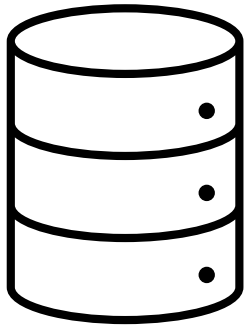
Deep Learning

DU IA Santé

Tutoriel

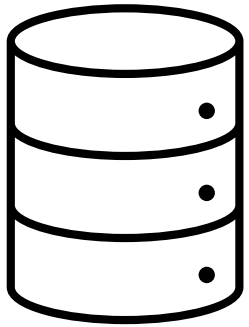
1. Les briques du Deep Learning (Aussi vrai pour beaucoup de méthode de ML)

Données

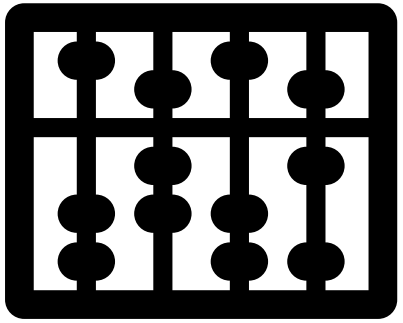


1. Les briques du Deep Learning (Aussi vrai pour beaucoup de méthode de ML)

Données

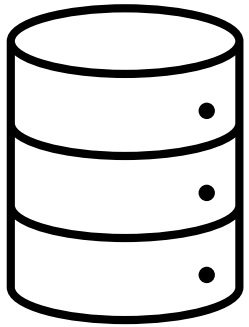


Modèle

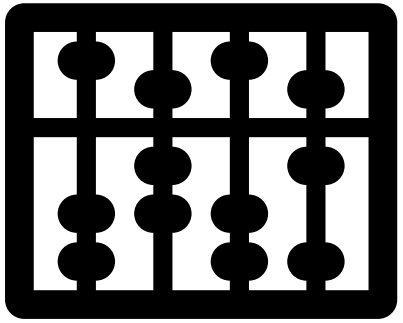


1. Les briques du Deep Learning (Aussi vrai pour beaucoup de méthode de ML)

Données



Modèle



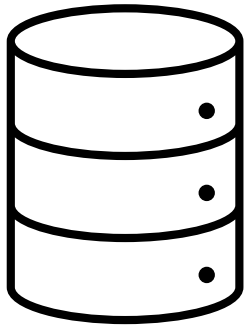
Fonction de perte

Risque

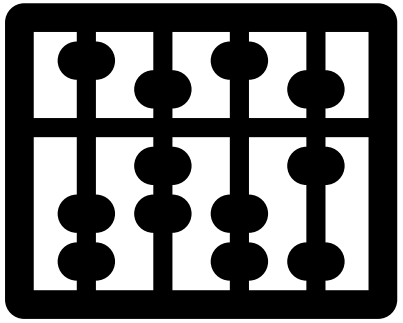
$$\mathcal{R}(\theta) = \mathbb{E}[L(\theta; x)]$$

1. Les briques du Deep Learning (Aussi vrai pour beaucoup de méthode de ML)

Données



Modèle

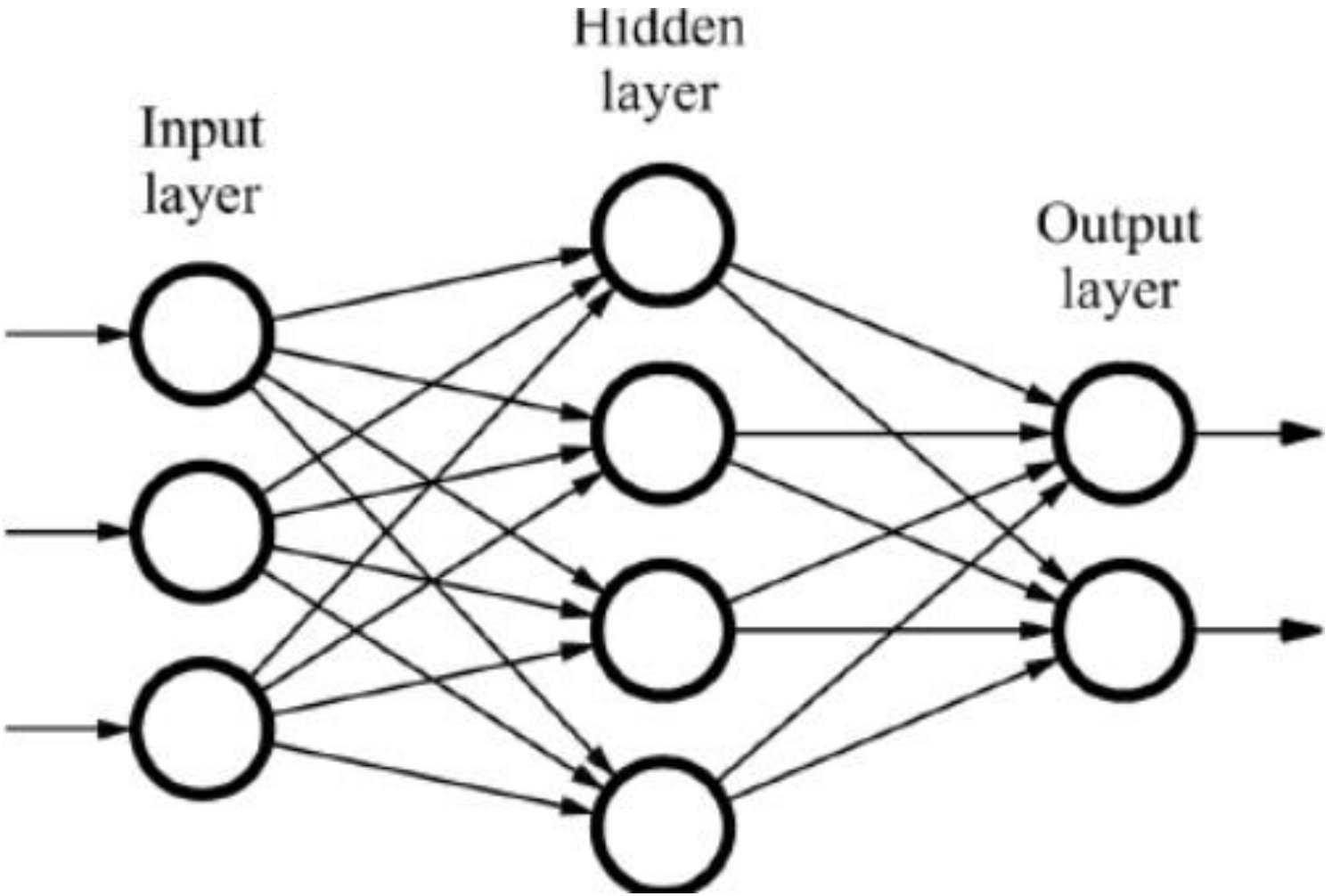


Fonction de perte
Risque

$$\mathcal{R}(\theta) = \mathbb{E}[L(\theta; x)]$$

Optimiseur

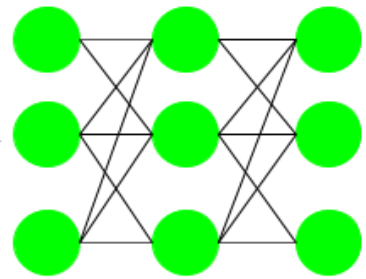
2. Réseaux de neurones



Machine Learning

Input

Feature Extraction



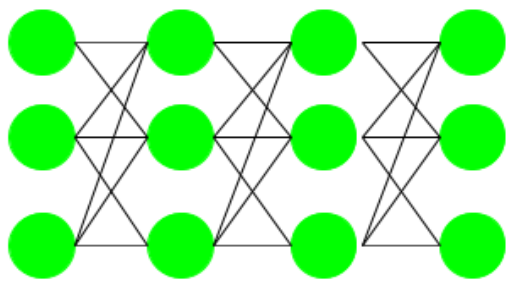
Classification

It's Car

Output

Deep Learning

Input



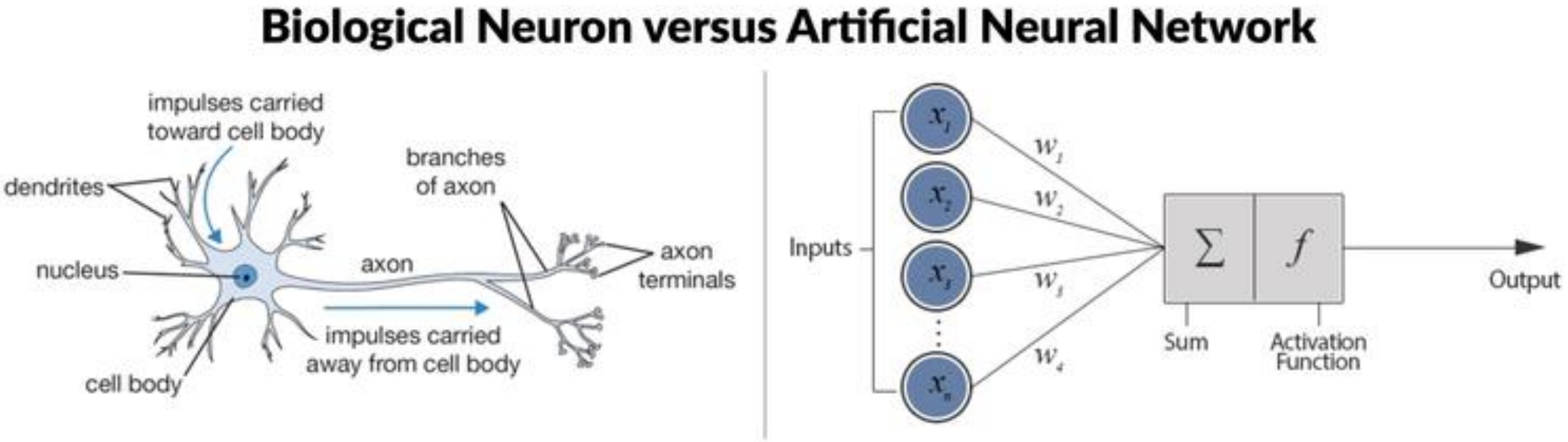
Feature Extraction and Classification

It's Car

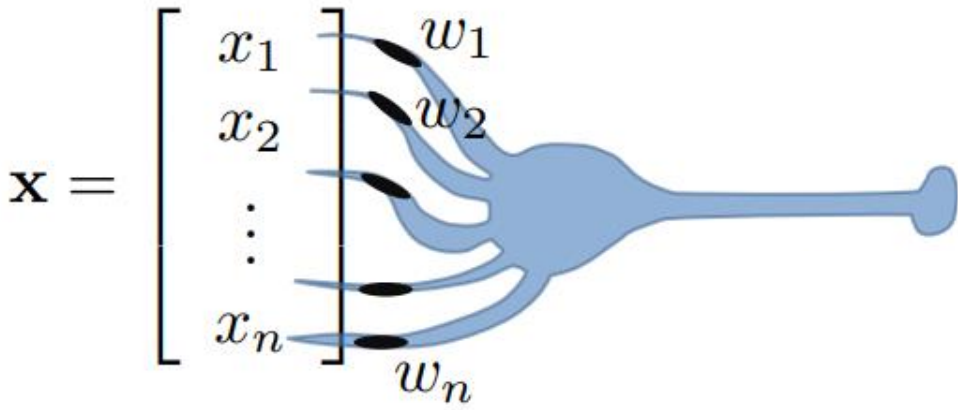
Output

2.a. Perceptron

Rosenblatt, 1957



2.a.Perceptron

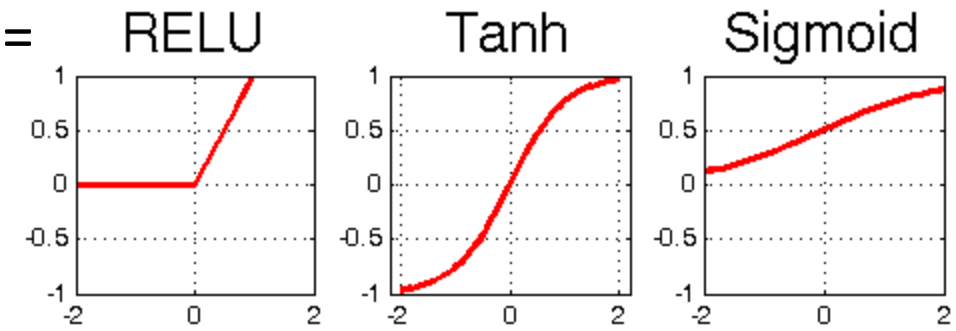


Comme une regression linéaire

$$output = g(w^T x + b)$$

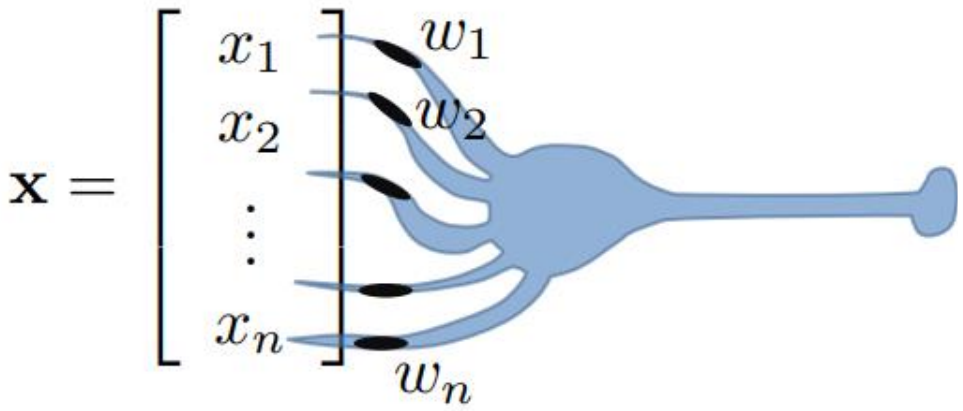
Fonction d'activation

Avec g =



Nombre de paramètres ?

2.a.Perceptron

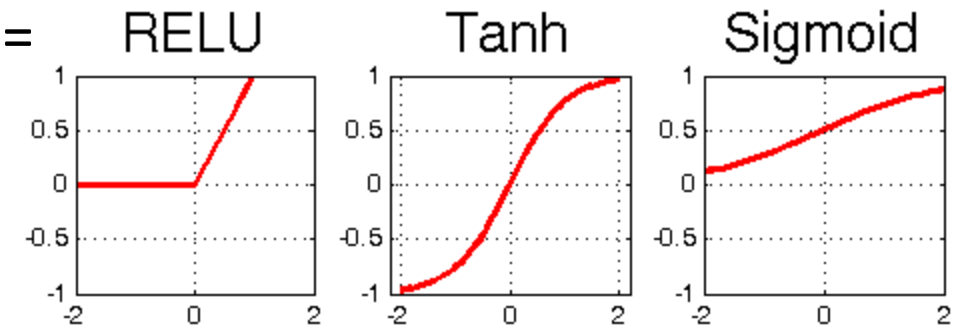


Comme une regression linéaire

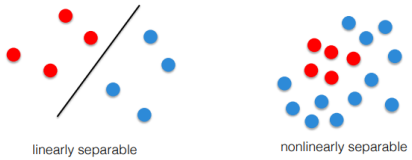
$$output = g(w^T x + b)$$

Fonction d'activation

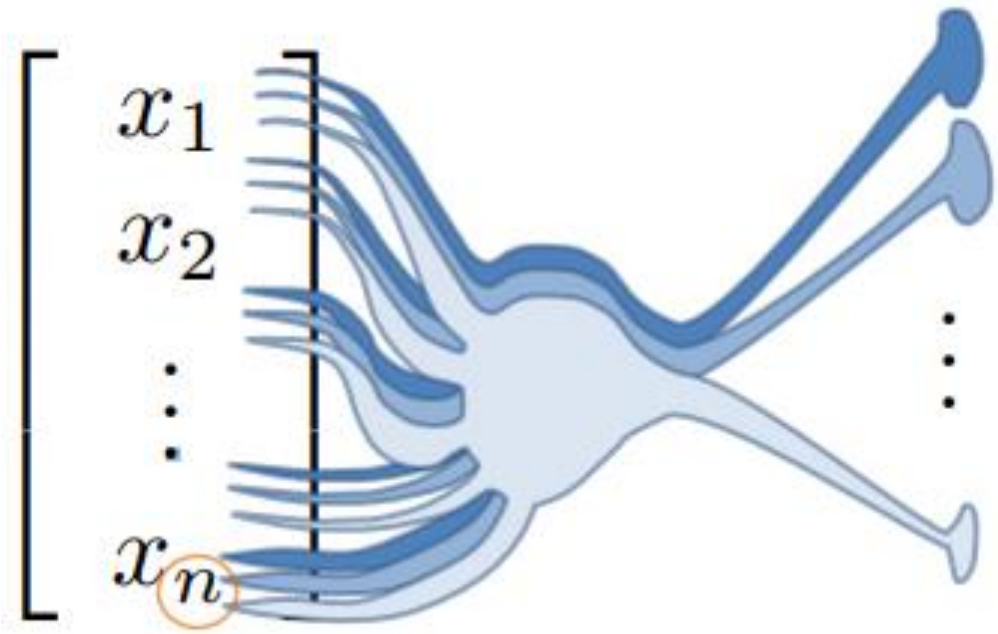
Avec g =



Pour l'instant, cela ressemble énormément à la regression linéaire. Peut-on vraiment gérer des problèmes non linéaire avec ce type de modèle?



2.b. Muti-layer Perceptron



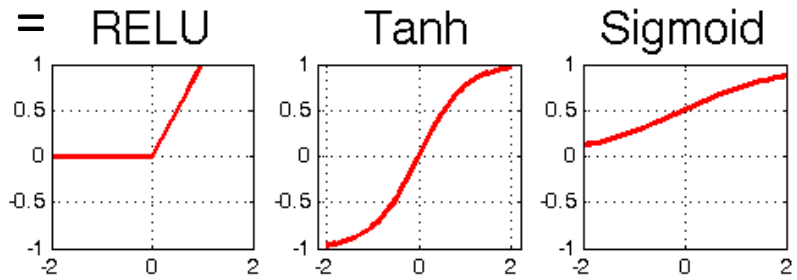
$$output_1 = g(w_1^T x + b_1)$$

$$output_2 = g(w_2^T x + b_2)$$

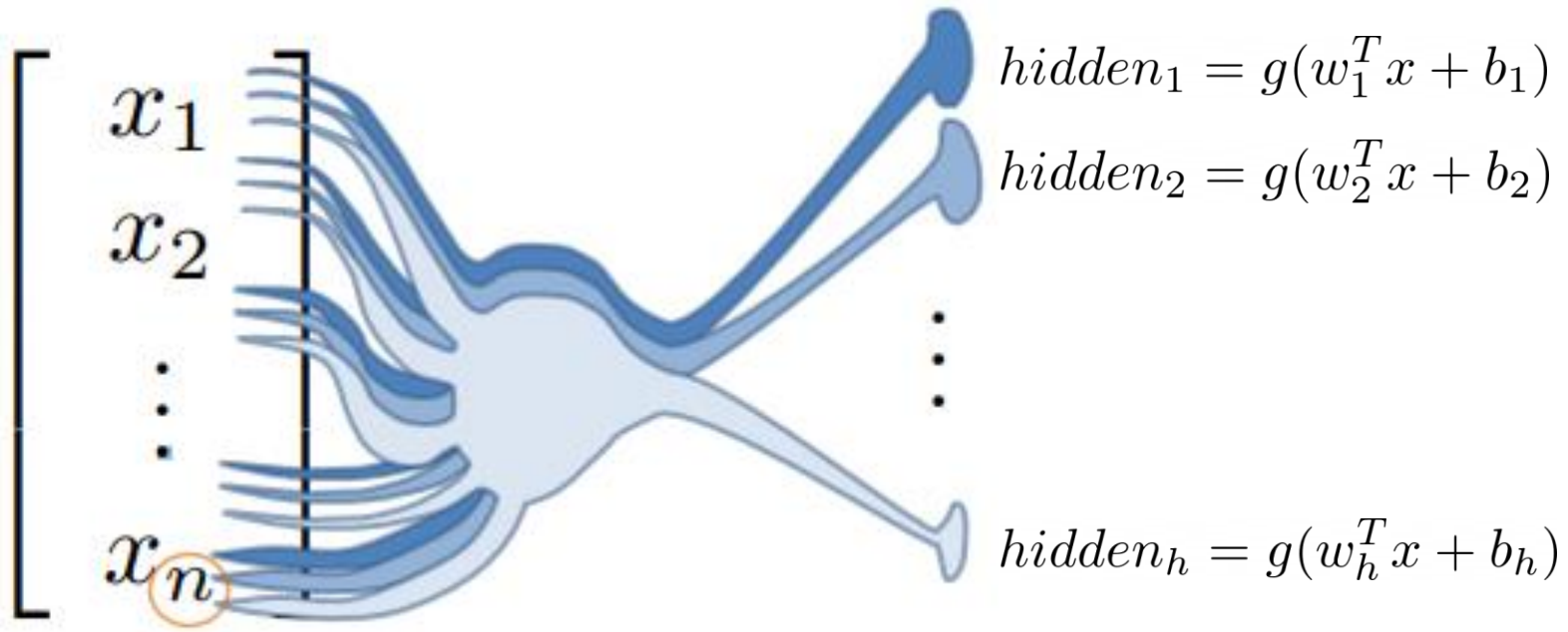
$$output_h = g(w_h^T x + b_h)$$

Nombre de paramètres ?

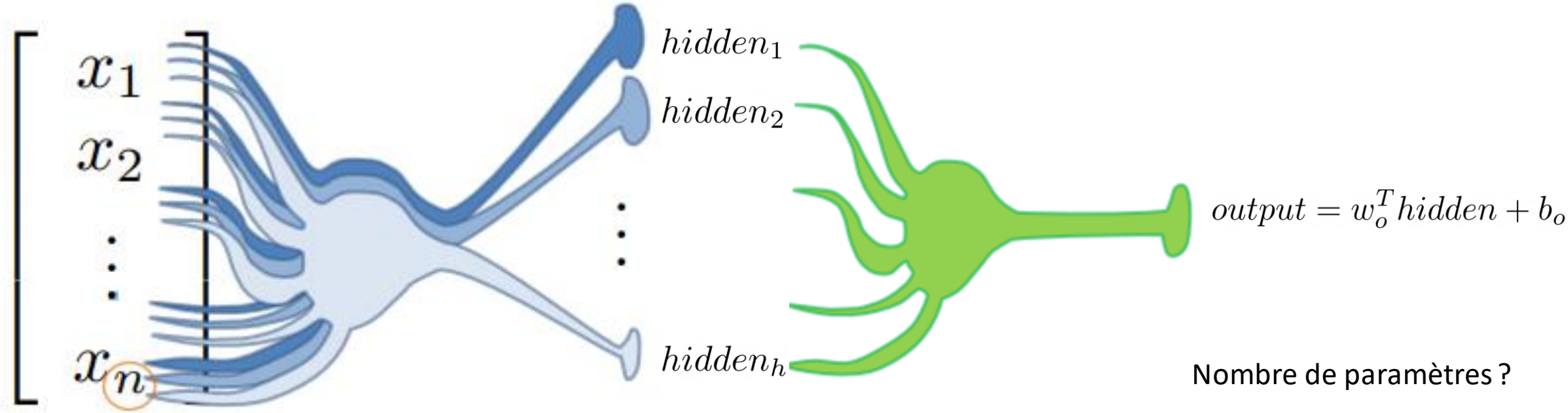
Avec g =



2.b. Muti-layer Perceptron



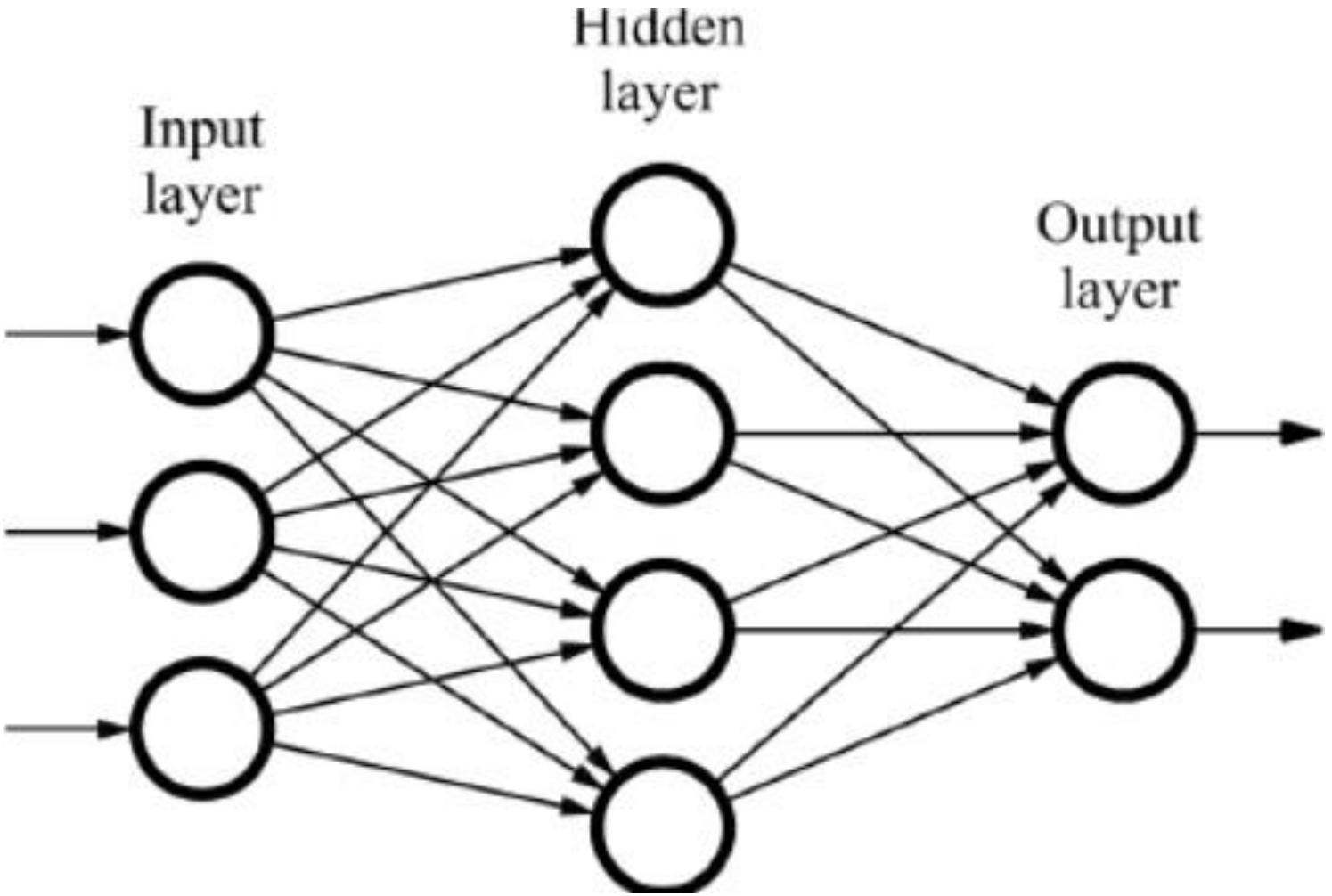
2.b. Muti-layer Perceptron



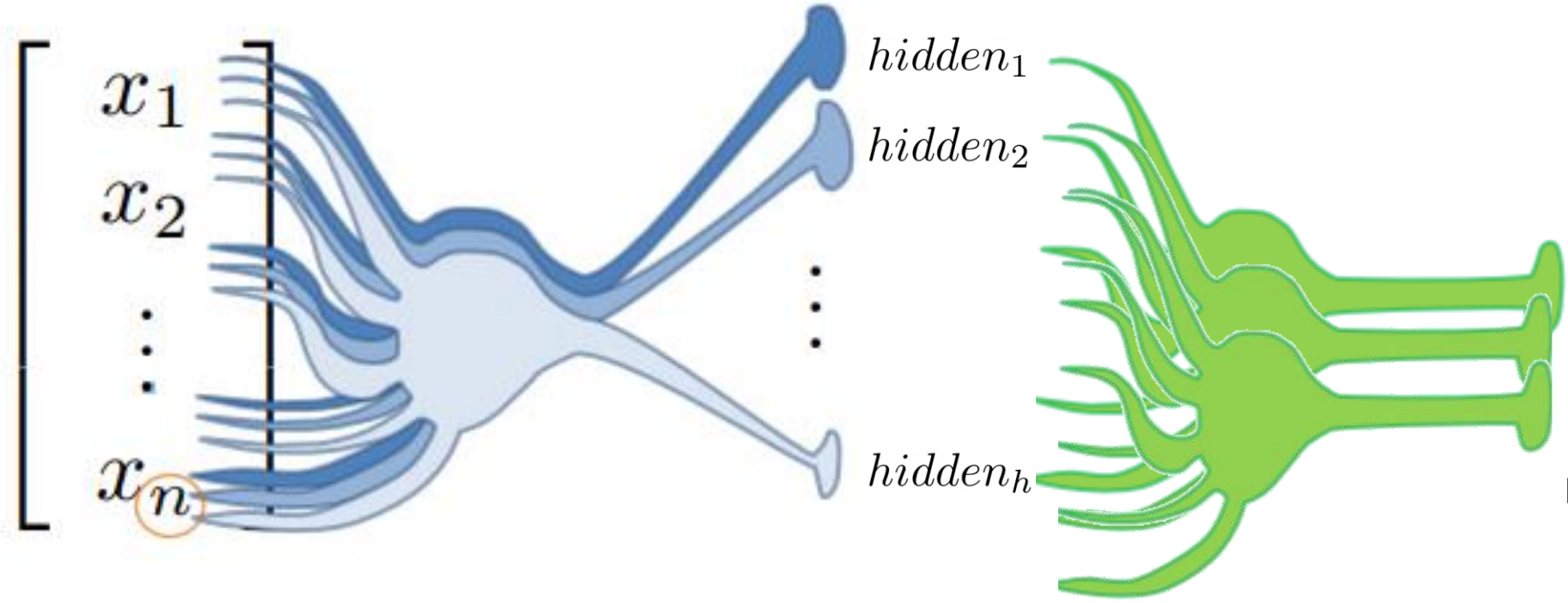
Nombre de paramètres ?

Un multi-layer perceptron peut résoudre des problemes non linéaires

Neural network



2.b. Muti-layer Perceptron - classification



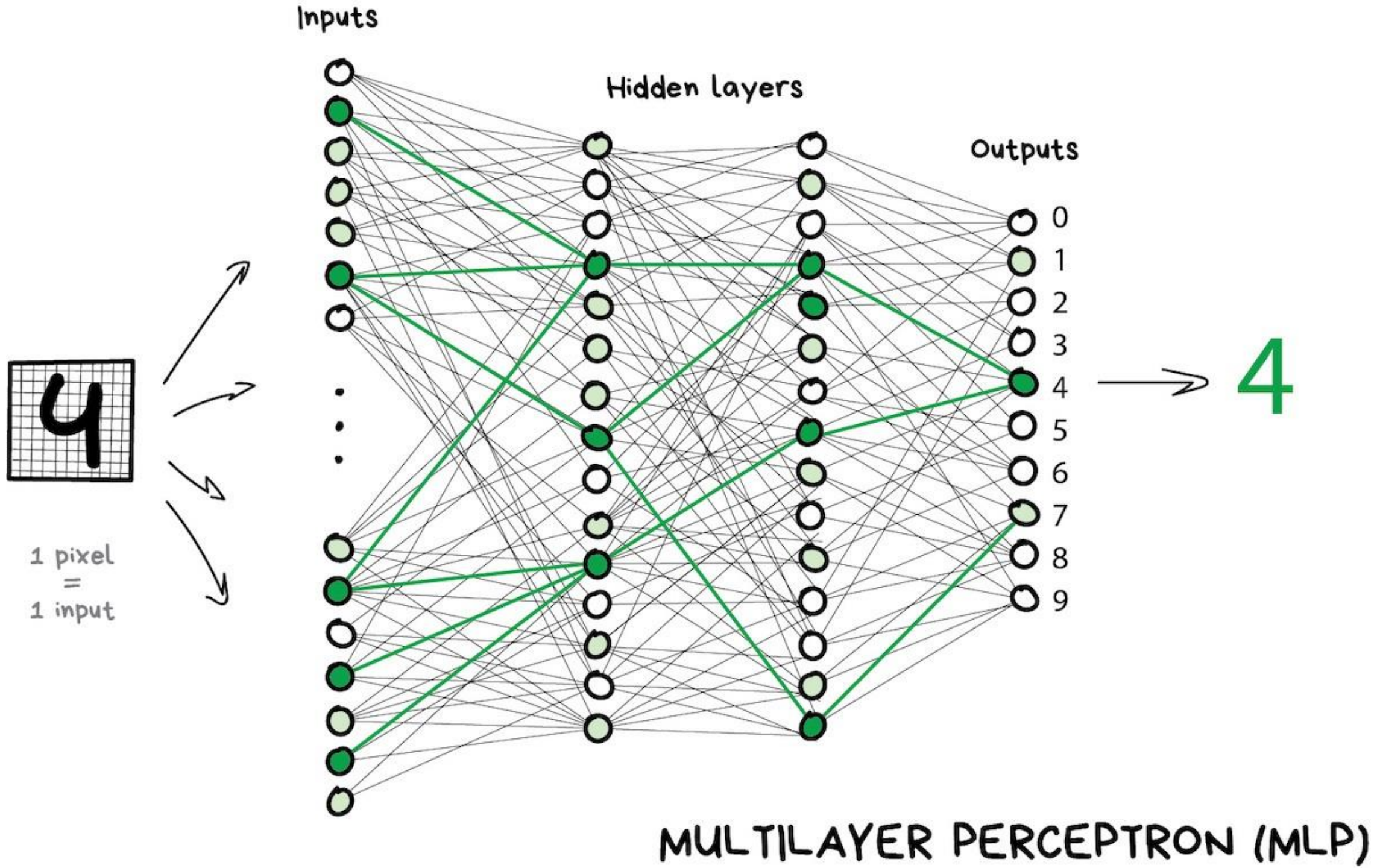
$P_1 \quad = w_o^T hidden + b_o$

$\ldots \quad = w_o^T hidden + b_o$

$P_3 \quad = w_o^T hidden + b_o$

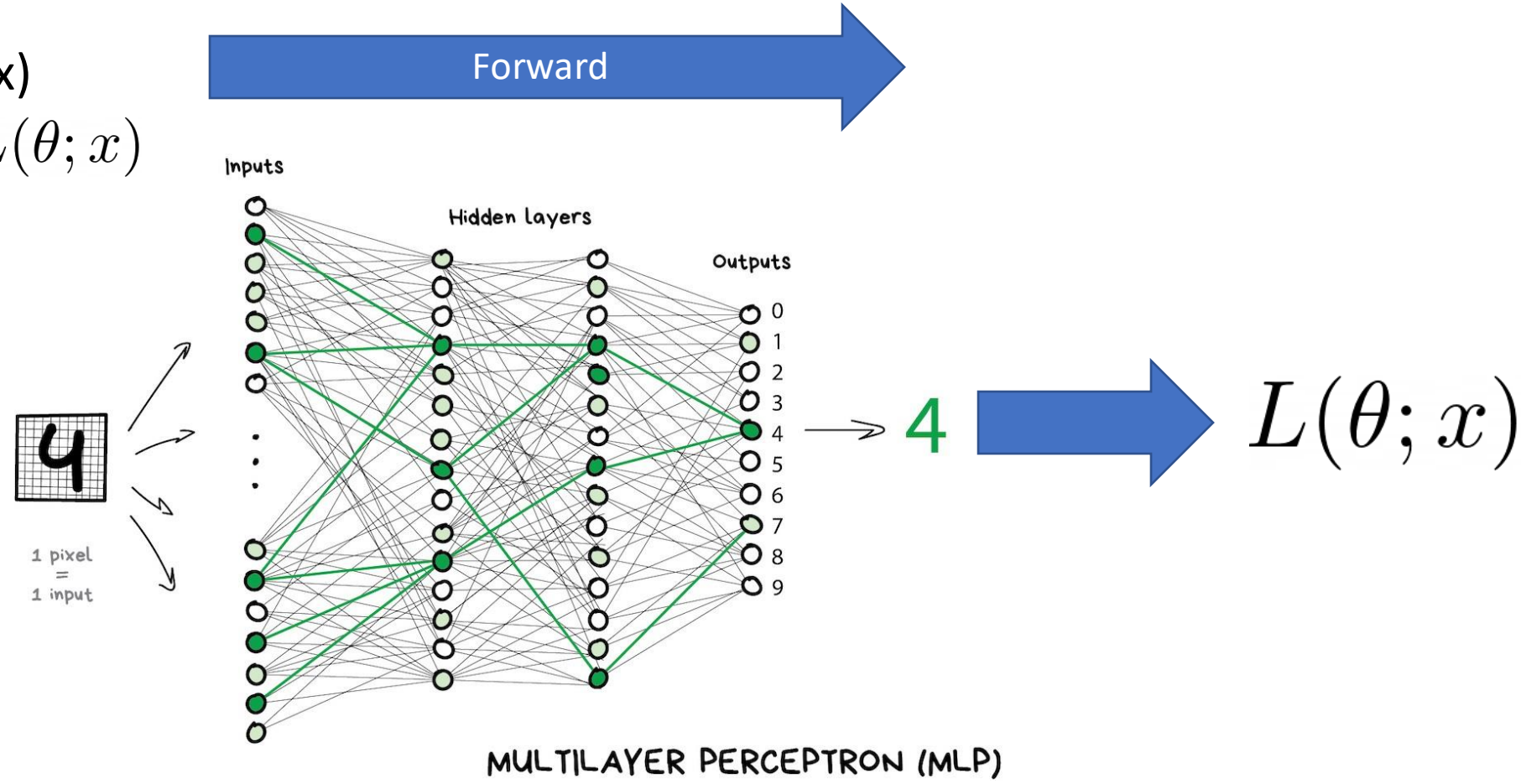
Nombre de paramètres ?

Un multi-layer perceptron peut résoudre des problemes non linéaires


Inputs
Hidden layers
Outputs
0
1
2
3
4
5
6
7
8
9
4
1 pixel
=
1 input
MULTILAYER PERCEPTRON (MLP)

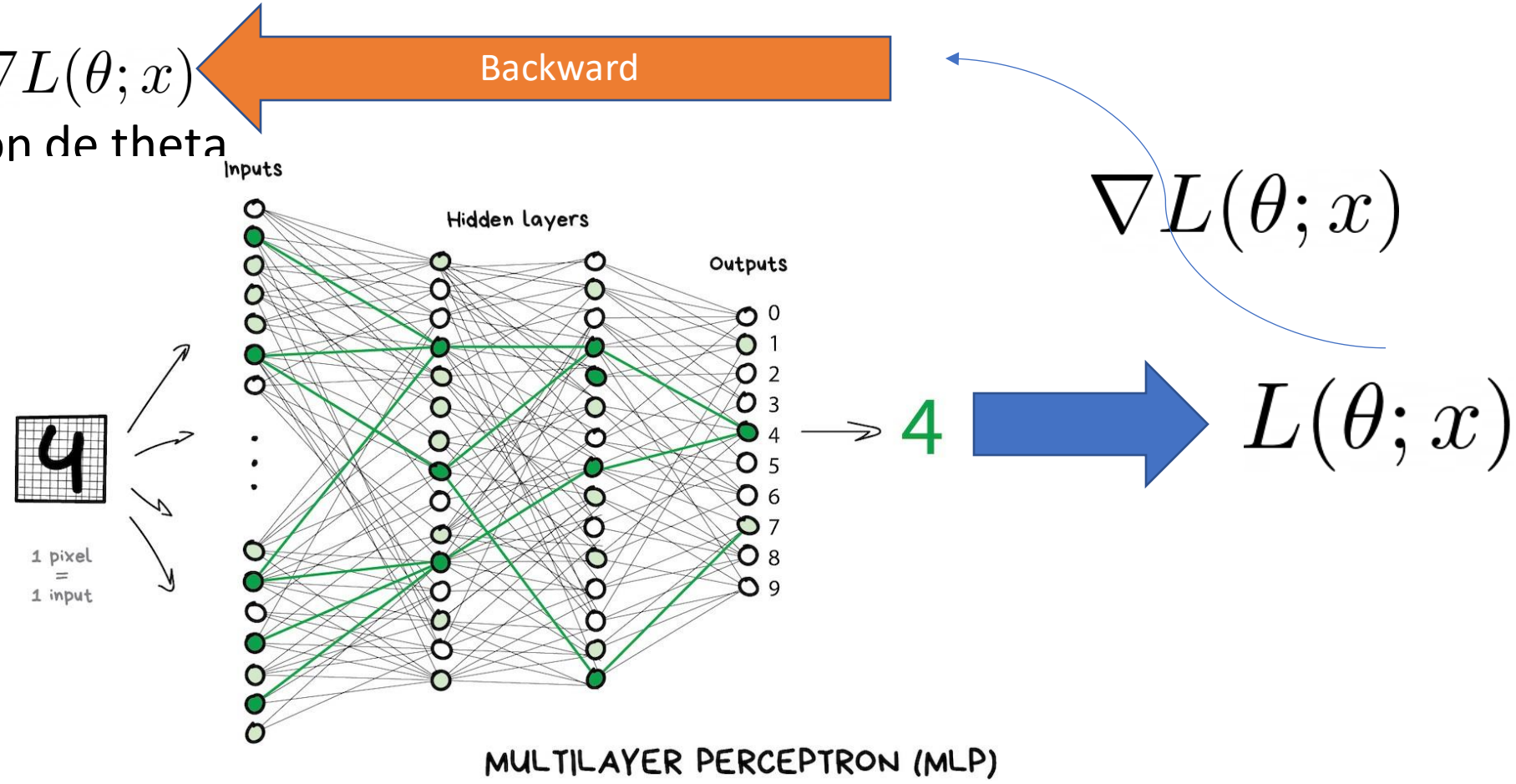
Forward et backward: deux termes propres au deep learning

- Forward:
  - Calcul de f(x)
  - Calcul de $L(\theta; x)$

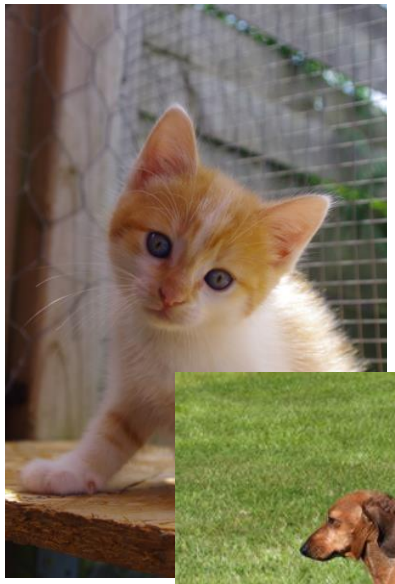


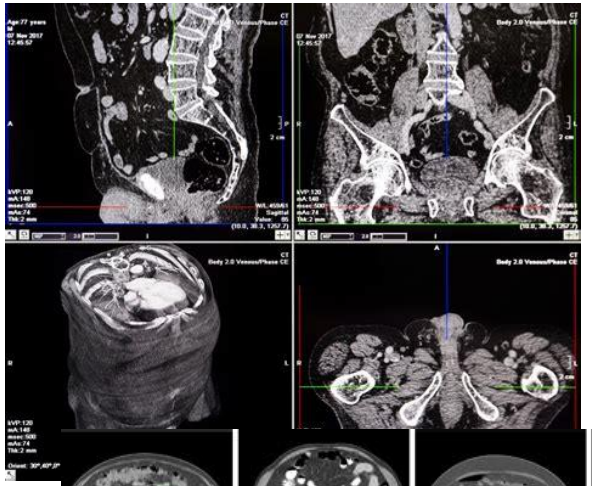
Forward et backward: deux termes propres au deep learning

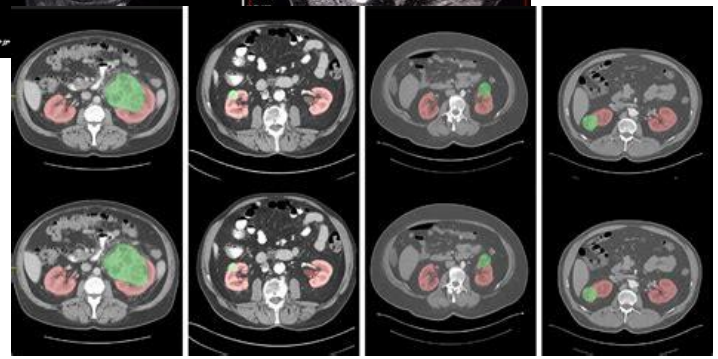
- Forward:
  - Calcul de $\nabla L(\theta; x)$
  - Optimisation de theta

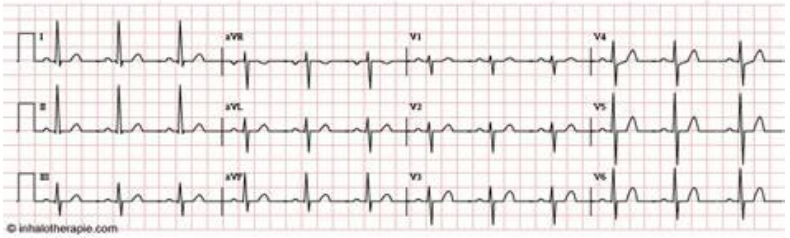


Quel est le principal problème du multi-layer perceptron?

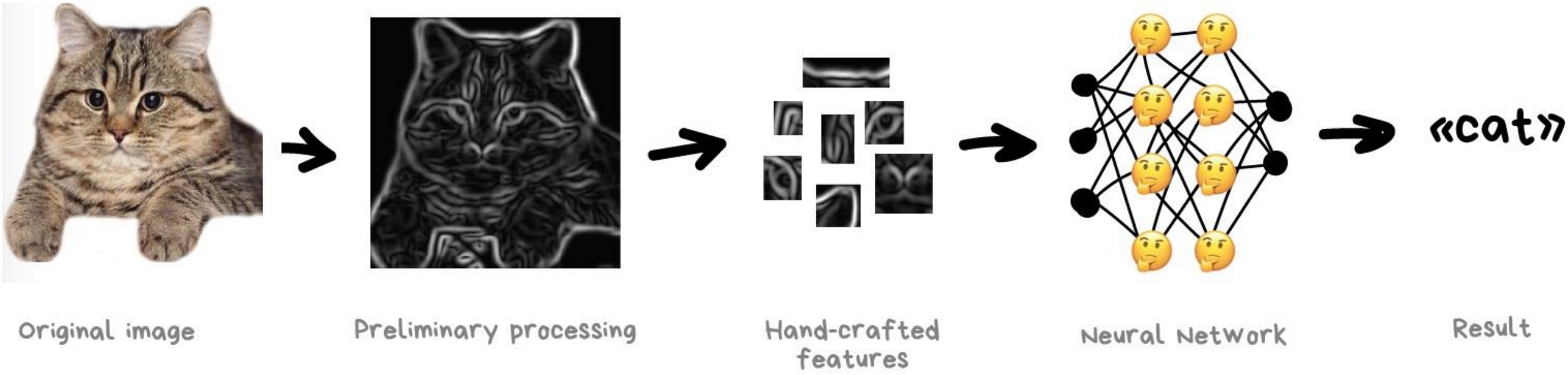








5. Convolutional Neural Network



Workflow de Machine Learning traditionnel: on crée les variables de nos donées à la main

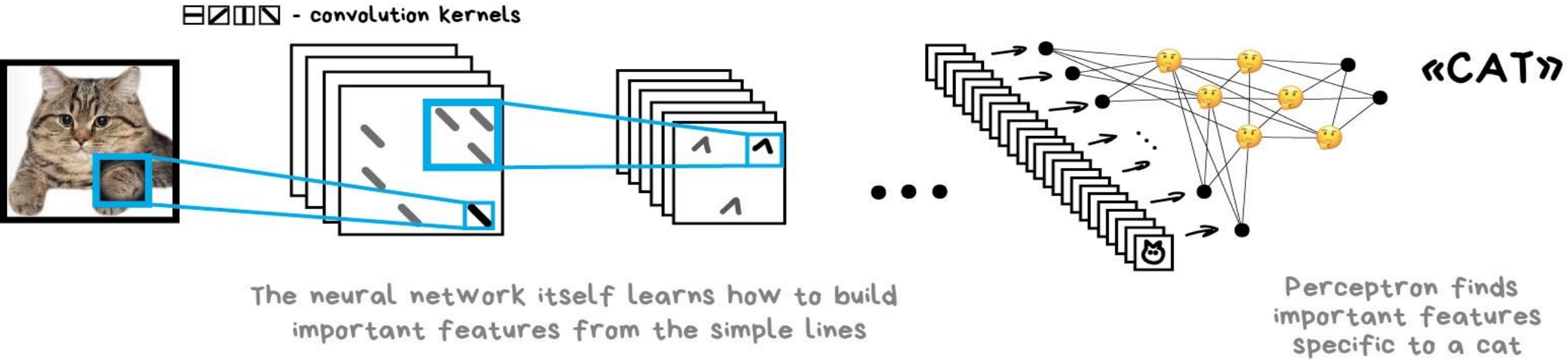
Exemple: on va indiqué ou sont les oreilles et la queue.

Problemes: Que se passe-t-il si les oreilles du chat ne sont pas sur la photo ?

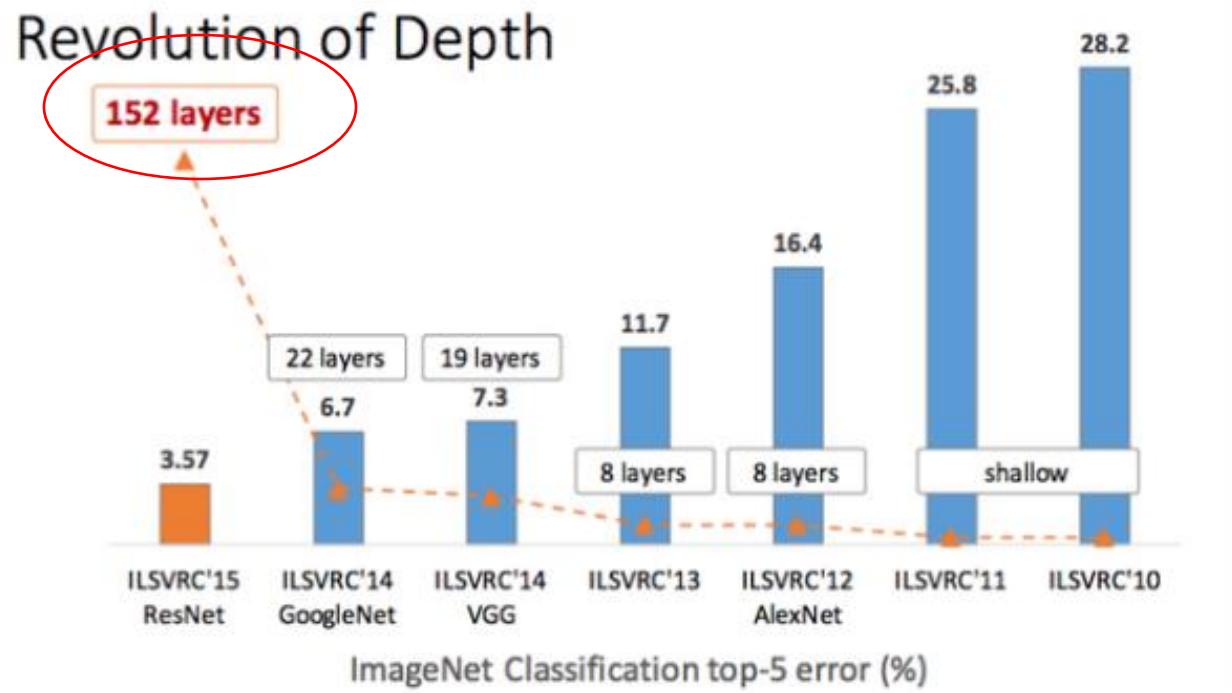
L'homme ne reconnait pas un chat d'un chien par la position ni la forme des oreilles.

5. Convolutional Neural Network

On veut que la “machine” crée elle même ses variables



ResNet [He et al, CVPR 2016]



Year	Model name	Accuracy (%)	Size (MB)	MACs (G)	#Params (M)	Time (ms)
2012	**AlexNet**	56.48	237.89	0.72	61.10	0.46
2014	**VGG19**	72.38	461.10	0.40	20.08	1.12
2014	**Inception_v3**	79.35	5212.60	1.53	6.25	5.80
2016	**ResNet18**	76.82	721.60	0.56	11.22	1.28
2016	**ResNet50**	78.07	4233.60	1.30	23.70	4.56
2016	**ResNet101**	77.35	6409.60	2.52	42.70	7.38
2016	**ResNet152**	78.28	9097.60	3.74	58.34	10.54
2018	**DenseNet121**	78.46	5025.60	0.90	7.05	5.23
2018	**DenseNet169**	75.56	6111.60	1.07	12.64	6.75
2018	**DenseNet201**	76.87	7947.60	1.38	18.28	9.24