Dauphine | PSL
UNIVERSITÉ PARIS

Deep Learning for Image Analysis

Mehyar MLAWEH

🏫 Master BDIA - Dauphine Tunis
PhD - Université PSL

👨‍💻 Generative AI Engineer - Wealthy Technology

Resources: https://mehyarmlaweh.github.io/mehyar-mlaweh/teaching

Course evaluation

- Final Project: **70%**
- Lab Session Reports: **20%**
- Quizzes: **10%**

Module Overview

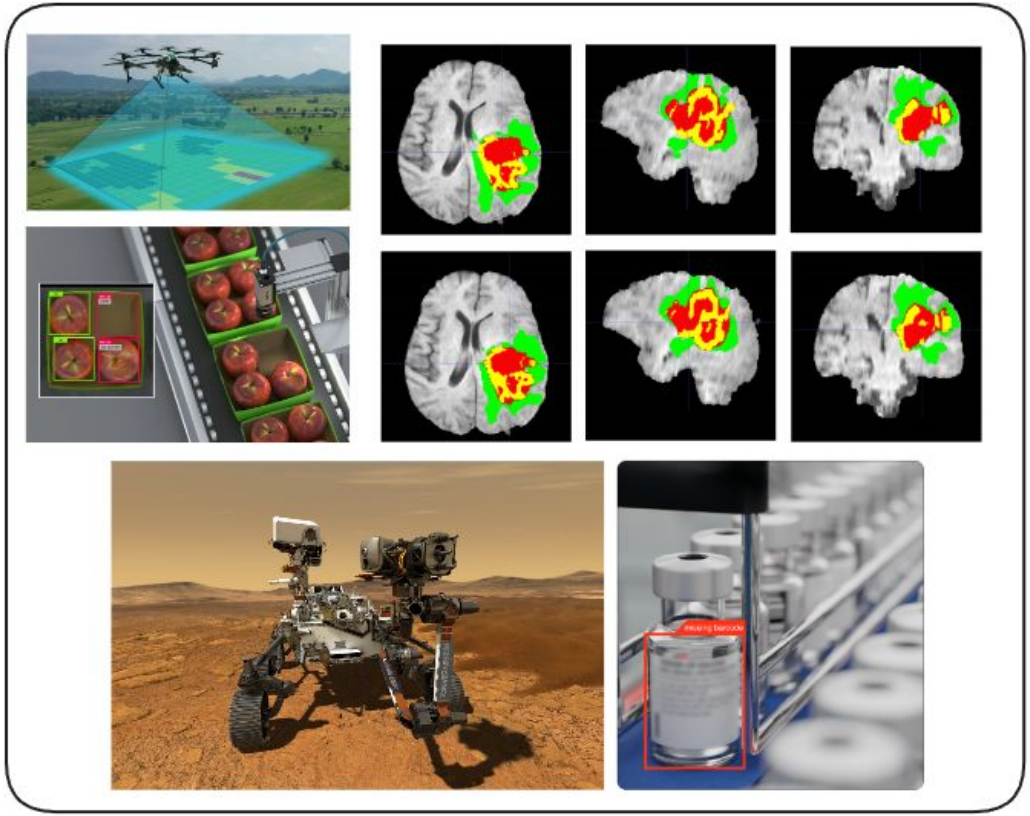
1. Computer Vision Foundations
2. CNNs
3. Transfer Learning
4. Vision Transformers
5. Object Detection
6. Segment Anything Model (SAM)
7. Final Project

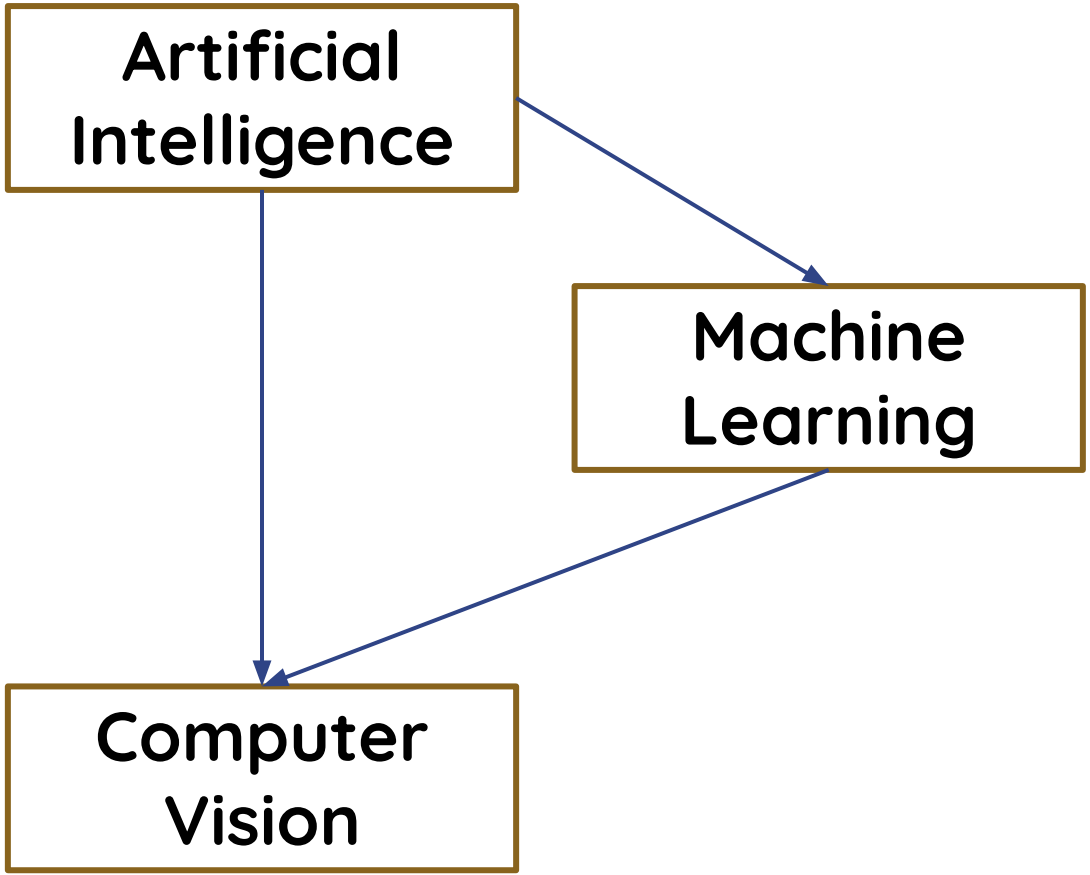
Dauphine | PSL
UNIVERSITÉ PARIS

Lecture 1:
Computer Vision Foundations

At an abstract level, the goal of computer vision problems is to use the observed image data to infer something about the world.

– Page 83, Computer Vision: Models, Learning, and Inference, 2012





Core Computer Vision Tasks

Image Classification

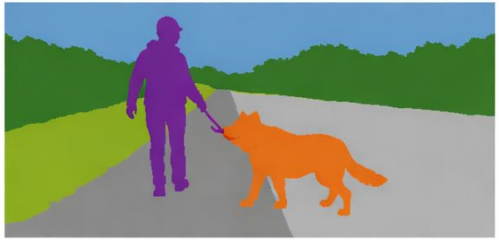
What is in the image?

Object Detection

What objects are present and where are they?

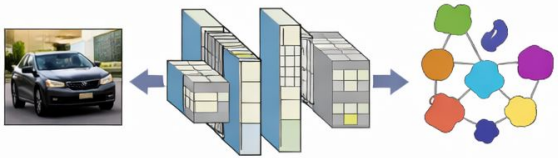
Image Segmentation

Which pixels belong to which object?

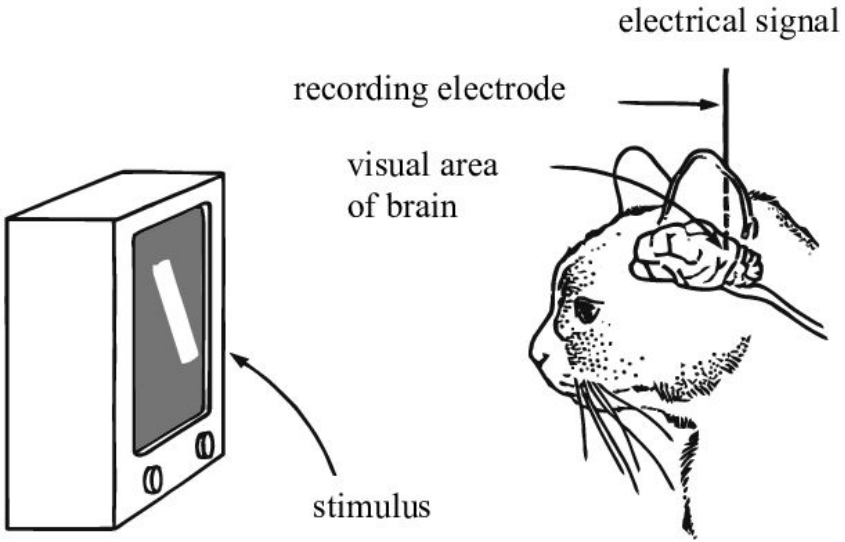


Representation Learning

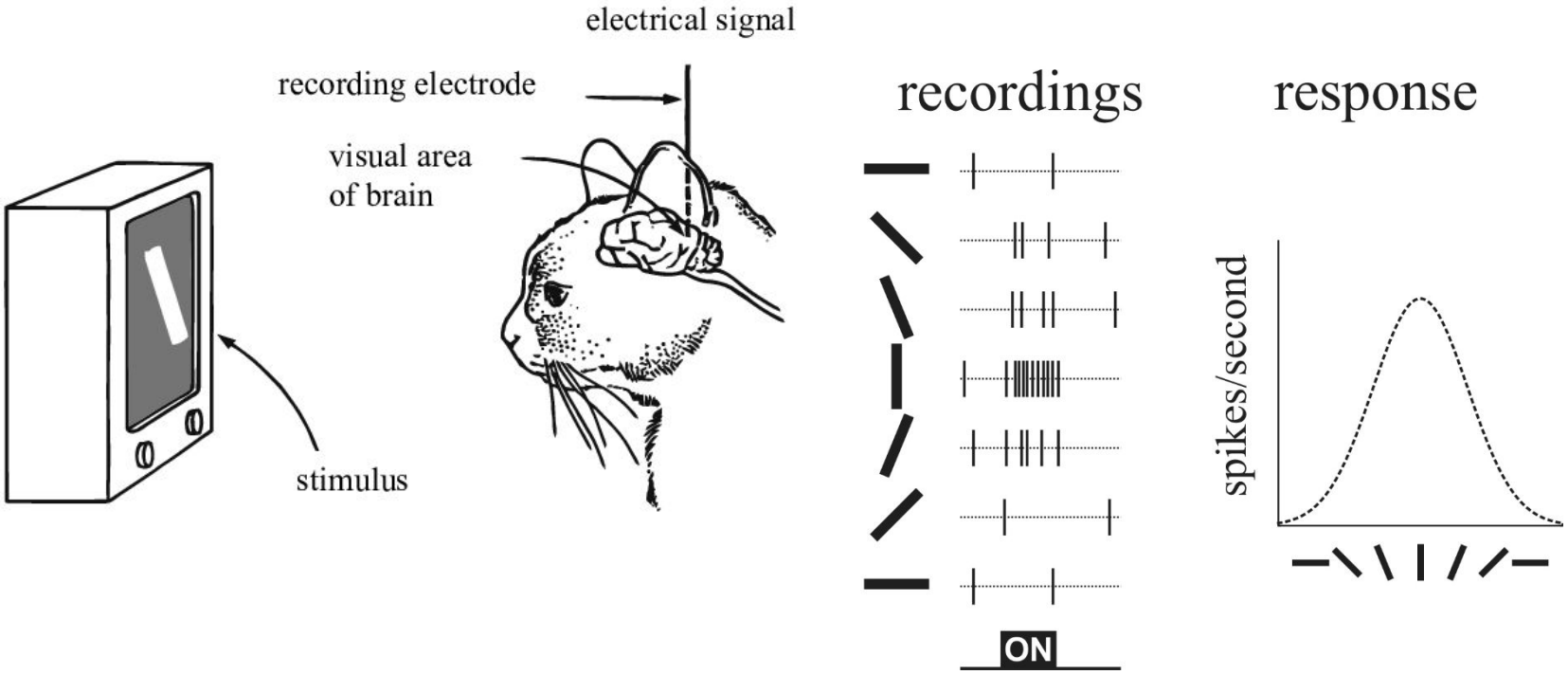
Learning visual features automatically?



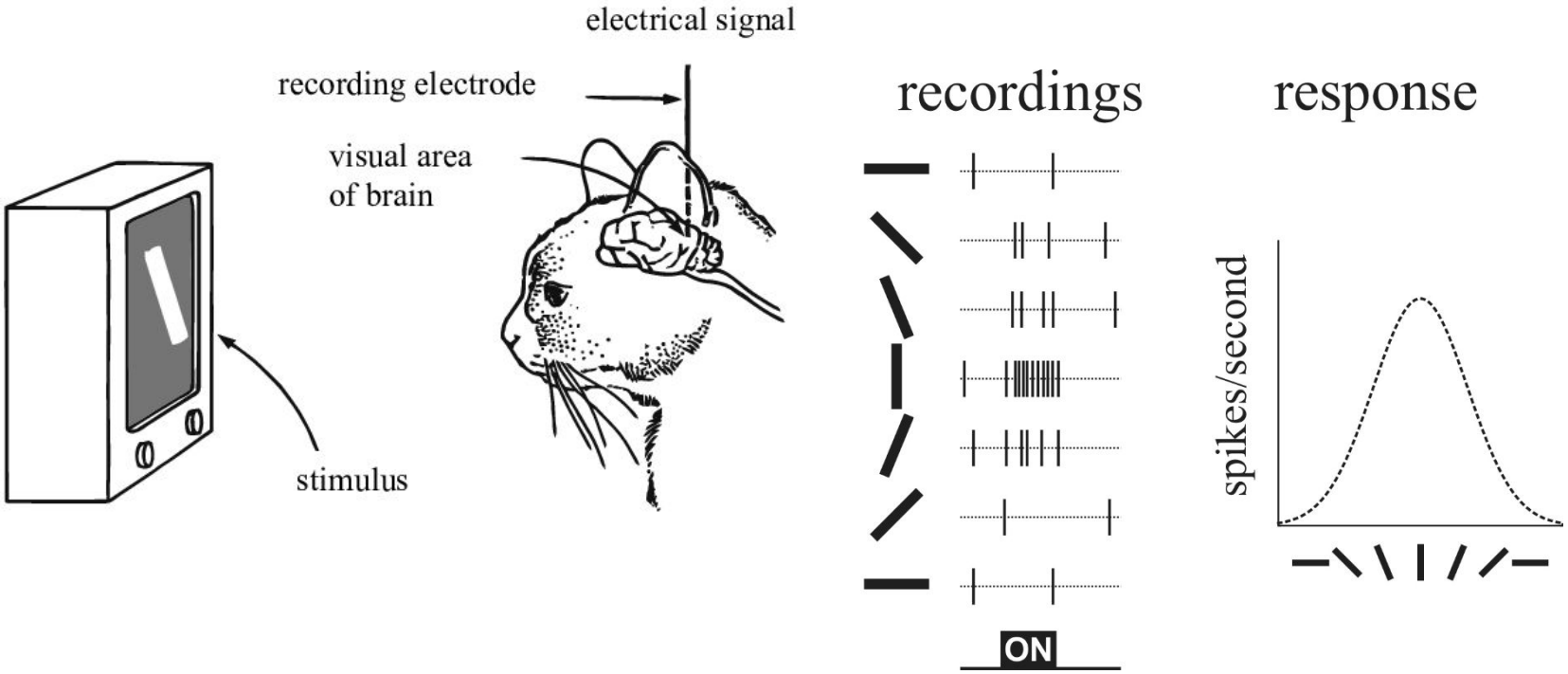
Hubel & Wiesel (1959)



Hubel & Wiesel (1959)



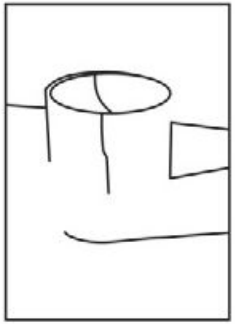
Hubel & Wiesel (1959)



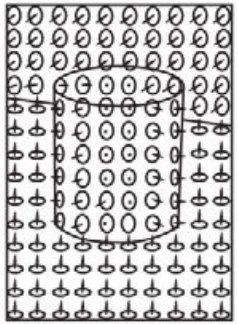
This discovery introduced a key idea: vision is hierarchical

David Marr (1970s)

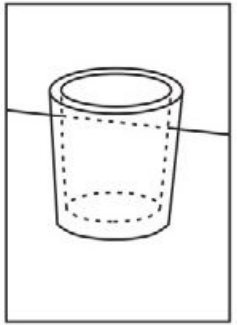
Retinal Image



Primal Sketch



2.5D Sketch



3D Sketch

This framework strongly influenced computer vision research for decades

Edges, Parts, and Features (1980s–2000s)

Researchers attempted to recognize objects using:

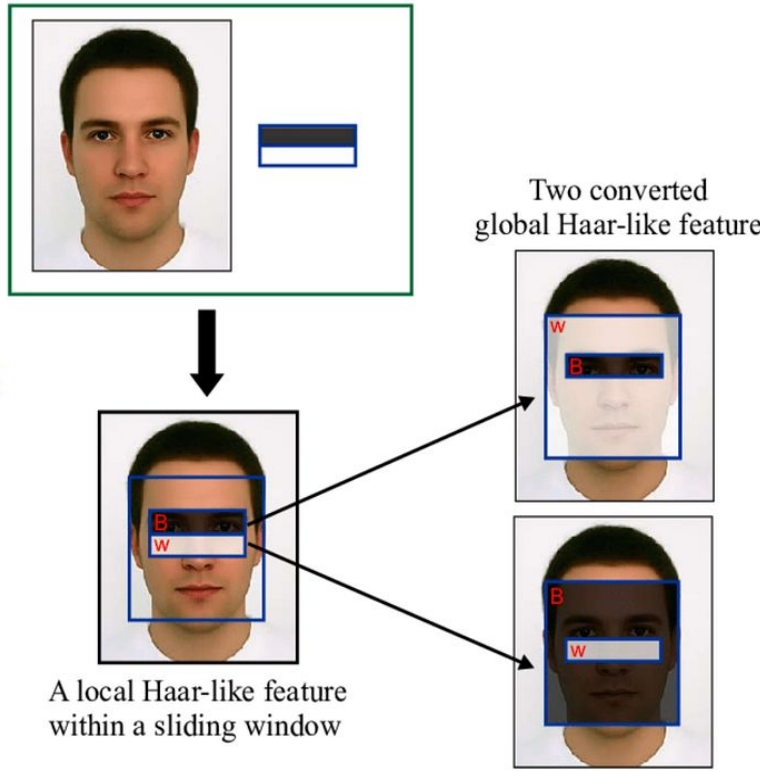
- Geometric primitives (generalized cylinders)
- Edges and contours

Notable milestones:

- Generalized Cylinders (Brooks & Binford, 1979)
- Canny Edge Detector (1986), one of the most influential algorithms in vision

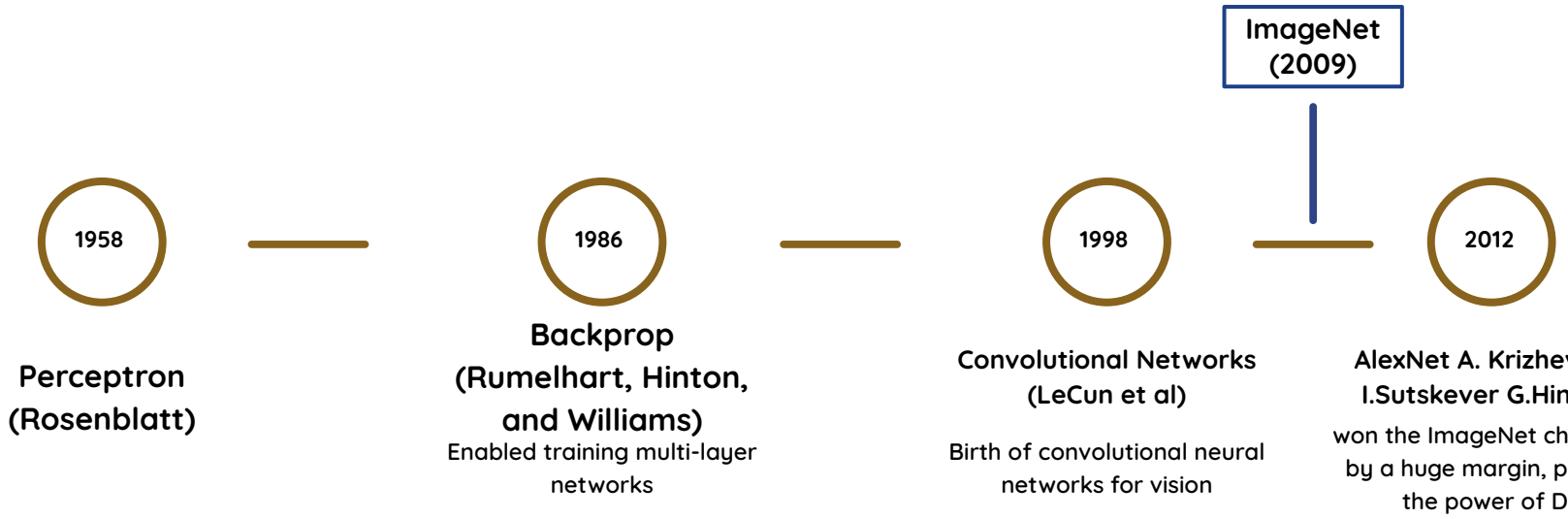
These methods were elegant and mathematically grounded, but sensitive to noise and fragile in real-world images.

Viola–Jones (2001)

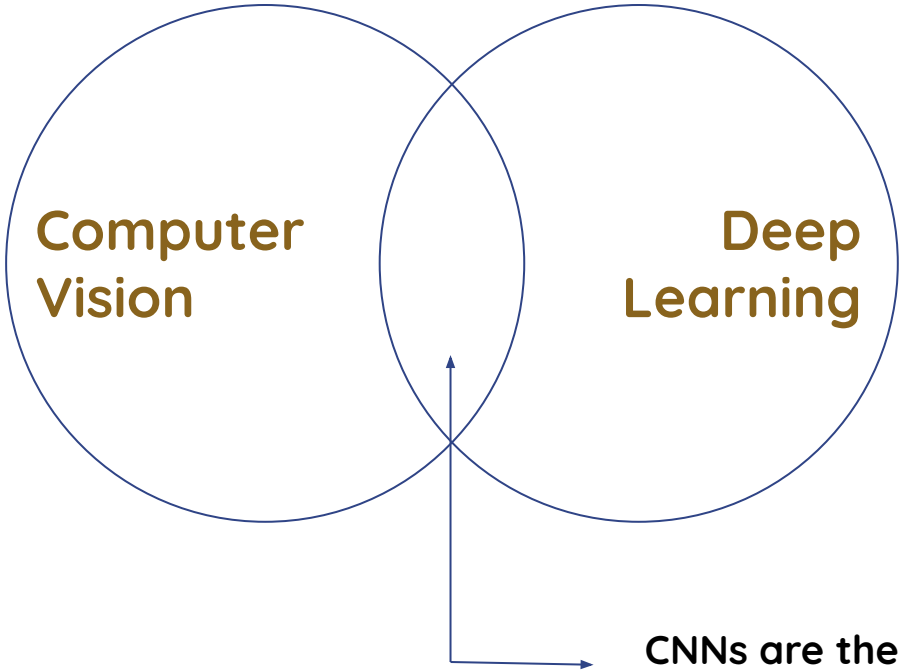


It relies on fixed Haar-like features rather than learning from data.

The Birth of Neural Networks (1950s–1980s)



Crossroads



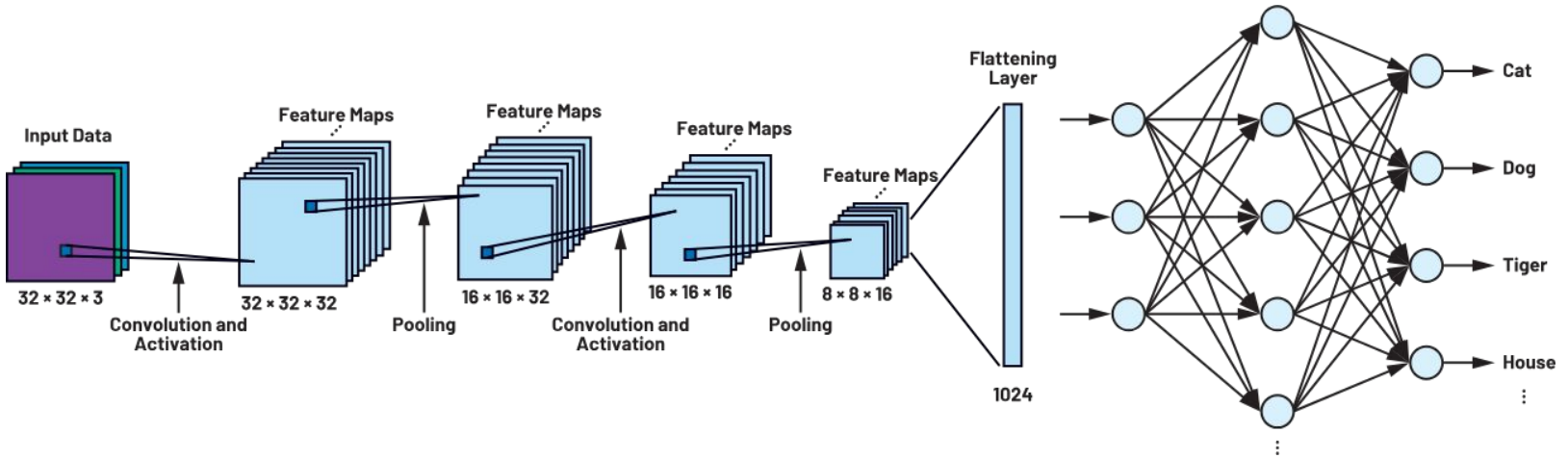
CNNs are the point where rule-based vision ideas became learnable.

Why didn’t LeNet already solve everything?

After 2010 :

- Large-scale data (ImageNet, 2009)
- GPUs & computation
- Deeper, trainable architectures

CNNs



[CIFAR CNN architecture]

📌 CNNs are rarely used alone today, they act as backbones, reused across tasks.

CNNs

What Are Convolutional Neural Networks (CNNs)?

CNNs

- **Convolutional Layers**
  Learn local patterns (edges, textures, shapes) using learnable filters.

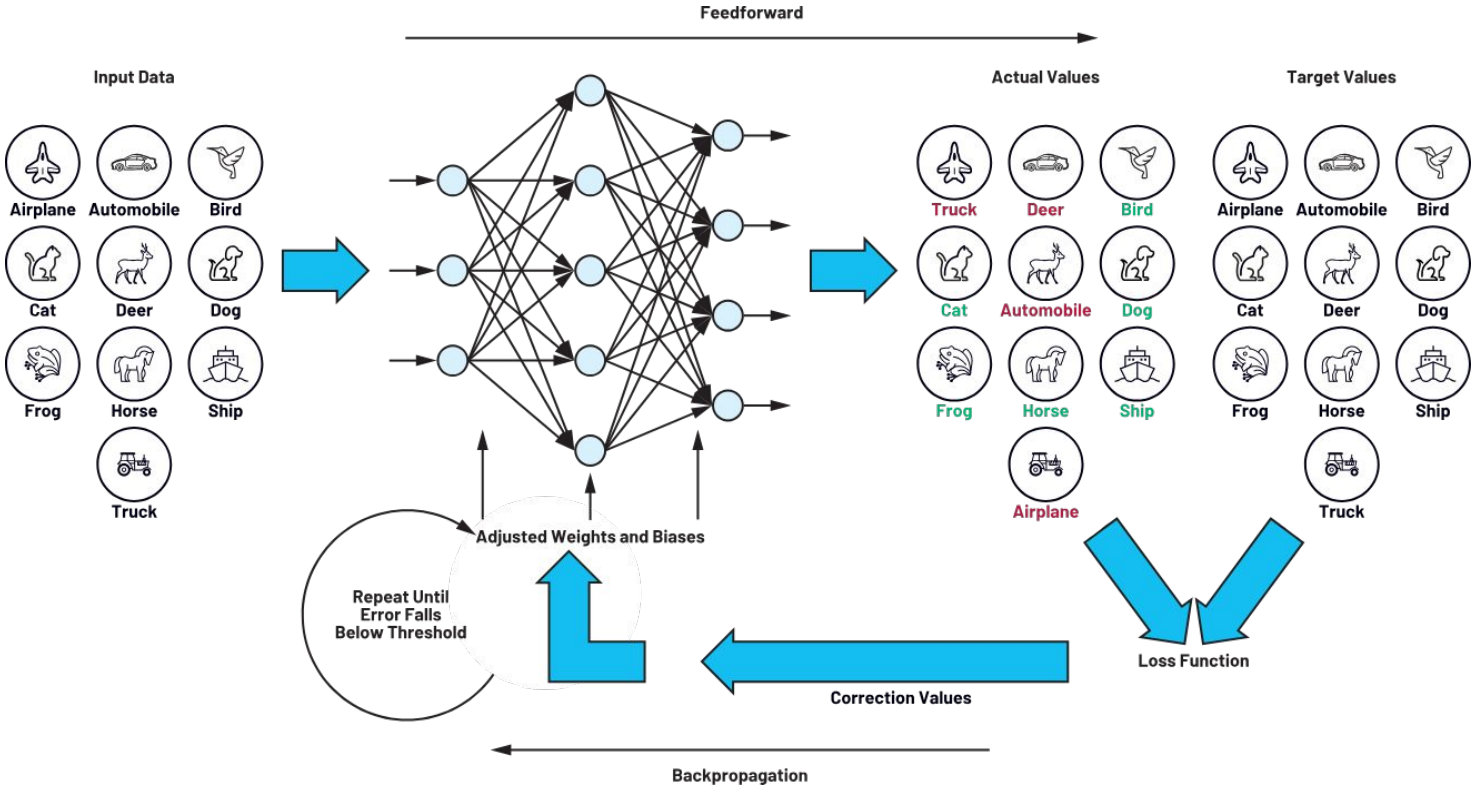
- **Activation Functions (e.g., ReLU)**
  Introduce non-linearity to model complex visual patterns.

- **Pooling Layers**
  Reduce spatial dimensions and improve robustness to small variations.

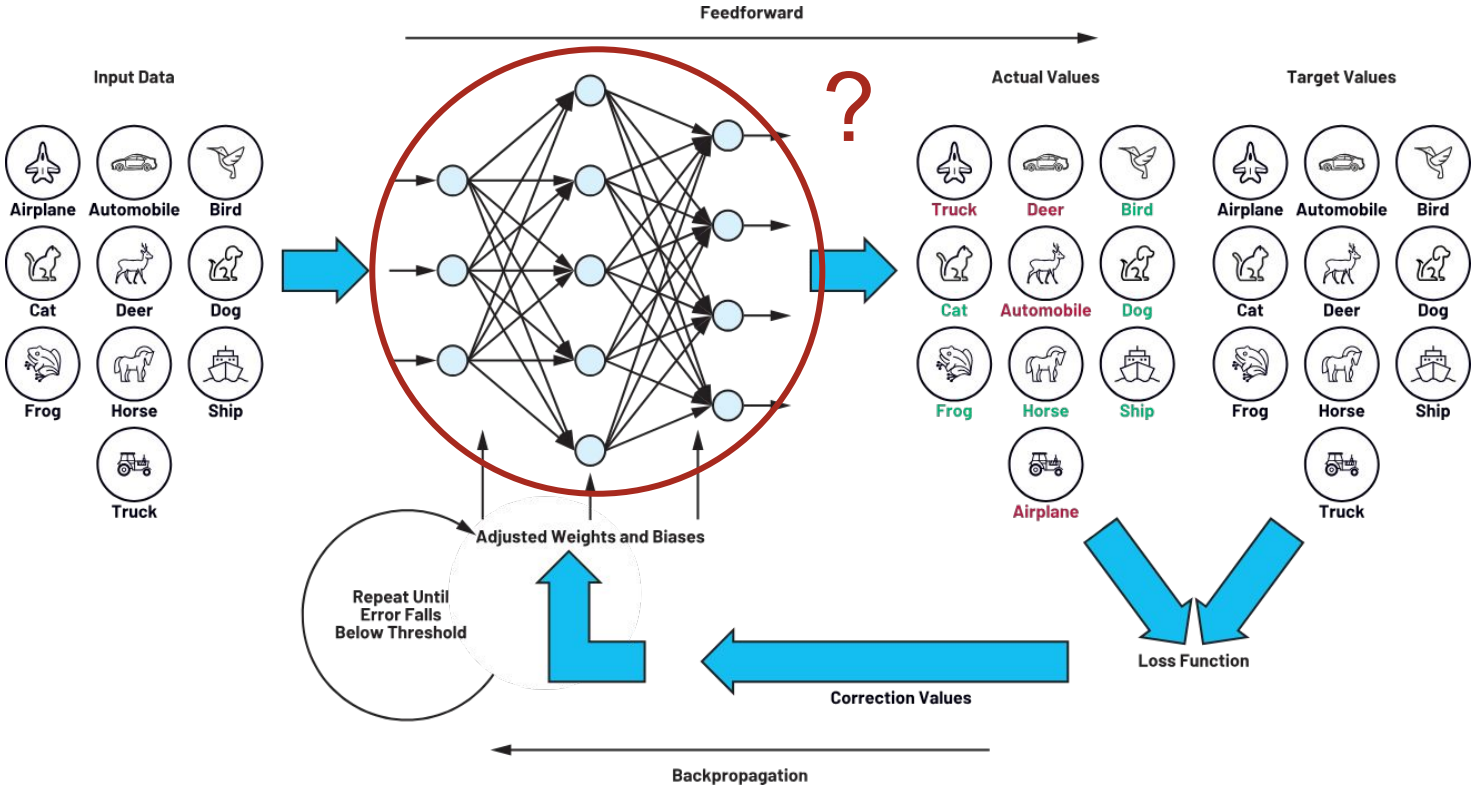
- **Fully Connected Layers**
  Combine learned features to make final predictions.

How Are CNNs Trained ?

Supervised learning approach

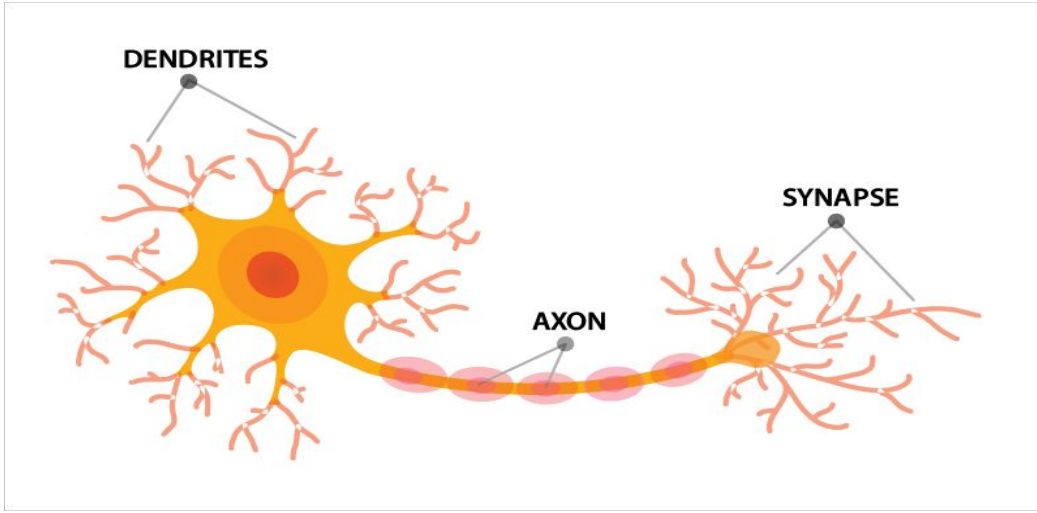


Supervised learning approach

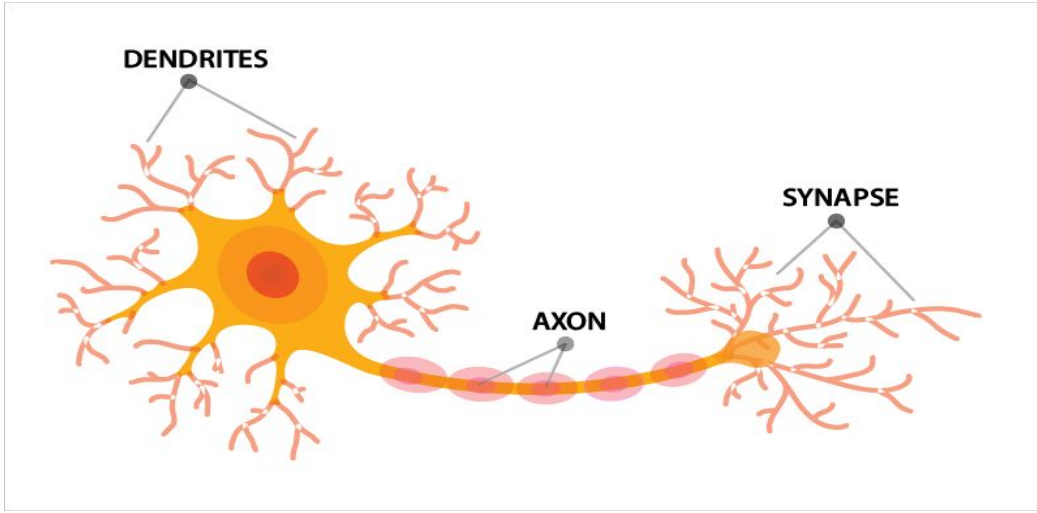


Backup

Biological neuron

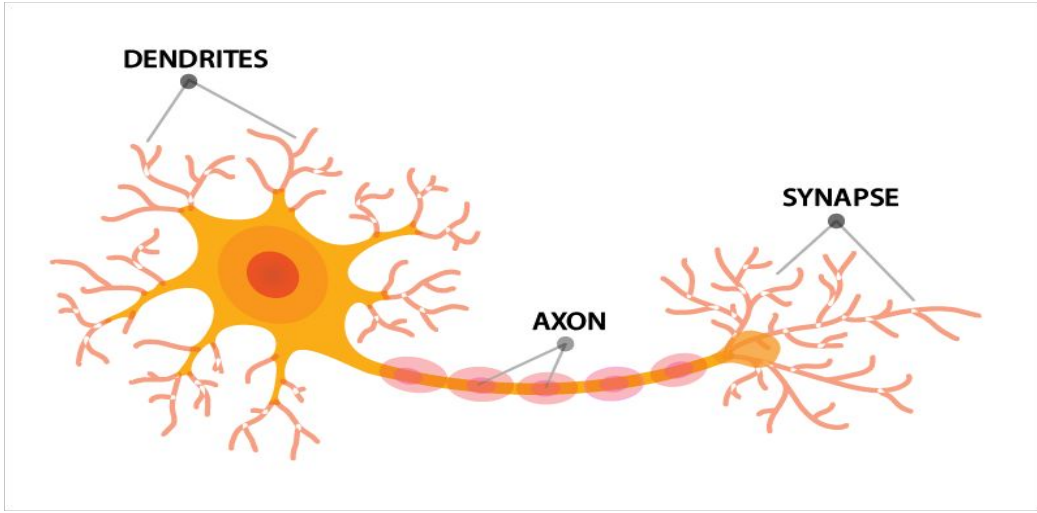


Biological neuron



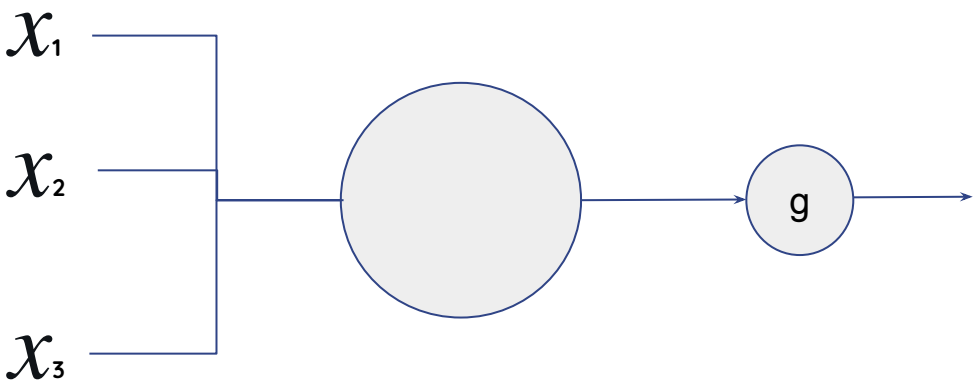
- A human neuron can have several thousand dendrites.

Biological neuron



- A human neuron can have several thousand dendrites.
- The neuron sends a signal through its axon if during a given interval of time the net input signal is larger than a threshold.

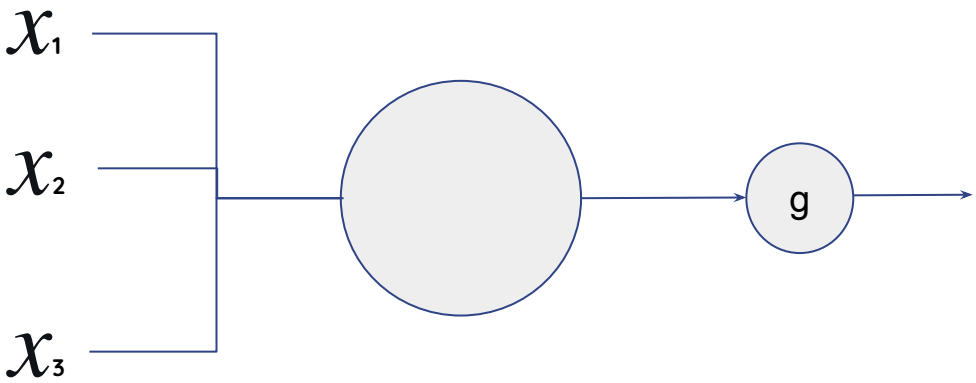
Artificial neuron



General principle

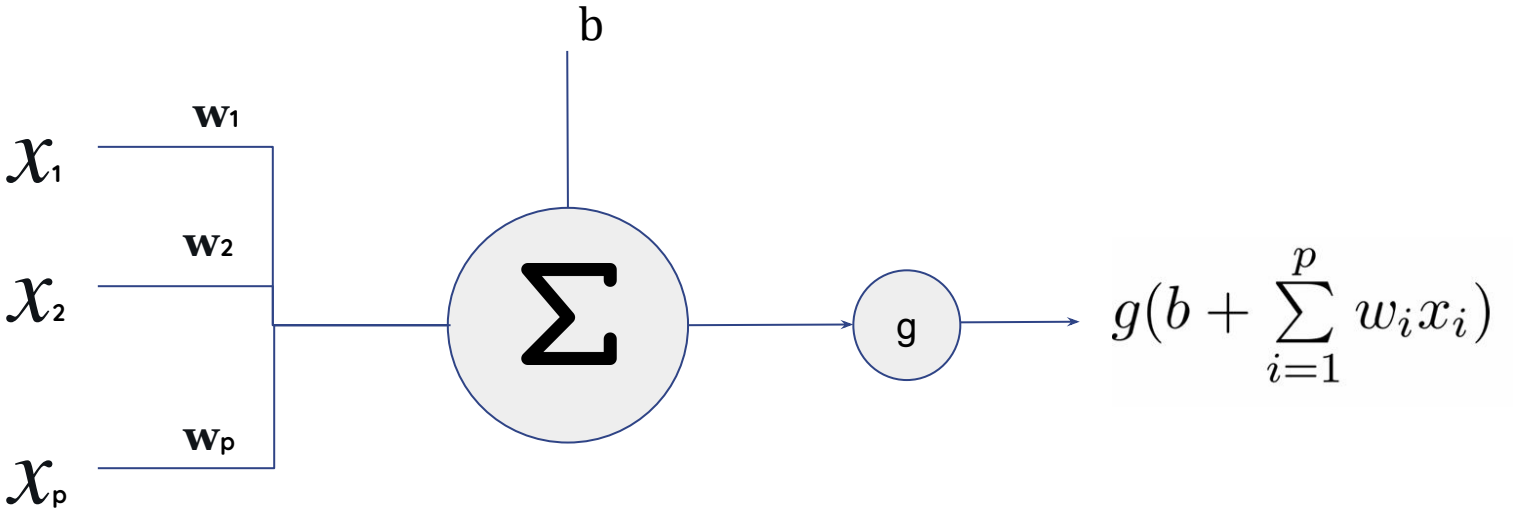
An artificial neuron takes p inputs {x_1, x_2, …, xp}, combines them into a single value, and applies an activation function g(·) to produce the output.

Artificial neuron

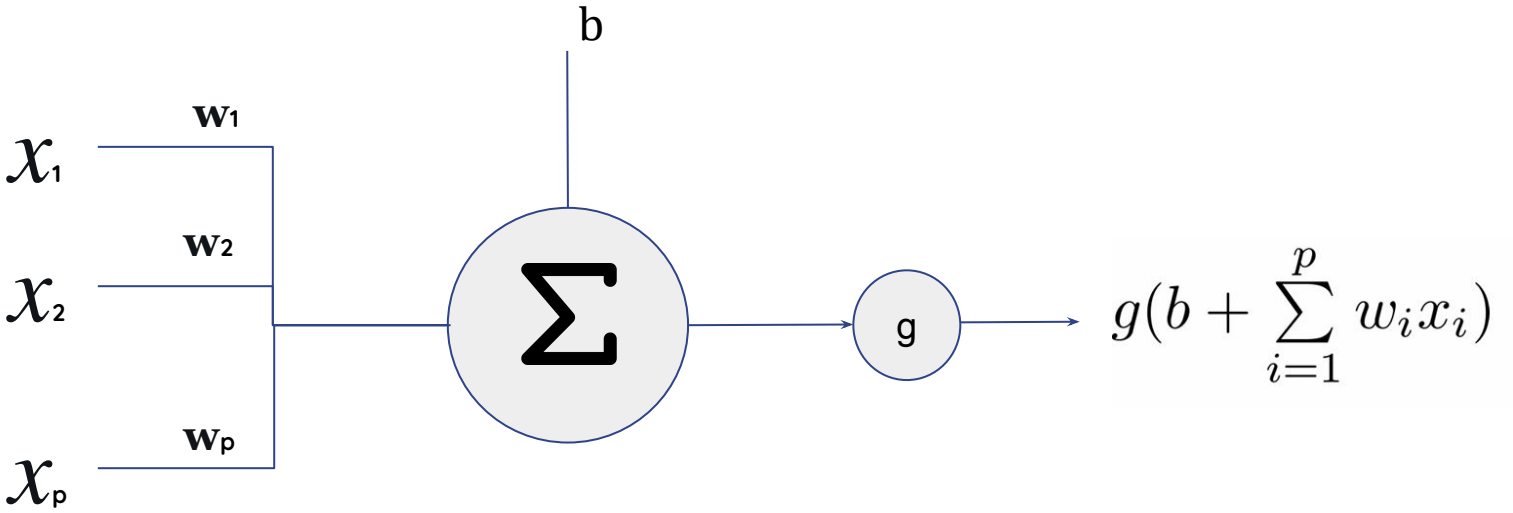


- The first artificial neuron model was proposed by [McCulloch and Pitts, 1943]
- Input and output signals were binary

Modern Artificial neuron



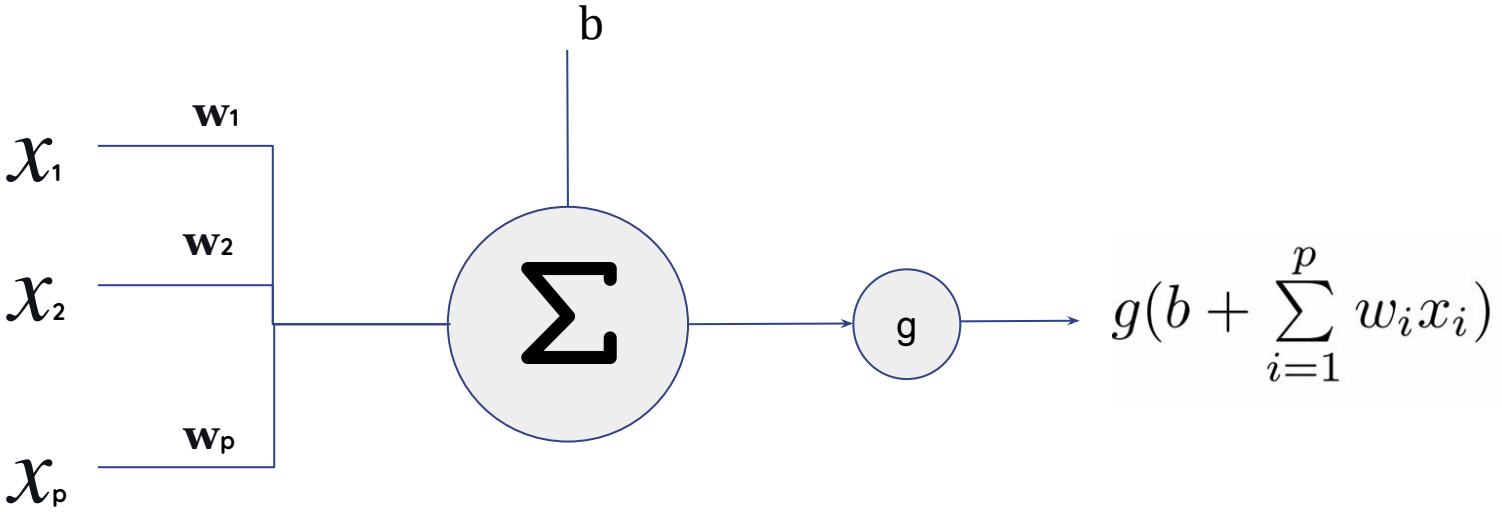
Modern Artificial neuron



The neuron computes a linear combination of the inputs xi

- The weights wi are multiplied with the inputs
- The bias b can be interpreted as a threshold on the sum

Modern Artificial neuron

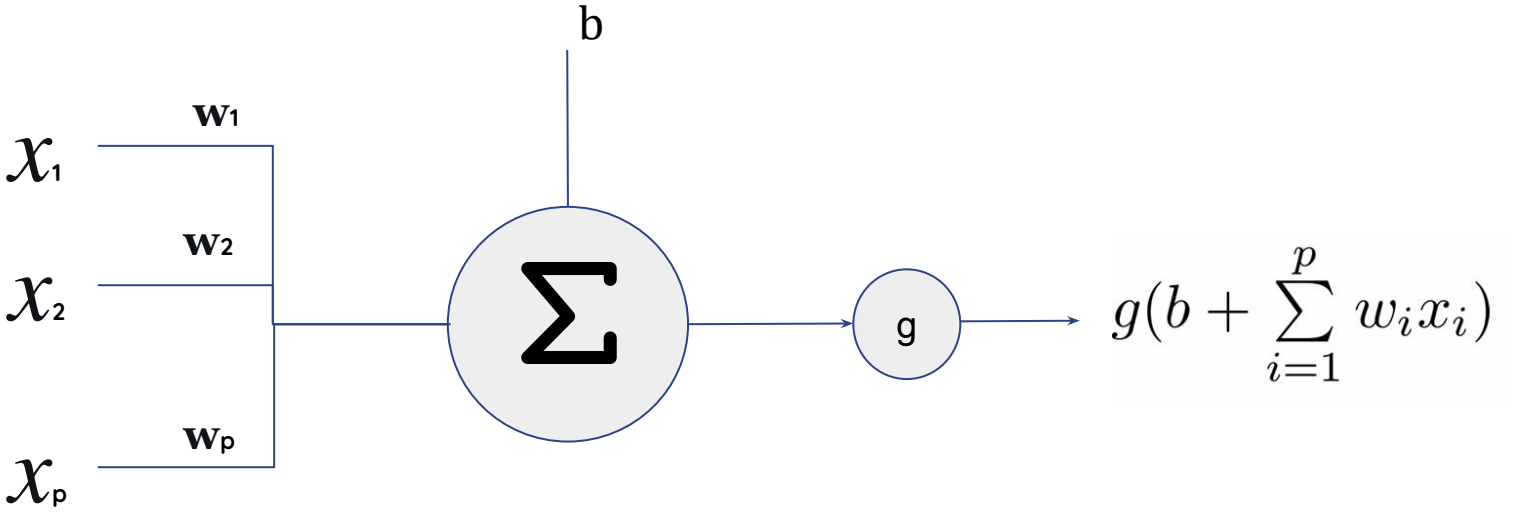


The neuron computes a linear combination of the inputs xi

- The weights wi are multiplied with the inputs
- The bias b can be interpreted as a threshold on the sum

The activation function g somehow decides, depending on its input, if a signal (the neuron's activation) is produced.

The role of the activation function

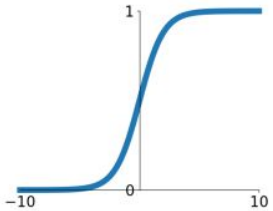


- It's like a gate
- If its input in "high enough", then the neuron is activated, i.e. a signal (other than zero) is produced
- It can be interpreted as a source of abstraction: information considered as unimportant is ignored (or reduced)

Some activation functions

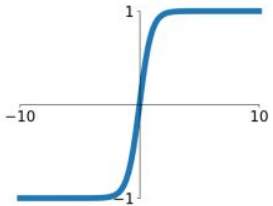
Sigmoid

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



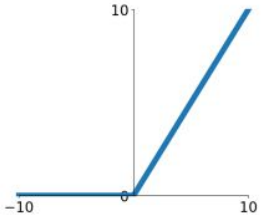
tanh

$$\tanh(x)$$



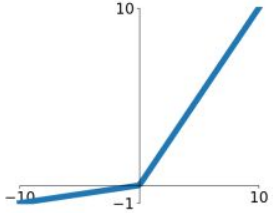
ReLU

$$\max(0, x)$$



Leaky ReLU

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

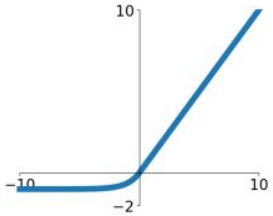


Maxout

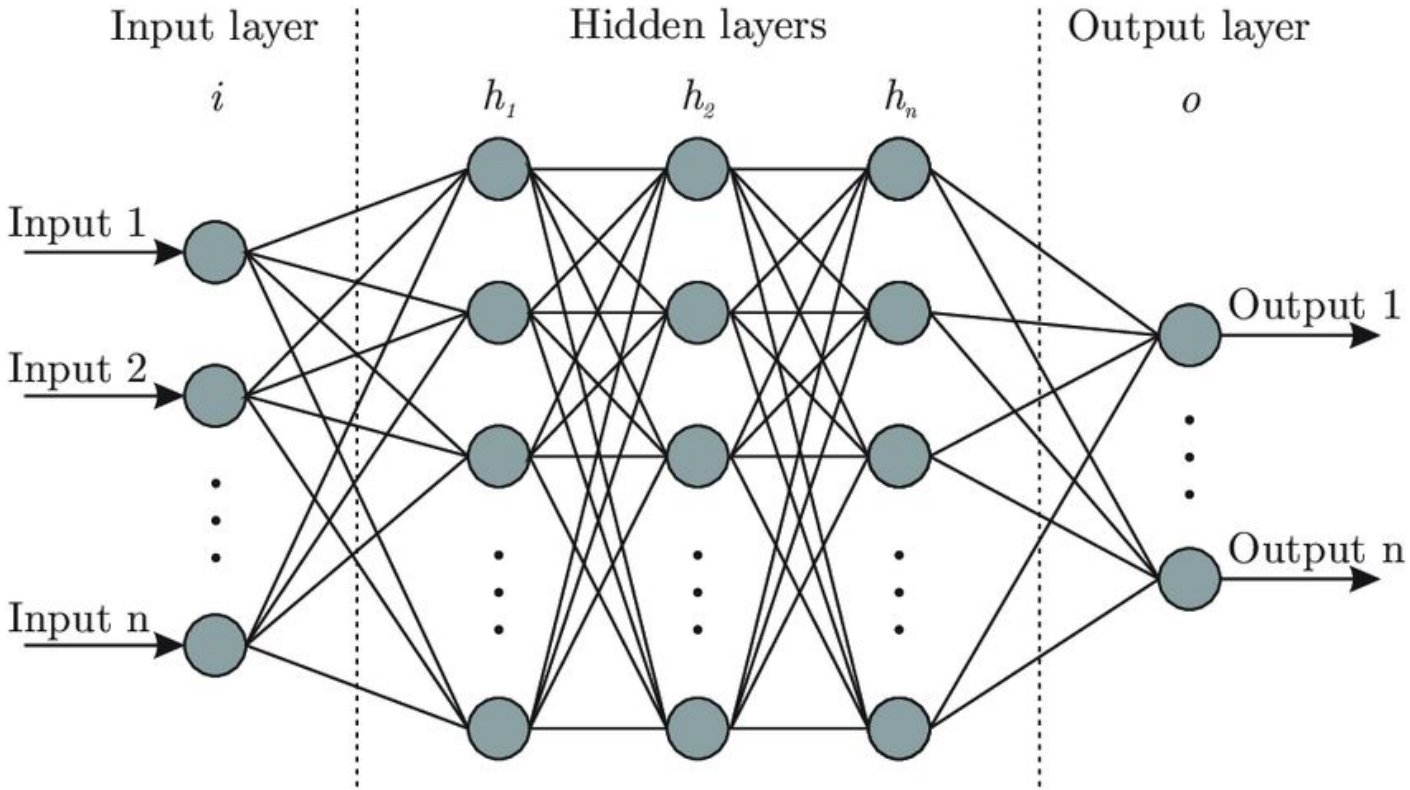
$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

ELU

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



Artificial neural networks

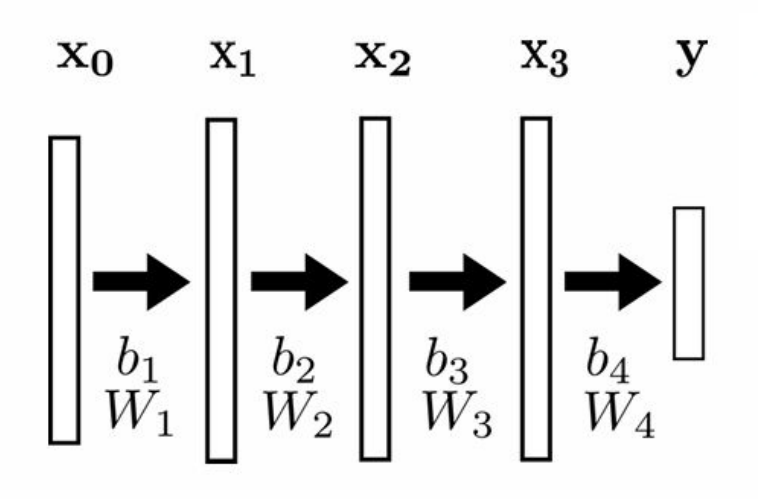


Artificial neural networks

Fully-connected layer :

- A layer is said to be fully-connected (FC) if each of its neurons is connected to all the neurons of the previous layer

The equation of a FC neural network :



$$\mathbf{x}_i = g_i\left(\mathbf{x}_{i-1}^\top \mathbf{W}_i + \mathbf{b}_i\right), \quad i \in \{1, 2, 3\}$$

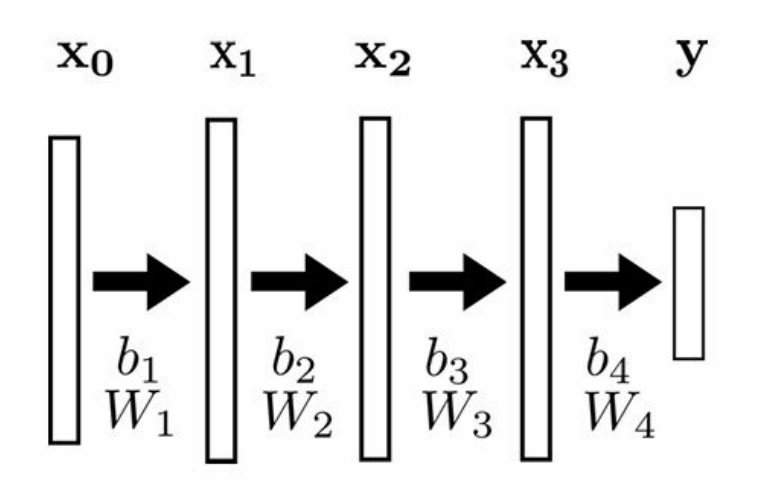
$$\mathbf{y} = g_4\left(\mathbf{x}_3^\top \mathbf{W}_4 + \mathbf{b}_4\right)$$

Artificial neural networks

Fully-connected layer :

- A layer is said to be fully-connected (FC) if each of its neurons is connected to all the neurons of the previous layer

The equation of a FC neural network :



$$\mathbf{x}_i = g_i\left(\mathbf{x}_{i-1}^\top \mathbf{W}_i + \mathbf{b}_i\right), \quad i \in \{1, 2, 3\}$$

$$\mathbf{y} = g_4\left(\mathbf{x}_3^\top \mathbf{W}_4 + \mathbf{b}_4\right)$$

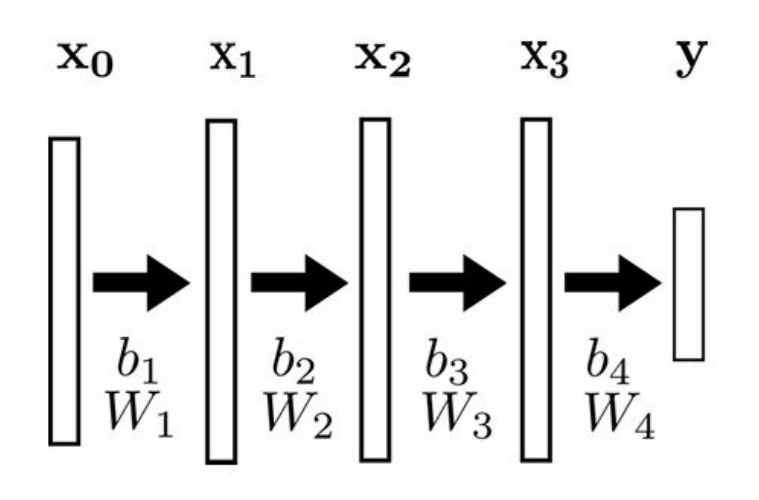
What happens if $g_i(z) = z$

Artificial neural networks

Fully-connected layer :

- A layer is said to be fully-connected (FC) if each of its neurons is connected to all the neurons of the previous layer

The equation of a FC neural network :

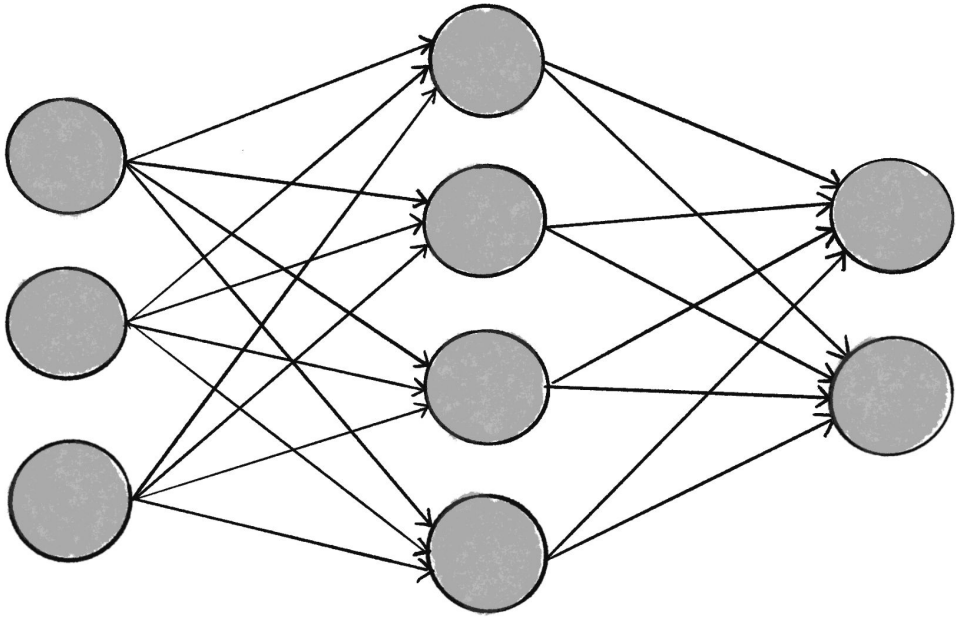


What happens if $g_i(z) = z$

Without nonlinear activations, a deep FC neural network is mathematically equivalent to a single linear (affine) layer

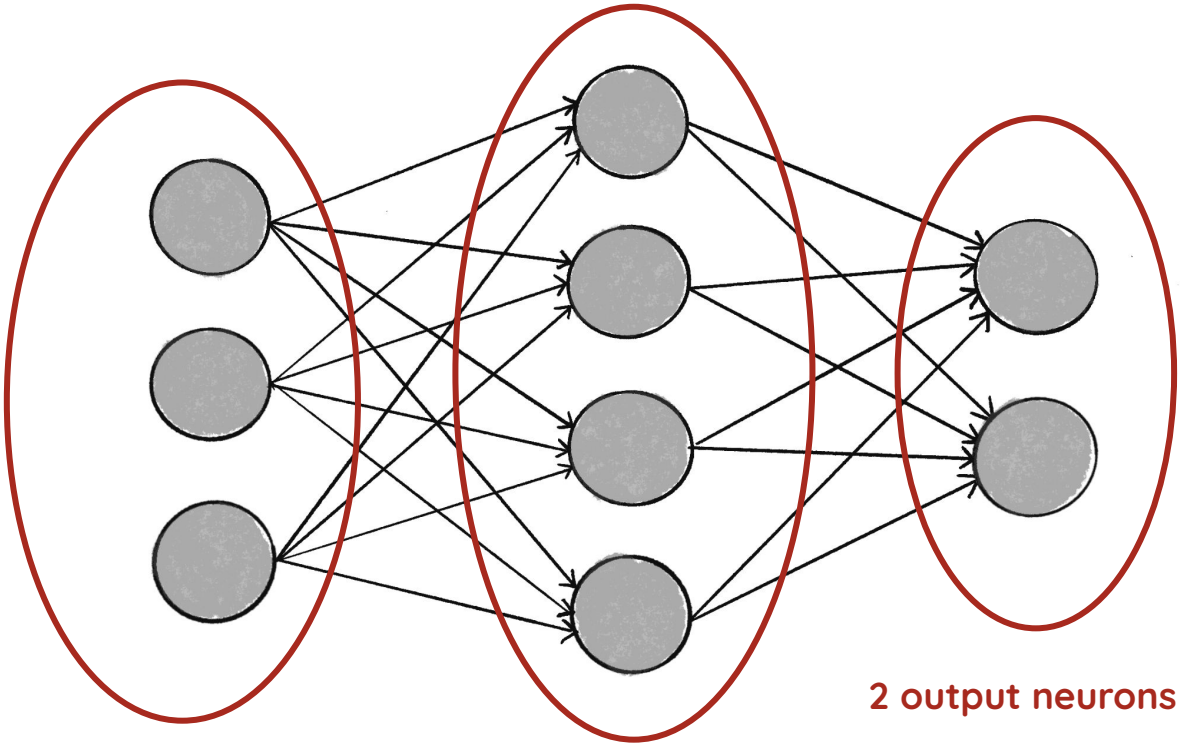
Artificial neural networks

Number of parameters ?



Artificial neural networks

Number of parameters ?



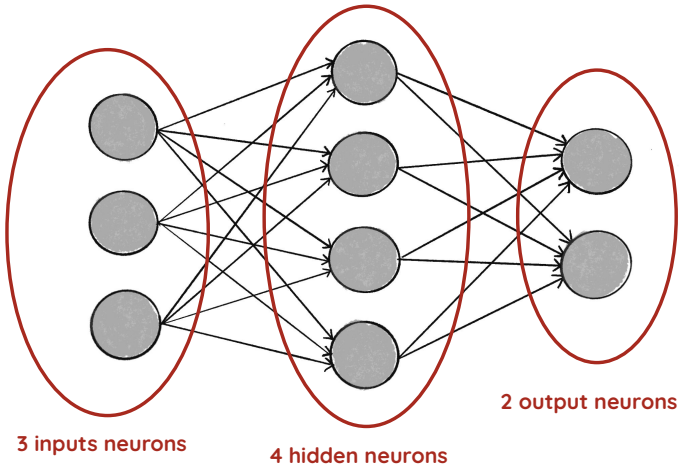
3 inputs neurons

4 hidden neurons

2 output neurons

Artificial neural networks

Number of parameters ?



Rule to remember !

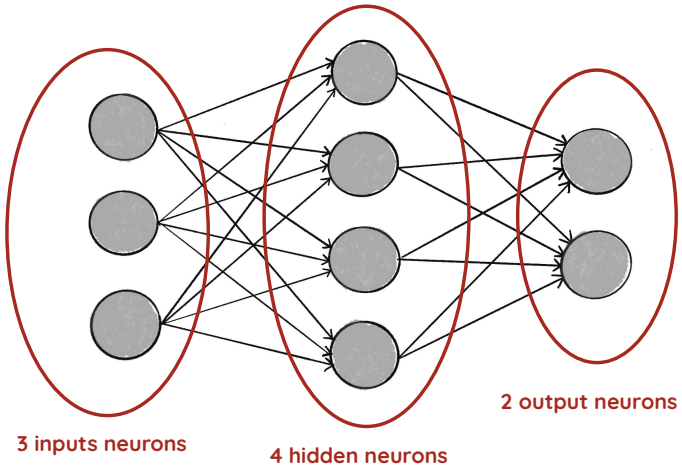
For two connected layers:

Parameters= (inputs×neurons) + (neurons)

Weights biases

Artificial neural networks

Number of parameters ?



Rule to remember !

For two connected layers:

Parameters= (inputs×neurons) + (neurons)

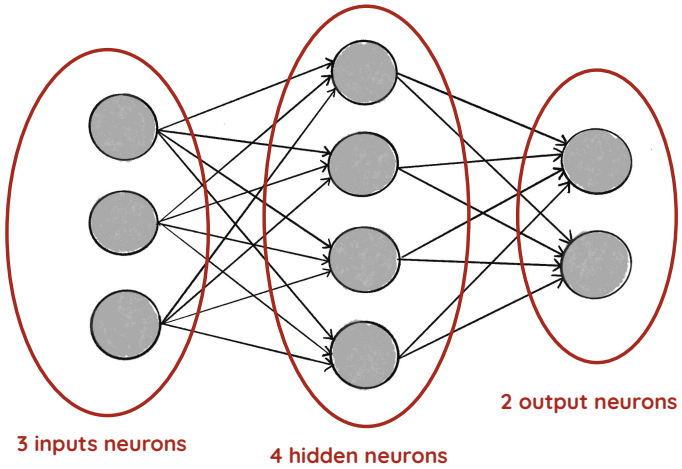
Weights biases

So what's the number of parameters ?

A/ 20
B/ 26
C/ 16

Artificial neural networks

Number of parameters ?



(3 x 4) + 4 = 16 (4 x 2) + 2 = 10

10 + 16 = 26

Rule to remember !

For two connected layers:

Parameters= (inputs×neurons) + (neurons)

Weights biases

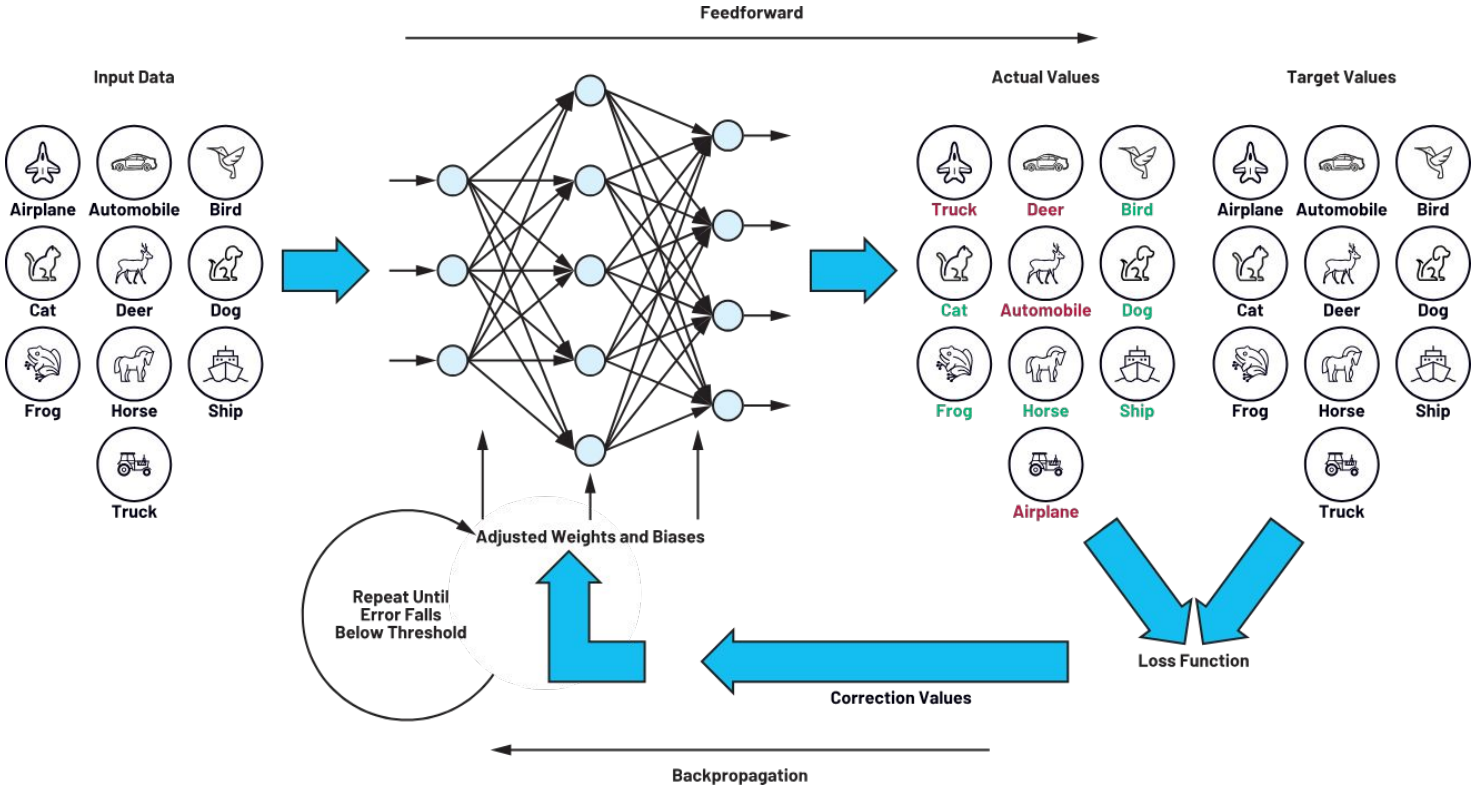
So what's the number of parameters ?

A/ 20

B/ 26

C/ 16

Training a neural network



Training a neural network

What is a loss function:

A loss that measures how wrong the network's prediction is.

- Mean Squared Error (MSE)
- Cross-Entropy Loss : used in image classifications problems

Training is an **optimization problem**, where we want to find the weights that minimize the **loss function**

The problem that :

- Networks have millions of parameters
- Each parameter affects the loss indirectly
- We need an efficient way to compute influence

Training a neural network

What is a Gradient?

- During training, we want to minimize the loss
- The gradient tells us:

  How much the loss changes when a parameter changes

- For a weight w : $\frac{\partial \mathrm{t}}{\partial \mathrm{W}}$

  If I slightly change w, how does the loss change?

Why Do We Need Gradients?

- Neural networks have millions of parameters
- We don’t know the best weights directly
- Gradients tell us:
  - Which direction reduces the loss
  - How strongly each parameter influences the loss

Training a neural network

Gradient Descent:

Parameters of the neural network are updated iteratively

$$\text{Repeat until converge } \{$$

$$\text{w} = \text{w} - \alpha \left[\frac{\partial Loss}{\partial w}\right]$$

$$\text{b} = \text{b} - \alpha \left[\frac{\partial Loss}{\partial b}\right]$$

$$\}$$

Training a neural network

Gradient Descent:

Parameters of the neural network are updated iteratively

Repeat Until Convergence {

    for i = 1...m {

        $\omega \leftarrow \omega - \alpha * \nabla_w L_m(w)$

    }

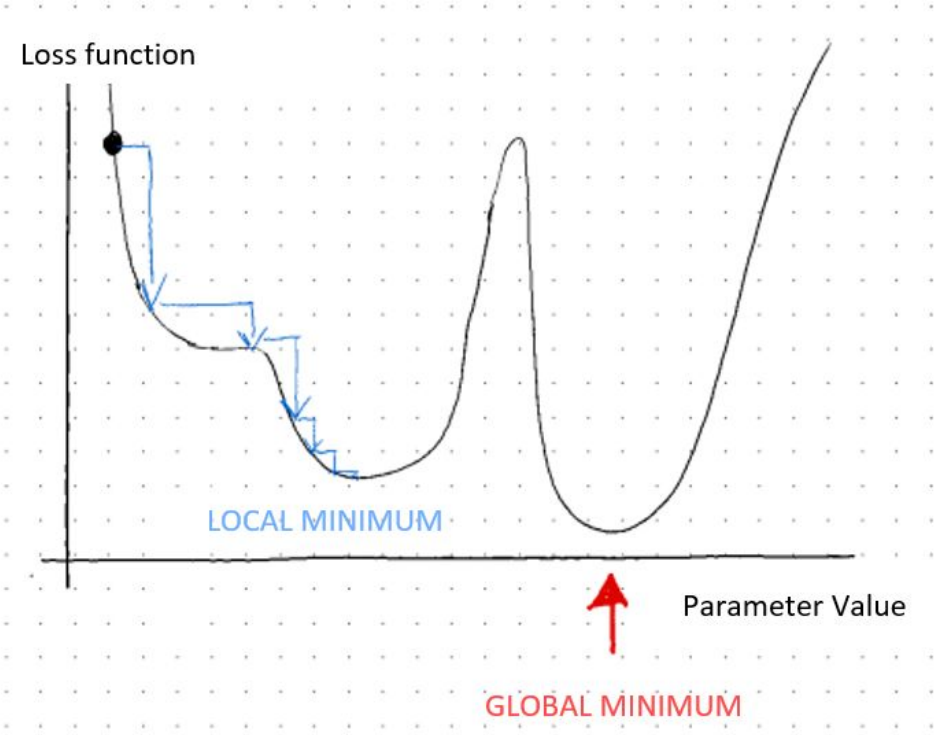
}

Learning rate (step size)

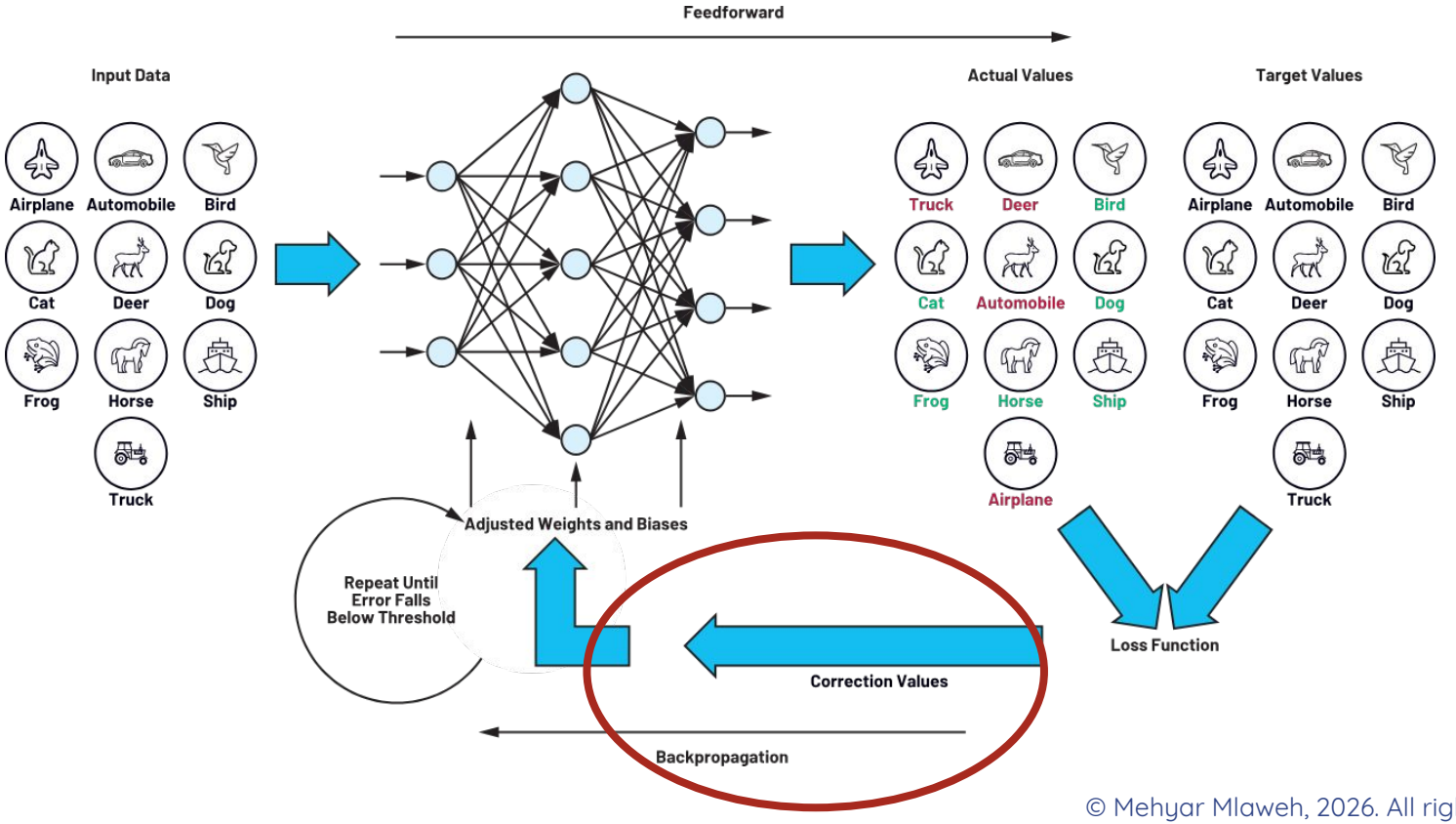
Training = repeatedly moving parameters in the direction that reduces loss

Training a neural network

Gradient Descent:



Training a neural network



Training a neural network

- Gradient descent updates parameters

- Backpropagation efficiently computes gradients for all parameters using the Chain Rule.

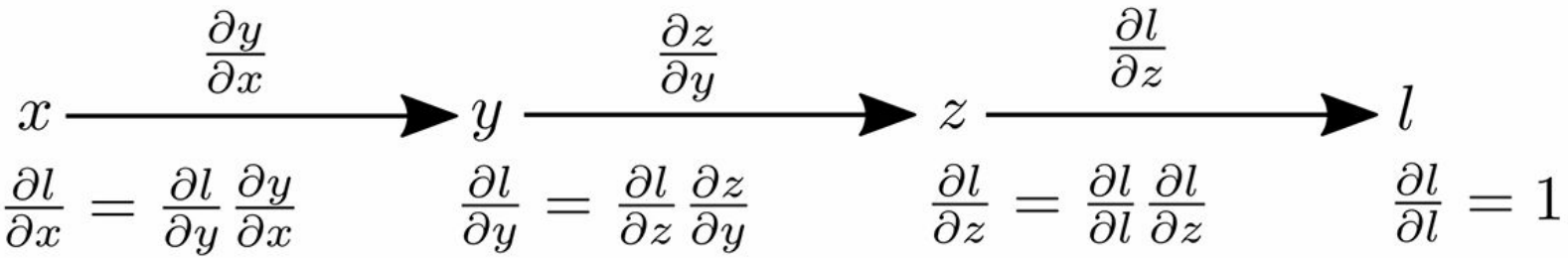
- It was first applied to NN in the eighties [Werbos, 1982, LeCun, 1985]

Chain Rule :

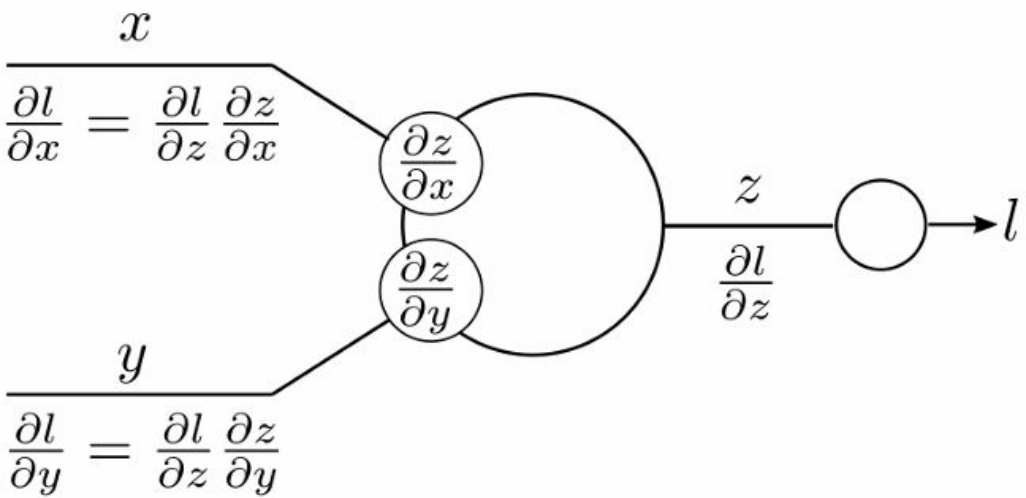
if f(x) and x(t) are functions, then

$$\frac{\mathrm{d}}{\mathrm{d}t} f(x(t)) = \frac{\mathrm{d}f}{\mathrm{d}x} \frac{\mathrm{d}x}{\mathrm{d}t}.$$

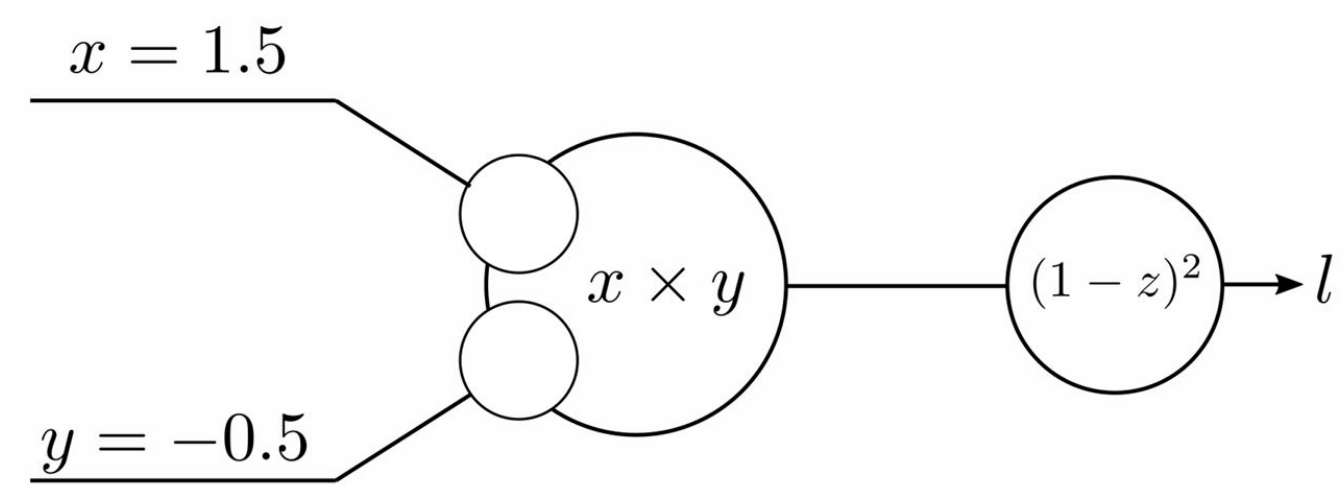
Simple example



Another example



Training a neural network



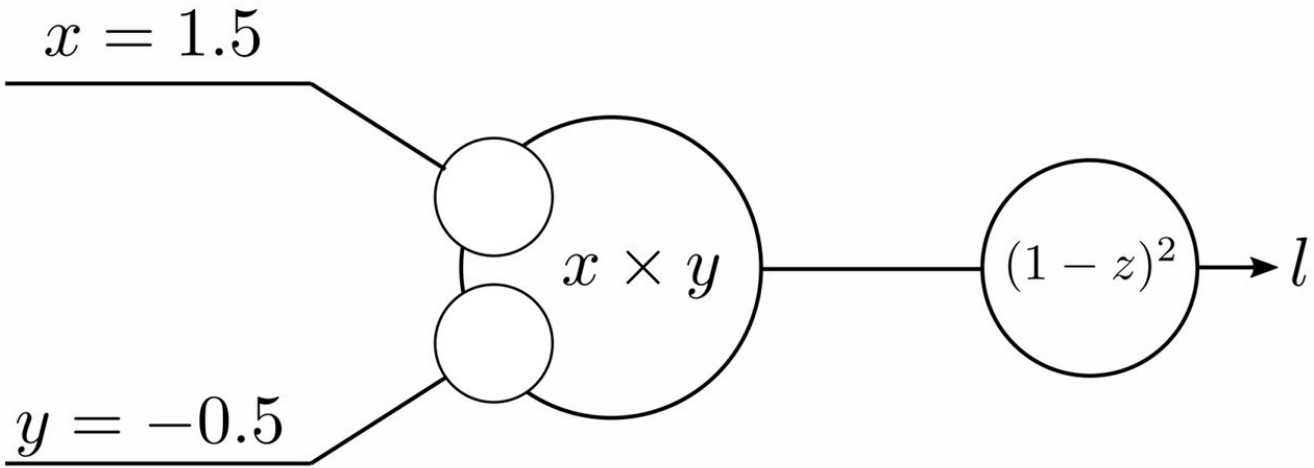
$\frac{\partial l}{\partial x}$ is equal to:

A/ 1.75

B/ -0.5

C/ 0.75

Training a neural network



$\frac{\partial l}{\partial x}$ is equal to:

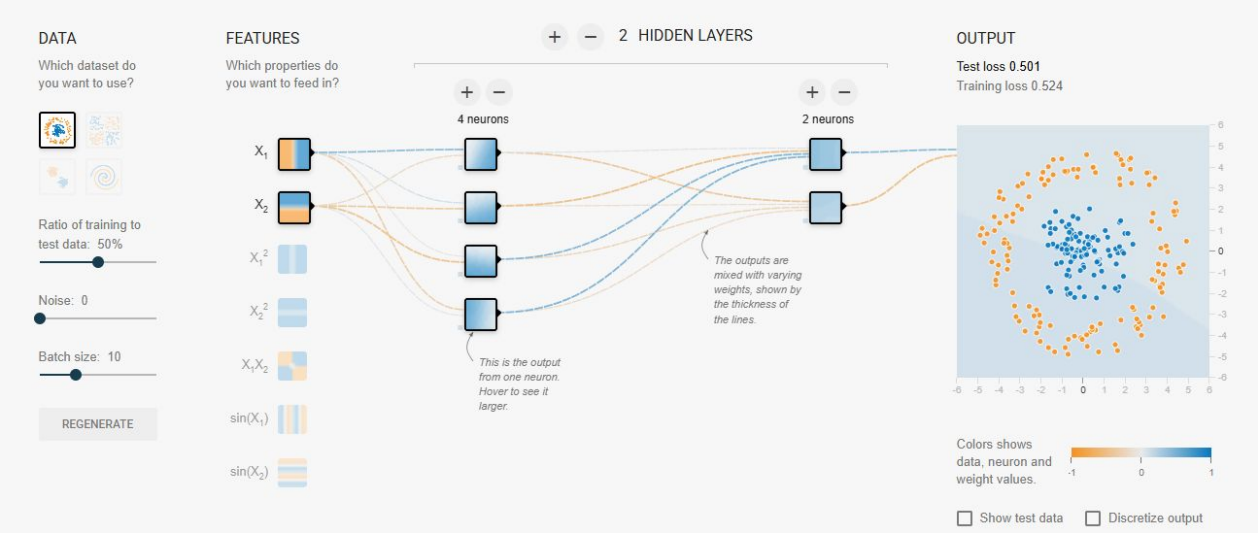
A/ 1.75

B/ -0.5

C/ 0.75

Let’s Play with Neural Networks

👉[Tensorflow Playground]



Next time ?

Convolutional Neural Networks

References

Books

• Computer Vision: Models, Learning, and Inference (2012). https://amzn.to/2rxrdOF
• Programming Computer Vision with Python (2012). https://amzn.to/2QKTaAL
• Multiple View Geometry in Computer Vision (2004). https://amzn.to/2LfHLE8
• Computer Vision: Algorithms and Applications (2010). https://amzn.to/2LcIt4J
• Computer Vision: A Modern Approach (2002). https://amzn.to/2rv7AqJ
• Deep Learning for Computer Vision : Image Classification, Object Detection and Face Recognition in Python(2019). https://machinelearningmastery.com/deep-learning-for-computer-vision/
• Hands-On Machine Learning with Scikit-Learn, Keras & TensorFlow — Aurélien Géron (2019) https://amzn.to/3uZbN6Z

Articles & Online Resources

• Krizhevsky, Sutskever, Hinton (2012) — ImageNet Classification with Deep CNNs (AlexNet — deep learning breakthrough)
• "Computer vision," Wikipedia. https://en.wikipedia.org/wiki/Computer_vision
• "Machine vision," Wikipedia. https://en.wikipedia.org/wiki/Machine_vision
• "Digital image processing," Wikipedia. https://en.wikipedia.org/wiki/Digital_image_processing
• "Training Convolutional Neural Networks: What Is Machine Learning" Analog Devices. https://www.analog.com/en/resources/analog-dialogue/articles/training-convolutional-neural-networks-what-is-machine-learning-part-2.html