AIDOS LAB
AI FOR DATA-ORIENTED SCIENCE

UNI FR

# Introduction to Deep Learning and Topological Deep Learning

Session 1 — Topological and Geometric Deep Learning: Theory, Methods and Applications

Universidad Complutense de Madrid

Inés García Redondo — May 11, 2026

# The course

What to expect

## *Introductory sessions*

Monday

**Session 1: Introduction to DL**

Tuesday

**Session 2: Introduction to TDA**

## *Current areas of research*

Wednesday

**Session 3: Learning from topological domains**

Thursday, morning

**Session 4: Generalization**

Thursday, afternoon

**Session 5: Interpretability**

Interventional: design new architectures or intervene the ones we have to inject *topological and geometric inductive biases*

Observational: use topology and geometry to better understand the behavior of DL systems

Interventional: incorporate this knowledge to influence their behavior in a desired way

# Session 1 — Introduction to Deep Learning and Topological Deep Learning

Outline

- What is deep learning?
  - Classification vs. regression tasks
- The multilayer-perceptron: the building block of DL
- Training neural networks
  - Gradient descent
  - Backpropagation
- Learning from images:
  - Convolutional neural networks
- Learning from text:
  - Transformers and the attention mechanism

# What is Deep Learning?

Session 1 — Topological and Geometric Deep Learning: Theory, Methods and Applications

Inés García Redondo — May 11, 2026

# The power of AI

And how it's shaping our world

AlphaGo

Article

**Olympiad-level formal mathematical reasoning with reinforcement learning**

https://doi.org/10.1038/s41586-025-09833-y
Received: 3 June 2025
Accepted: 30 October 2025
Published online: 12 November 2025
Open access
Check for updates

Thomas Hubert[1]✉, Rishi Mehta[1], Laurent Sartran[1], Miklós Z. Horváth[1], Goran Žužić[1], Eric Wieser[1]✉, Aja Huang[1], Julian Schrittwieser[1], Yannick Schroecker[1], Hussain Masoom[1], Ottavia Bertolli[1], Tom Zahavy[1], Amol Mandhane[1], Jessica Yung[1], Iuliya Beloshapka[1], Borja Ibarz[1], Vivek Veeriah[1], Lei Yu[1], Oliver Nash[1], Paul Lezeau[1], Salvatore Mercuri[1], Calle Sönne[1], Bhavik Mehta[1], Alex Davies[1], Daniel Zheng[1], Fabian Pedregosa[1], Yin Li[1], Ingrid von Glehn[1], Mark Rowland[1], Samuel Albanie[1], Ameya Velingker[1], Simon Schmitt[1], Edward Lockhart[1], Edward Hughes[1], Henryk Michalewski[1], Nicolas Sonnerat[1], Demis Hassabis[1], Pushmeet Kohli[1]✉ & David Silver[1]

AlphaFold 2/3


AlphaGeometry and AlphaProof

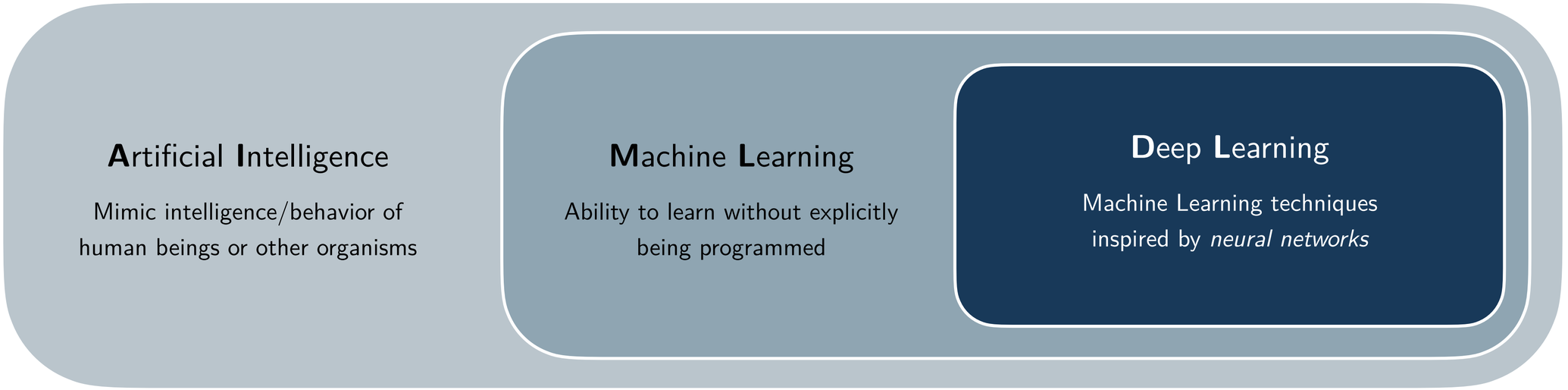
Gemini

Large Language Models

DALL·E


Image and video generation

# What is Deep Learning?




Teaching computers how to **learn a task** directly from **raw data**

# What is Deep Learning?

**A**rtificial **I**ntelligence: are we there yet?

> *Intelligence is not a scalar quantity. The space of problems is gigantic and [...] any intelligent system will only excel at a tiny subset of them. Any intelligent system has blind spots, hence vulnerabilities.*
>
> Yann LeCun

Best to think of AI as *PI* (pseudo-intelligence) or *SI* (simulated intelligence).

Our systems are *not even close* to real intelligence.

# Learning a task

The set up for supervised learning

**Data space:** Probability space: $(Z, \mathcal{F}, \mu)$ $Z = X \times Y$

Features, Labels

**Training data:** $\{x_1, \ldots, x_n\} \subset X$ $\{y_1, \ldots, y_n\} \subset Y$

**Task:**

Model

Learn $\hat{y} = f_\theta(x), \quad \theta \in \mathbb{R}^D$

s.t. $\mathcal{R} = \sum_{i=1}^{n} \mathcal{L}\left(f_\theta(x_i), y_i\right)$ is minimized

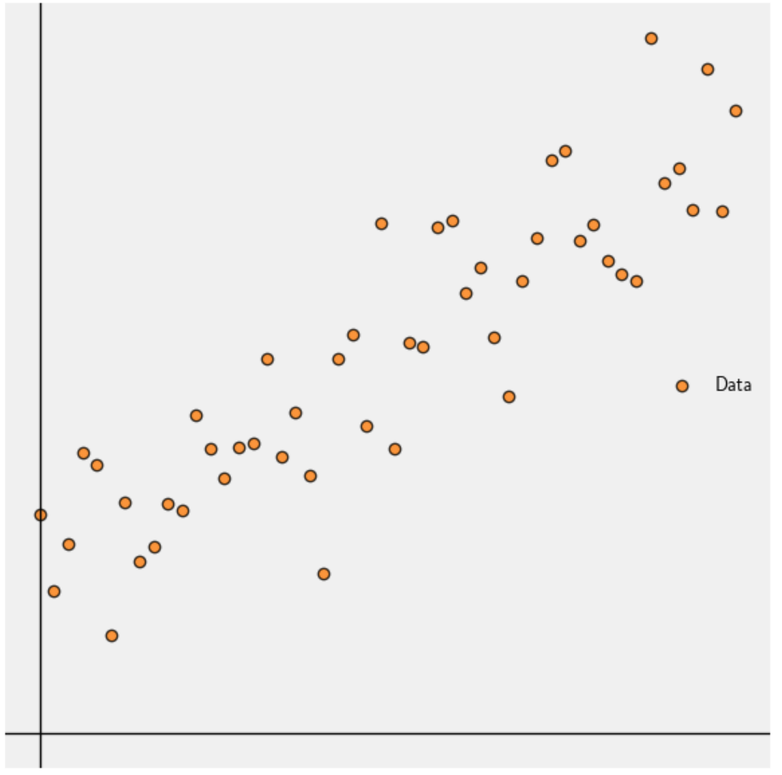
Empirical risk

Loss

# Learning a task

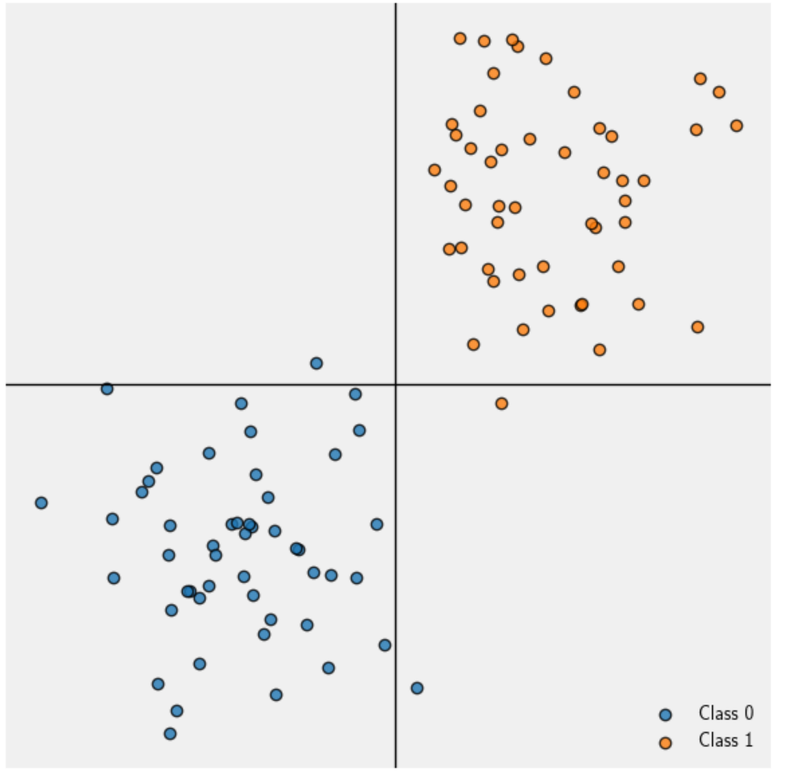
The set up for supervised learning

## Regression




$X = \mathbb{R}, \quad Y = \mathbb{R}$

## Classification

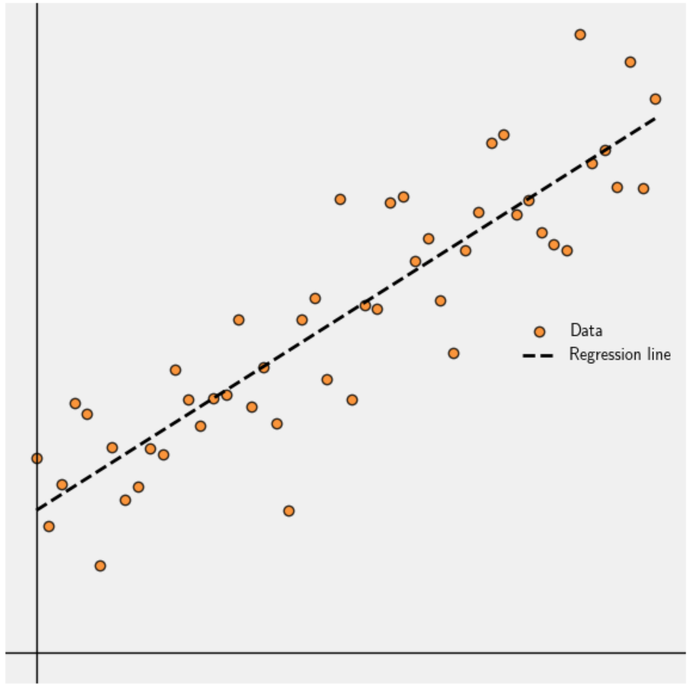



$X = \mathbb{R}^2, \quad Y = \{0, 1\}$

# Learning a task

The set up for supervised learning

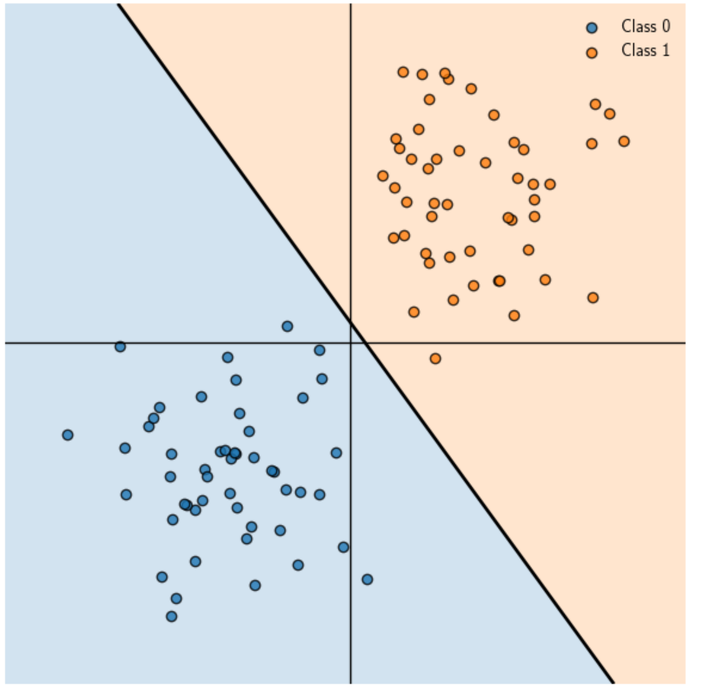
## Regression




Linear regression $f_{m,b}(x) = mx + b$

Mean Squared Error $\mathscr{R} = MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$

## Classification



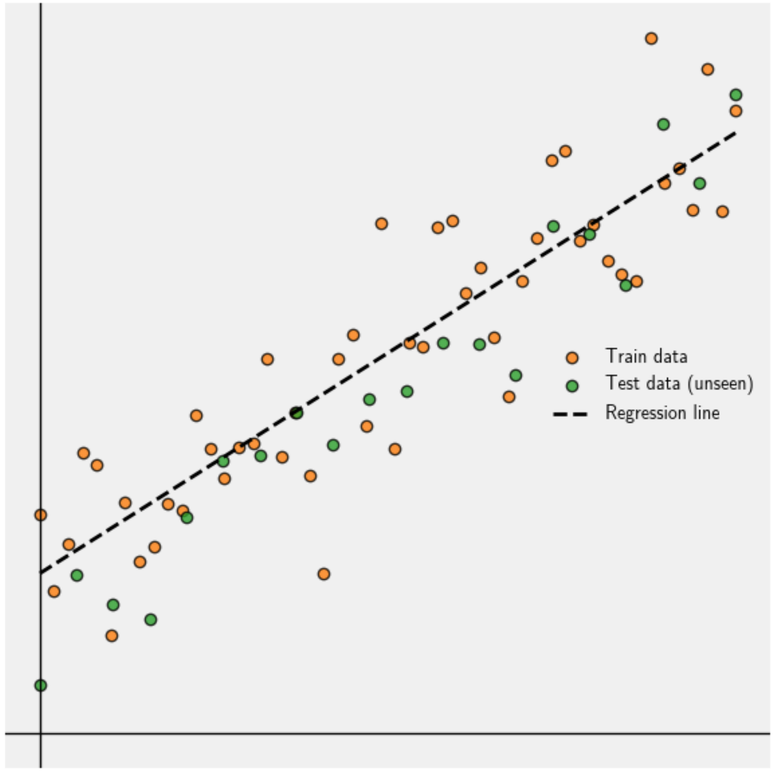

Logistic regression $f_{w,b}(\mathbf{x}) = \sigma(\mathbf{w}^\top \mathbf{x} + b)$

Cross-Entropy Loss $\mathscr{R} = -\frac{1}{n}\sum_{i=1}^{n}\left[y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)\right]$

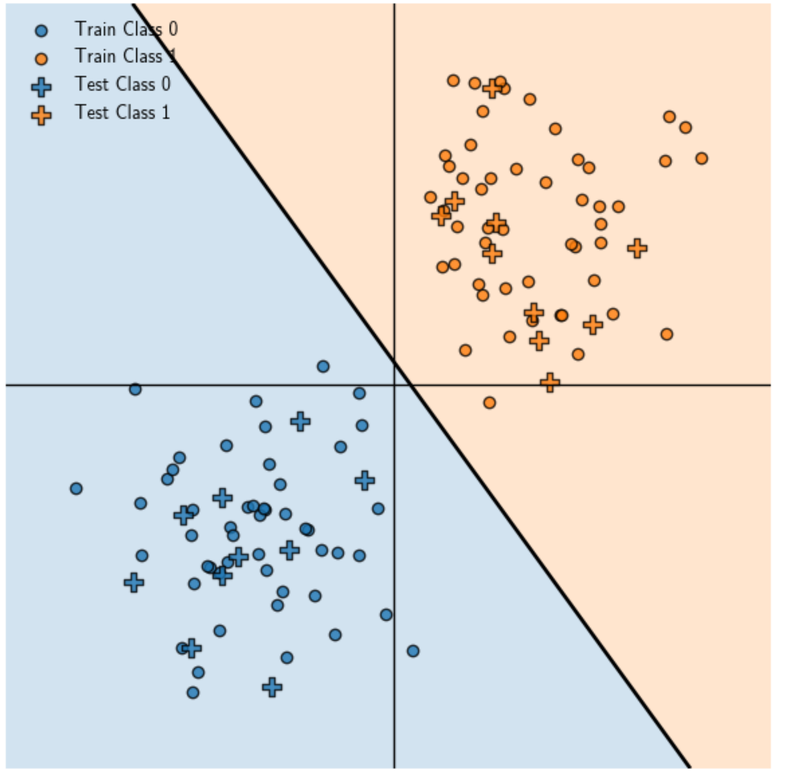
# Learning a task

Generalization

## Regression




## Classification

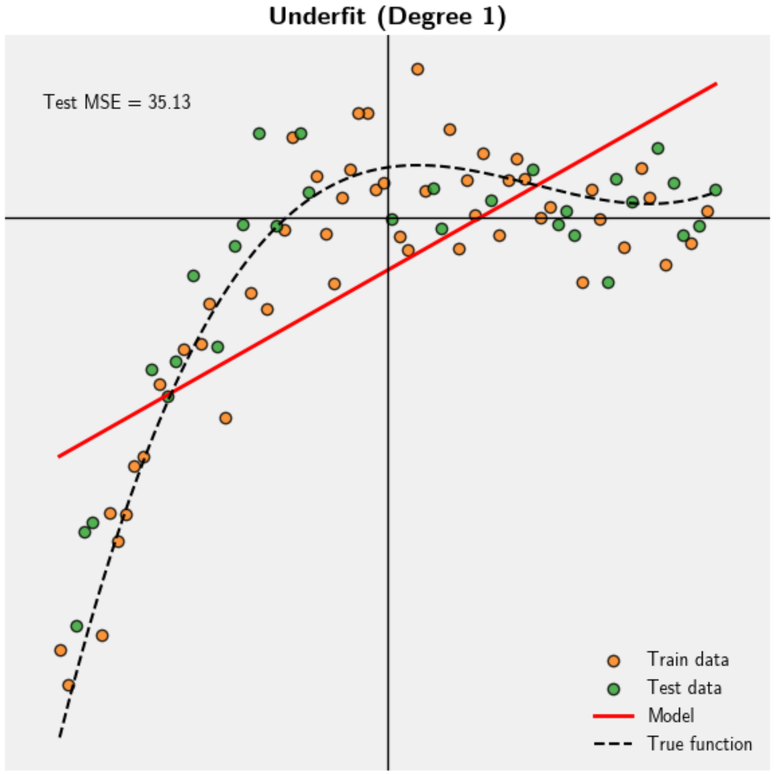


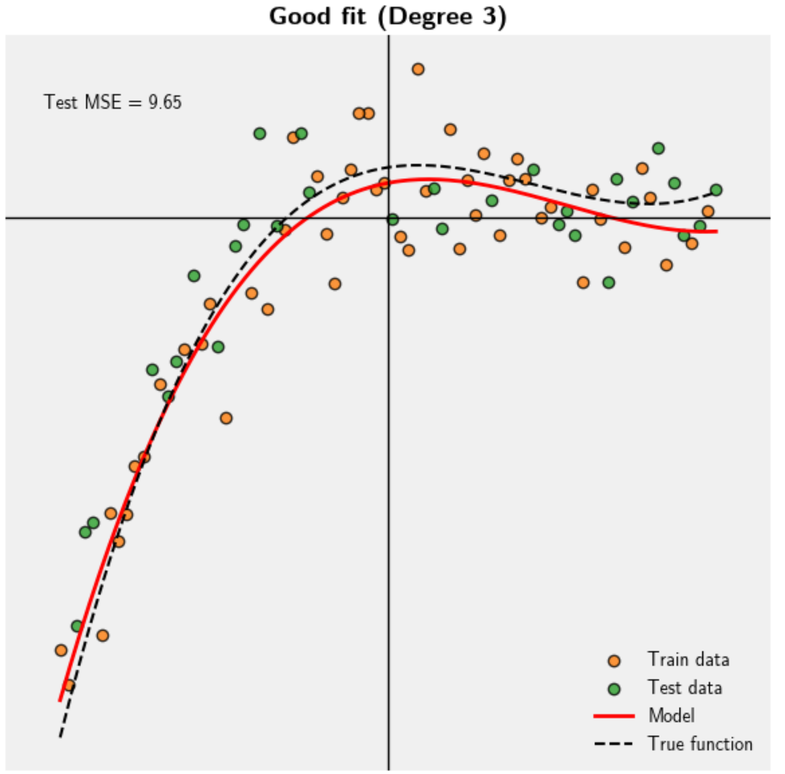

We seek that the model *generalizes* well: provides good predictions for unseen data

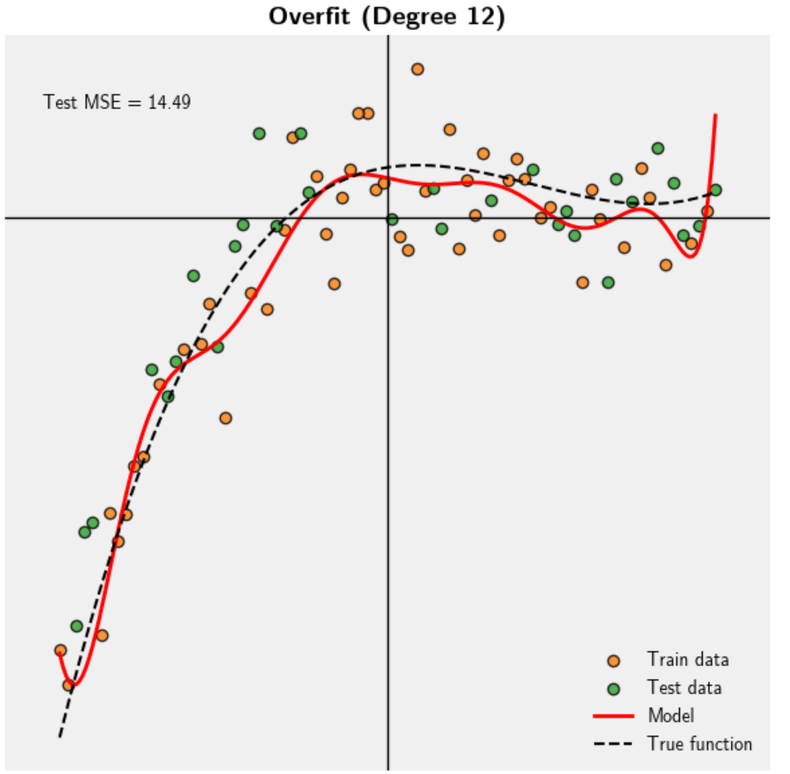
# Learning a task

Underfitting and overfitting

Fit $\frac{1}{2}x^3 - 2x^2 + x + 3$



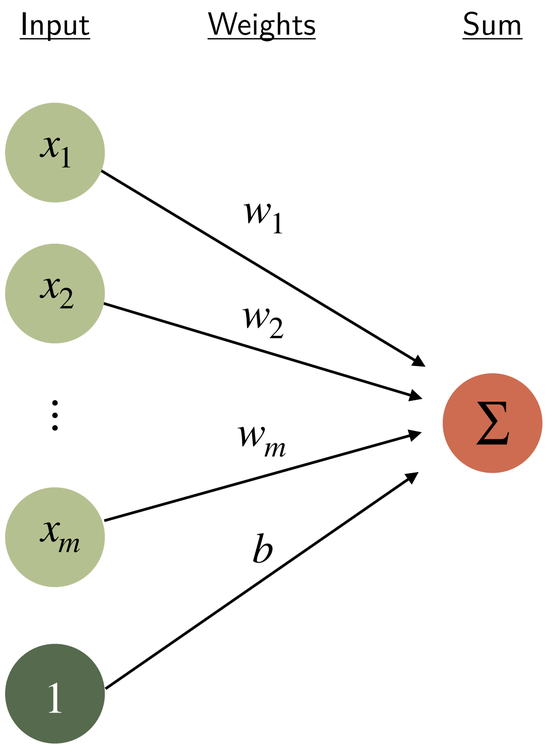
# Neural Networks: the Multilayer Perceptron

Session 1 — Topological and Geometric Deep Learning: Theory, Methods and Applications

Inés García Redondo — May 11, 2026

# The Perceptron

The structural building block of deep learning

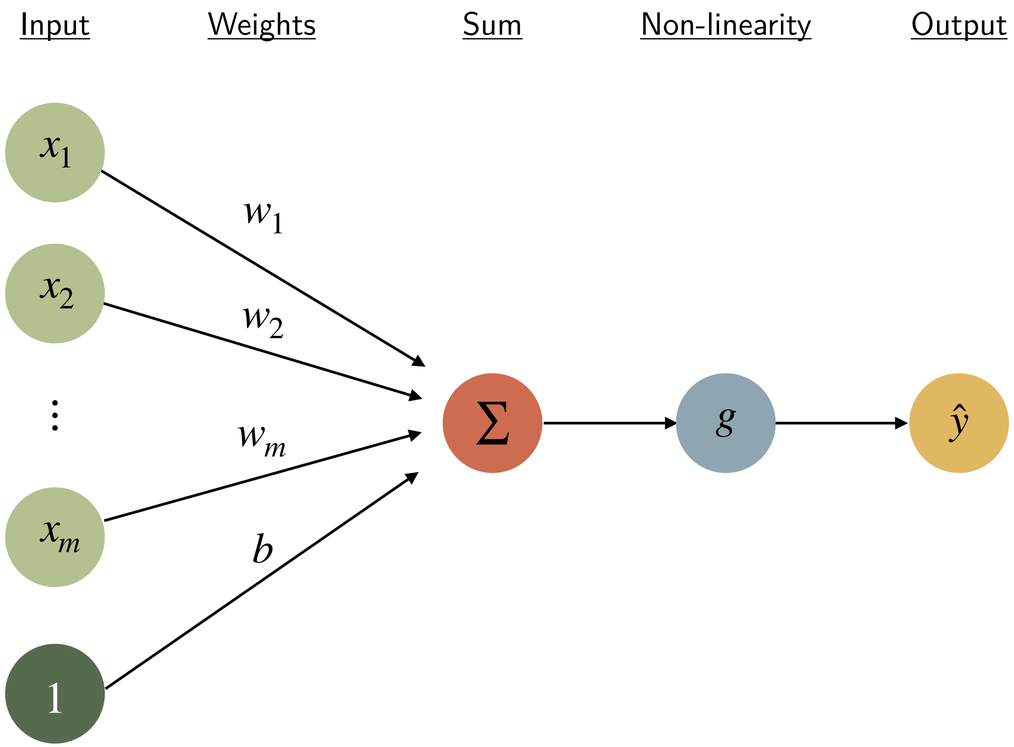



$$\mathbf{w} = (w_1, \ldots, w_m)$$

$$\mathbf{x} = (x_1, \ldots, x_m)$$

# The Perceptron

The structural building block of deep learning




Non-linearity

Bias

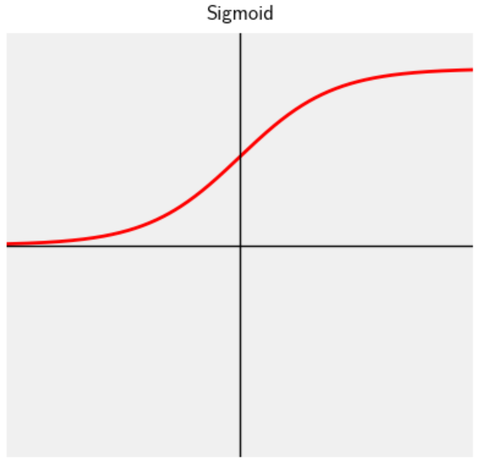
$$\hat{y} = g\left(\sum_{i=1}^{m} w_i x_i + b\right) = g\left(\mathbf{w} \cdot \mathbf{x} + b\right)$$

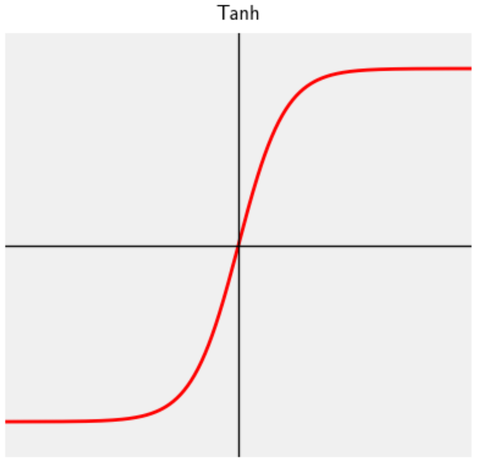
Weigths

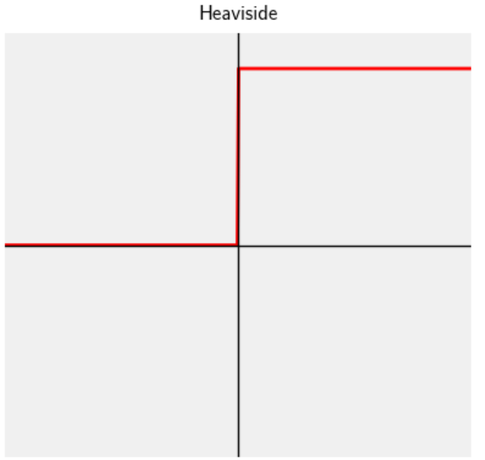
$$\mathbf{w} = (w_1, \ldots, w_m) \qquad \mathbf{x} = (x_1, \ldots, x_m)$$

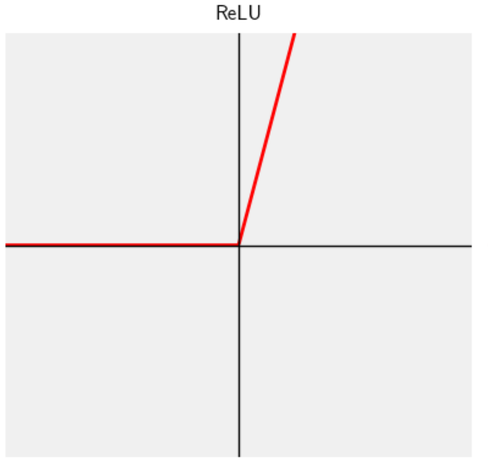
# Common activation functions

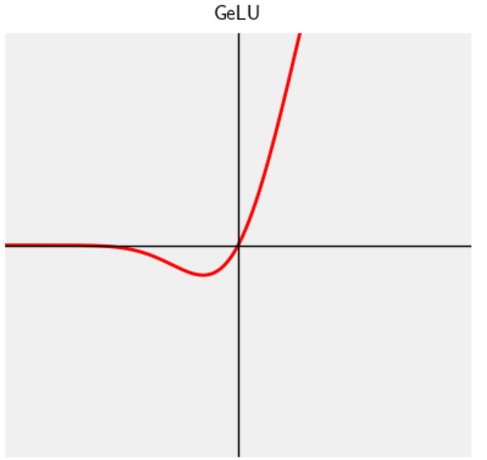
$$\hat{y} = g\left(\sum_{i=1}^{m} w_i x_i + b\right) = g\left(\mathbf{w} \cdot \mathbf{x} + b\right)$$




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

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

$$g(z) = \begin{cases} 0 & \text{if } z < 0 \\ 0.5 & \text{if } z = 0 \\ 1 & \text{if } 1x > 0 \end{cases}$$

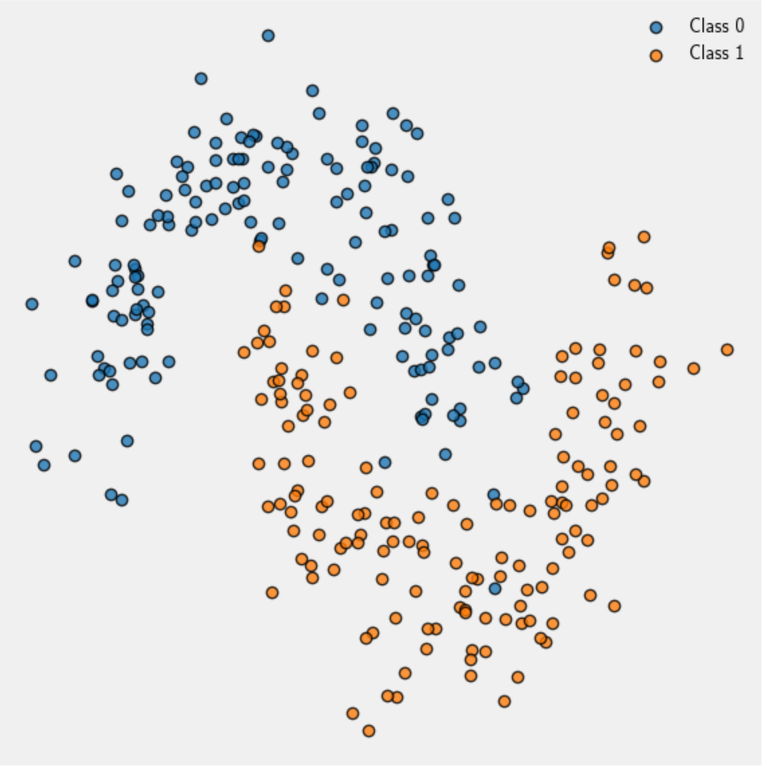
$$g(z) = \max(0, z)$$

$$g(z) = z \cdot \Phi(z)$$

Cumulative distribution function of the standard normal distribution

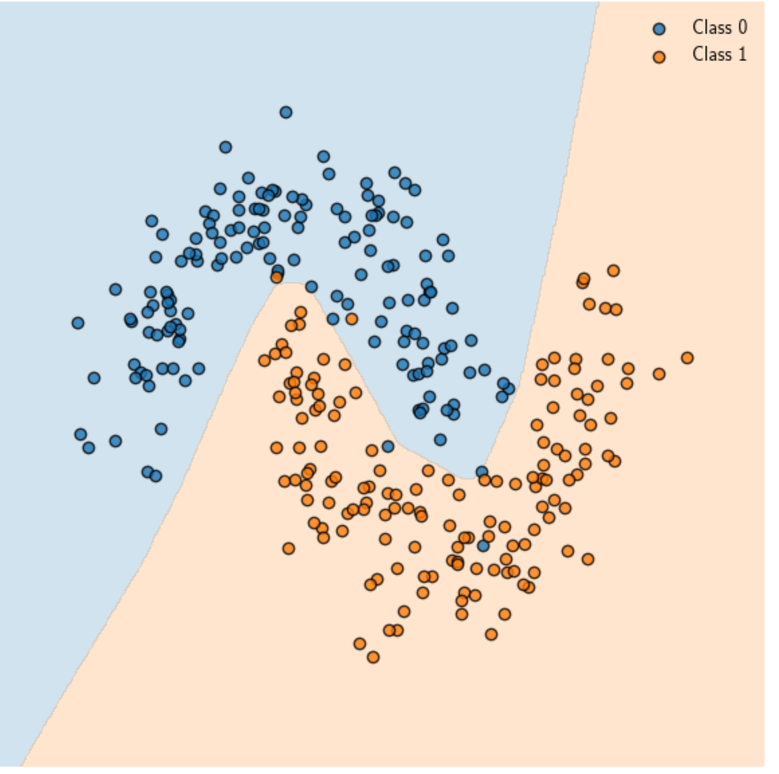
Note: they must be non-linear

# Why non linear activations?




What happens if the data is non linearly separable?

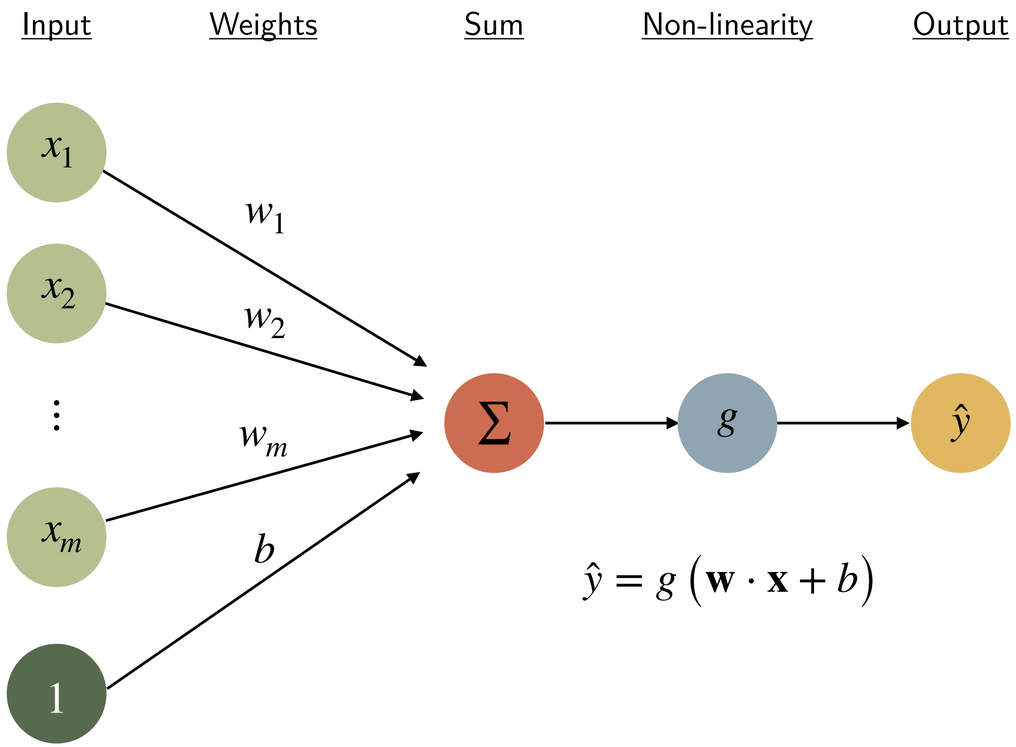
# Why non linear activations?




Non-linearities allow us to approximate arbitrarily complex functions

# The Perceptron

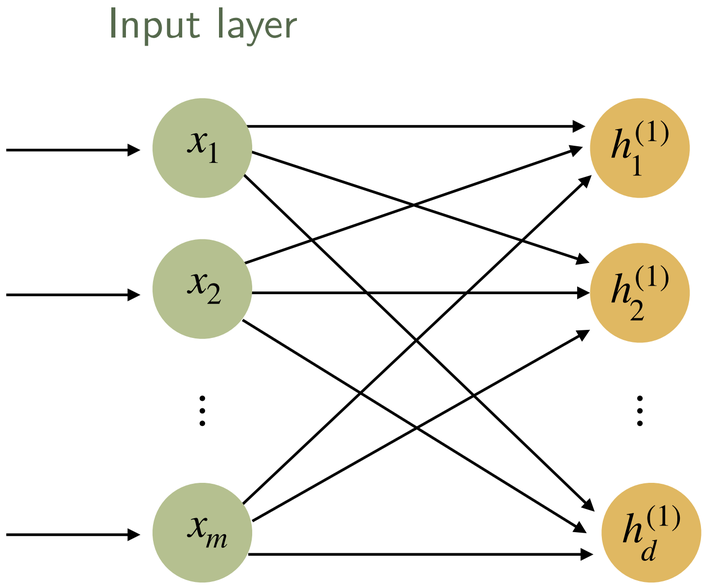
The structural building block of deep learning




The perceptron is a *single* neuron, which is *activated* based on the output of an *activation function*, evaluated on a weighted sum of inputs

# The Multi-Layer Perceptron

Components




$$\mathbf{h}^{(1)} = g\left(\mathbf{W}^{(1)} \cdot \mathbf{x} + \mathbf{b}^{(1)}\right)$$

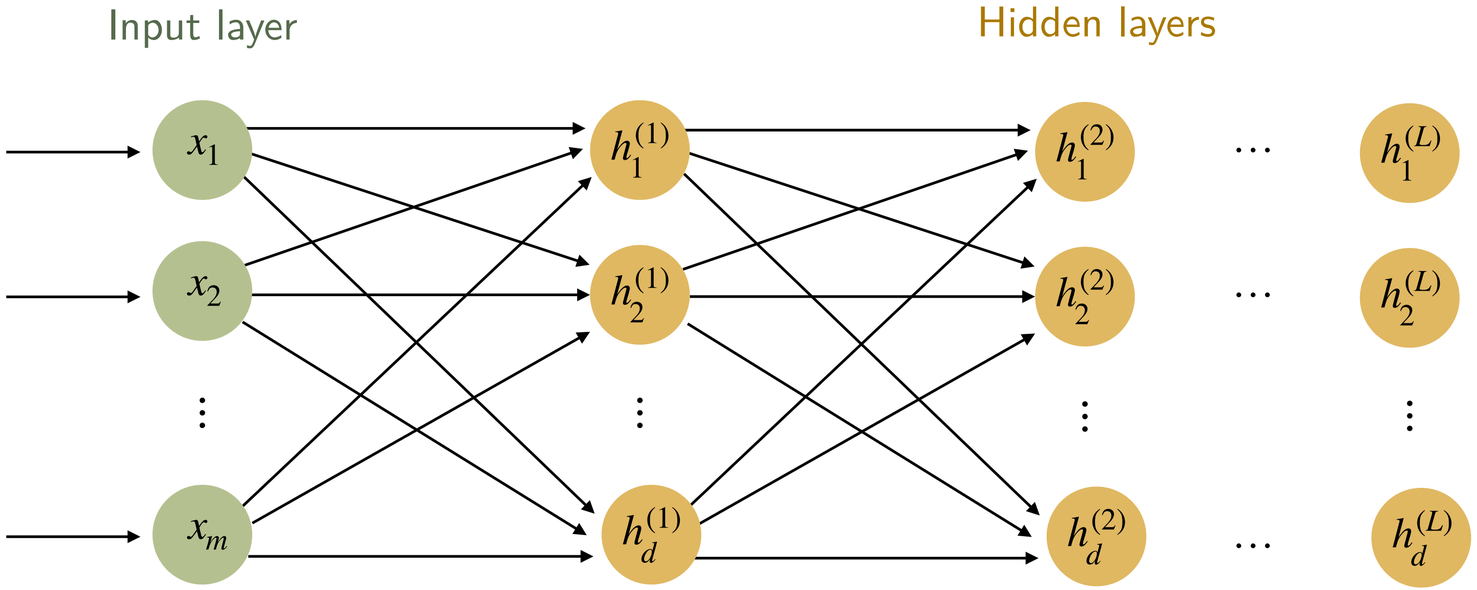
$$\mathbf{h}^{(1)} = (h_1^{(1)}, \ldots, h_d^{(1)})$$

$$\mathbf{W}^{(1)} = \begin{pmatrix} w_{1,1}^{(1)} & \cdots & w_{1,m}^{(1)} \\ \vdots & \ddots & \vdots \\ w_{d,1}^{(1)} & \cdots & w_{d,m}^{(1)} \end{pmatrix}$$

$$\mathbf{b}^{(1)} = (b_1^{(1)}, \ldots, b_d^{(1)})$$

# The Multi-Layer Perceptron

Components




$$\mathbf{h}^{(1)} = g\left(\mathbf{W}^{(1)} \cdot \mathbf{x} + \mathbf{b}^{(1)}\right)$$

$$\mathbf{h}^{(l)} = g\left(\mathbf{W}^{(l)} \cdot \mathbf{h}^{(l-1)} + \mathbf{b}^{(l)}\right)$$

$$\mathbf{h}^{(1)} = (h_1^{(1)}, \ldots, h_d^{(1)})$$

$$\mathbf{h}^{(l)} = (h_1^{(l)}, \ldots, h_d^{(l)})$$

$$\mathbf{W}^{(1)} = \begin{pmatrix} w_{1,1}^{(1)} & \cdots & w_{1,m}^{(1)} \\ \vdots & \ddots & \vdots \\ w_{d,1}^{(1)} & \cdots & w_{d,m}^{(1)} \end{pmatrix}$$

$$\mathbf{W}^{(l)} = \begin{pmatrix} w_{1,1}^{(l)} & \cdots & w_{1,d}^{(l)} \\ \vdots & \ddots & \vdots \\ w_{d,1}^{(l)} & \cdots & w_{d,d}^{(l)} \end{pmatrix}$$

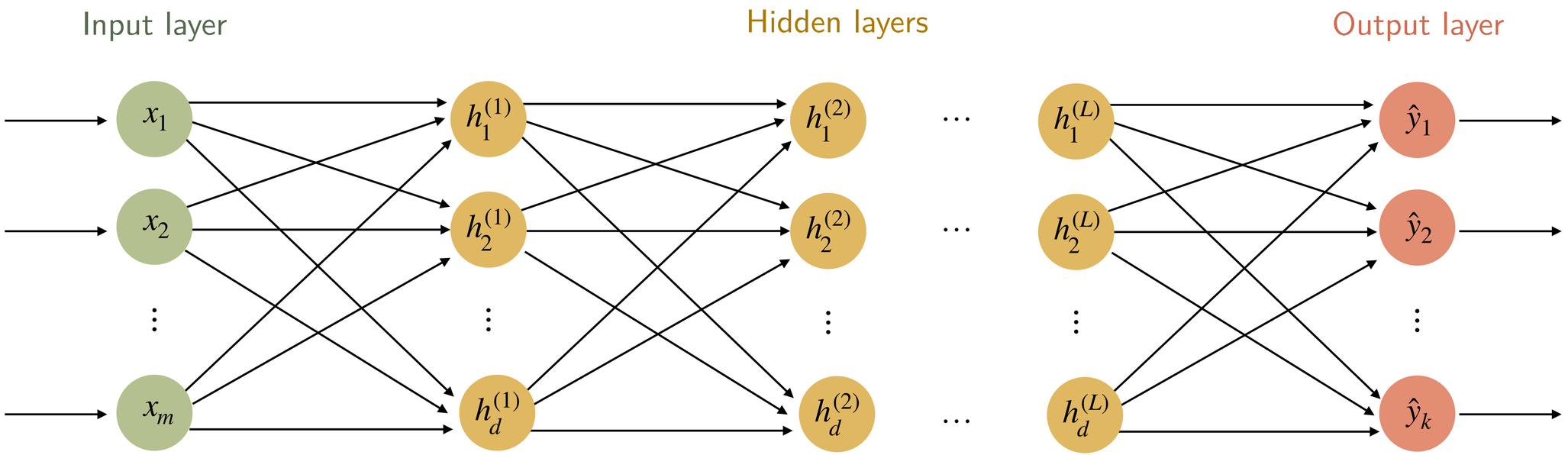
$$\mathbf{b}^{(1)} = (b_1^{(1)}, \ldots, b_d^{(1)})$$

$$\mathbf{b}^{(l)} = (b_1^{(l)}, \ldots, b_d^{(l)})$$

# The Multi-Layer Perceptron

Components

$$f_\theta : \mathbb{R}^m \to \mathbb{R}^k, \quad \theta \in \mathbb{R}^D$$




$$\mathbf{h}^{(1)} = g\left(\mathbf{W}^{(1)} \cdot \mathbf{x} + \mathbf{b}^{(1)}\right) \qquad \mathbf{h}^{(l)} = g\left(\mathbf{W}^{(l)} \cdot \mathbf{h}^{(l-1)} + \mathbf{b}^{(l)}\right) \qquad \hat{\mathbf{y}} = g\left(\mathbf{W}^{(L+1)} \cdot \mathbf{h}^{(L)} + \mathbf{b}^{(L+1)}\right)$$

$$\mathbf{h}^{(1)} = (h_1^{(1)}, \ldots, h_d^{(1)}) \qquad \mathbf{h}^{(l)} = (h_1^{(l)}, \ldots, h_d^{(l)}) \qquad \hat{\mathbf{y}} = (\hat{y}_1, \ldots, \hat{y}_k)$$

$$\mathbf{W}^{(1)} = \begin{pmatrix} w_{1,1}^{(1)} & \cdots & w_{1,m}^{(1)} \\ \vdots & \ddots & \vdots \\ w_{d,1}^{(1)} & \cdots & w_{d,m}^{(1)} \end{pmatrix} \qquad \mathbf{W}^{(l)} = \begin{pmatrix} w_{1,1}^{(l)} & \cdots & w_{1,d}^{(l)} \\ \vdots & \ddots & \vdots \\ w_{d,1}^{(l)} & \cdots & w_{d,d}^{(l)} \end{pmatrix} \qquad \mathbf{W}^{(L+1)} = \begin{pmatrix} w_{1,1}^{(L+1)} & \cdots & w_{1,d}^{(L+1)} \\ \vdots & \ddots & \vdots \\ w_{k,1}^{(L+1)} & \cdots & w_{k,d}^{(L+1)} \end{pmatrix}$$

$$\mathbf{b}^{(1)} = (b_1^{(1)}, \ldots, b_d^{(1)}) \qquad \mathbf{b}^{(l)} = (b_1^{(l)}, \ldots, b_d^{(l)}) \qquad \mathbf{b}^{(L+1)} = (b_1^{(L+1)}, \ldots, b_k^{(L+1)})$$

# Universal Function Approximation

Why MLPs work for non-linear activations

**Theorem (Universal function approximation)**

Let $\sigma \in C(\mathbb{R}, \mathbb{R})$ be a non-polynomial activation function. For every $m, k \in \mathbb{N}$, every compact subset $K \subset \mathbb{R}^m$, every function $f \in C(K, \mathbb{R}^k)$ and $\epsilon > 0$, there exists $d \in \mathbb{R}$, $\mathbf{A} \in \mathbb{R}^{d \times m}$, $\mathbf{b} \in \mathbb{R}^d$, and $\mathbf{C} \in \mathbb{R}^{k \times d}$ such that

$$\sup_{x \in K} \|f(x) - g(x)\| < \epsilon,$$

where $g(x) = \mathbf{C}\sigma(\mathbf{A}x + b)$.

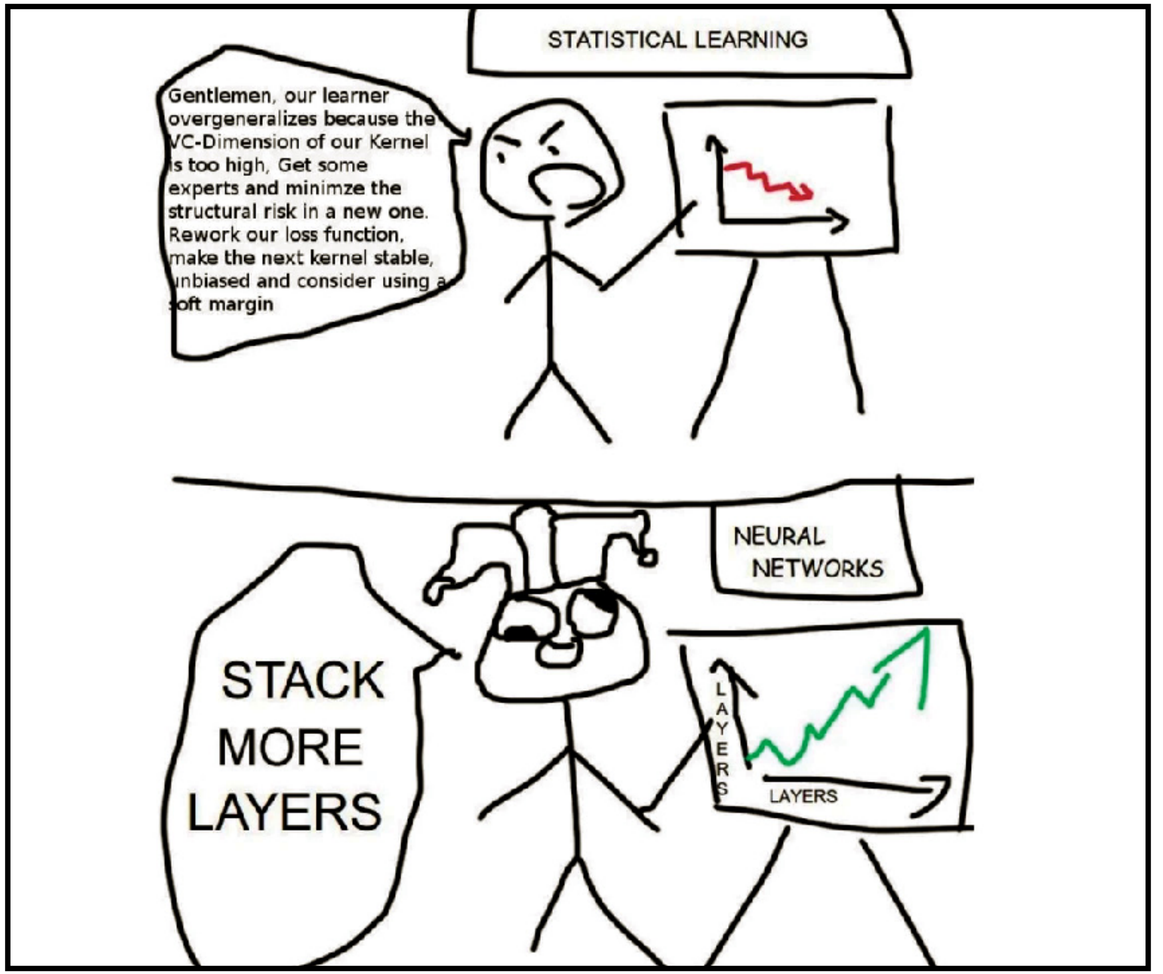
**Implication**

*Theoretically* we can approximate any continuous function if we make no restrictions on the *width* $d$ of our neural network.

A. Pinkus, "Approximation theory of the MLP model in neural networks", *Acta Numerica 8, 1999, pp.143-195*

# Universal Function Approximation

Why MLPs work for non-linear activations



# Training a neural network

Session 1 — Topological and Geometric Deep Learning: Theory, Methods and Applications

Inés García Redondo — May 11, 2026

# Training a neural network

Gradient descent

Training point

**Input**: $f_\theta : \mathbb{R}^m \to \mathbb{R}^k, \quad \mathscr{L}(\mathbf{x}, \theta)$

1. Initialize weights randomly: $\theta_0$
2. Loop until convergence:
    1. Compute gradient: $\frac{\partial \mathscr{L}}{\partial \theta} \in \mathbb{R}^D$
    2. Update weights:

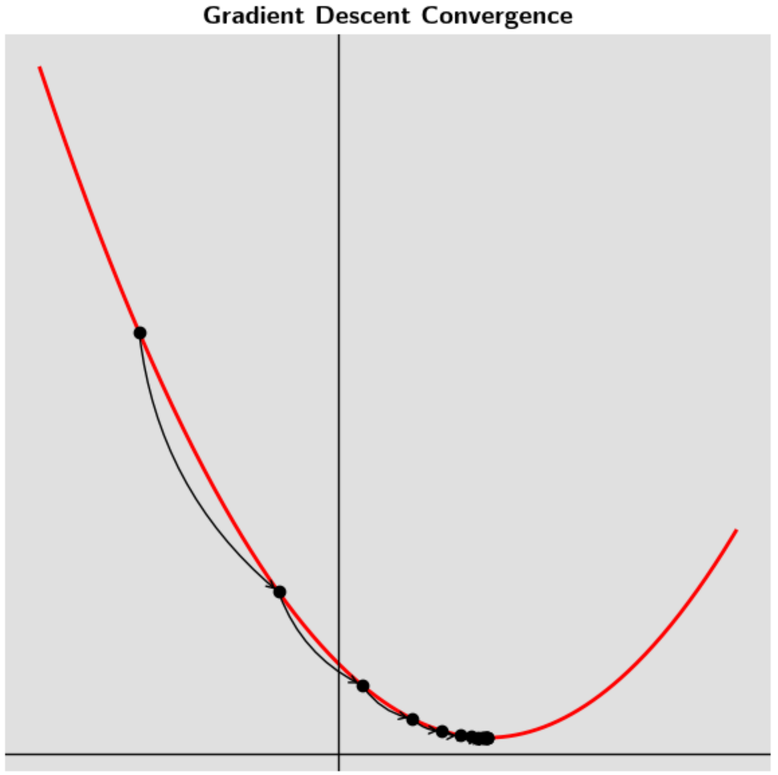
Learning rate

Single sample: $\theta_{t+1} \leftarrow \theta_t + \eta \frac{\partial \mathscr{L}}{\partial \theta}(\mathbf{x}, \theta_t)$

All training data: $\theta_{t+1} \leftarrow \theta_t + \eta \frac{1}{n} \sum_{i=1}^{n} \frac{\partial \mathscr{L}}{\partial \theta}(\mathbf{x}_i, \theta_t)$

**Output:** Optimal weights after convergence $\theta_T$

**Computational cost:** $O(n \cdot D)$



# Training a neural network

*Stochastic* gradient descent

Training point

**Input**: $f_\theta : \mathbb{R}^m \to \mathbb{R}^k, \quad \mathscr{L}(\mathbf{x}, \theta)$

1. Initialize weights randomly: $\theta_0$
2. Loop until convergence:
    1. Compute gradient: $\dfrac{\partial \mathscr{L}}{\partial \theta} \in \mathbb{R}^D$
    2. Update weights:

Learning rate

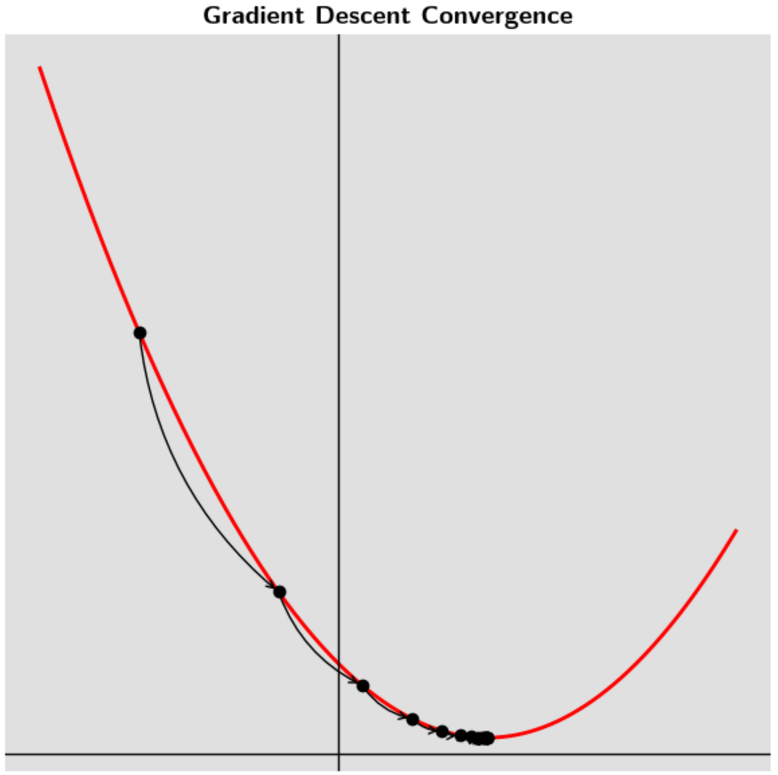
$$\text{Single sample: } \theta_{t+1} \leftarrow \theta_t + \eta \frac{\partial \mathscr{L}}{\partial \theta}(\mathbf{x}, \theta_t)$$

$$\text{Batch of data: } \theta_{t+1} \leftarrow \theta_t + \eta \frac{1}{B} \sum_{i=1}^{B} \frac{\partial \mathscr{L}}{\partial \theta}(\mathbf{x}_i, \theta_t)$$

$B$ is called the *batch size*

**Output:** Optimal weights after convergence $\theta_T$

**Computational cost:** $O(B \cdot D)$



# Training a neural network

*Stochastic* gradient descent

Training point

**Input**: $f_\theta : \mathbb{R}^m \to \mathbb{R}^k, \quad \mathscr{L}(\mathbf{x}, \theta)$

1. Initialize weights randomly: $\theta_0$
2. Loop until convergence:
    1. Compute gradient: $\dfrac{\partial \mathscr{L}}{\partial \theta} \in \mathbb{R}^D$
    2. Update weights:

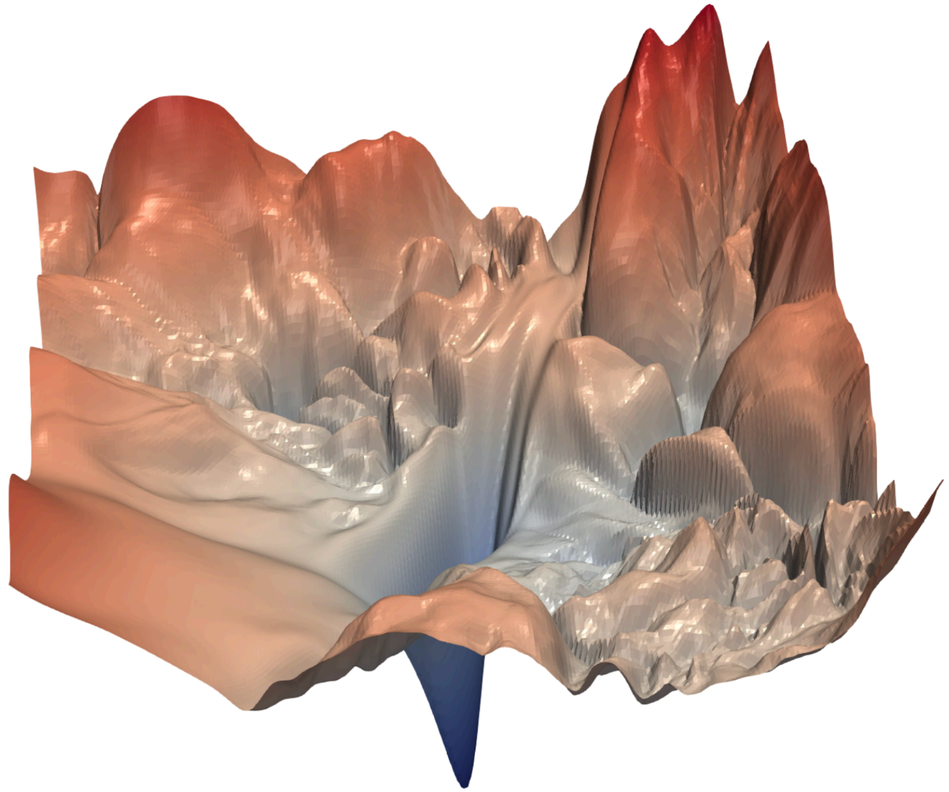
Learning rate

$$\text{Single sample: } \theta_{t+1} \leftarrow \theta_t + \eta \frac{\partial \mathscr{L}}{\partial \theta}(\mathbf{x}, \theta_t)$$

$$\text{Batch of data: } \theta_{t+1} \leftarrow \theta_t + \eta \frac{1}{B} \sum_{i=1}^{B} \frac{\partial \mathscr{L}}{\partial \theta}(\mathbf{x}_i, \theta_t)$$

$B$ is called the *batch size*

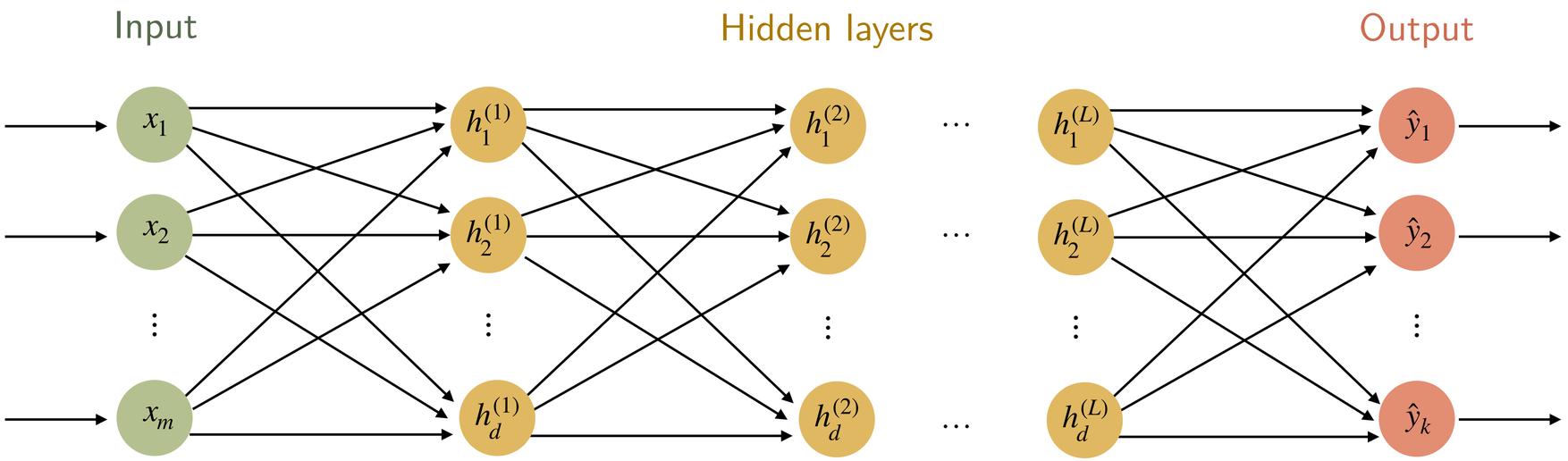
**Output:** Optimal weights after convergence $\theta_T$



*H. Li et al., Visualizing the Loss Landscape of Neural Nets, NeurIPS 2018*

# Training a neural network

Backpropagation




Want the gradients: $\dfrac{\partial \mathscr{L}}{\partial \mathbf{W}^{(l)}}, \dfrac{\partial \mathscr{L}}{\partial \mathbf{b}^{(l)}}$

Use chain rule:

$$\mathbf{z}^{(l)} = \mathbf{W}^{(l)} \cdot \mathbf{h}^{(l-1)} + \mathbf{b}^{(l)}$$

$$\mathbf{h}^{(l)} = g\left(\mathbf{z}^{(l)}\right)$$

Error signal $\quad \delta^{(l)} = \dfrac{\partial \mathscr{L}}{\partial \mathbf{z}^{(l)}}$

$$\frac{\partial \mathscr{L}}{\partial \mathbf{W}^{(l)}} = \delta^{(l)} \cdot \left(\mathbf{h}^{(l-1)}\right)$$

$$\frac{\partial \mathscr{L}}{\partial \mathbf{b}^{(l)}} = \delta^{(l)}$$

$$\delta^{(L)} = \nabla_{\hat{\mathbf{y}}} \mathscr{L} \odot g'(\mathbf{z}^{(L)})$$

$$\boxed{\delta^{(l)} = \left(\mathbf{W}^{(l+1)}\right)^{\top} \delta^{(l+1)} \odot g'(\mathbf{z}^{(L)})}$$

Careful with *vanishing* and *exploding* gradients!

# Learning from images

Session 1 — Topological and Geometric Deep Learning: Theory, Methods and Applications

Inés García Redondo — May 11, 2026

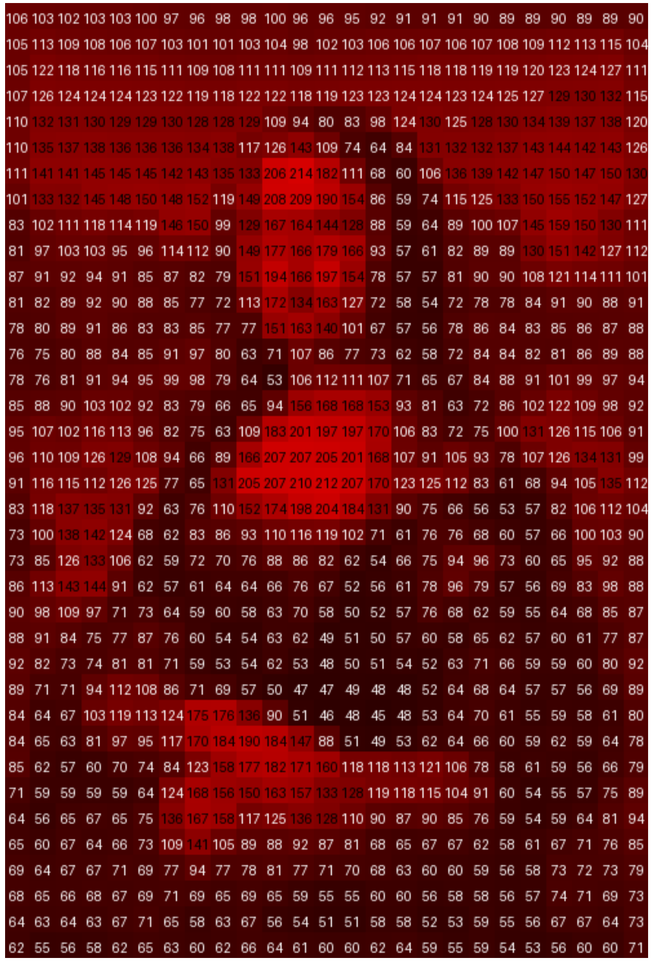
# Learning from images

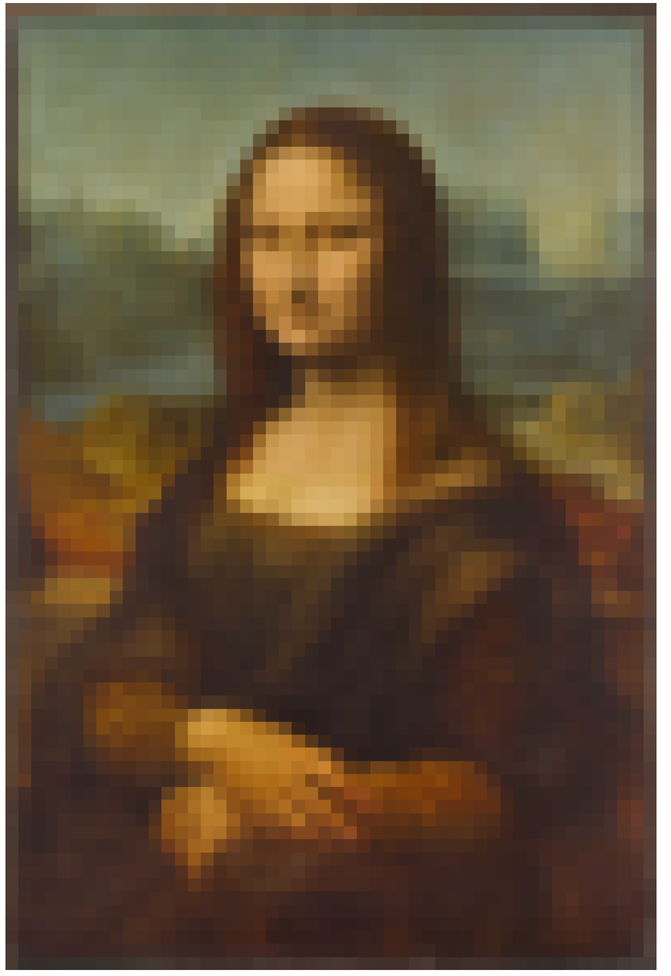
What the computer sees

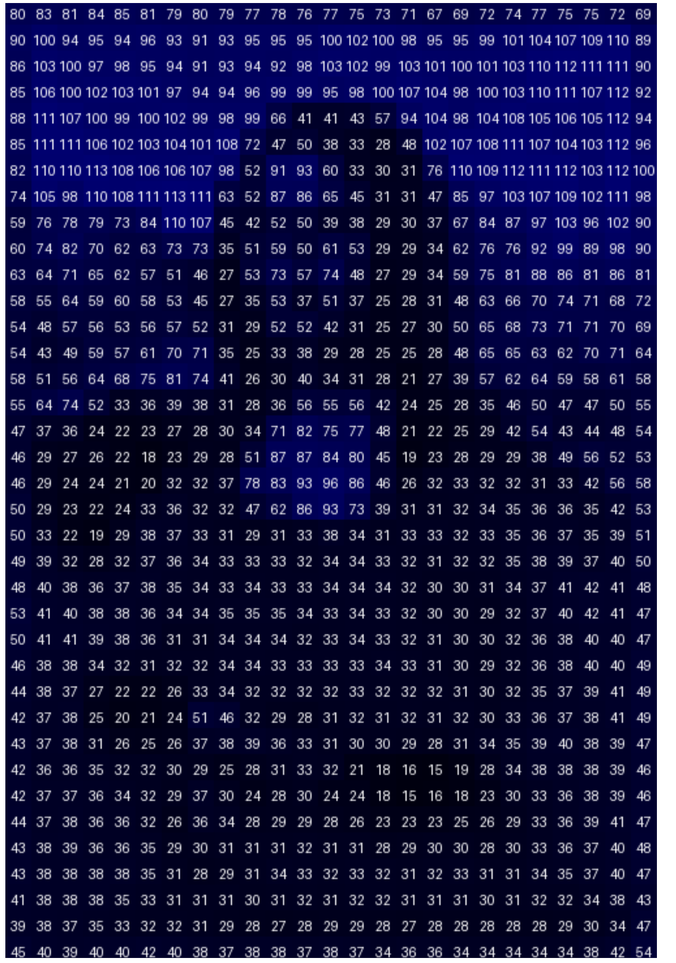
| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 96 | 98 | 97 | 98 | 101 | 102 | 98 | 95 | 94 | 96 | 95 | 95 | 96 | 94 | 90 | 94 | 89 | 86 | 85 | 83 | 83 | 83 | 83 | 84 | 83 | 83 | 80 | 79 | 78 |
| 100 | 116 | 112 | 112 | 111 | 110 | 112 | 109 | 105 | 106 | 106 | 106 | 108 | 104 | 106 | 108 | 108 | 110 | 109 | 108 | 107 | 108 | 109 | 110 | 111 | 114 | 114 | 115 | 116 | 96 |
| 100 | 125 | 121 | 119 | 116 | 117 | 116 | 114 | 110 | 110 | 111 | 114 | 113 | 110 | 112 | 114 | 114 | 115 | 118 | 118 | 116 | 117 | 119 | 119 | 120 | 125 | 125 | 128 | 127 | 100 |
| 96 | 128 | 125 | 124 | 124 | 125 | 122 | 119 | 117 | 115 | 117 | 118 | 116 | 119 | 122 | 124 | 120 | 118 | 121 | 124 | 121 | 121 | 123 | 123 | 127 | 129 | 128 | 129 | 131 | 103 |
| 100 | 133 | 128 | 129 | 129 | 127 | 127 | 126 | 125 | 121 | 122 | 126 | 127 | 116 | 106 | 108 | 116 | 123 | 128 | 128 | 123 | 123 | 124 | 126 | 129 | 131 | 132 | 131 | 136 | 105 |
| 100 | 136 | 134 | 132 | 132 | 130 | 129 | 132 | 131 | 128 | 130 | 128 | 93 | 69 | 57 | 55 | 60 | 77 | 117 | 133 | 126 | 125 | 129 | 132 | 132 | 136 | 136 | 135 | 139 | 109 |
| 100 | 135 | 139 | 138 | 137 | 135 | 135 | 136 | 132 | 135 | 138 | 99 | 87 | 107 | 89 | 63 | 51 | 47 | 64 | 125 | 134 | 130 | 133 | 137 | 139 | 139 | 137 | 137 | 141 | 113 |
| 99 | 138 | 140 | 141 | 142 | 140 | 140 | 140 | 138 | 143 | 124 | 101 | 160 | 173 | 156 | 108 | 63 | 48 | 46 | 86 | 138 | 137 | 139 | 141 | 142 | 144 | 139 | 139 | 146 | 116 |
| 93 | 141 | 142 | 140 | 151 | 150 | 149 | 143 | 143 | 145 | 94 | 116 | 167 | 171 | 160 | 130 | 90 | 55 | 46 | 56 | 116 | 127 | 134 | 138 | 143 | 148 | 146 | 142 | 149 | 119 |
| 81 | 115 | 116 | 120 | 133 | 129 | 137 | 147 | 153 | 138 | 71 | 122 | 153 | 158 | 137 | 130 | 105 | 58 | 45 | 47 | 86 | 105 | 117 | 122 | 142 | 148 | 149 | 136 | 136 | 106 |
| 70 | 91 | 103 | 109 | 106 | 99 | 104 | 129 | 144 | 118 | 62 | 92 | 105 | 119 | 98 | 87 | 91 | 60 | 44 | 44 | 72 | 89 | 98 | 100 | 134 | 150 | 147 | 131 | 128 | 101 |
| 73 | 92 | 101 | 103 | 96 | 89 | 89 | 98 | 99 | 90 | 61 | 131 | 145 | 132 | 141 | 148 | 121 | 58 | 43 | 44 | 69 | 88 | 92 | 90 | 120 | 142 | 137 | 127 | 124 | 104 |
| 78 | 87 | 88 | 91 | 92 | 89 | 82 | 80 | 77 | 65 | 55 | 124 | 153 | 127 | 142 | 147 | 95 | 51 | 44 | 42 | 64 | 86 | 93 | 96 | 109 | 119 | 112 | 108 | 108 | 93 |
| 74 | 75 | 82 | 88 | 86 | 87 | 84 | 82 | 74 | 60 | 49 | 90 | 133 | 93 | 113 | 120 | 79 | 49 | 44 | 41 | 54 | 73 | 76 | 78 | 84 | 89 | 93 | 85 | 86 | 86 |
| 70 | 71 | 82 | 86 | 85 | 81 | 79 | 78 | 77 | 67 | 51 | 63 | 124 | 118 | 109 | 105 | 70 | 47 | 43 | 42 | 56 | 78 | 81 | 81 | 84 | 87 | 85 | 83 | 85 | 82 |
| 69 | 67 | 73 | 81 | 81 | 76 | 77 | 81 | 87 | 77 | 53 | 44 | 74 | 120 | 92 | 68 | 55 | 47 | 44 | 42 | 55 | 80 | 87 | 83 | 83 | 82 | 83 | 87 | 84 | 78 |
| 69 | 65 | 69 | 76 | 84 | 84 | 88 | 95 | 101 | 87 | 53 | 43 | 39 | 58 | 63 | 53 | 60 | 58 | 45 | 44 | 51 | 74 | 82 | 80 | 77 | 75 | 79 | 84 | 87 | 78 |
| 72 | 70 | 77 | 83 | 90 | 90 | 91 | 96 | 96 | 84 | 54 | 44 | 41 | 70 | 104 | 87 | 94 | 82 | 51 | 49 | 49 | 65 | 80 | 85 | 88 | 96 | 96 | 88 | 87 | 79 |
| 73 | 79 | 89 | 91 | 89 | 79 | 75 | 66 | 66 | 58 | 46 | 45 | 68 | 115 | 132 | 128 | 132 | 114 | 64 | 60 | 50 | 49 | 64 | 77 | 84 | 104 | 92 | 88 | 79 | 76 |
| 74 | 78 | 78 | 81 | 84 | 84 | 73 | 62 | 60 | 51 | 47 | 76 | 139 | 159 | 155 | 154 | 153 | 124 | 70 | 63 | 51 | 51 | 60 | 90 | 113 | 105 | 94 | 91 | 83 | 74 |
| 75 | 78 | 79 | 84 | 98 | 93 | 76 | 72 | 60 | 47 | 61 | 118 | 164 | 167 | 161 | 164 | 155 | 121 | 69 | 58 | 56 | 68 | 57 | 65 | 93 | 108 | 108 | 109 | 95 | 75 |
| 71 | 78 | 78 | 76 | 84 | 96 | 88 | 70 | 49 | 50 | 106 | 157 | 168 | 168 | 168 | 167 | 158 | 118 | 75 | 83 | 96 | 89 | 64 | 48 | 56 | 76 | 87 | 105 | 111 | 85 |
| 67 | 78 | 82 | 83 | 82 | 91 | 91 | 57 | 49 | 57 | 118 | 166 | 164 | 174 | 180 | 178 | 163 | 117 | 86 | 79 | 65 | 53 | 47 | 43 | 45 | 55 | 63 | 80 | 105 | 87 |
| 65 | 72 | 89 | 93 | 92 | 89 | 60 | 48 | 53 | 67 | 78 | 93 | 113 | 140 | 152 | 151 | 117 | 77 | 55 | 51 | 51 | 49 | 43 | 43 | 44 | 49 | 68 | 70 | 72 | 74 |
| 61 | 59 | 79 | 89 | 92 | 84 | 52 | 48 | 55 | 66 | 62 | 68 | 78 | 78 | 77 | 80 | 65 | 48 | 45 | 56 | 60 | 56 | 52 | 48 | 44 | 45 | 59 | 66 | 67 | 69 |
| 61 | 56 | 73 | 83 | 83 | 68 | 49 | 46 | 53 | 56 | 54 | 60 | 69 | 65 | 64 | 59 | 45 | 42 | 50 | 54 | 62 | 74 | 67 | 54 | 47 | 48 | 57 | 71 | 61 | 69 |
| 65 | 72 | 100 | 107 | 103 | 66 | 49 | 45 | 46 | 49 | 49 | 51 | 52 | 57 | 59 | 47 | 41 | 44 | 47 | 51 | 65 | 71 | 56 | 45 | 44 | 52 | 53 | 68 | 62 | 67 |
| 75 | 78 | 90 | 94 | 82 | 54 | 51 | 48 | 45 | 48 | 48 | 48 | 49 | 55 | 51 | 43 | 41 | 43 | 43 | 55 | 60 | 52 | 43 | 43 | 44 | 52 | 52 | 67 | 71 | 70 |
| 72 | 69 | 73 | 68 | 56 | 54 | 61 | 55 | 46 | 44 | 45 | 45 | 49 | 52 | 47 | 40 | 41 | 40 | 42 | 51 | 45 | 42 | 45 | 44 | 43 | 48 | 47 | 51 | 56 | 65 |
| 70 | 68 | 63 | 56 | 57 | 60 | 64 | 59 | 48 | 42 | 42 | 43 | 49 | 48 | 40 | 40 | 41 | 40 | 44 | 43 | 40 | 46 | 46 | 45 | 44 | 47 | 47 | 51 | 60 | 69 |
| 73 | 63 | 56 | 53 | 54 | 57 | 58 | 52 | 47 | 41 | 42 | 43 | 48 | 43 | 40 | 39 | 40 | 41 | 42 | 40 | 42 | 47 | 50 | 47 | 45 | 47 | 46 | 51 | 62 | 73 |
| 71 | 56 | 52 | 55 | 69 | 74 | 73 | 62 | 50 | 49 | 47 | 42 | 40 | 39 | 38 | 38 | 40 | 38 | 39 | 39 | 43 | 47 | 46 | 45 | 44 | 45 | 45 | 47 | 56 | 71 |
| 67 | 51 | 49 | 56 | 77 | 81 | 78 | 79 | 106 | 135 | 114 | 79 | 52 | 36 | 35 | 39 | 38 | 36 | 39 | 40 | 43 | 47 | 47 | 44 | 43 | 45 | 45 | 46 | 50 | 65 |
| 67 | 50 | 49 | 53 | 67 | 70 | 66 | 73 | 124 | 142 | 141 | 135 | 124 | 86 | 52 | 39 | 37 | 36 | 37 | 38 | 41 | 46 | 47 | 44 | 46 | 48 | 46 | 46 | 51 | 64 |
| 68 | 50 | 48 | 46 | 54 | 57 | 60 | 61 | 84 | 103 | 123 | 134 | 134 | 131 | 122 | 72 | 49 | 54 | 52 | 62 | 58 | 53 | 46 | 45 | 49 | 49 | 45 | 46 | 51 | 61 |
| 65 | 48 | 45 | 43 | 45 | 46 | 48 | 56 | 81 | 103 | 105 | 116 | 119 | 114 | 98 | 97 | 86 | 83 | 76 | 76 | 74 | 65 | 52 | 42 | 45 | 45 | 43 | 46 | 54 | 62 |
| 55 | 44 | 46 | 46 | 47 | 44 | 43 | 82 | 112 | 129 | 98 | 96 | 109 | 106 | 99 | 75 | 76 | 69 | 69 | 71 | 63 | 60 | 55 | 42 | 40 | 43 | 44 | 48 | 60 | 67 |
| 55 | 44 | 48 | 50 | 50 | 49 | 50 | 84 | 105 | 125 | 99 | 75 | 80 | 93 | 81 | 83 | 64 | 56 | 55 | 58 | 57 | 55 | 49 | 41 | 41 | 44 | 47 | 51 | 62 | 68 |
| 54 | 45 | 49 | 49 | 47 | 49 | 52 | 67 | 97 | 94 | 70 | 60 | 59 | 62 | 61 | 58 | 52 | 46 | 45 | 47 | 48 | 46 | 43 | 42 | 44 | 48 | 50 | 52 | 58 | 65 |
| 56 | 49 | 51 | 51 | 51 | 52 | 51 | 51 | 71 | 60 | 56 | 56 | 59 | 60 | 52 | 50 | 49 | 46 | 44 | 43 | 44 | 43 | 42 | 41 | 43 | 50 | 52 | 52 | 55 | 63 |
| 55 | 47 | 49 | 50 | 51 | 51 | 49 | 50 | 53 | 50 | 48 | 52 | 52 | 48 | 44 | 43 | 44 | 45 | 44 | 41 | 43 | 42 | 42 | 41 | 41 | 51 | 52 | 50 | 53 | 57 |
| 53 | 49 | 50 | 51 | 49 | 48 | 50 | 50 | 46 | 45 | 45 | 46 | 42 | 40 | 39 | 37 | 39 | 42 | 41 | 39 | 41 | 40 | 42 | 40 | 40 | 48 | 50 | 47 | 49 | 56 |
| 50 | 46 | 47 | 46 | 45 | 50 | 50 | 46 | 42 | 40 | 43 | 45 | 38 | 37 | 37 | 36 | 38 | 41 | 41 | 38 | 37 | 38 | 42 | 38 | 39 | 44 | 46 | 44 | 46 | 61 |
| 55 | 46 | 46 | 47 | 49 | 51 | 54 | 52 | 50 | 47 | 50 | 52 | 51 | 48 | 49 | 49 | 47 | 47 | 49 | 50 | 43 | 45 | 46 | 43 | 42 | 43 | 47 | 49 | 52 | 64 |

# Learning from images

What the computer sees

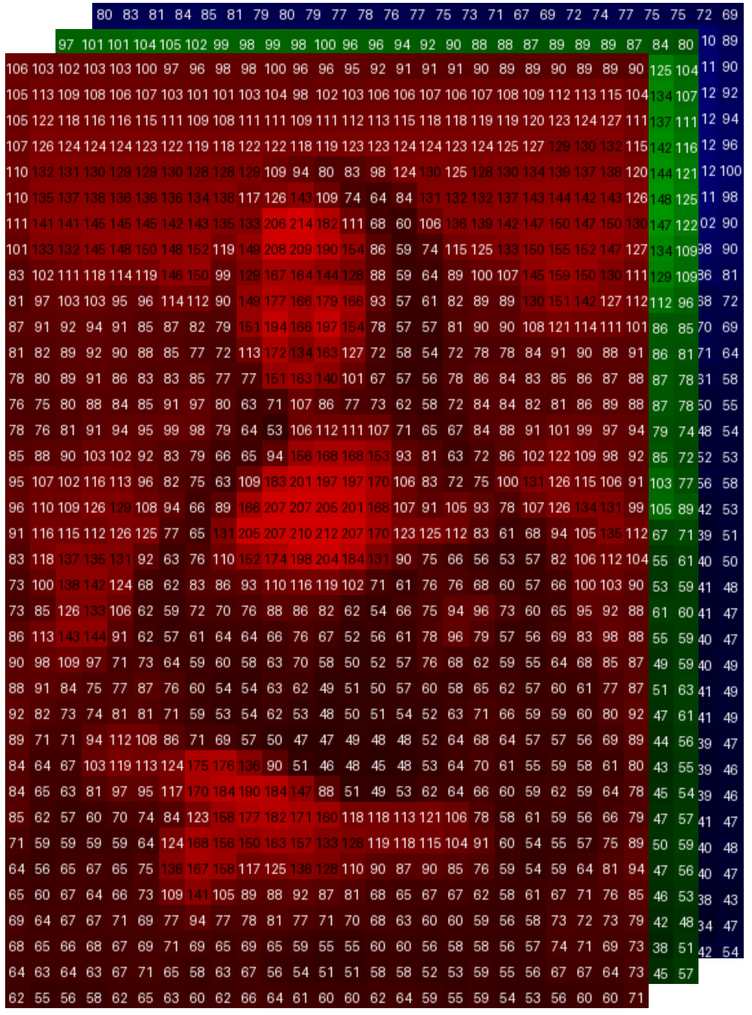






# Learning from images

What the computer sees



# Learning from images

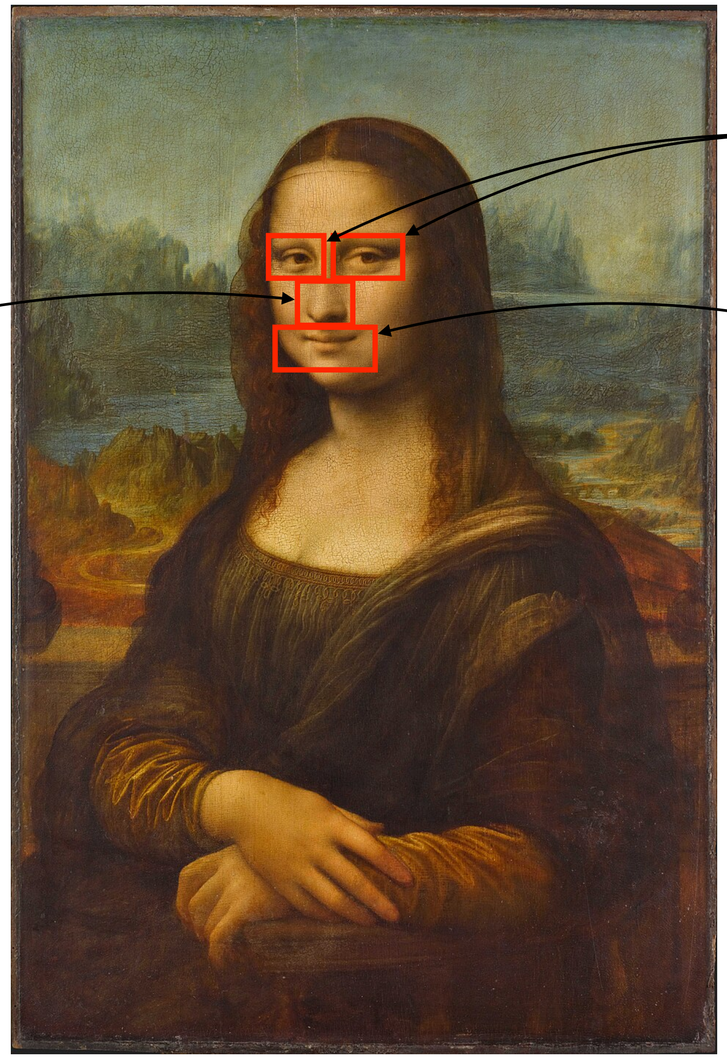
Challenges with using MLPs

- If I want to produce an image from an image with resolution 512 using a MLP I need $(512 \times 512 \times 3)^2 \approx 6.18 \cdot 10^{11}$ parameters
- With 4 bytes per parameter, this requires about *2.5 terabytes* in storage.
- This clearly does *not* scale.

- In addition, images have a lot of *structure* that we are not using.
- Images often vary *continuously*.
- Classification of images should *not* depend on a single pixel *but* in groups of pixels.

# Learning from images

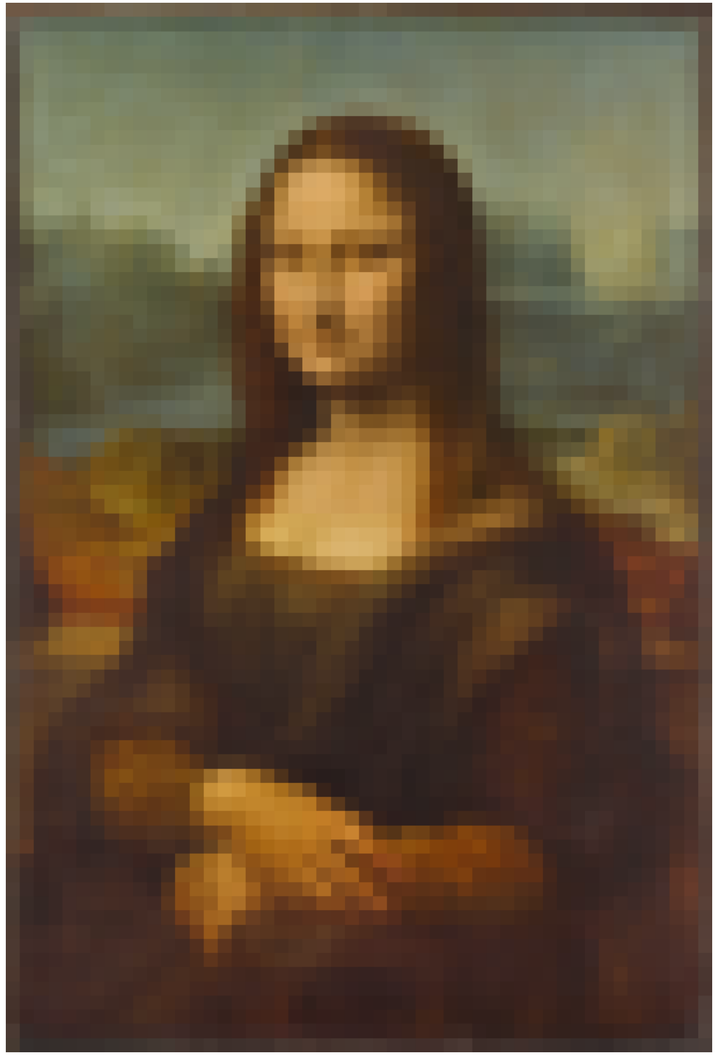
Recognition is based in detecting lower level features




This must be a face!

# Learning from images

We can still recognize with lower resolution



This looks like the Mona Lisa!

# Convolutions

| | | | | | | |
|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 | 1 | 0 |
| 0 | 0 | 0 | 1 | 1 | 1 | 0 |
| 0 | 0 | 0 | 1 | 1 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 | 0 | 0 |

Input image $I$

| | | |
|---|---|---|
| 1 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 1 |

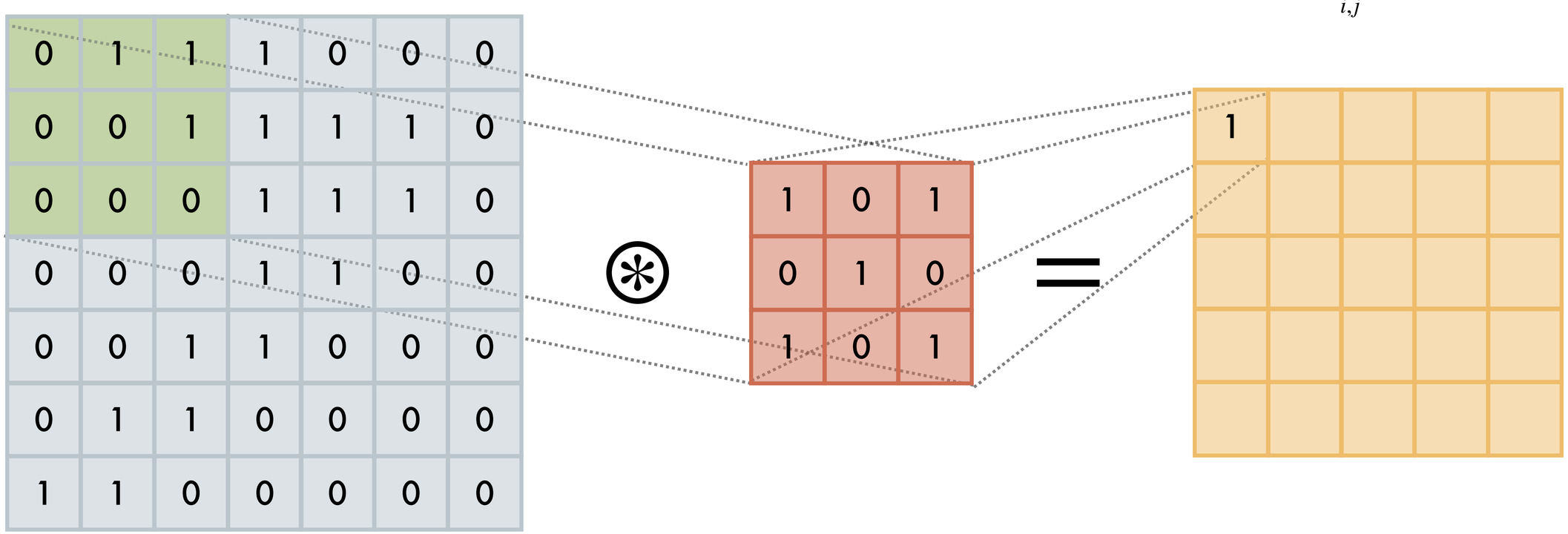
Filter or kernel $K$

# Convolutions

Need to also choose a *pooling* operation

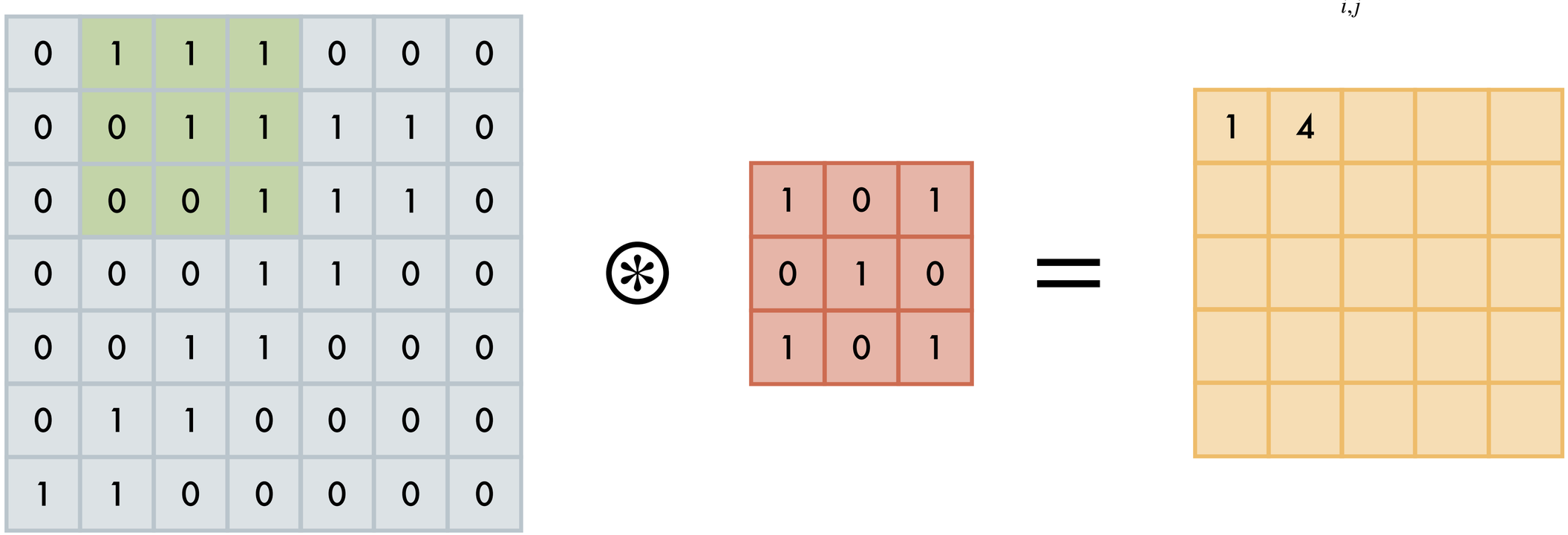
Here we sum

$$(I \circledast K)(x, y) = \sum_{i,j} K(i, j)\ I(x + i, y + j)$$



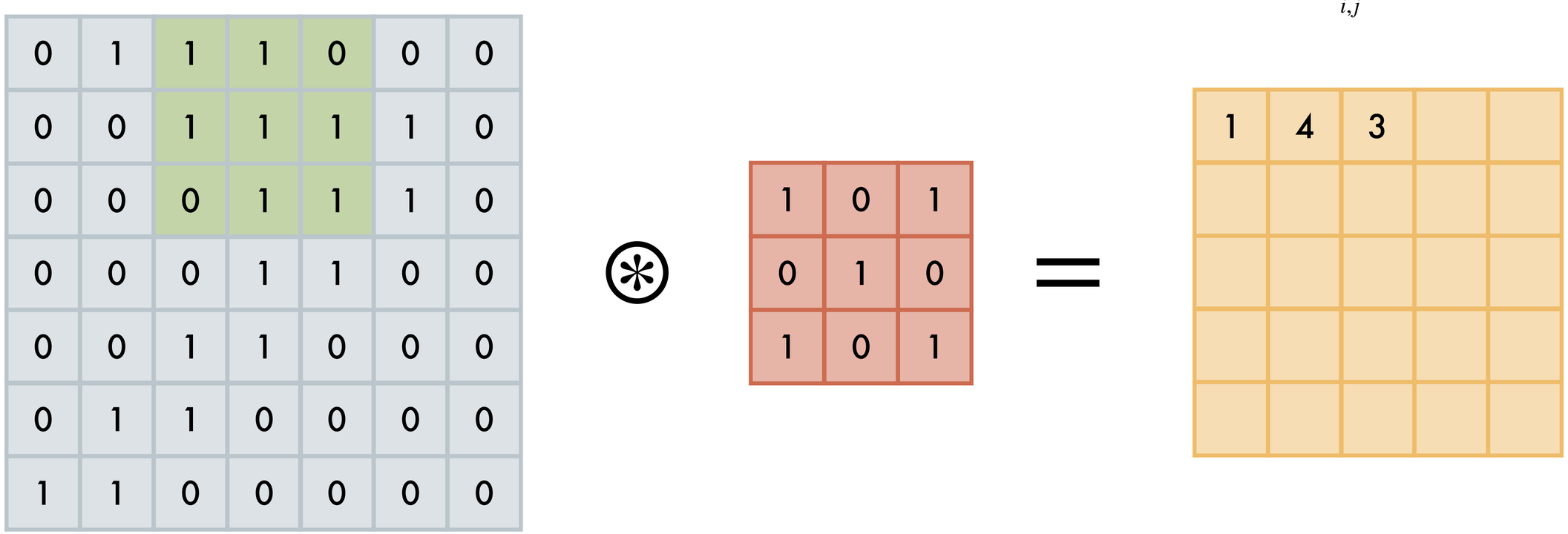
# Convolutions

$$(I \circledast K)(x, y) = \sum_{i,j} K(i, j)\ I(x + i, y + j)$$



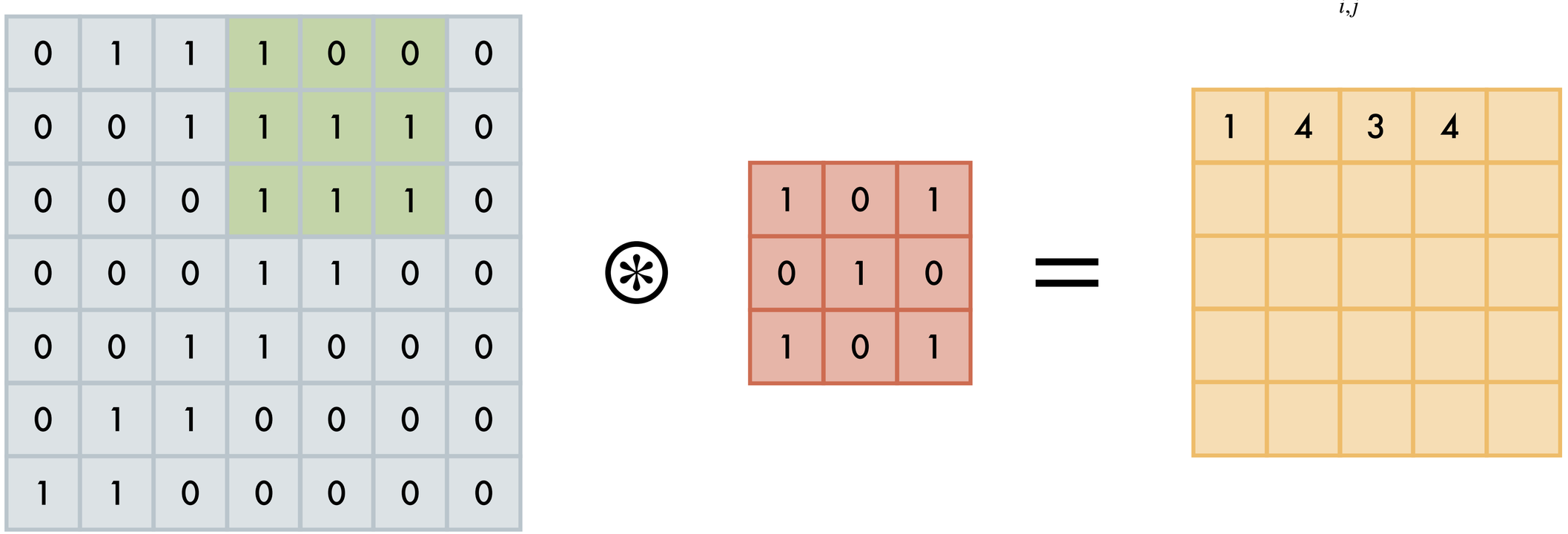
# Convolutions

$$(I \circledast K)(x, y) = \sum_{i,j} K(i, j)\ I(x + i, y + j)$$



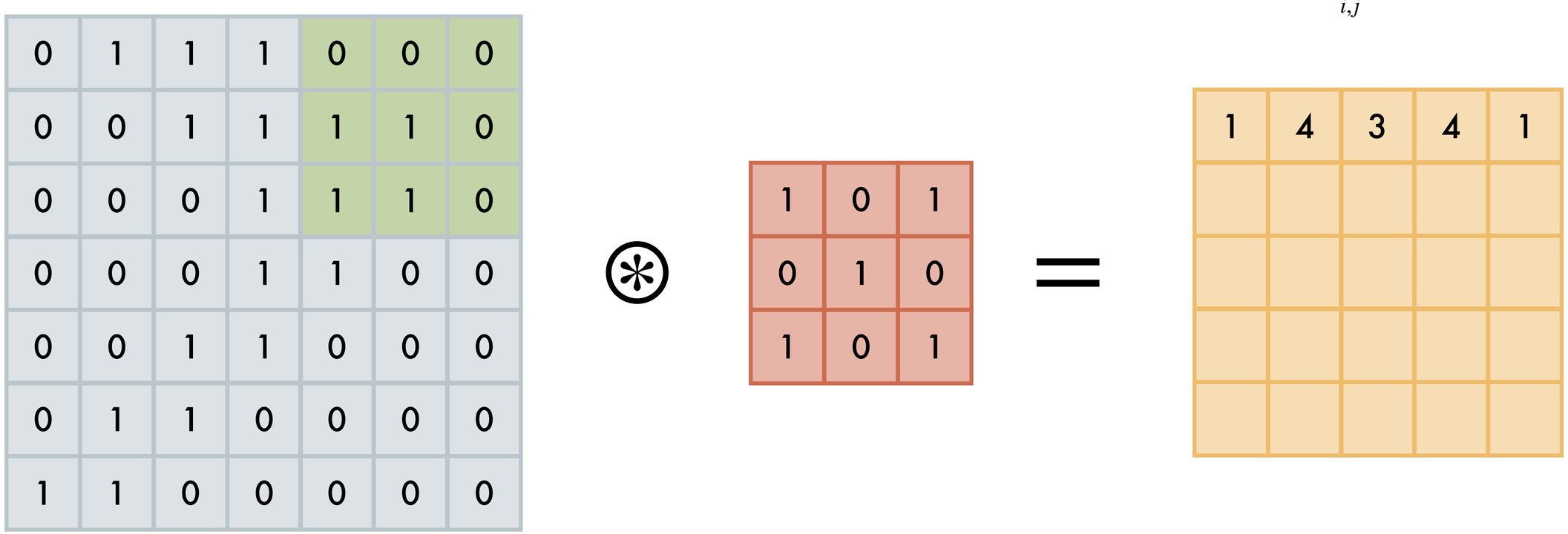
# Convolutions

$$(I \circledast K)(x, y) = \sum_{i,j} K(i, j)\ I(x + i, y + j)$$



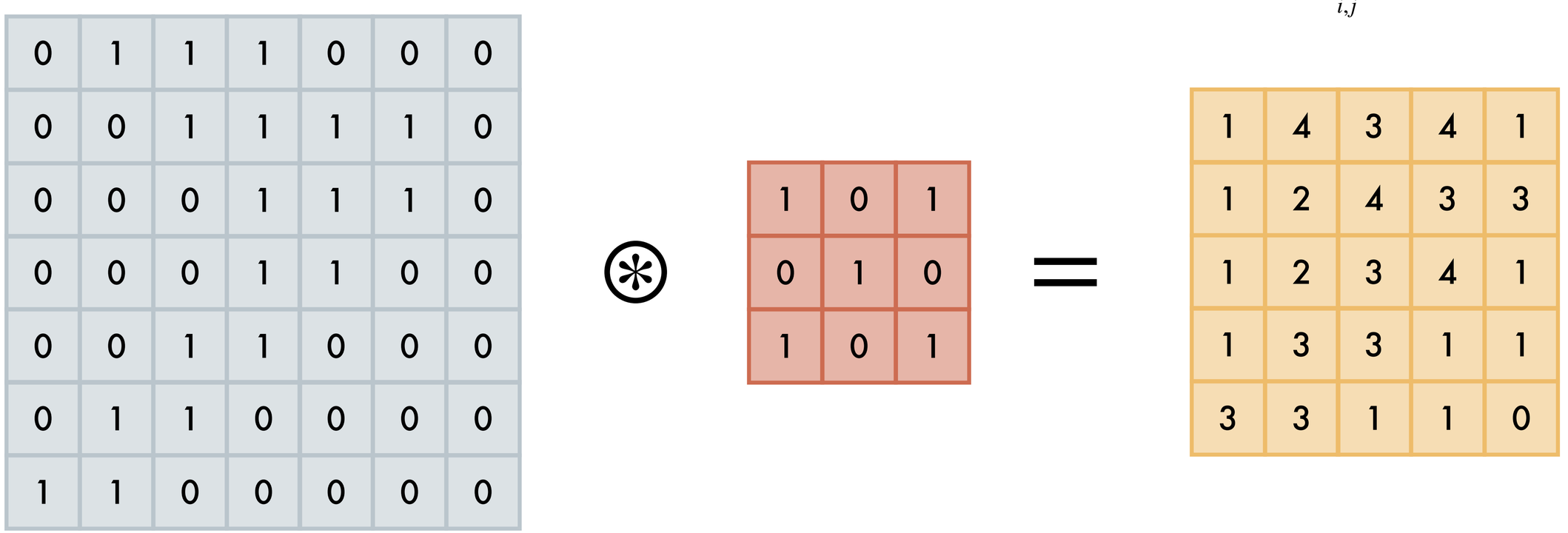
# Convolutions

$$(I \circledast K)(x, y) = \sum_{i,j} K(i, j)\ I(x + i, y + j)$$



# Convolutions

$$(I \circledast K)(x, y) = \sum_{i,j} K(i, j)\ I(x + i, y + j)$$



# Filter in computer vision

Gaussian blur

## Gaussian blur

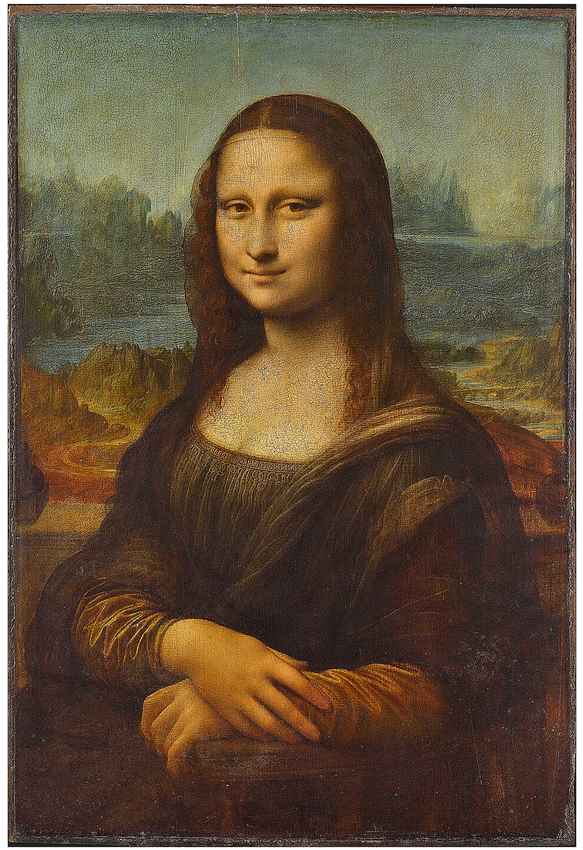
$$K(i,j) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{i^2+j^2}{2\sigma^2}\right)$$

$$\sigma \approx 1.7$$

$$K = \begin{bmatrix}
0.000 & 0.000 & 0.001 & 0.002 & 0.003 & 0.004 & 0.003 & 0.002 & 0.001 & 0.000 & 0.000 \\
0.000 & 0.001 & 0.003 & 0.006 & 0.010 & 0.012 & 0.010 & 0.006 & 0.003 & 0.001 & 0.000 \\
0.001 & 0.003 & 0.008 & 0.015 & 0.024 & 0.029 & 0.024 & 0.015 & 0.008 & 0.003 & 0.001 \\
0.002 & 0.006 & 0.015 & 0.028 & 0.045 & 0.054 & 0.045 & 0.028 & 0.015 & 0.006 & 0.002 \\
0.003 & 0.010 & 0.024 & 0.045 & 0.072 & 0.087 & 0.072 & 0.045 & 0.024 & 0.010 & 0.003 \\
0.004 & 0.012 & 0.029 & 0.054 & 0.087 & 0.106 & 0.087 & 0.054 & 0.029 & 0.012 & 0.004 \\
0.003 & 0.010 & 0.024 & 0.045 & 0.072 & 0.087 & 0.072 & 0.045 & 0.024 & 0.010 & 0.003 \\
0.002 & 0.006 & 0.015 & 0.028 & 0.045 & 0.054 & 0.045 & 0.028 & 0.015 & 0.006 & 0.002 \\
0.001 & 0.003 & 0.008 & 0.015 & 0.024 & 0.029 & 0.024 & 0.015 & 0.008 & 0.003 & 0.001 \\
0.000 & 0.001 & 0.003 & 0.006 & 0.010 & 0.012 & 0.010 & 0.006 & 0.003 & 0.001 & 0.000 \\
0.000 & 0.000 & 0.001 & 0.002 & 0.003 & 0.004 & 0.003 & 0.002 & 0.001 & 0.000 & 0.000
\end{bmatrix}$$

# Filter in computer vision

Sharpening filter



**Sharpening kernel**

$$K = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

# Filter in computer vision

Sobel filter

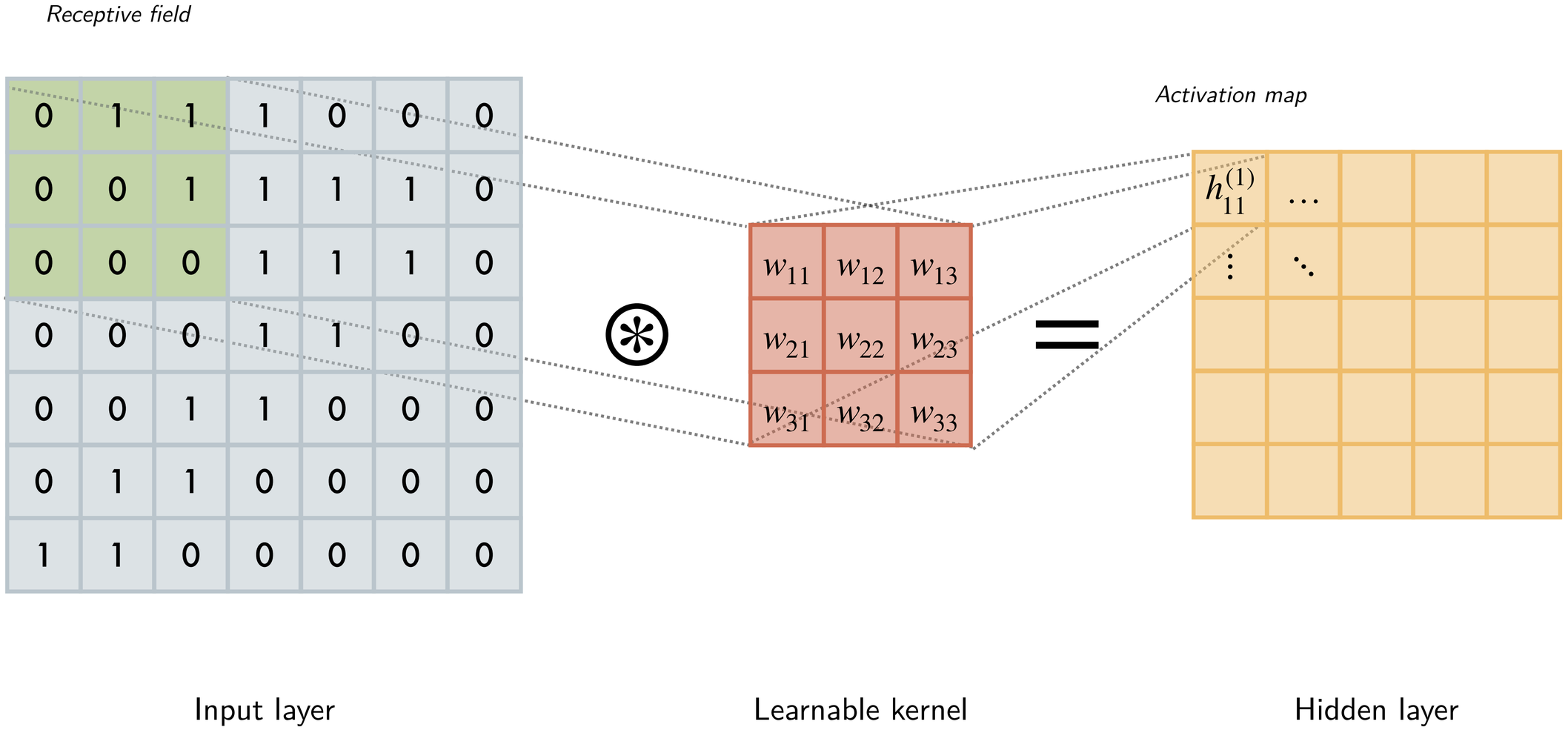
**Sobel filter (horizontal)**

$$K = \begin{bmatrix} -1 & -2 & 0 & 2 & 1 \\ -4 & -8 & 0 & 8 & 4 \\ -6 & -12 & 0 & 12 & 6 \\ -4 & -8 & 0 & 8 & 4 \\ -1 & -2 & 0 & 2 & 1 \end{bmatrix}$$

# Convolutional neural networks

Learnable kernels

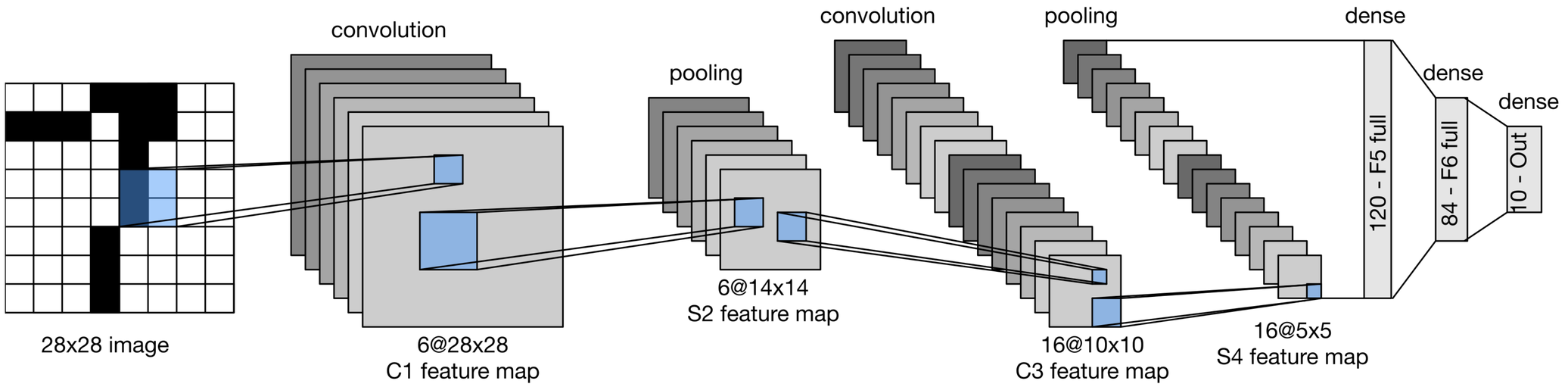
$$h_{11}^{(1)} = g\left(R_{11} \circledast W_{11} + b_{11}\right)$$



# Convolutional neural networks

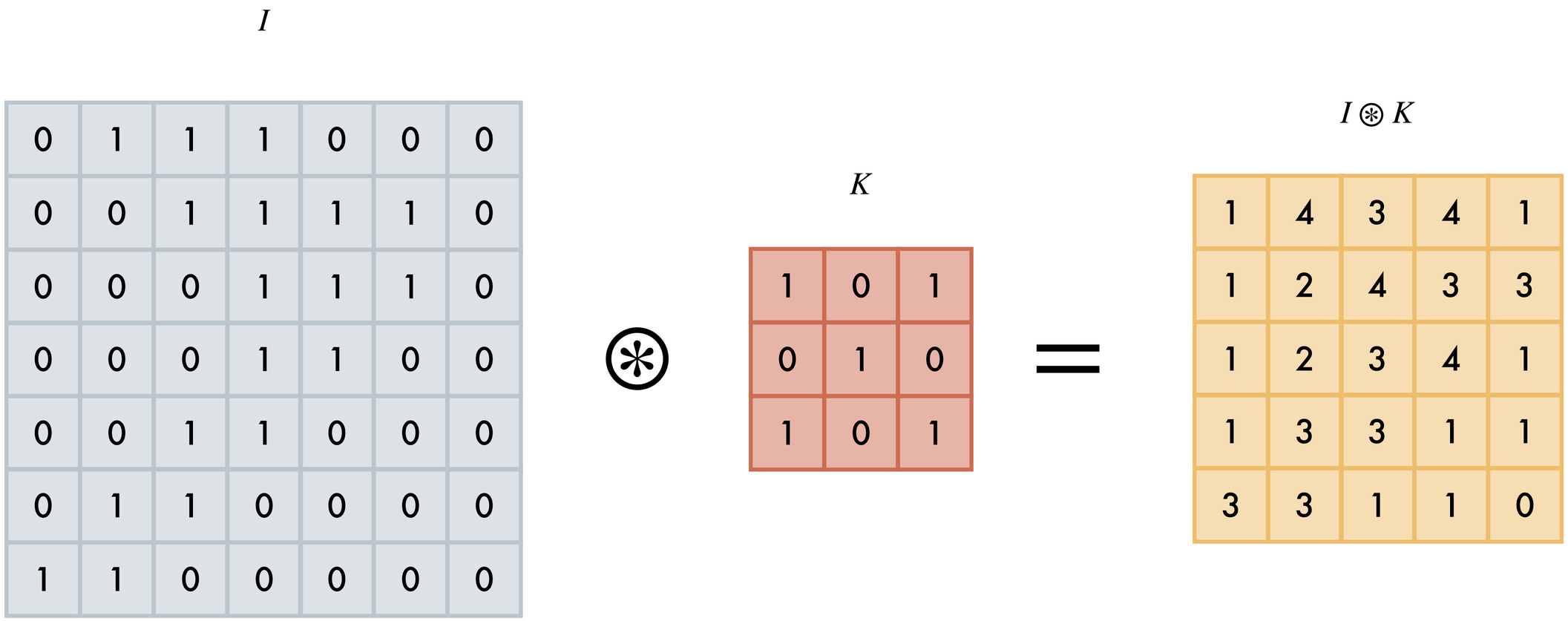
LeNet

https://en.wikipedia.org/wiki/LeNet




Y. LeCun, L. Bottou, Y. Bengio and P. Haffner, *Gradient-based learning applied to document recognition*, Proceedings of the IEEE 86.11, 1998, pp. 2278–2324

# Convolution is translation equivariant



# Convolution is translation equivariant

$T(I)$

| | | | | | | |
|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 | 1 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 0 | 0 | 0 | 1 |

$\circledast$

$K$

| | | |
|---|---|---|
| 1 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 1 |

$=$

$T(I) \circledast K = T(I \circledast K)$

| | | | | |
|---|---|---|---|---|
| 4 | 3 | 4 | 1 | 1 |
| 2 | 4 | 3 | 3 | 1 |
| 2 | 3 | 4 | 1 | 1 |
| 3 | 3 | 1 | 1 | 1 |
| 3 | 1 | 1 | 0 | 3 |

# Learning from text

Session 1 — Topological and Geometric Deep Learning: Theory, Methods and Applications

Inés García Redondo — May 11, 2026

# Learning from text

What the computer sees

Input

The cat ate the mouse

Tokenization: break down into semantic pieces

[[The], [cat], [ate], [the], [mouse]]

Size of vocabulary

Hidden dimension embeddings

Embedding: Map each token to a vocabulary dictionary and learn an embedding matrix $E \in \mathbb{R}^{V \times d}$

[1996, 4937, 8823, 1996, 8000]

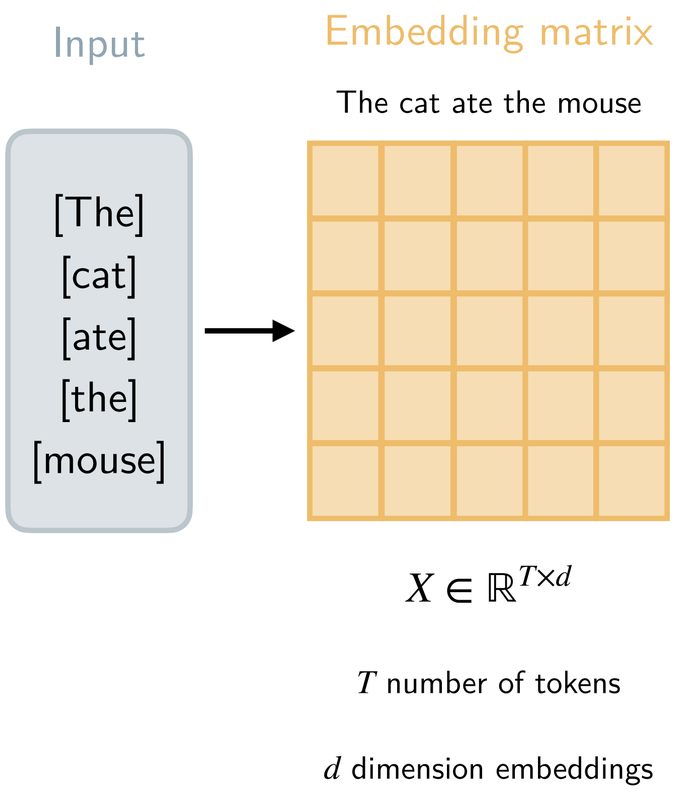
$$1996 \xrightarrow{E} v \in \mathbb{R}^d$$

Add positional encoding: the order of the words is important

Not the same to say: "The mouse ate the cat" than "The cat ate the mouse"

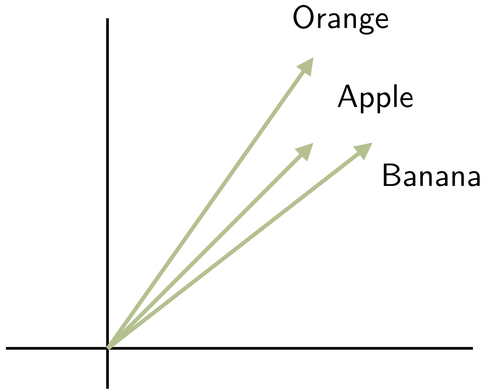
# Learning from text

Word embeddings



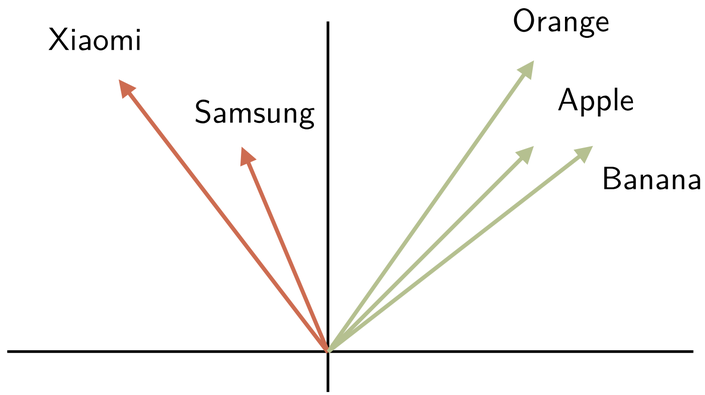

**How do these embeddings look like?**

They should capture **semantic** meaning




$$X_{\text{king}} - X_{\text{man}} + X_{\text{woman}} \approx X_{\text{queen}}$$

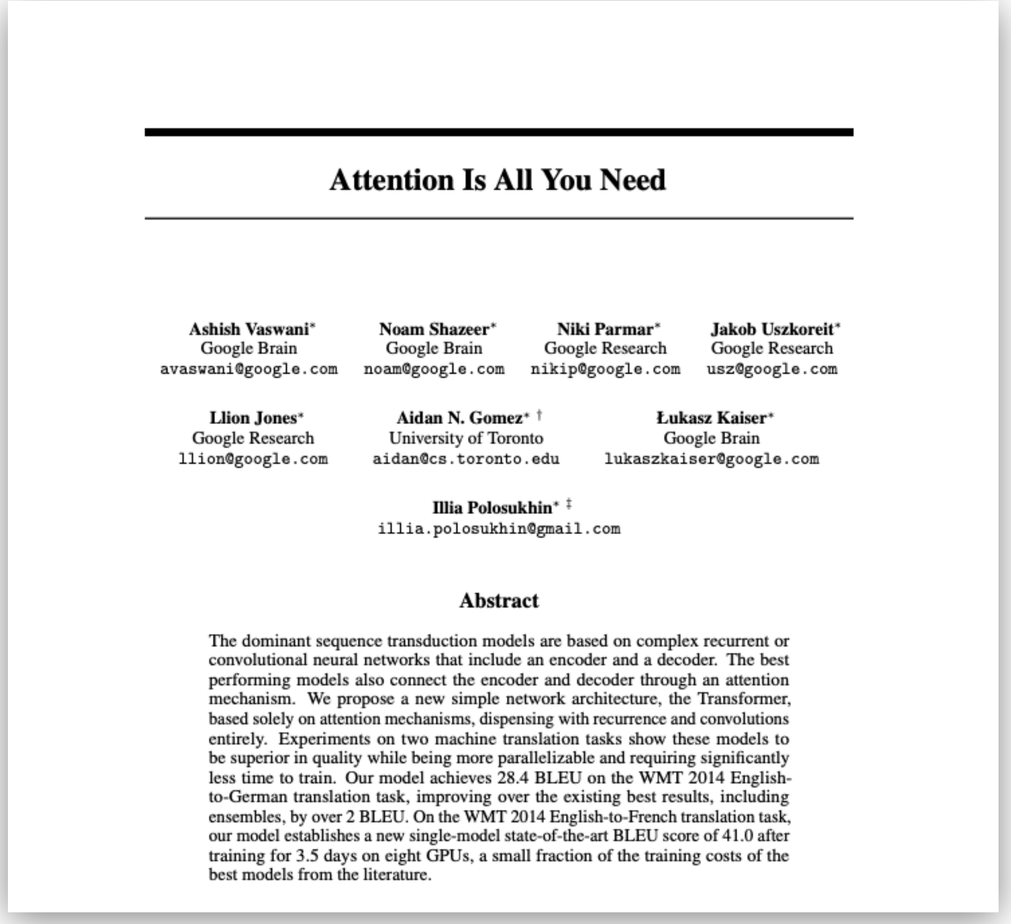
However, they still do not capture **contextual** meaning




*Apple has released a new phone*

# Learning from text

Transformers and the attention mechanism



**Attention Is All You Need**


**Ashish Vaswani*** Google Brain avaswani@google.com

**Noam Shazeer*** Google Brain noam@google.com

**Niki Parmar*** Google Research nikip@google.com

**Jakob Uszkoreit*** Google Research usz@google.com

**Llion Jones*** Google Research llion@google.com

**Aidan N. Gomez*** † University of Toronto aidan@cs.toronto.edu

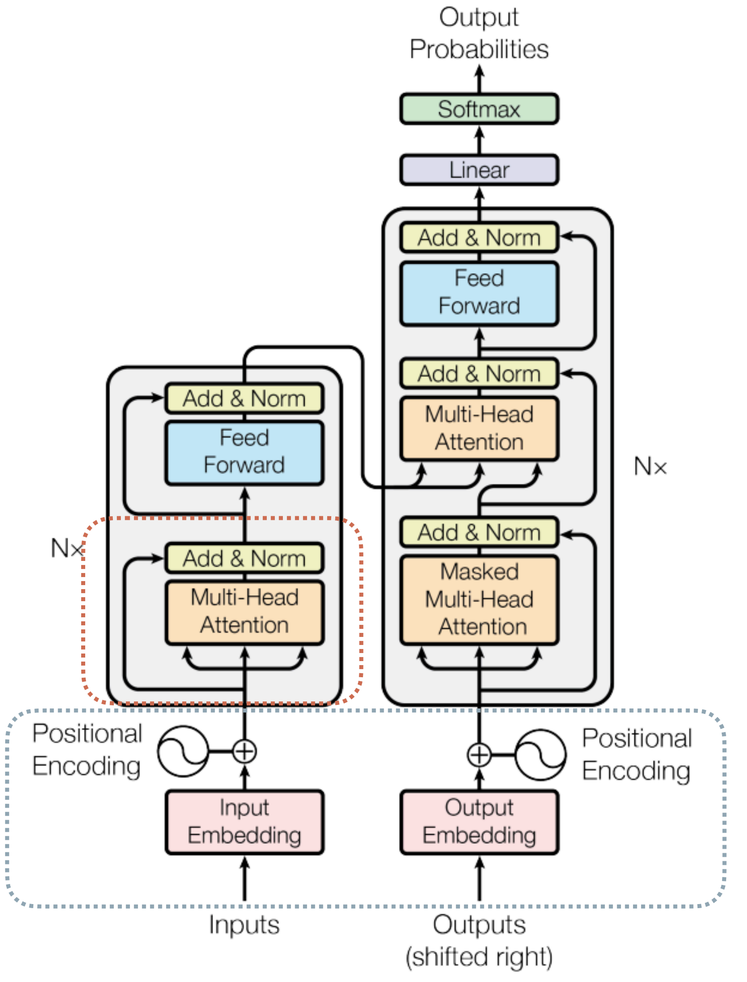
**Łukasz Kaiser*** Google Brain lukaszkaiser@google.com

**Illia Polosukhin*** ‡ illia.polosukhin@gmail.com



**Abstract**

The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions entirely. Experiments on two machine translation tasks show these models to be superior in quality while being more parallelizable and requiring significantly less time to train. Our model achieves 28.4 BLEU on the WMT 2014 English-to-German translation task, improving over the existing best results, including ensembles, by over 2 BLEU. On the WMT 2014 English-to-French translation task, our model establishes a new single-model state-of-the-art BLEU score of 41.0 after training for 3.5 days on eight GPUs, a small fraction of the training costs of the best models from the literature.

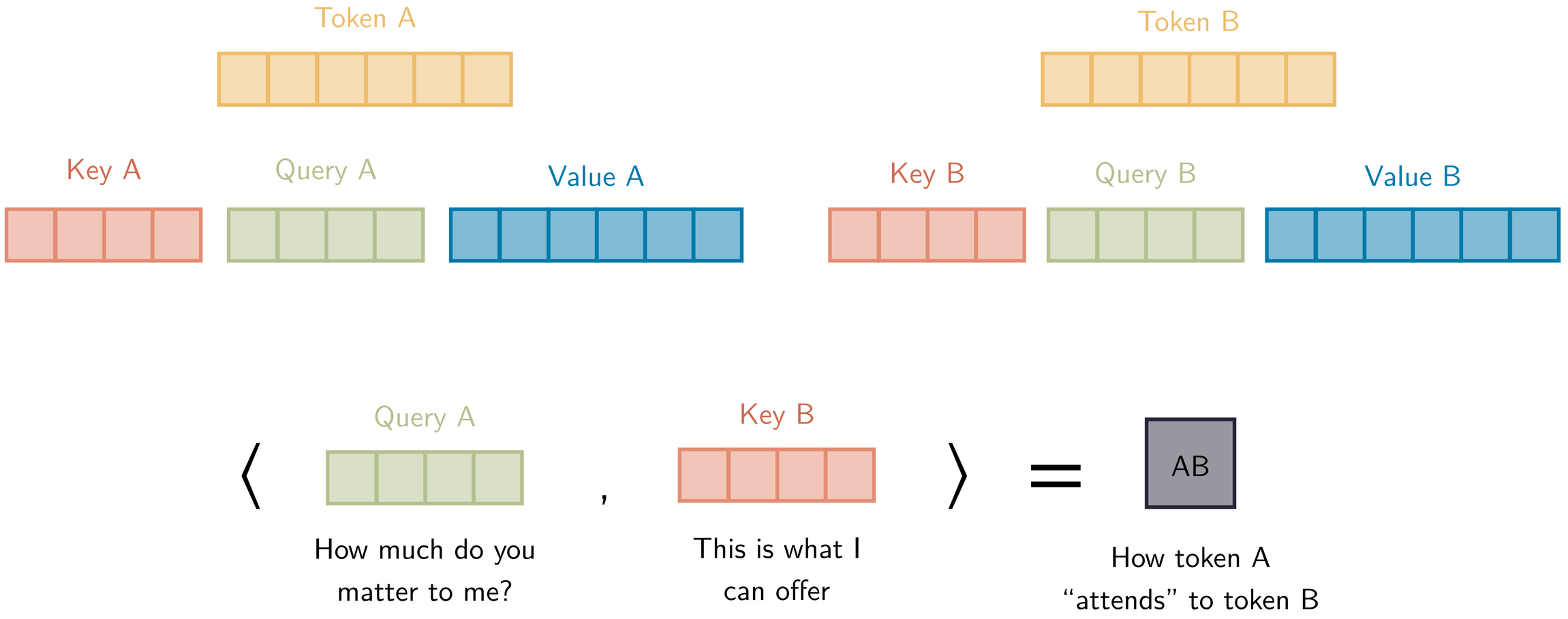




A. Vaswani, S. Noam, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser and I. Polosukhin

*Attention is All You Need*, NeurIPS 2017

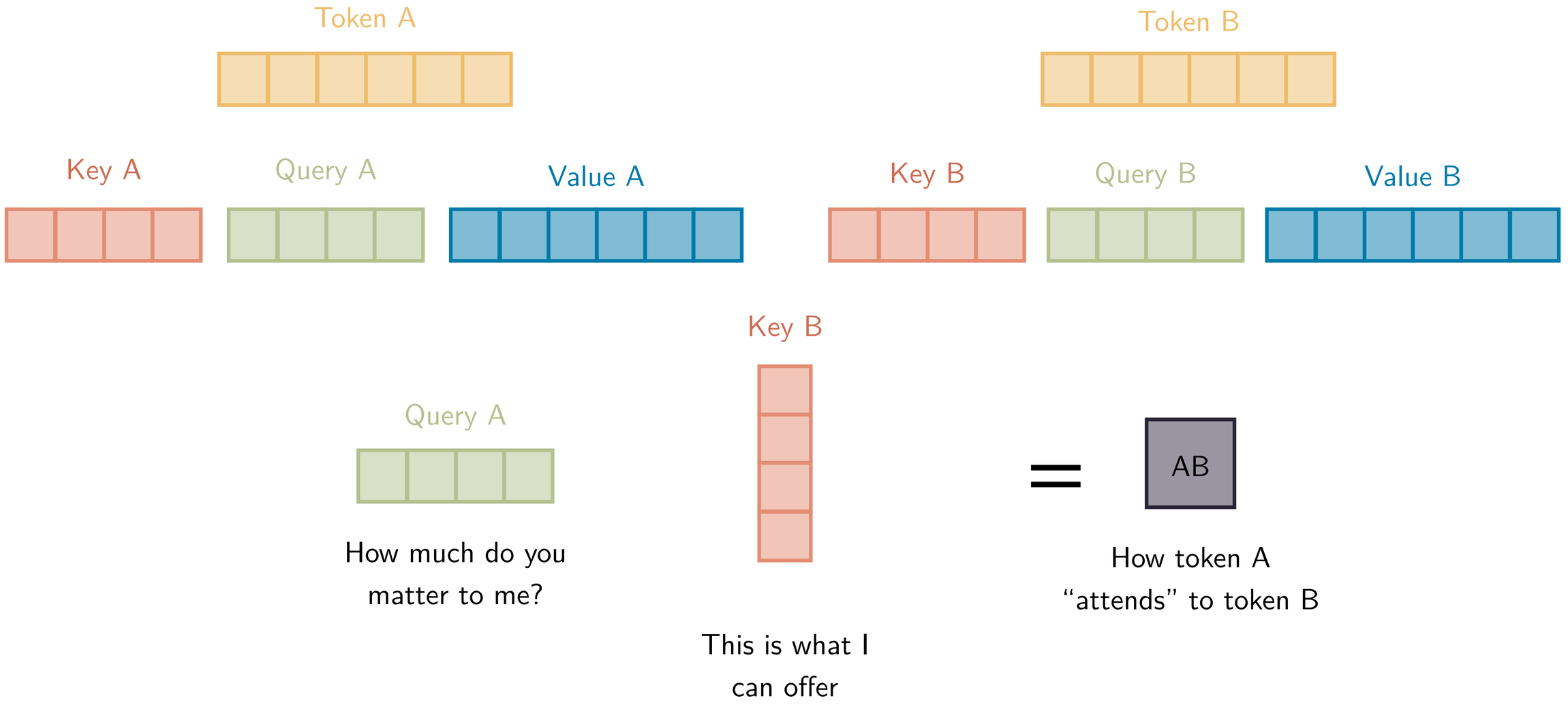
# Learning from text

The attention mechanism



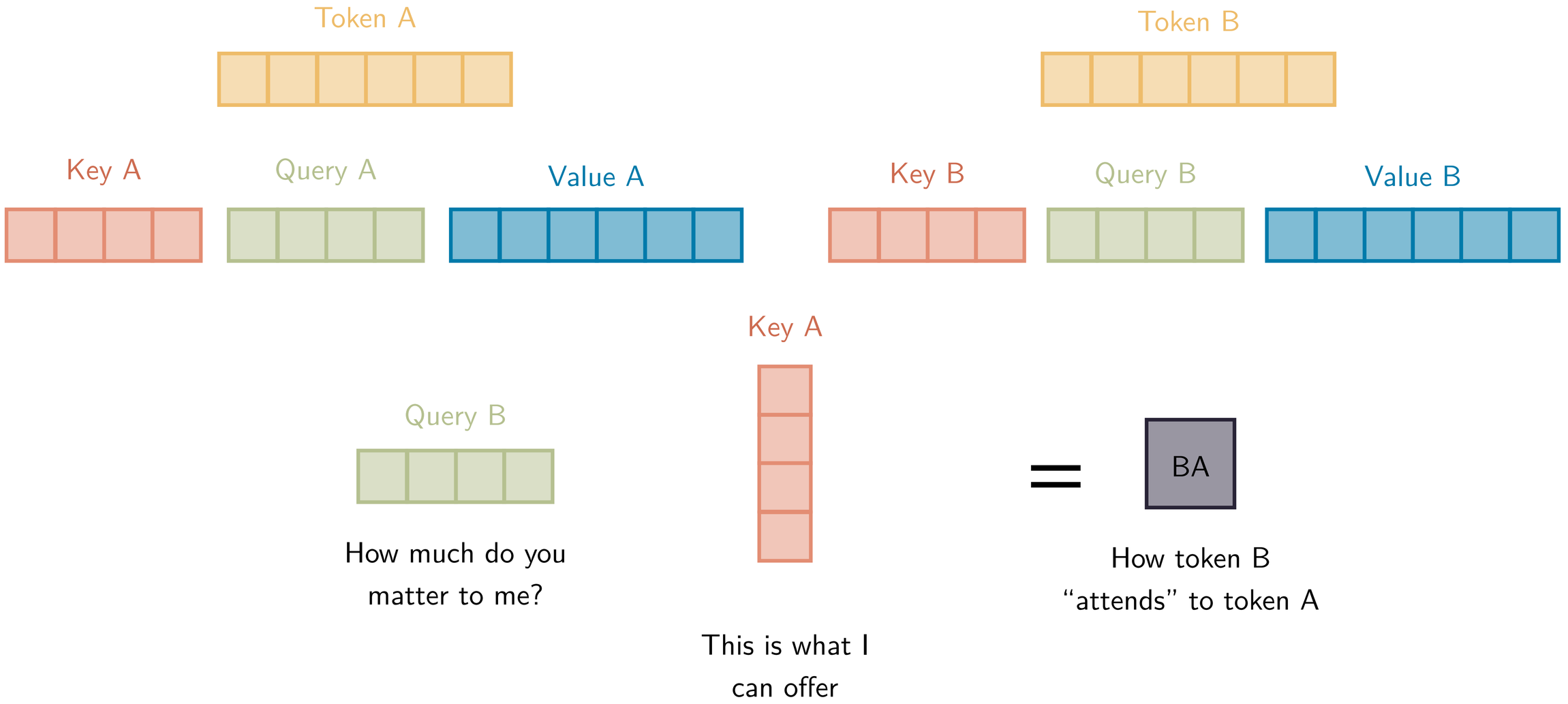
# Learning from text

The attention mechanism



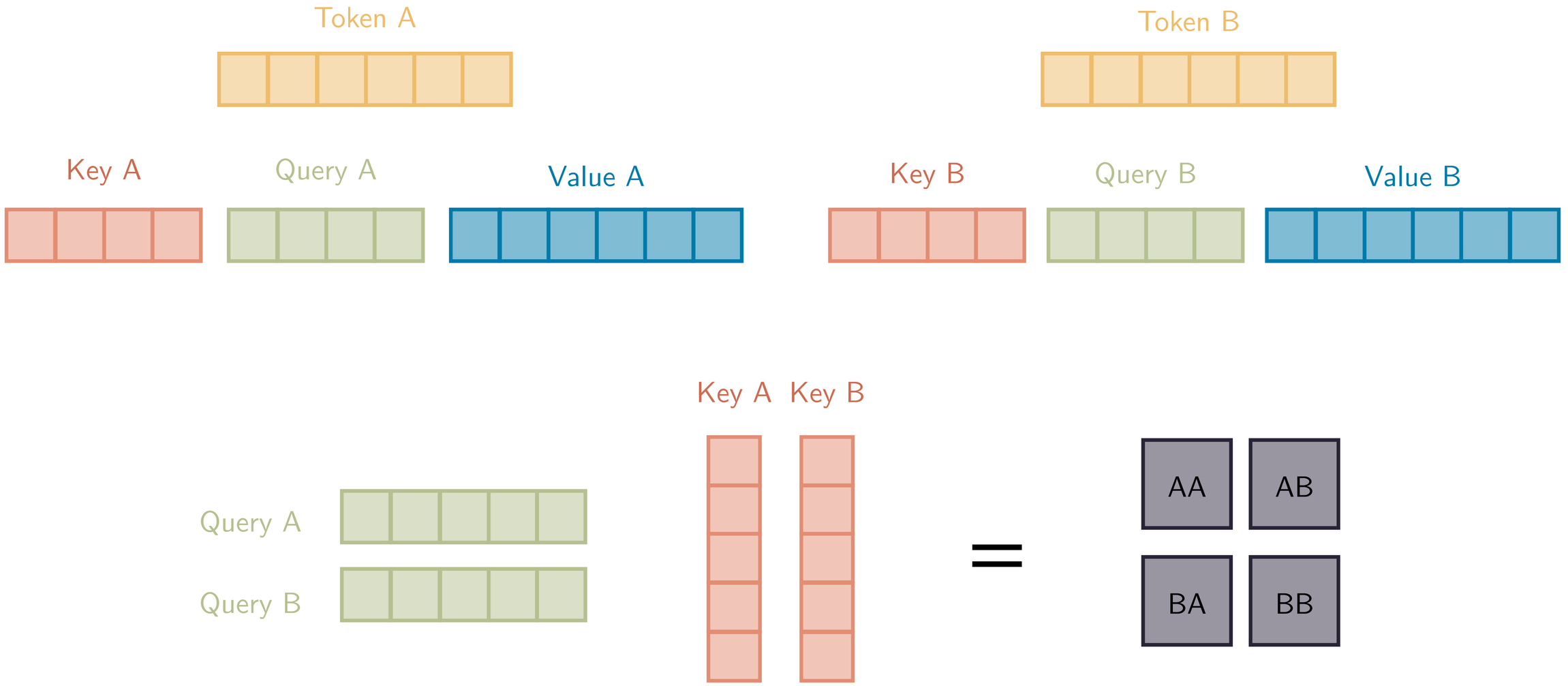
# Learning from text

The attention mechanism



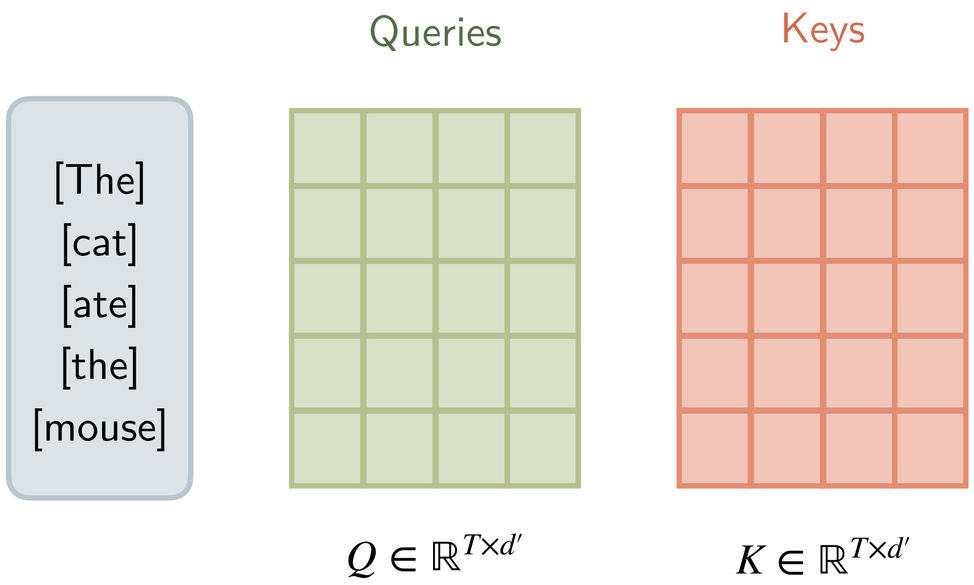
# Learning from text

The attention mechanism



# Learning from text

The attention mechanism



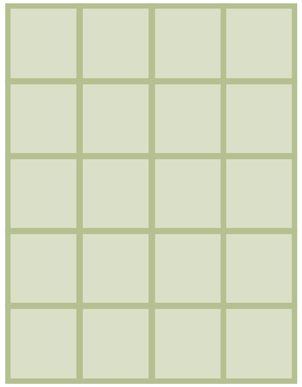
# Learning from text

The attention mechanism

Input

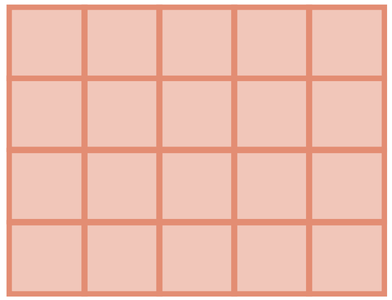
[The]
[cat]
[ate]
[the]
[mouse]

Queries



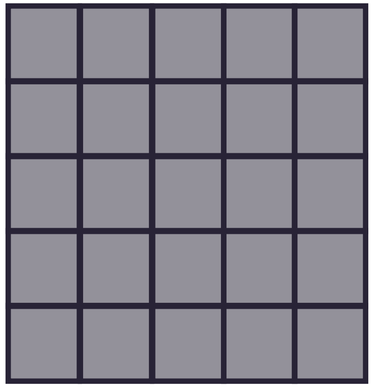
$Q \in \mathbb{R}^{T \times d'}$

Keys



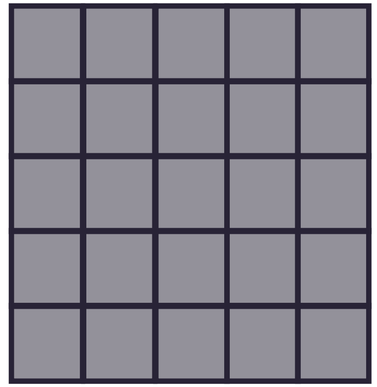
$K^\top \in \mathbb{R}^{d' \times T}$

Attention scores



$S := QK^\top \in \mathbb{R}^{T \times T}$

Attention weights

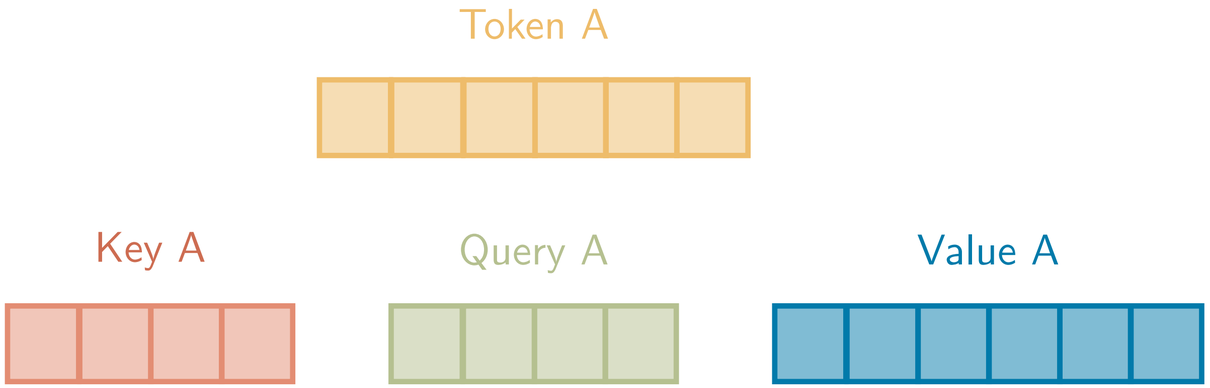


$$\text{softmax}\left(\frac{QK^\top}{\sqrt{d'}}\right) \in \mathbb{R}^{T \times T}$$

$$\text{softmax}\,(S_{i:})_j = \frac{e^{S_{ij}}}{\sum_{k=1}^{T} e^{S_{ik}}}$$

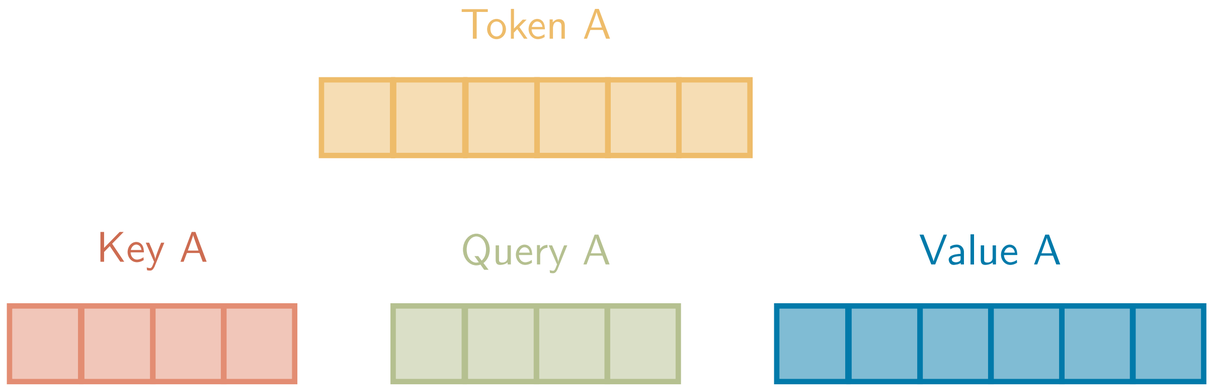
# Learning from text

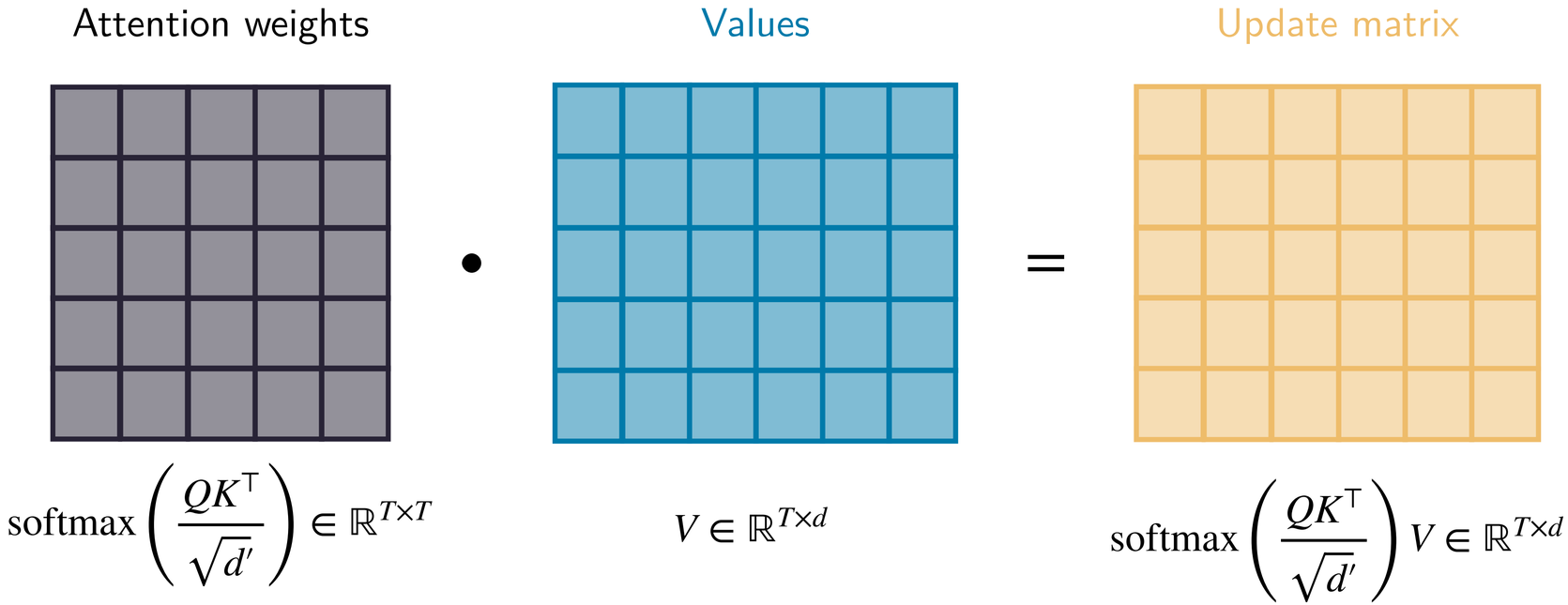
The attention mechanism



# Learning from text

The attention mechanism



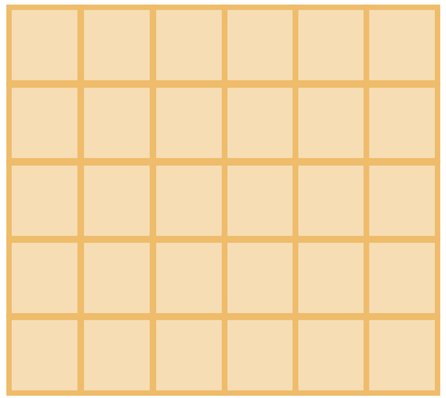
# Learning from text

The attention mechanism

Input

[The]
[cat]
[ate]
[the]
[mouse]

Embedding matrix

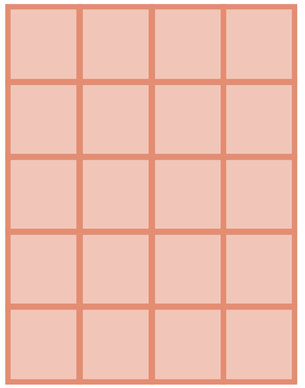


$X \in \mathbb{R}^{T \times d}$

$T$ number of tokens

$d$ dimension embeddings

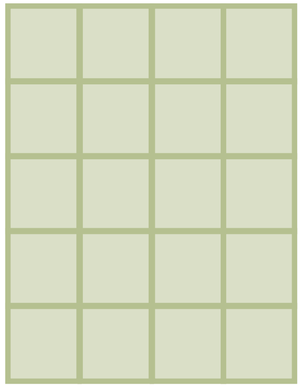
Keys



$K \in \mathbb{R}^{T \times d'}$

$K = XW_K$

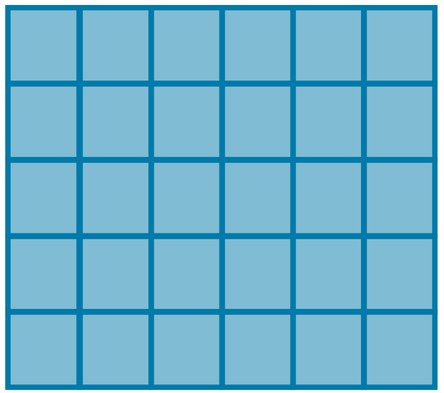
Queries



$Q \in \mathbb{R}^{d \times T}$

$Q = XW_Q$

Values

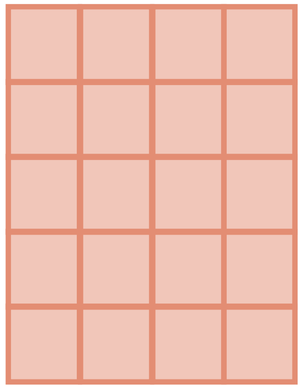


$V \in \mathbb{R}^{d \times T}$

$V = XW_V$

# Learning from text

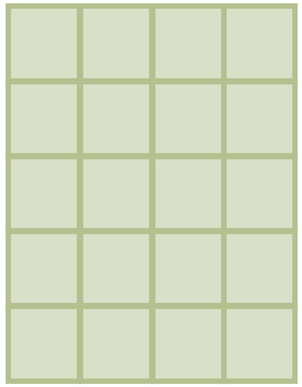
The attention mechanism



Keys

$K \in \mathbb{R}^{T \times d'}$

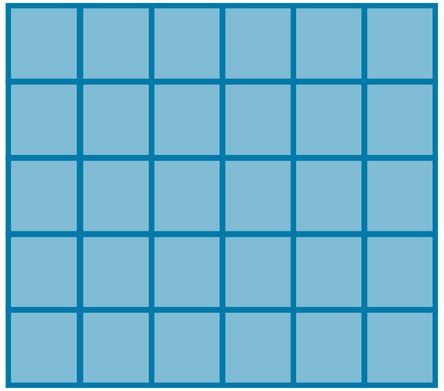
$K = XW_K$



Queries

$Q \in \mathbb{R}^{d \times T}$

$Q = XW_Q$

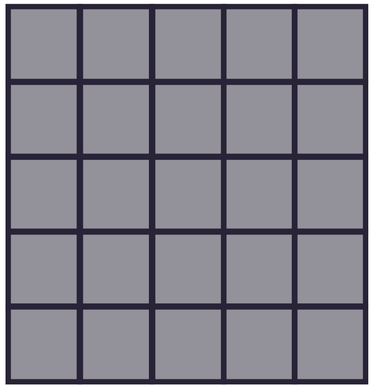


Values

$V \in \mathbb{R}^{d \times T}$

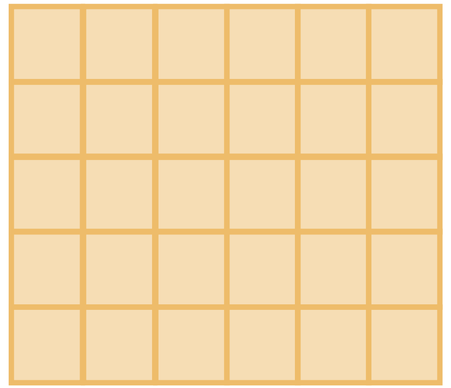
$V = XW_V$

Attention weights



$$\text{softmax}\left(\frac{QK^\top}{\sqrt{d'}}\right) \in \mathbb{R}^{T \times T}$$

Update matrix



$$\Delta X = \text{softmax}\left(\frac{QK^\top}{\sqrt{d'}}\right) V \in \mathbb{R}^{T \times d}$$

# Learning from text

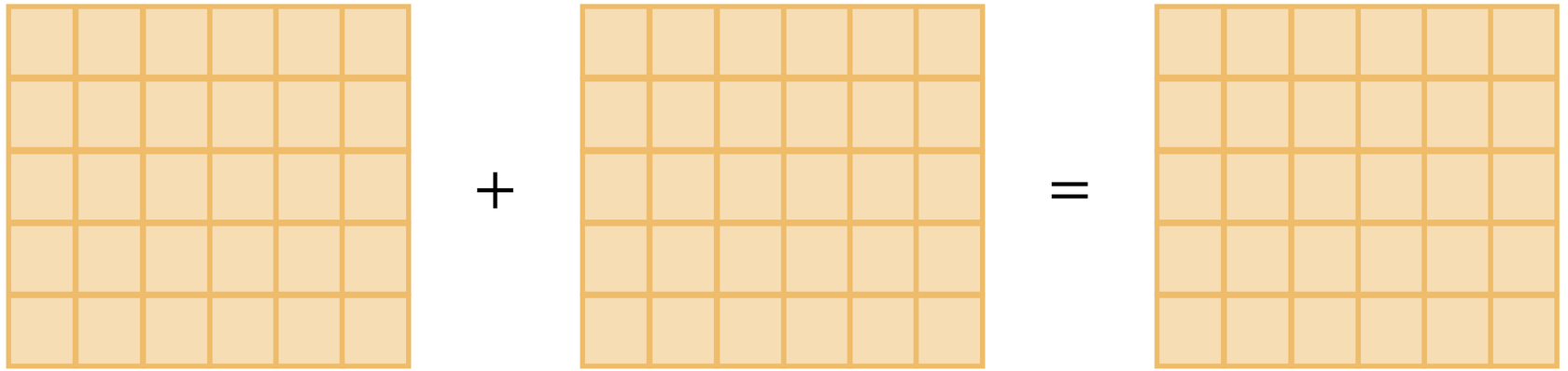
The attention mechanism

**Attention head**

Embedding matrix

Update matrix

Updated embeddings

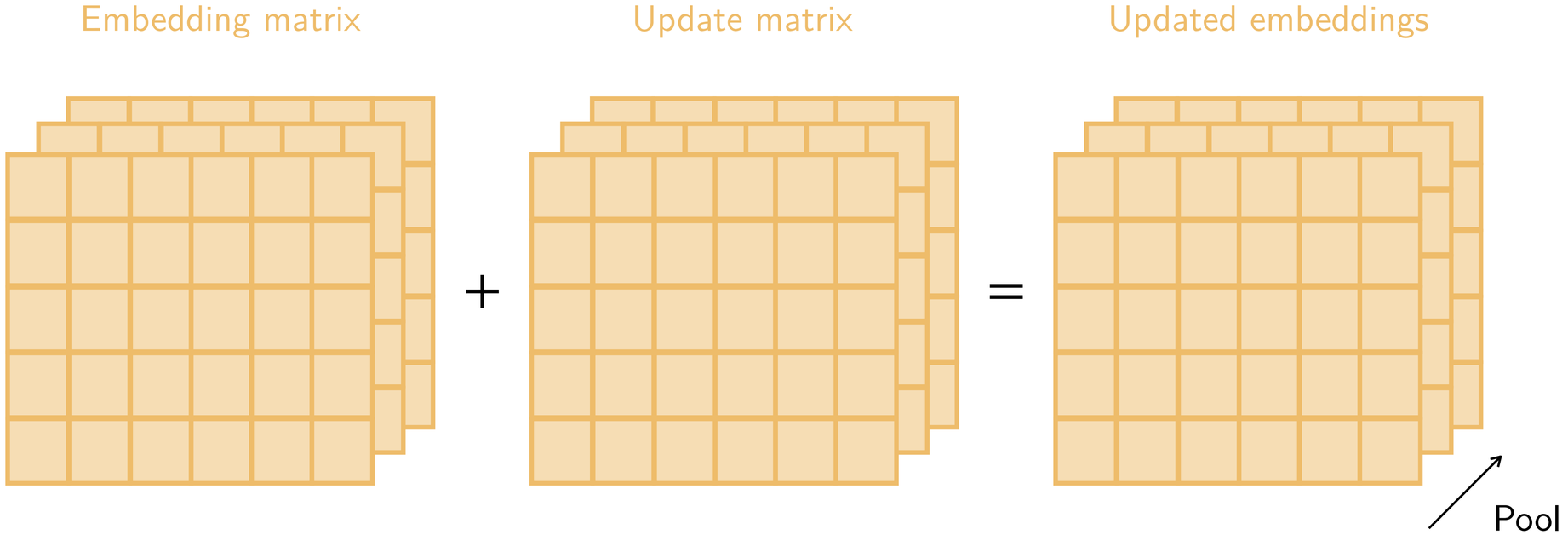


$$X \in \mathbb{R}^{T \times d} \qquad \Delta X = \text{softmax}\left(\frac{QK^\top}{\sqrt{d'}}\right) V \in \mathbb{R}^{T \times d} \qquad X' \in \mathbb{R}^{T \times d}$$

# Learning from text

The attention mechanism

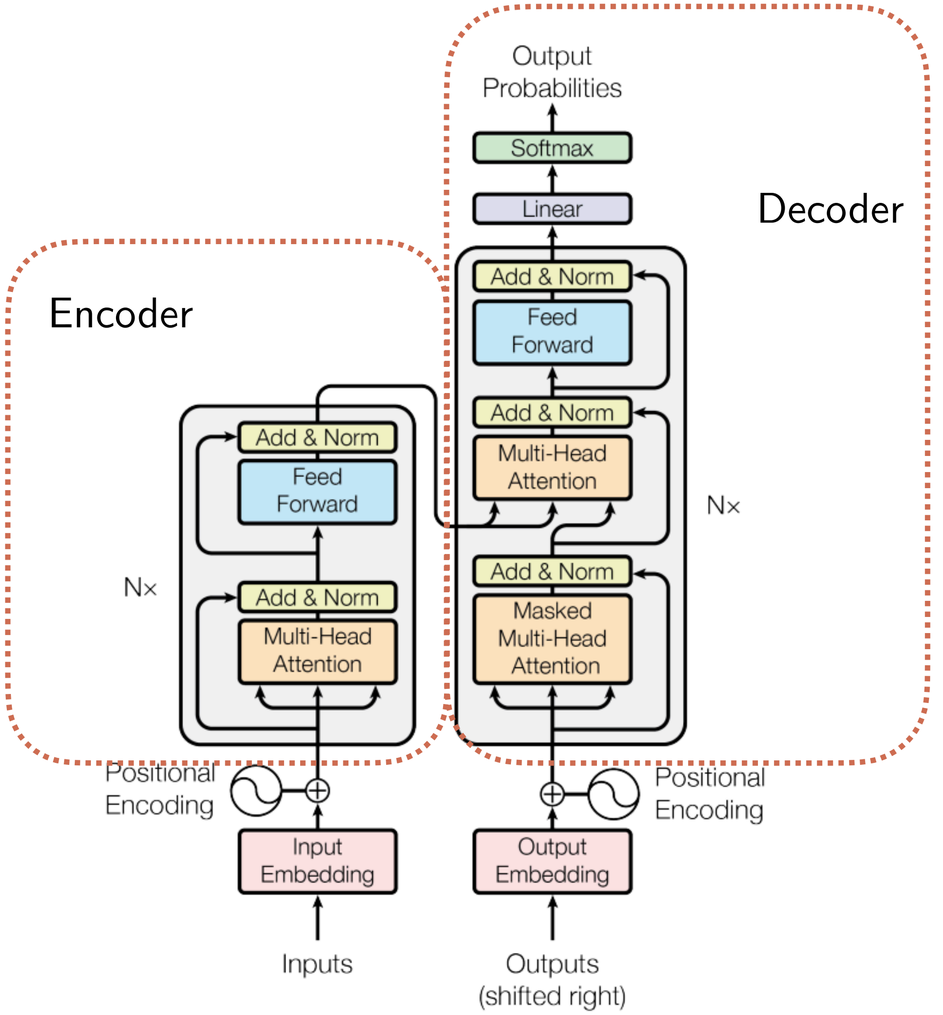
## Multi-Head Attention




$$X_i \in \mathbb{R}^{T \times d} \qquad \Delta X_i = \text{softmax}\left(\frac{QK^\top}{\sqrt{d'}}\right) V \in \mathbb{R}^{T \times d} \qquad X' \in \mathbb{R}^{T \times d}$$

# Learning from text

Transformers and the attention mechanism




1. **Predict masked tokens:**
   *The [mask] ate the mouse*
2. **Next token prediction:**
   The cat ate the...
3. **Translation**

A. Vaswani, S. Noam, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser and I. Polosukhin
*Attention is All You Need*, NeurIPS 2017

# The bitter lesson

Session 1 — Topological and Geometric Deep Learning: Theory, Methods and Applications

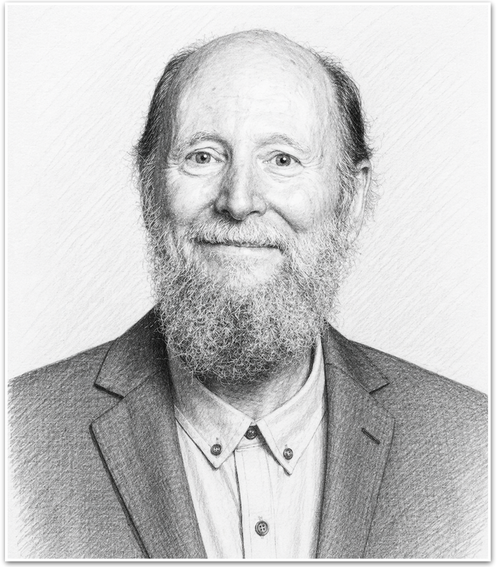
Inés García Redondo — May 11, 2026

# The bitter lesson

## The Bitter Lesson

**Rich Sutton**

**March 13, 2019**



The biggest lesson that can be read from 70 years of AI research is that general methods that leverage computation are ultimately the most effective, and by a large margin. […]

One thing that should be learned from the bitter lesson is the great power of general purpose methods, of methods that continue to scale with increased computation even as the available computation becomes very great. The two methods that seem to scale arbitrarily in this way are *search* and *learning*.

The second general point to be learned from the bitter lesson is that the actual contents of minds are tremendously, irredeemably complex; we should stop trying to find simple ways to think about the contents of minds, such as simple ways to think about space, objects, multiple agents, or symmetries. All these are part of the arbitrary, intrinsically-complex, outside world. They are not what should be built in, as their complexity is endless; instead we should build in only the meta-methods that can find and capture this arbitrary complexity. Essential to these methods is that they can find good approximations, but the search for them should be by our methods, not by us. We want AI agents that can discover like we can, not which contain what we have discovered. Building in our discoveries only makes it harder to see how the discovering process can be done.

# Thank you for your attention!

AIDOS LAB
AI FOR DATA-ORIENTED SCIENCE

UNI
FR

Session 1 — Introduction to Deep Learning and Topological Deep Learning

Inés García Redondo — May 11, 2026