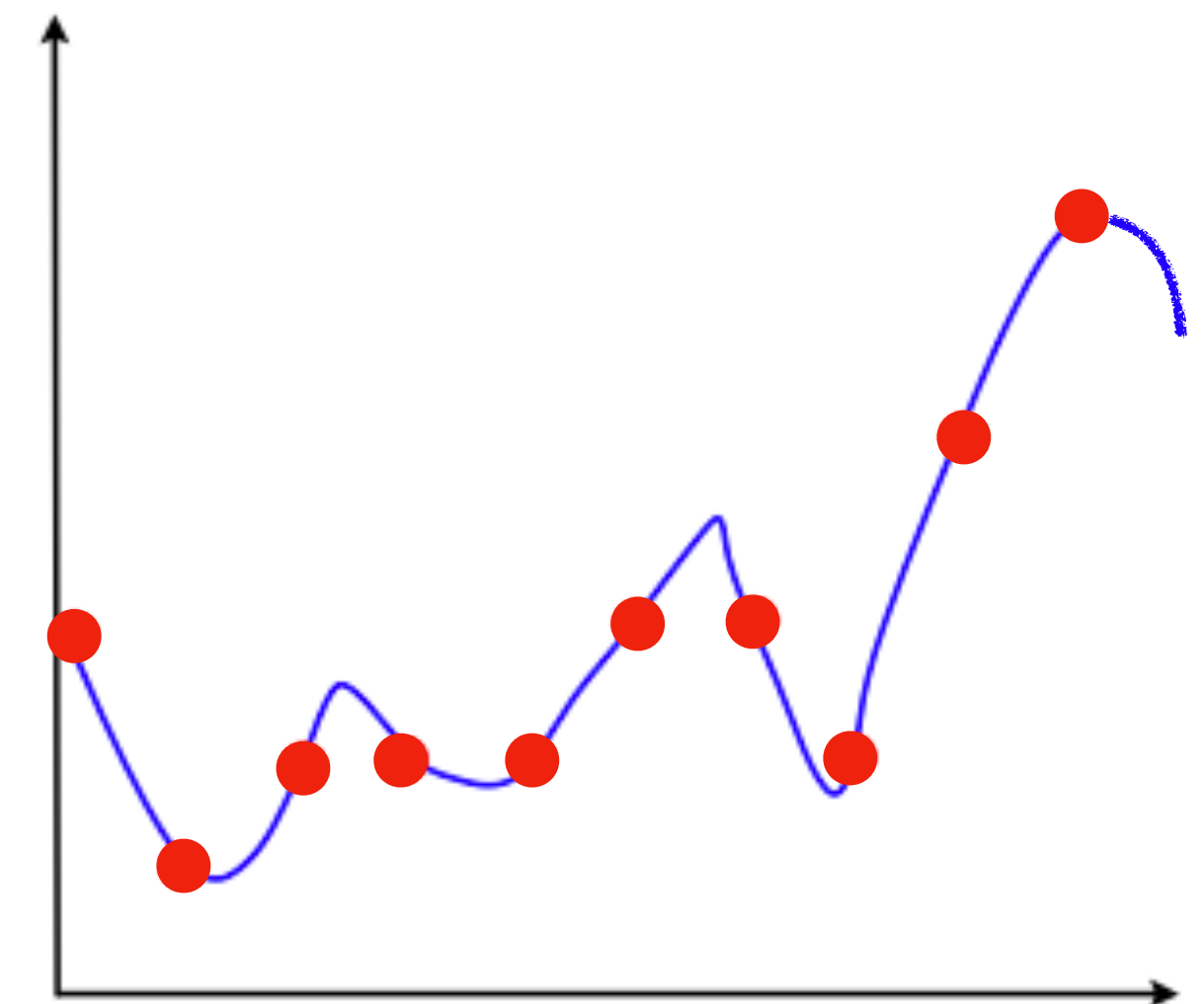


Haut Bias

Underfitting

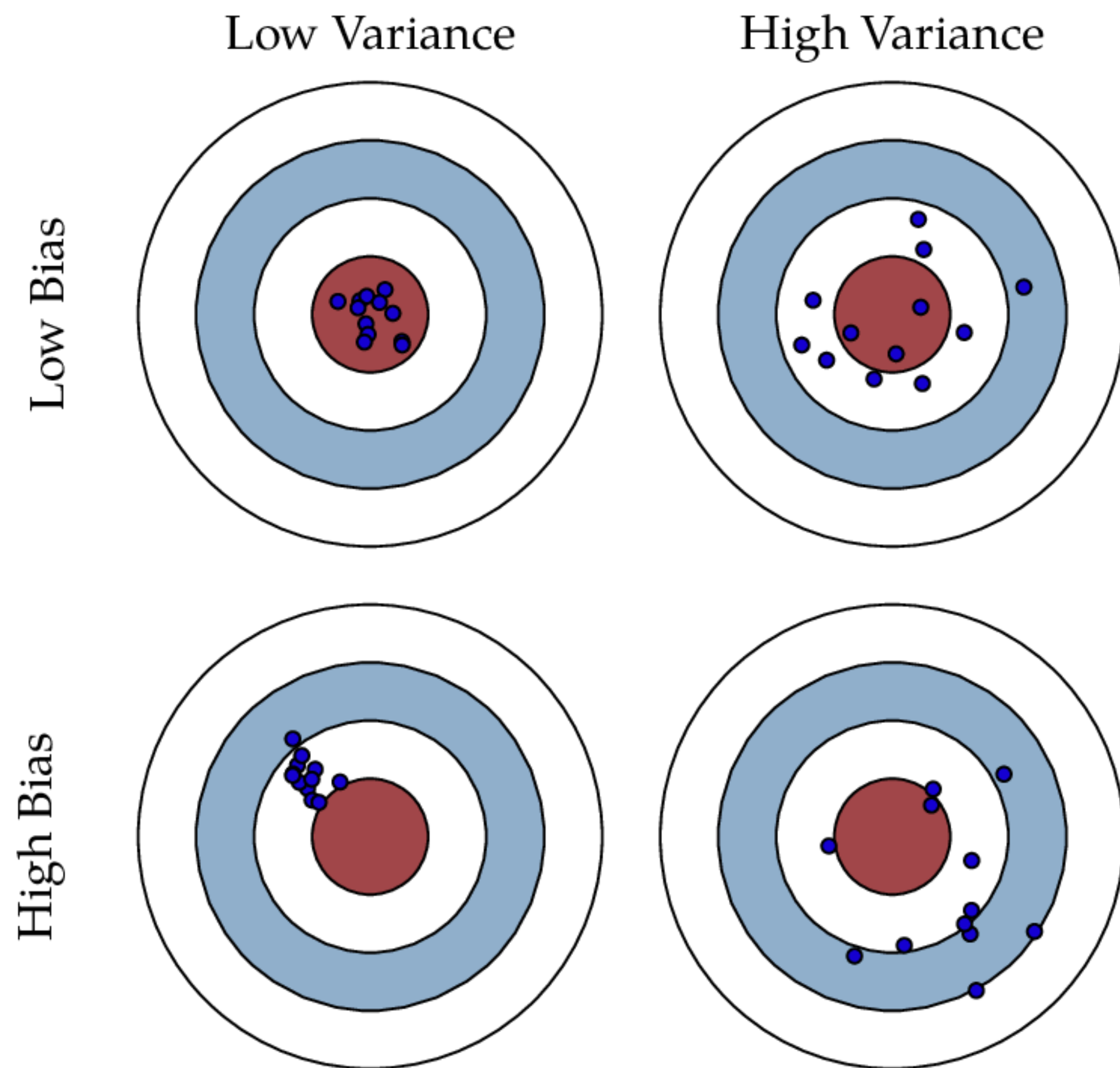


"Just Right"



Haute Variance


Overfitting



Construction & Optimisation

III. Biais et Variance : Des mesures de généralisation

- Reconnaître le biais et la variance à travers la performance d'un modèle.
- Illustration :

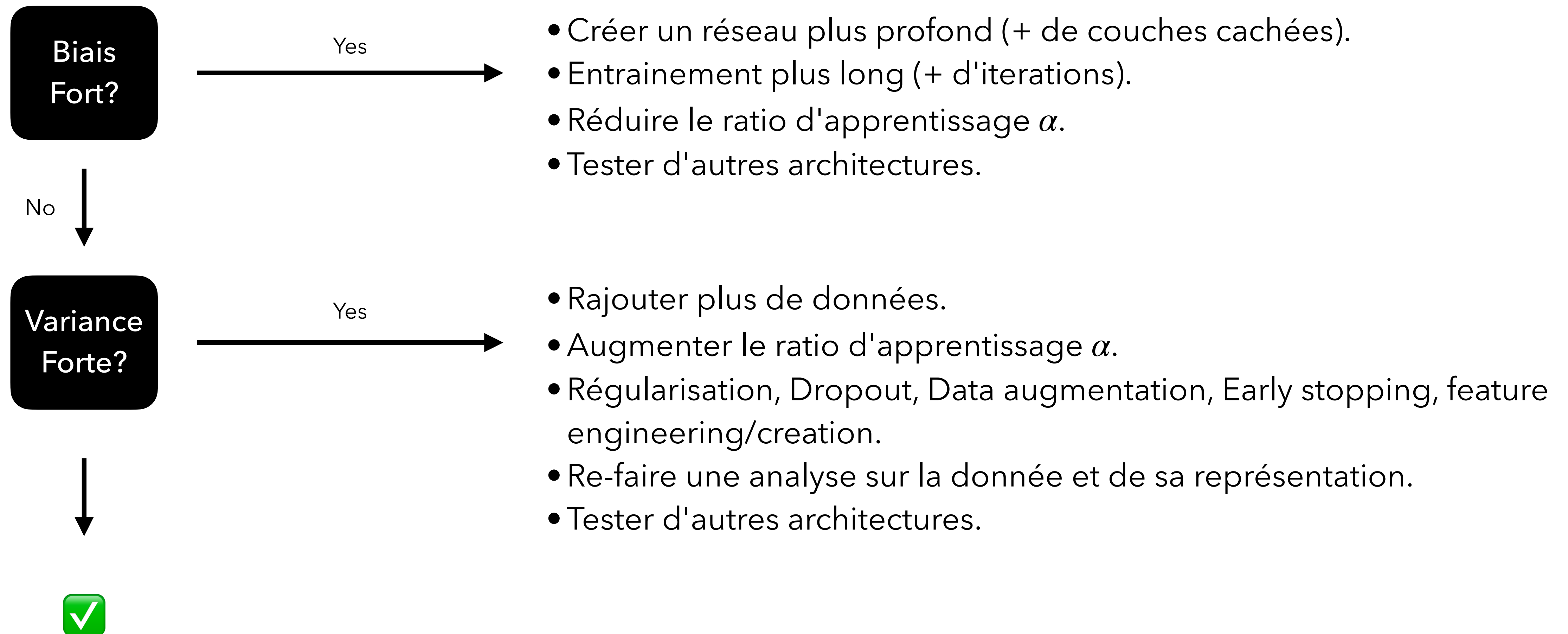
	Structure 1	Structure 2	Structure 3	Structure 4
Erreur au set d' <i>Entrainement</i>	1 %	15 %	15 %	0.5 %
Erreur au set de <i>Dev</i>	11 %	16 %	30 %	1.5 %
Analyse	Variance Forte Overfit a l'entrainement	Biais Fort Underfit a l'entrainement	Variance Forte & Biais Fort	

Recap

1. Datasets
2. Famille d'algorithme
3. **Biais et Variance**
 1. Des mesures de généralisation
 2. **Un arbre de décisions**
 3. Se comparer aux humains
4. Choisir parmi différentes structures
5. Et beaucoup d'autres

Construction & Optimisation

III. Biais et Variance : Un arbre de décisions



Construction & Optimisation

III. Biais et Variance : Se comparer aux humains

- La performance humaine (le taux d'erreurs moyen d'humains à une tâche) est souvent l'objectif pour la plupart des modèles :
 - Le niveau humain est souvent une référence pour l'erreur de Bayes.
 - C'est aux alentours de ce niveau que le modèle devient intéressant à utiliser.
 - Si le niveau maximale est dépassé alors il devient difficile de savoir si l'erreur vient de la variance ou du biais.

Erreur d'Humains	1 %	7,5 %	7,5 %
Erreur d'Entrainement	8 %	8 %	4 %
Erreur de Dev	10 %	15 %	5 %

Se concentrer sur le Biais.

Se concentrer sur la Variance.

Difficile a dire.
Une analyse des erreurs est a faire.

Construction & Optimisation

IV. Choisir parmi différentes structures

- Il existe plusieurs métriques pour évaluer la performance d'un modèle, et le pourcentage d'erreur n'est qu'une moyenne.

$$Precision = \frac{VP}{VP + FP}$$

$$Recall = \frac{VP}{VP + FN}$$

$$F_{Score} = 2 \frac{Pre * Rec}{Pre + Rec}$$

	1	0
$\hat{1}$	Vrai Positif	Faux Positif
$\hat{0}$	Faux Négatif	Vrai Négatif

- Exemple : Quel classifieur est le meilleur?

Classifieur	Precision	Recall	F Score	Temps d'exécution
A	0,95	0,90	0,92	80 ms
B	0,98	0,85	0,91	95 ms
C	0,97	0,93	0,95	1,500 ms

Recap

1. Datasets
2. Famille d'algorithmes
3. Biais et Variance
 1. Des mesures de généralisation
 2. Un arbre de décisions
 3. Se comparer aux humains
4. Choisir parmi différentes structures
5. Et beaucoup d'autres

Construction & Optimisation

V. Et beaucoup d'autres

- Normalisation des inputs
- Initialisation aléatoire des poids
- Checker les gradients
- Entraînements par batch
- Gradient Descent avec momentum
- Learning rate decay
- Utiliser des échelles appropriés pour chaque hyperparamètre
- Exploding and Vanishing Gradient
- Dead ReLU units