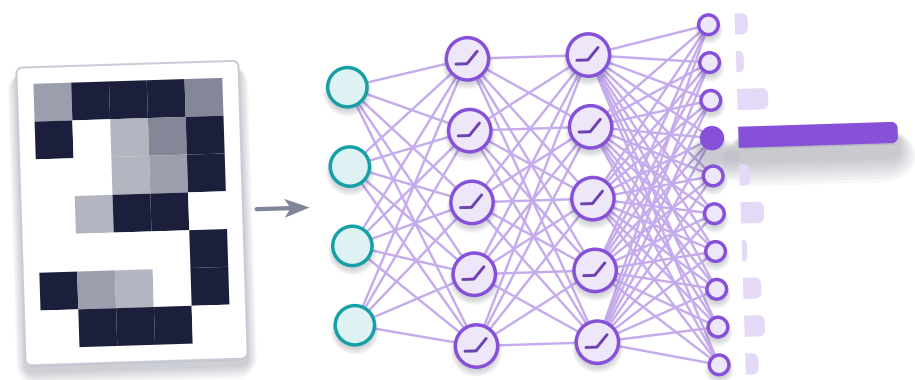
Prof. Aviv Zohar

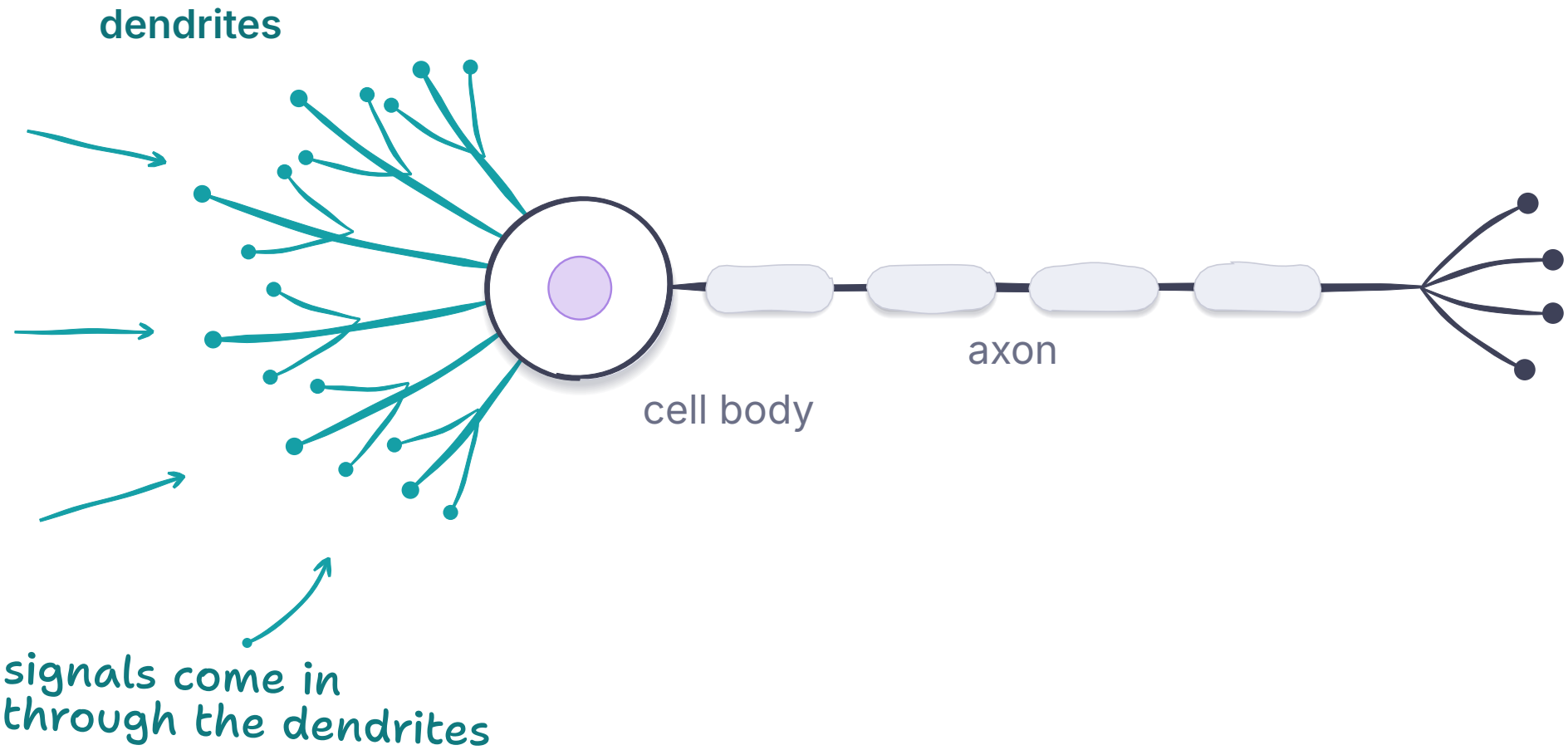
Semester A, 2026–7

Inside the Box: Neural Networks

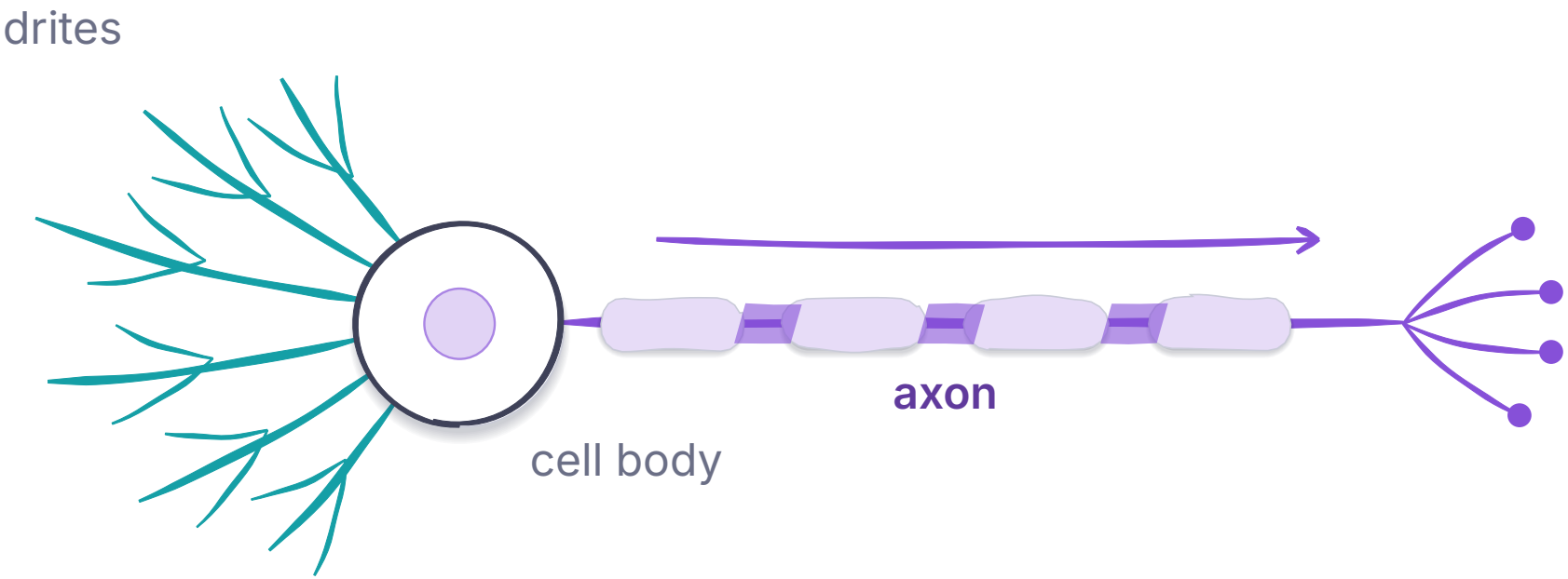
From a single neuron to a network that learns



Loosely inspired by the brain

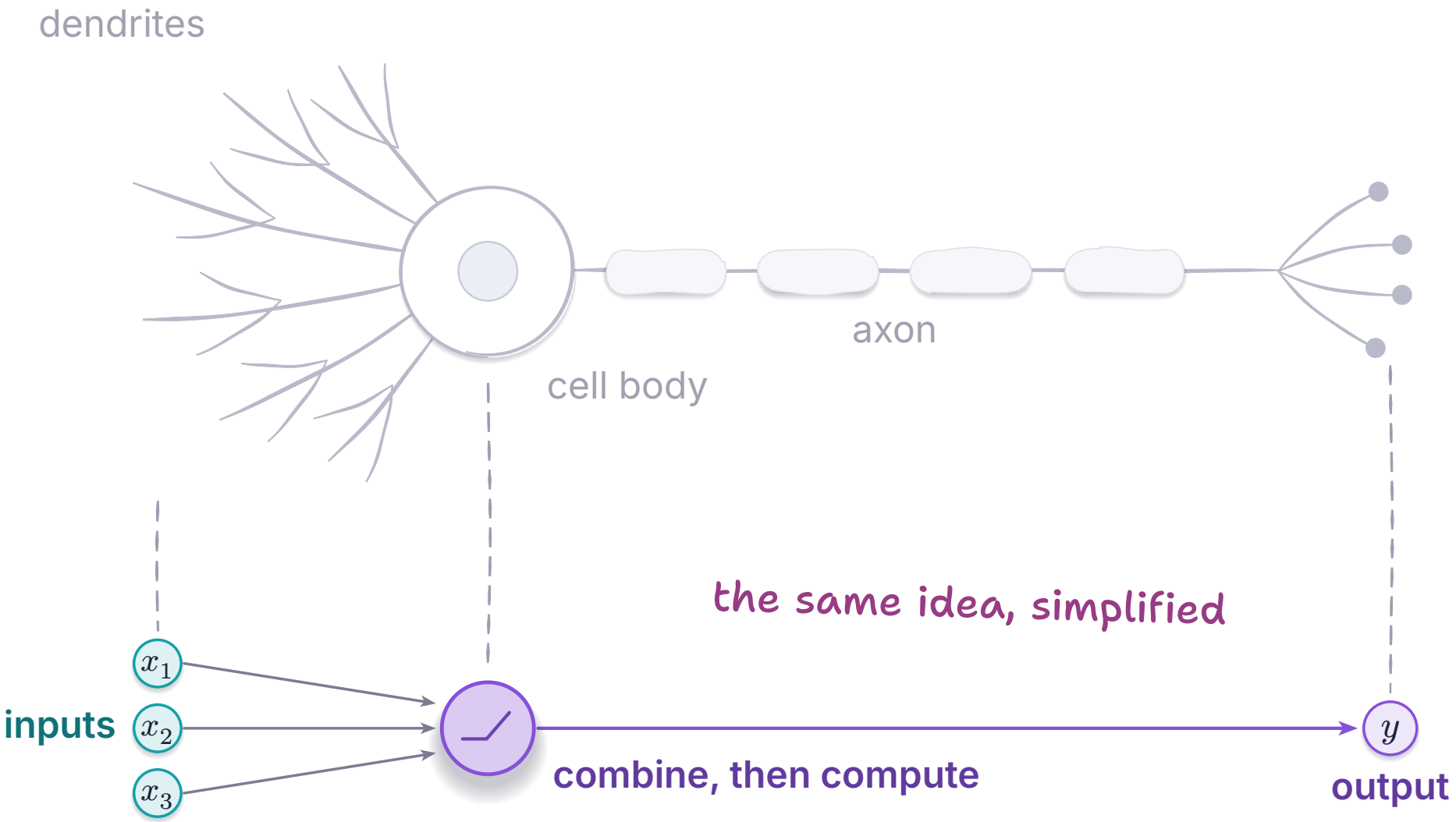


Loosely inspired by the brain



fires once the combined signal crosses a threshold

Loosely inspired by the brain

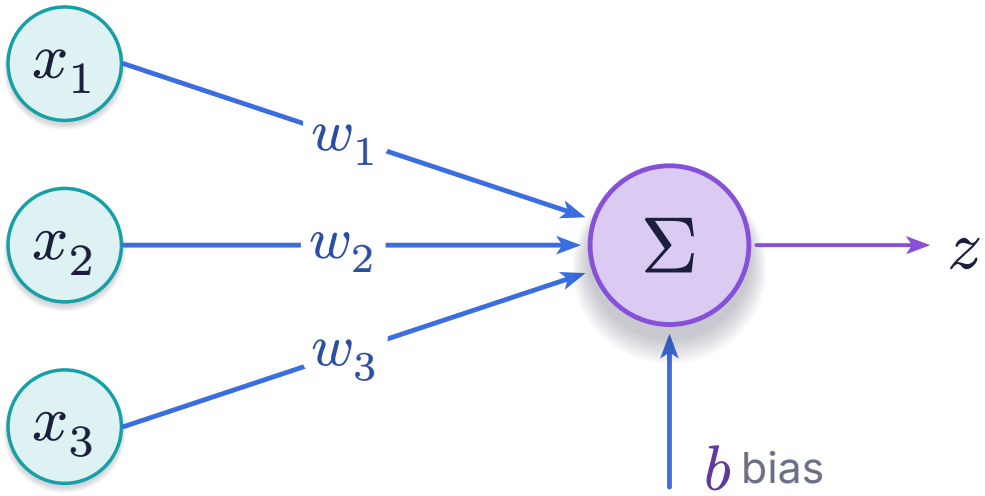


One artificial neuron



inputs: real numbers

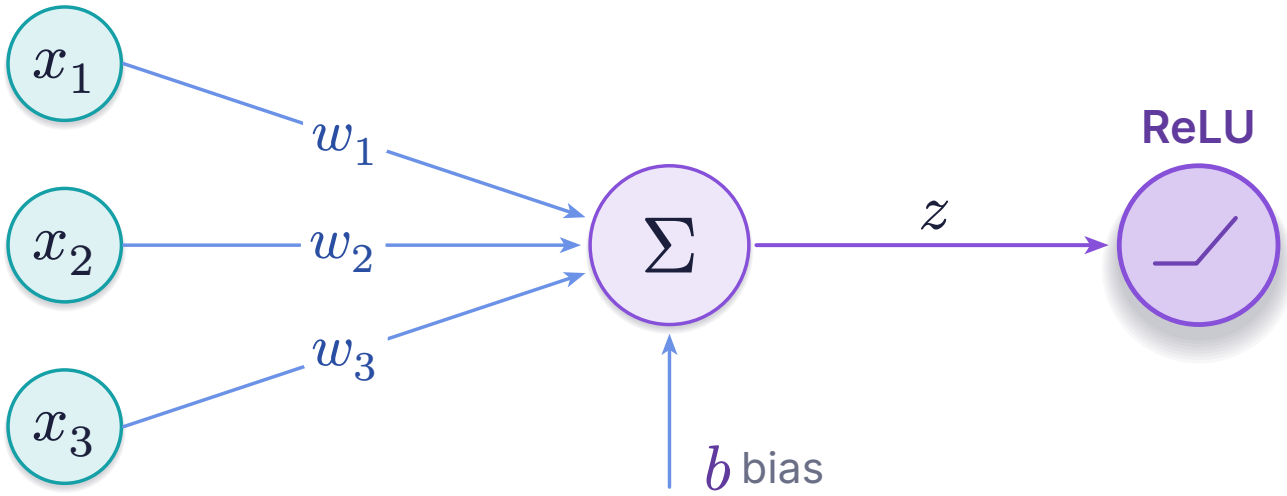
One artificial neuron



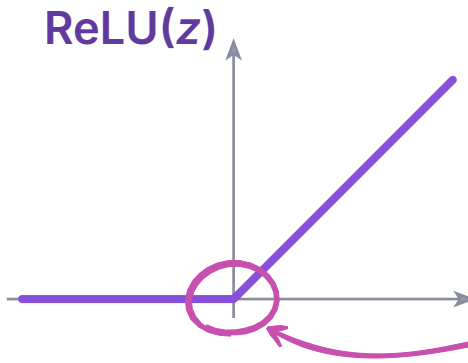
$$z = w_1 x_1 + w_2 x_2 + w_3 x_3 + b$$

Weight 0: the input adds nothing.
Negative weight: more input, smaller z.

One artificial neuron



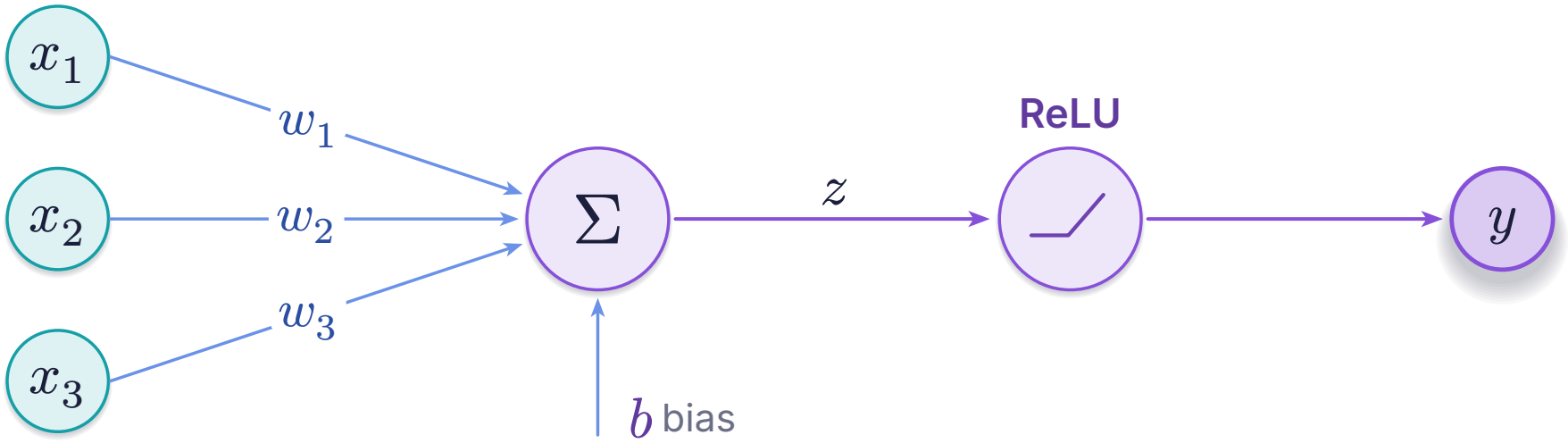
$$z = w_1 x_1 + w_2 x_2 + w_3 x_3 + b$$



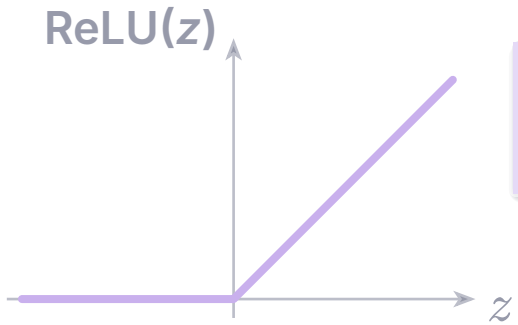
$$\mathrm{ReLU}(z) = \max(z, 0)$$

the bend makes it non-linear

One artificial neuron

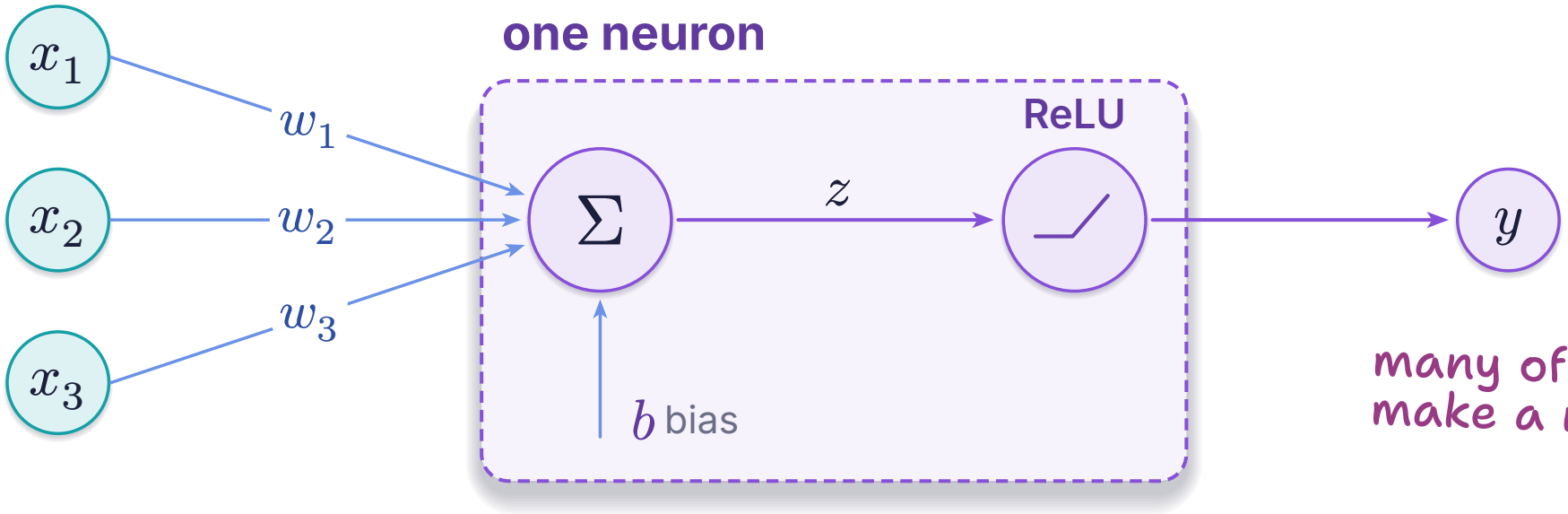


$$z = w_1 x_1 + w_2 x_2 + w_3 x_3 + b$$



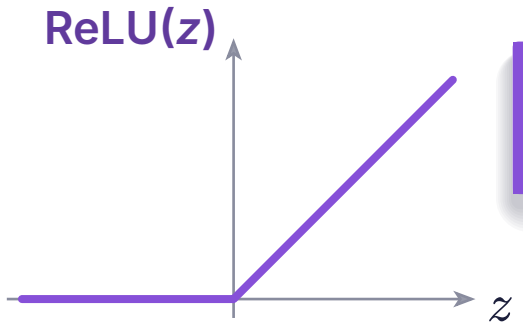
$$\mathrm{ReLU}(z) = \max(z, 0)$$

One artificial neuron



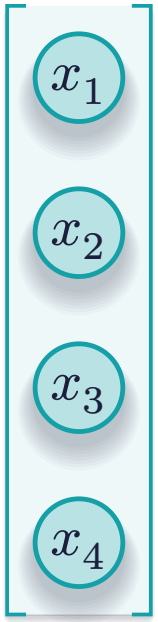
many of these make a network

$$z = w_1 x_1 + w_2 x_2 + w_3 x_3 + b$$



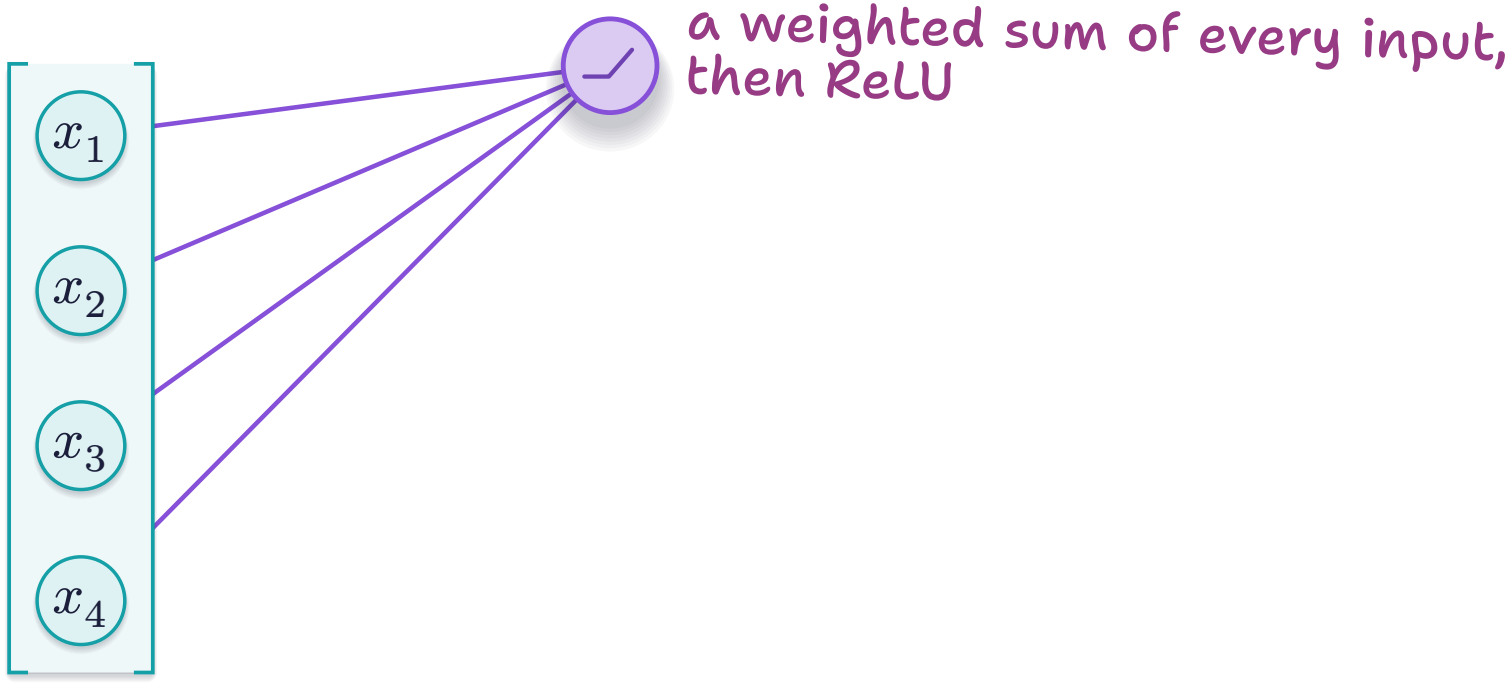
$$\mathrm{ReLU}(z) = \max(z, 0)$$

Neurons in layers make a network



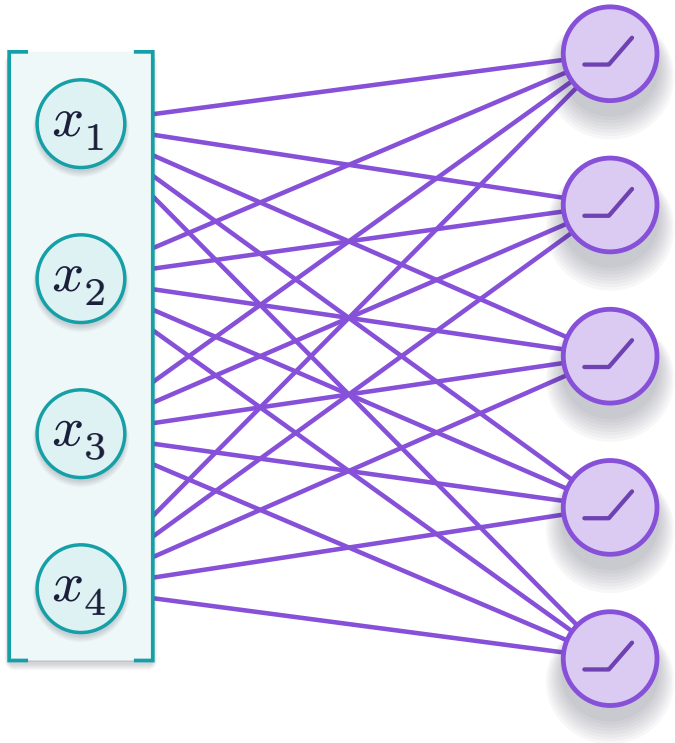
input vector

Neurons in layers make a network



input vector

Neurons in layers make a network

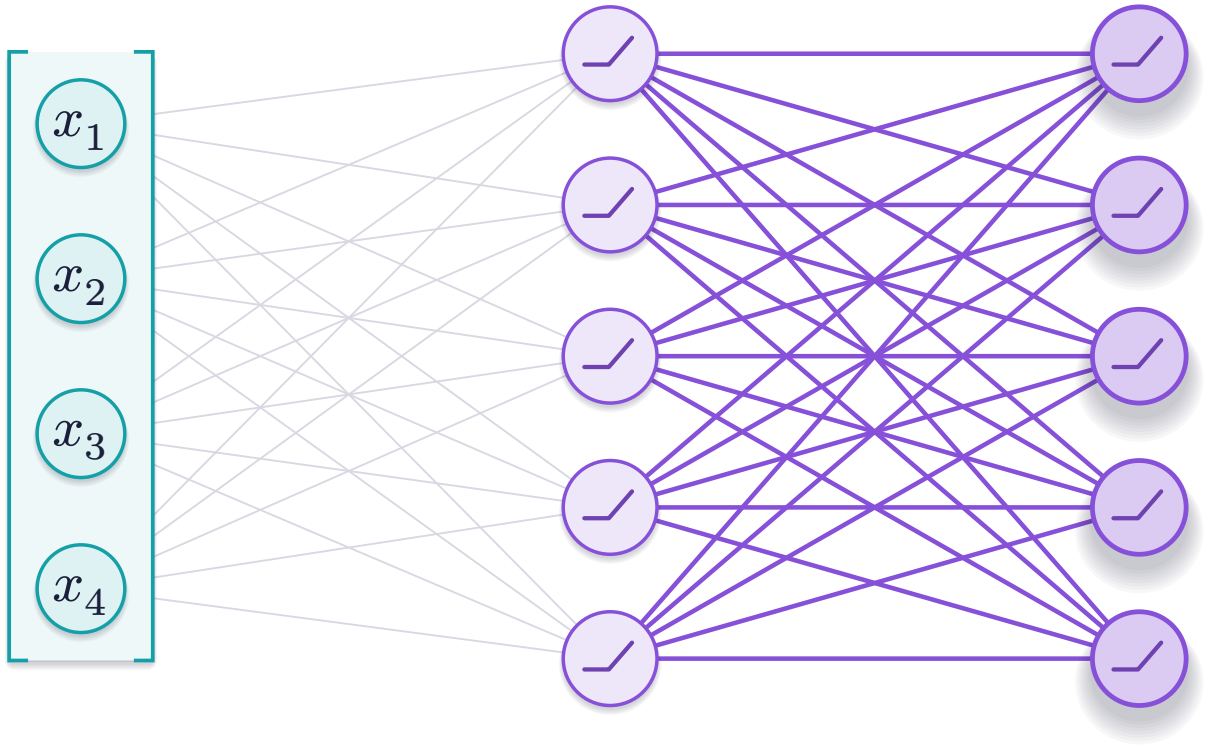


input vector

layer 1

Same inputs,
own weights and bias,
so a different function
each.

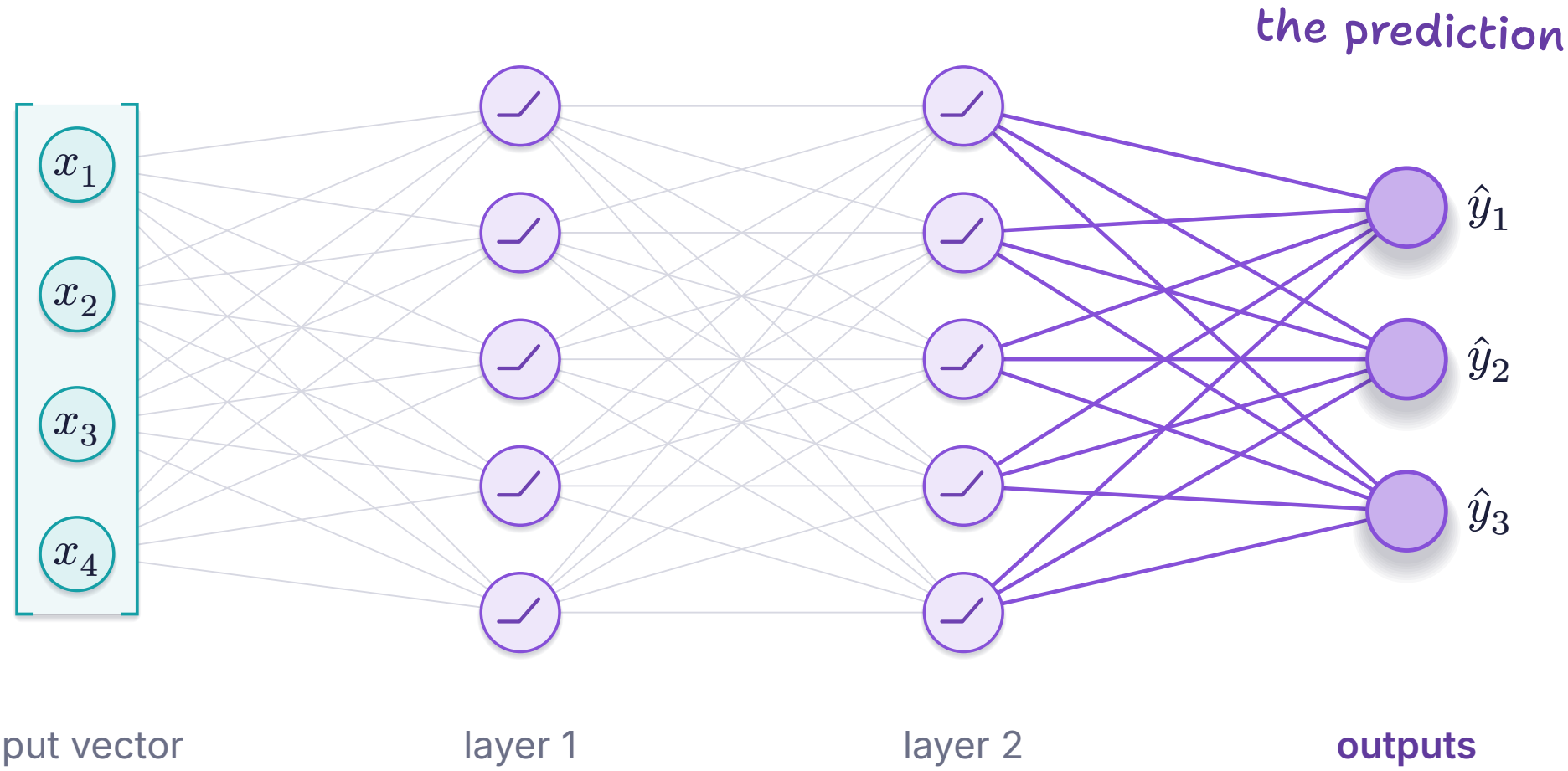
Neurons in layers make a network



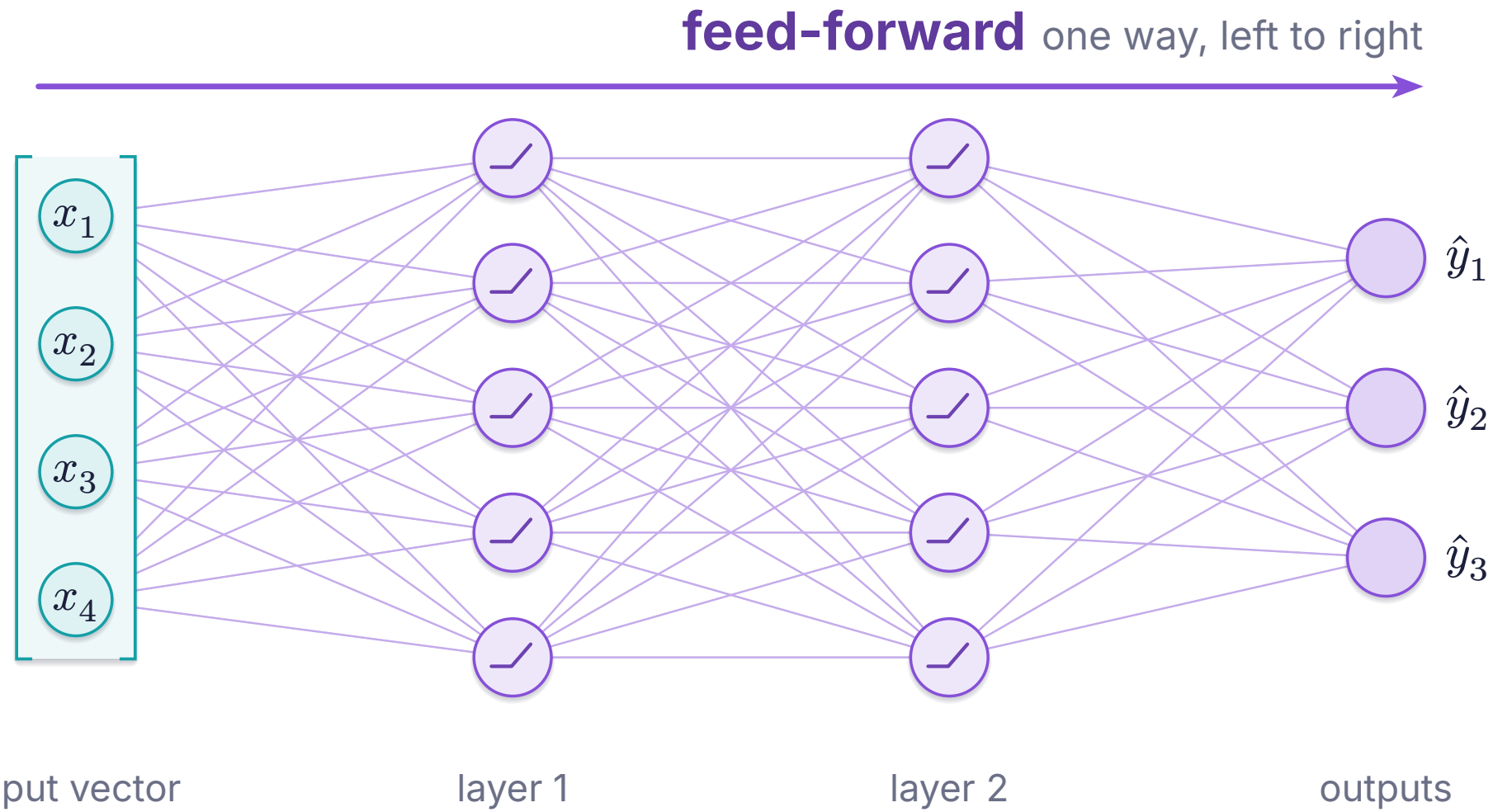
input vector | layer 1 | **layer 2**

layer 1's outputs
are layer 2's inputs

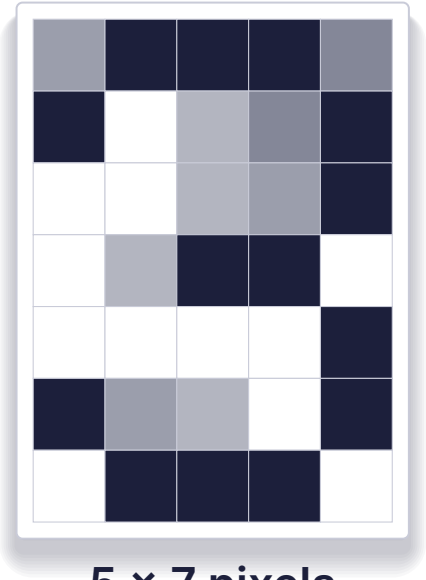
Neurons in layers make a network



Neurons in layers make a network



A task: recognize a handwritten digit



5 × 7 pixels

a scan is just
a grid of pixels

A task: recognize a handwritten digit

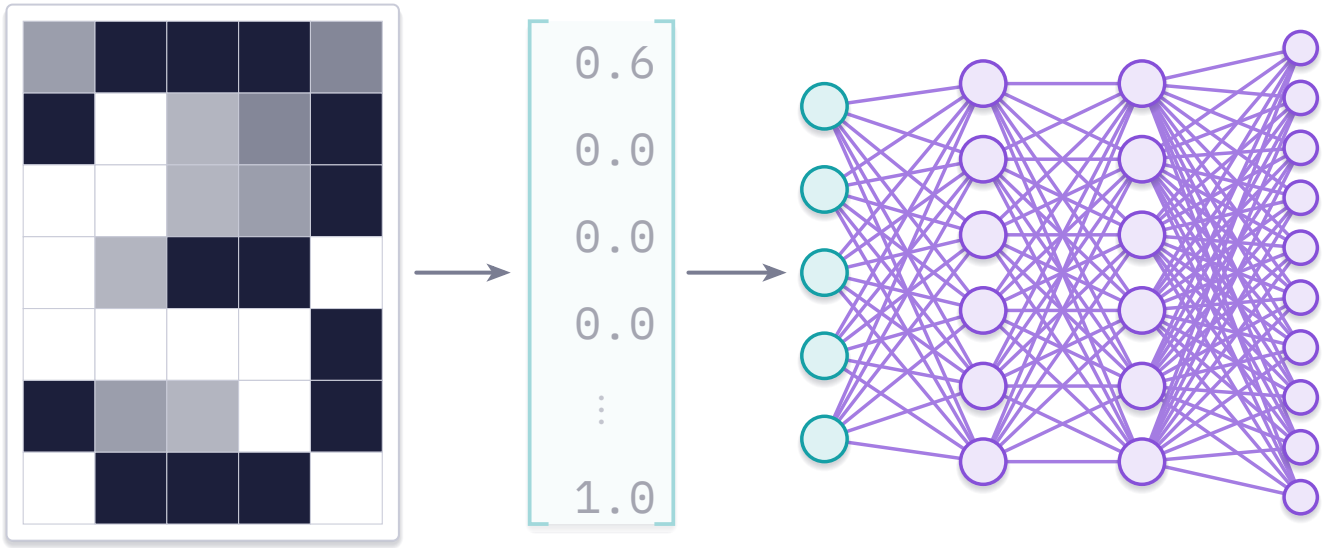


5 × 7 pixels
0 = black · 1 = white

35 numbers

Only brightness. Nothing here says which digit it is: the network will learn that from examples.

A task: recognize a handwritten digit

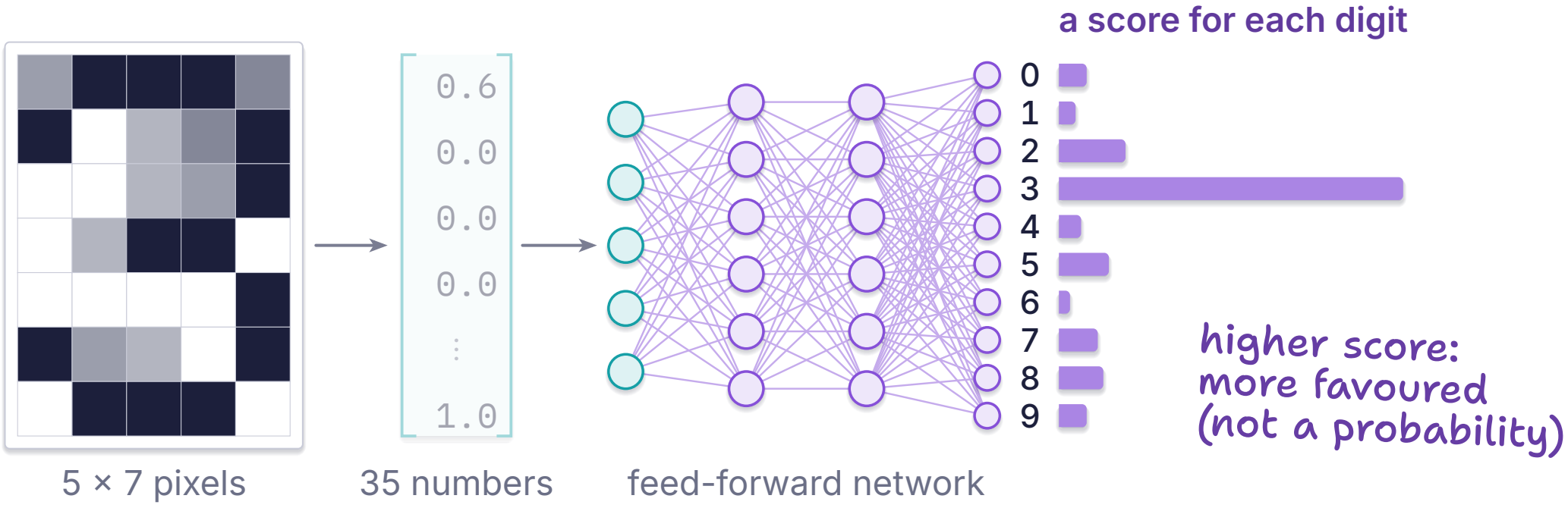


5 × 7 pixels

35 numbers

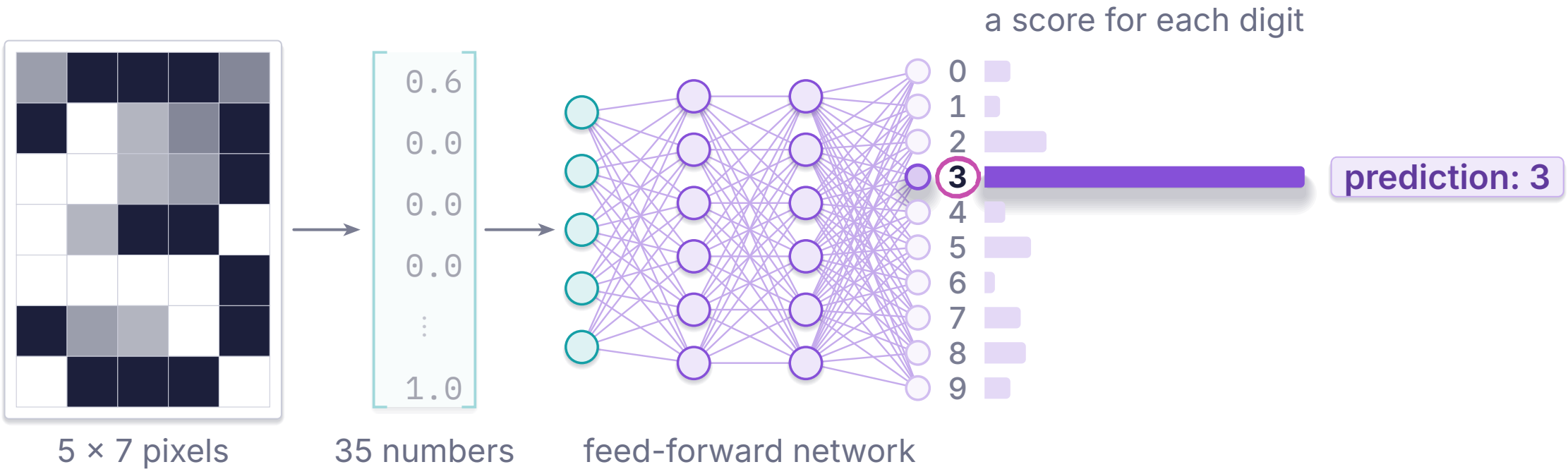
feed-forward network

A task: recognize a handwritten digit



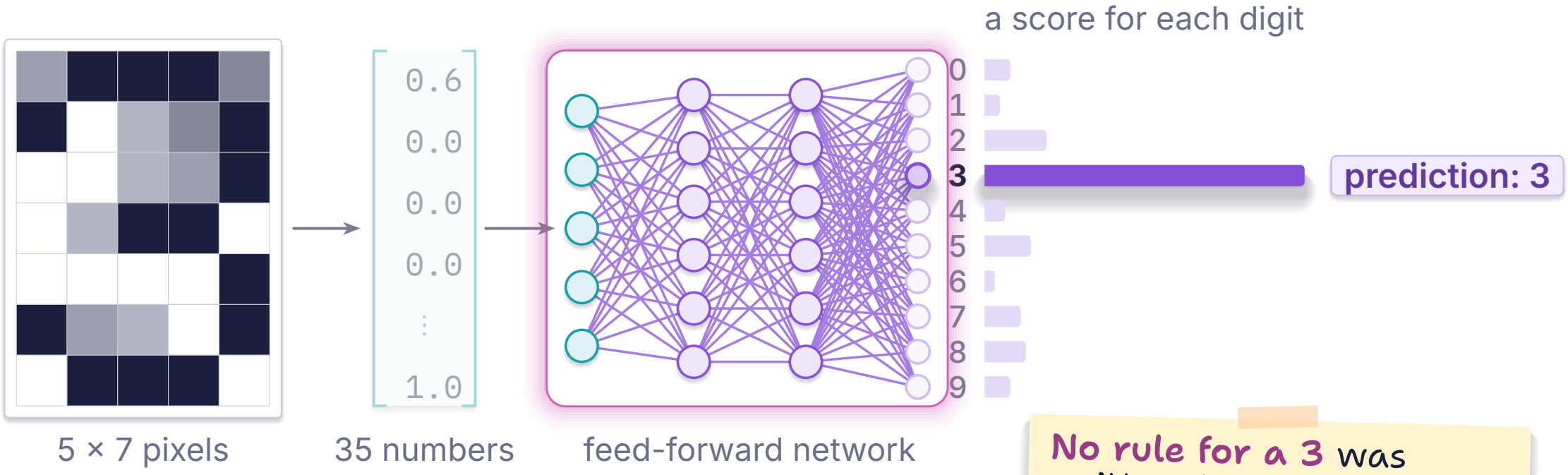
scores are illustrative

A task: recognize a handwritten digit



scores are illustrative

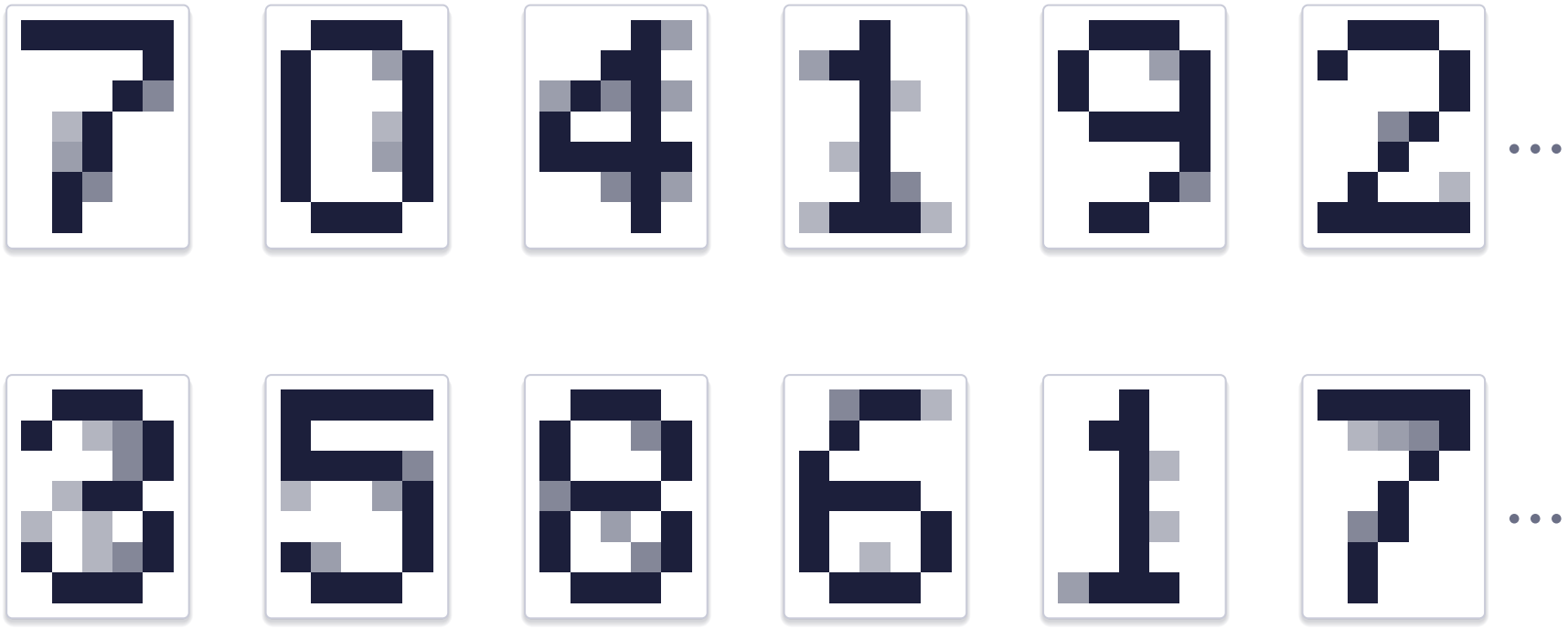
A task: recognize a handwritten digit



No rule for a 3 was written: it is spread across the weights.

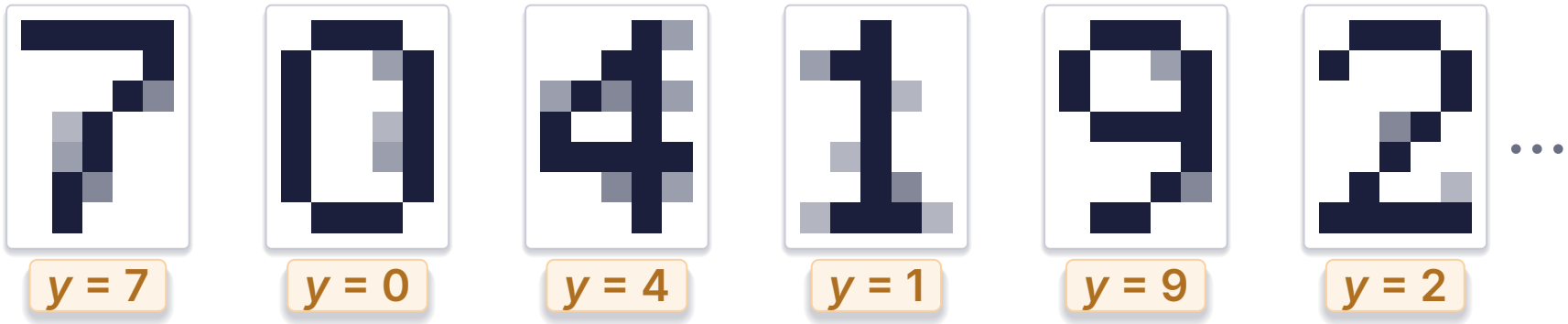
scores are illustrative

Learning from examples



lots of
scanned digits

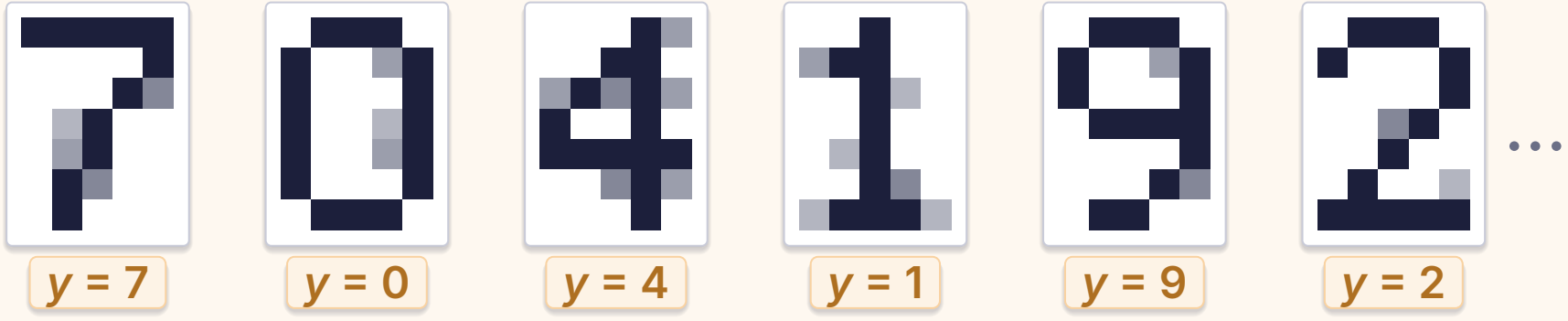
Learning from examples



each with its correct label

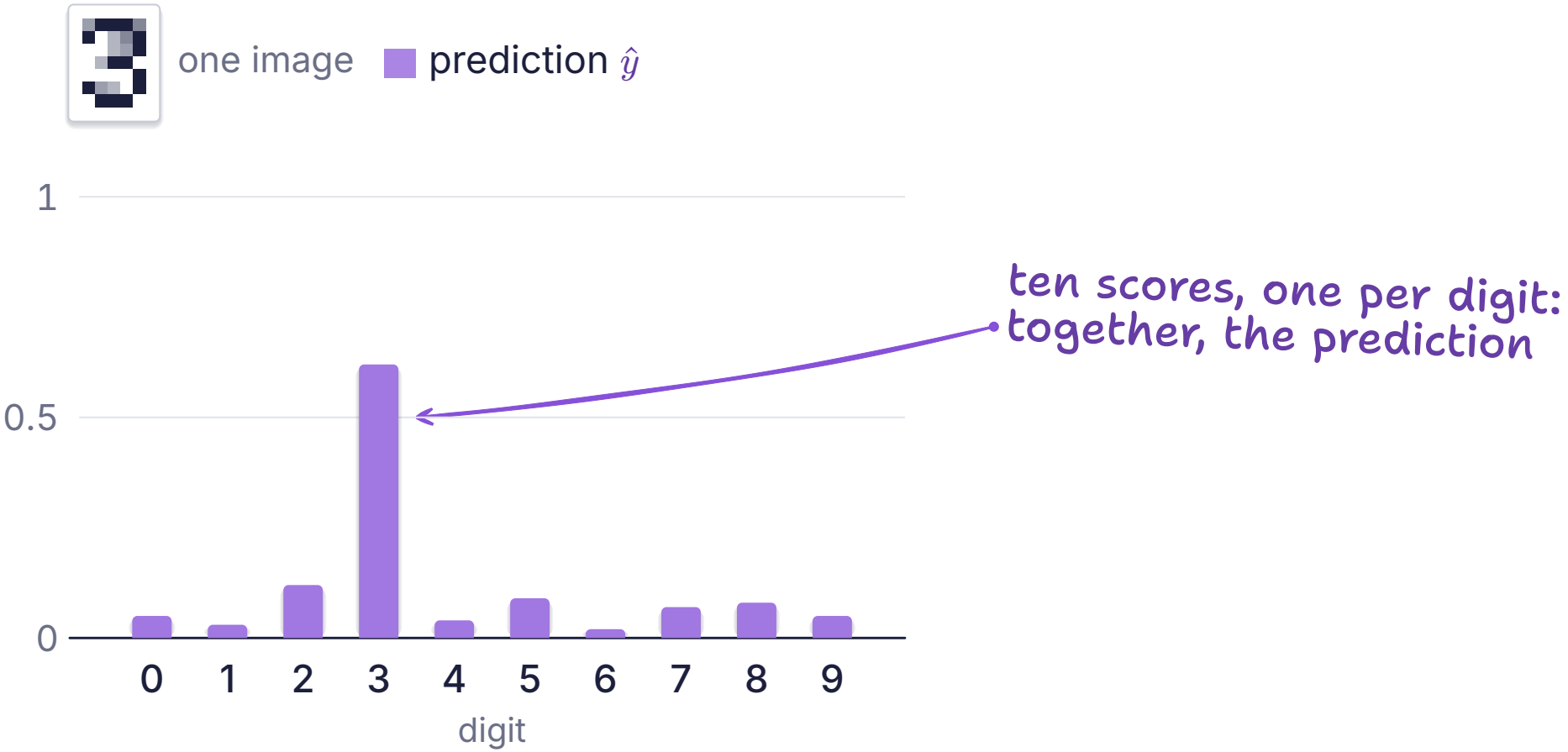
Learning from examples

training set: (image, label) pairs

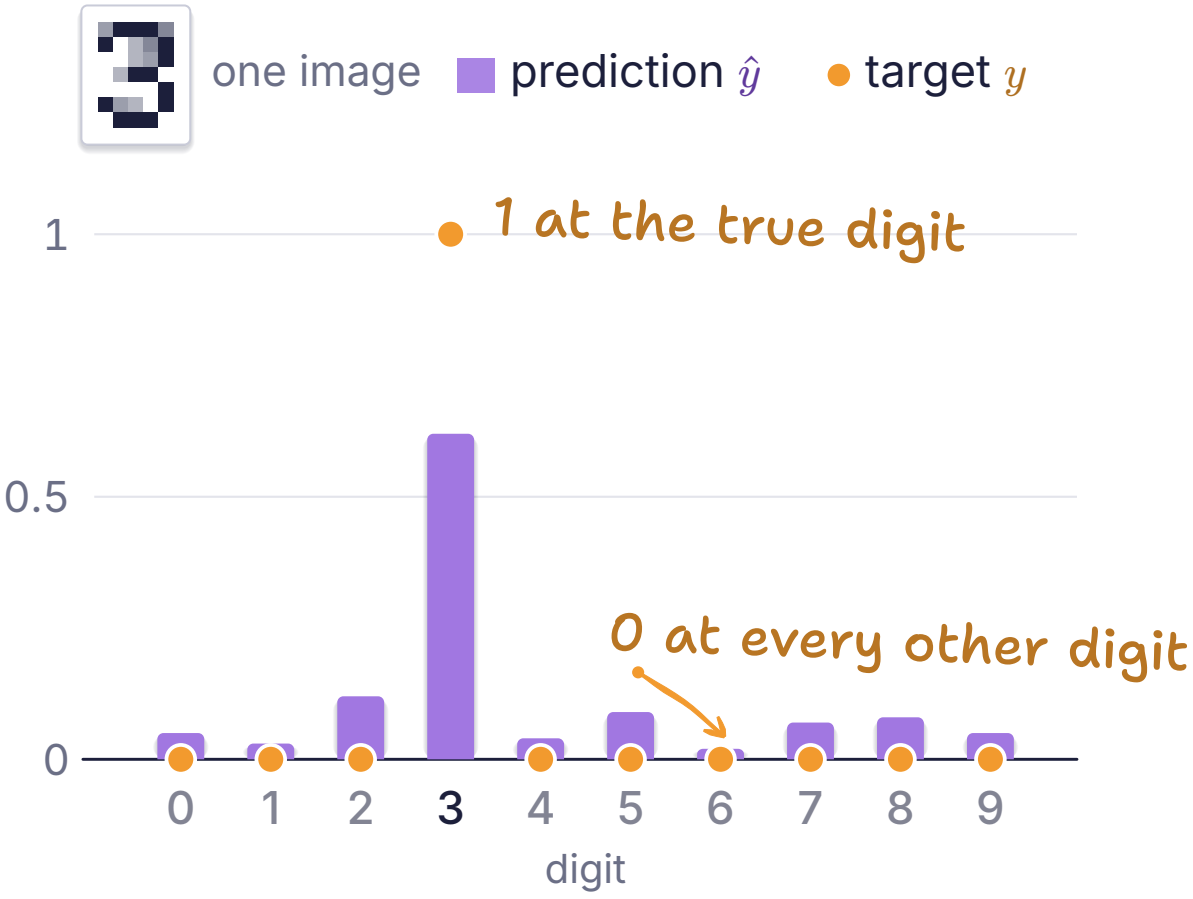


More varied, correctly labelled examples: more to learn from.

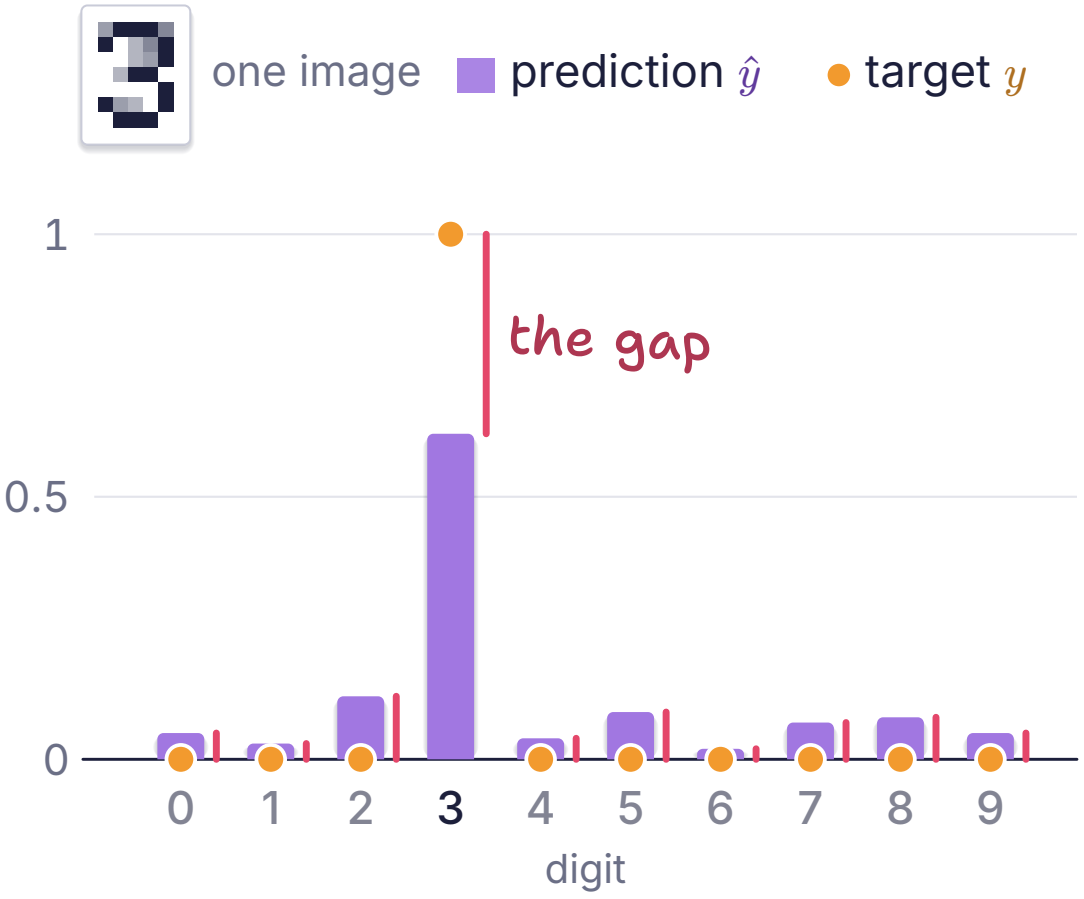
The loss measures how wrong a prediction is



The loss measures how wrong a prediction is



The loss measures how wrong a prediction is



$$L = (\hat{y}_0 - y_0)^2 + \cdots + (\hat{y}_9 - y_9)^2$$

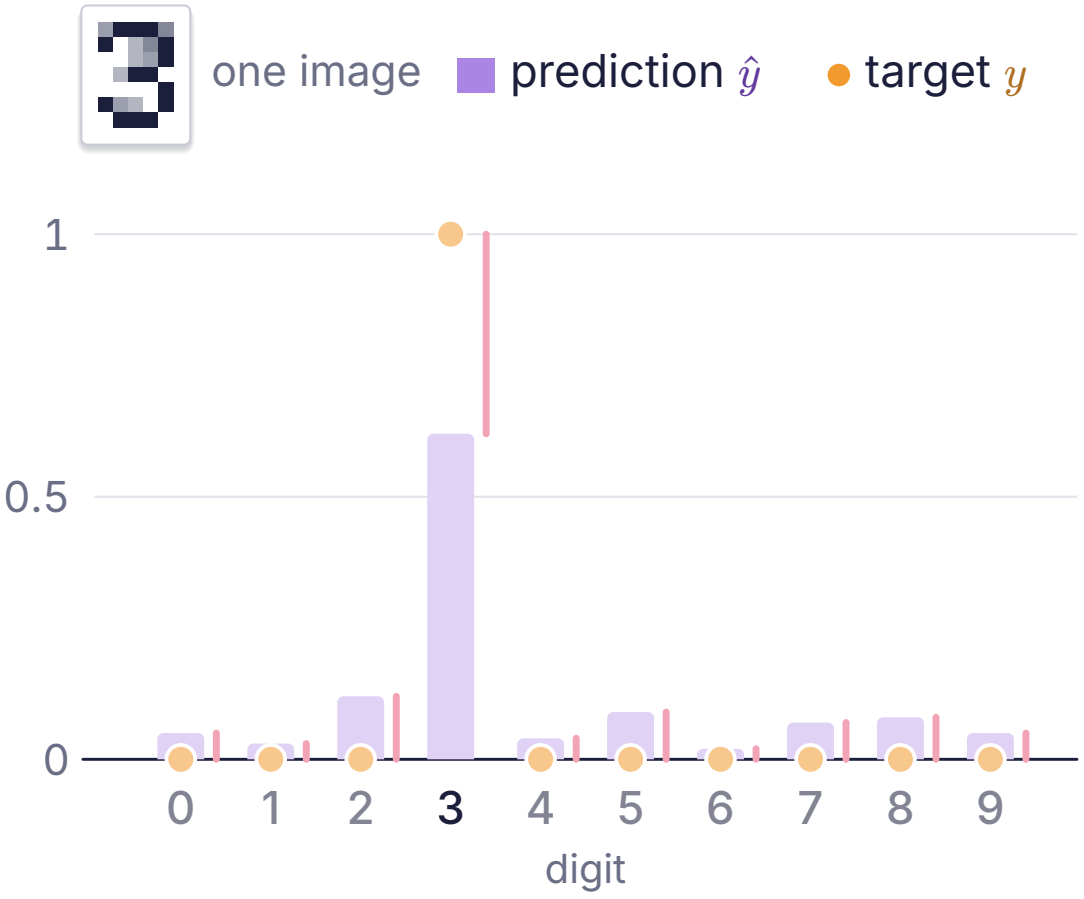
$L = 0.19$ for this image

Squared: every gap counts as positive, and big gaps count much more.

L = 0 only when every score matches its target.

scores and loss are illustrative

The loss measures how wrong a prediction is



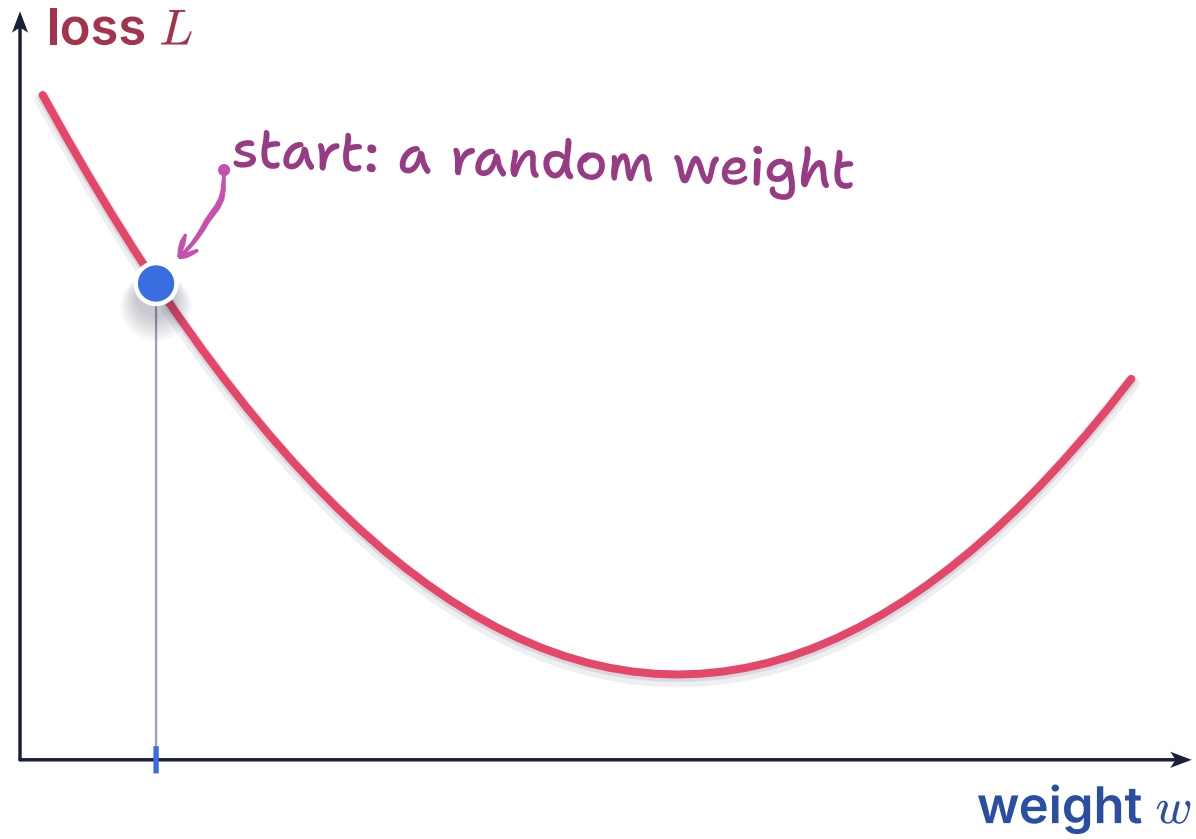
$$L = (\hat{y}_0 - y_0)^2 + \cdots + (\hat{y}_9 - y_9)^2$$

$$\text{average loss} = (L_1 + \cdots + L_N)/N$$

one loss per image, over all *N* images

training adjusts the weights to bring this down

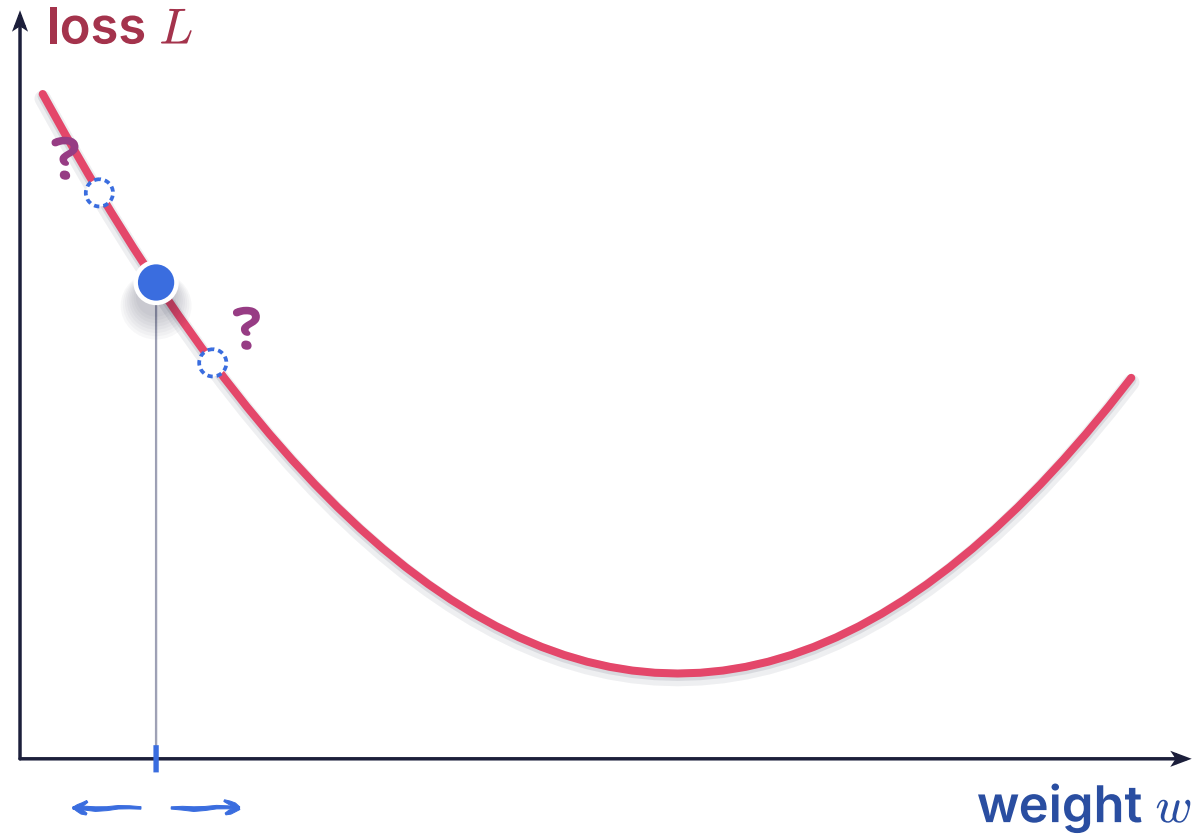
Training follows the slope downhill



Training set fixed: the loss depends on the weights, $L(w)$

a simple loss curve, for illustration

Training follows the slope downhill

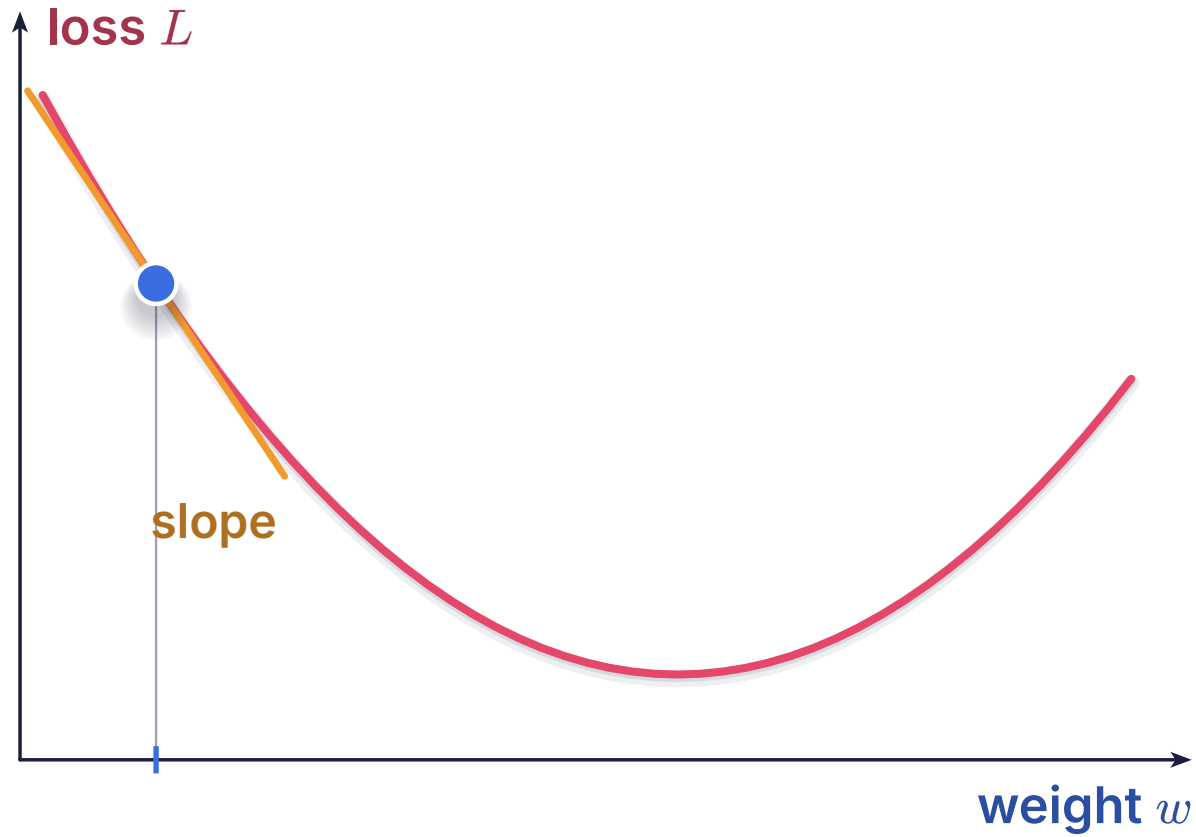


Training set fixed: the loss depends on the weights, $L(w)$

Nudge w a little. Does the loss grow or shrink?

a simple loss curve, for illustration

Training follows the slope downhill



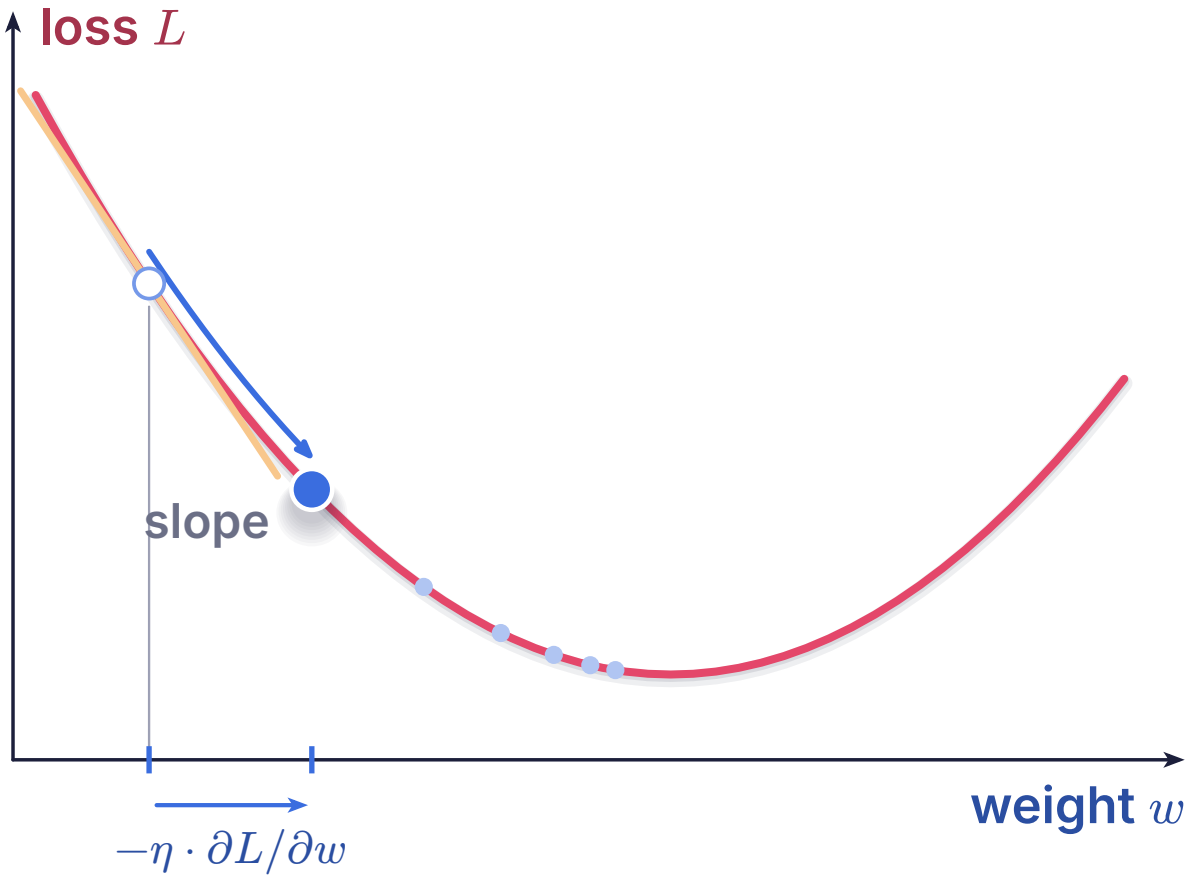
Training set fixed: the loss depends on the weights, $L(w)$

slope $= \partial L / \partial w$

sign: which way the loss grows
size: how steep the change is

a simple loss curve, for illustration

Training follows the slope downhill



Training set fixed: the loss depends on the weights, $L(w)$

slope $= \partial L/\partial w$

sign: which way the loss grows
size: how steep the change is

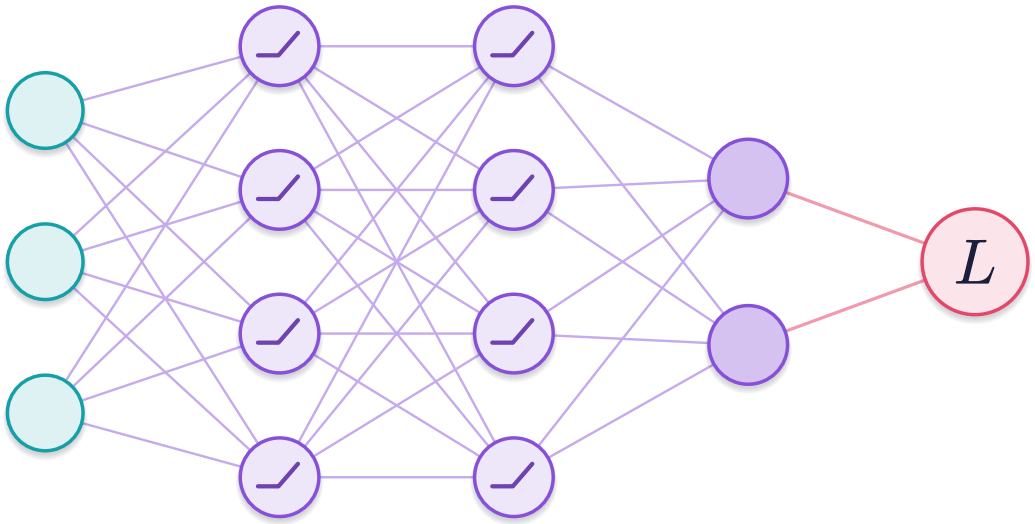
gradient descent

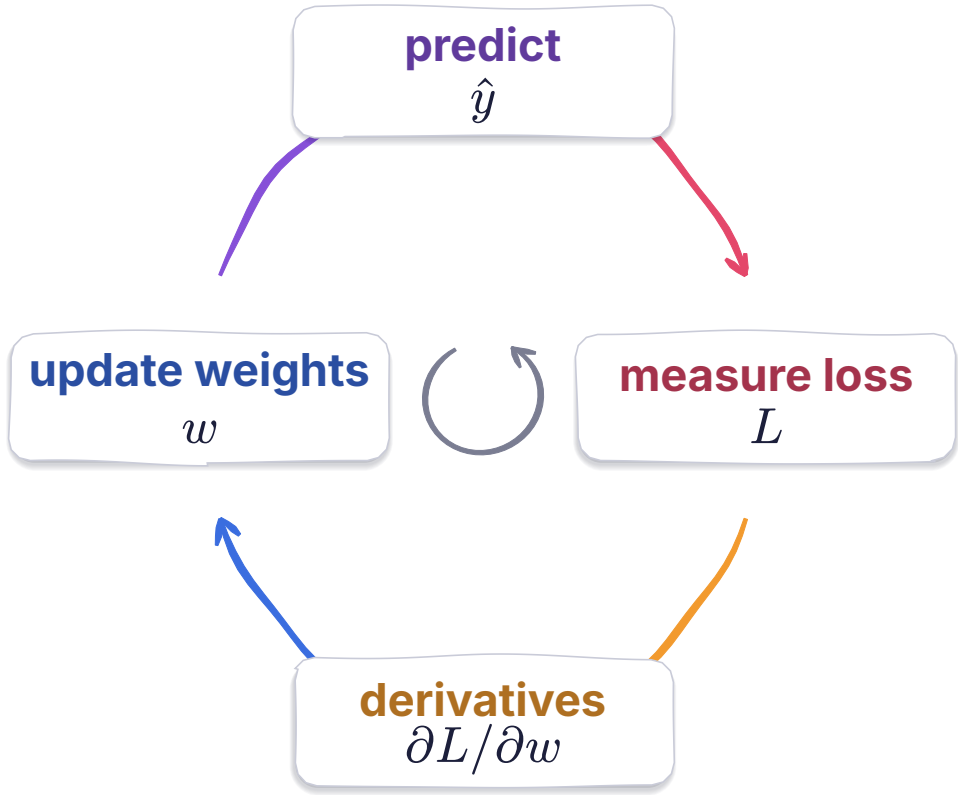
$$w \leftarrow w - \eta \cdot \partial L/\partial w$$

η, the learning rate: a small positive number

a simple loss curve, for illustration

Repeat: predict, measure, adjust



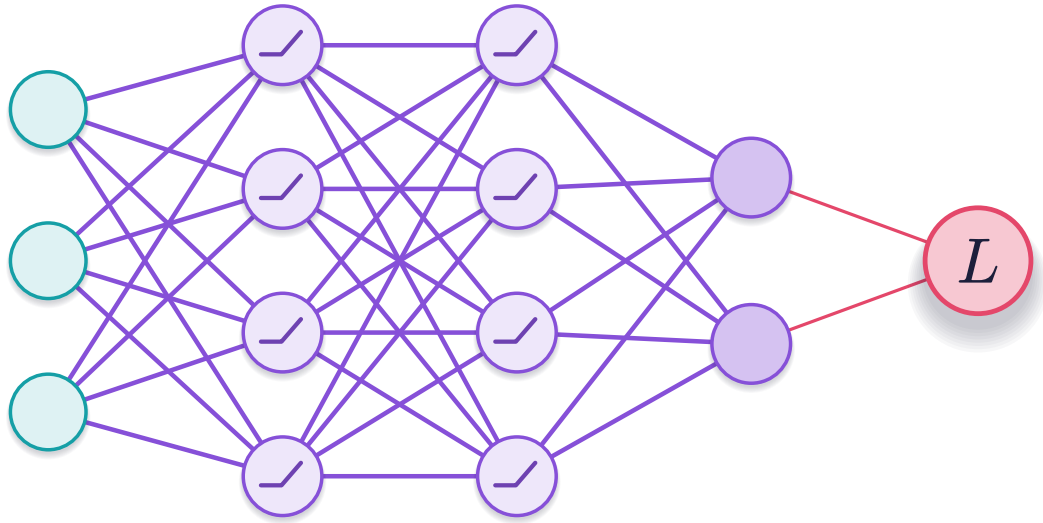


many examples, many passes over the training set

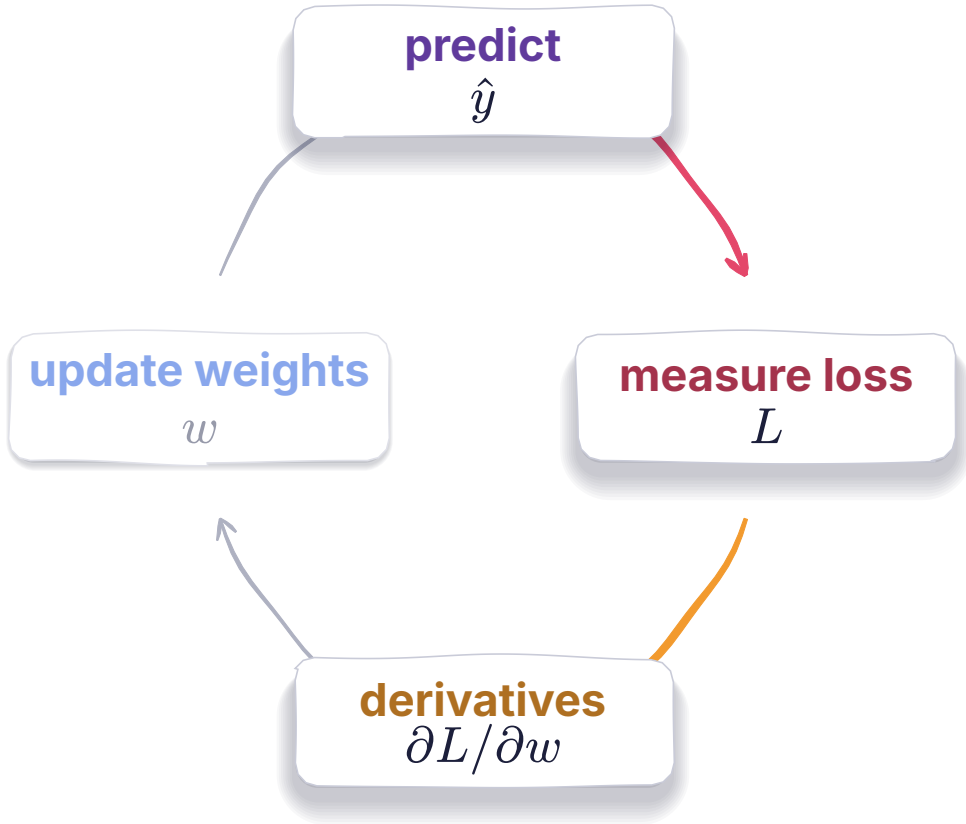
Repeat: predict, measure, adjust

backpropagation

forward pass



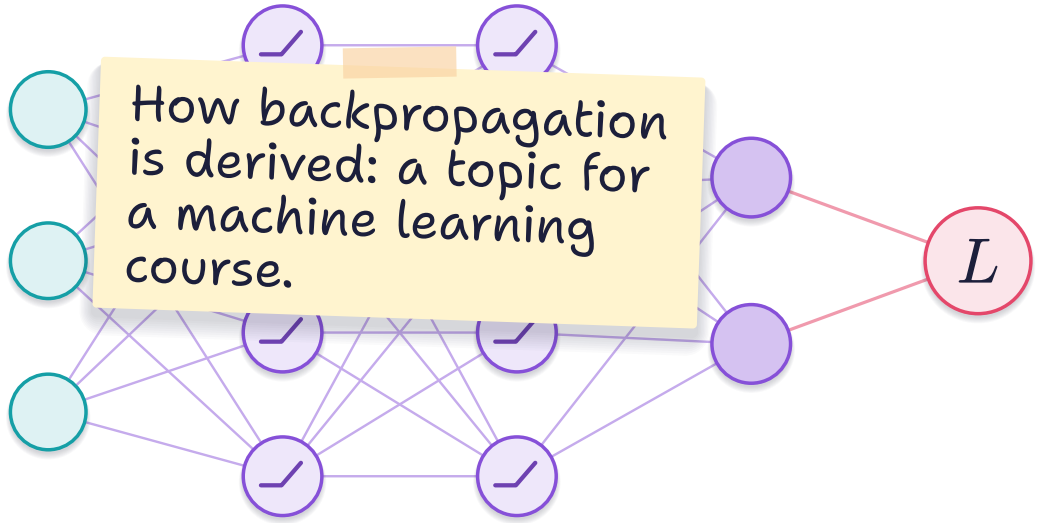
backward pass



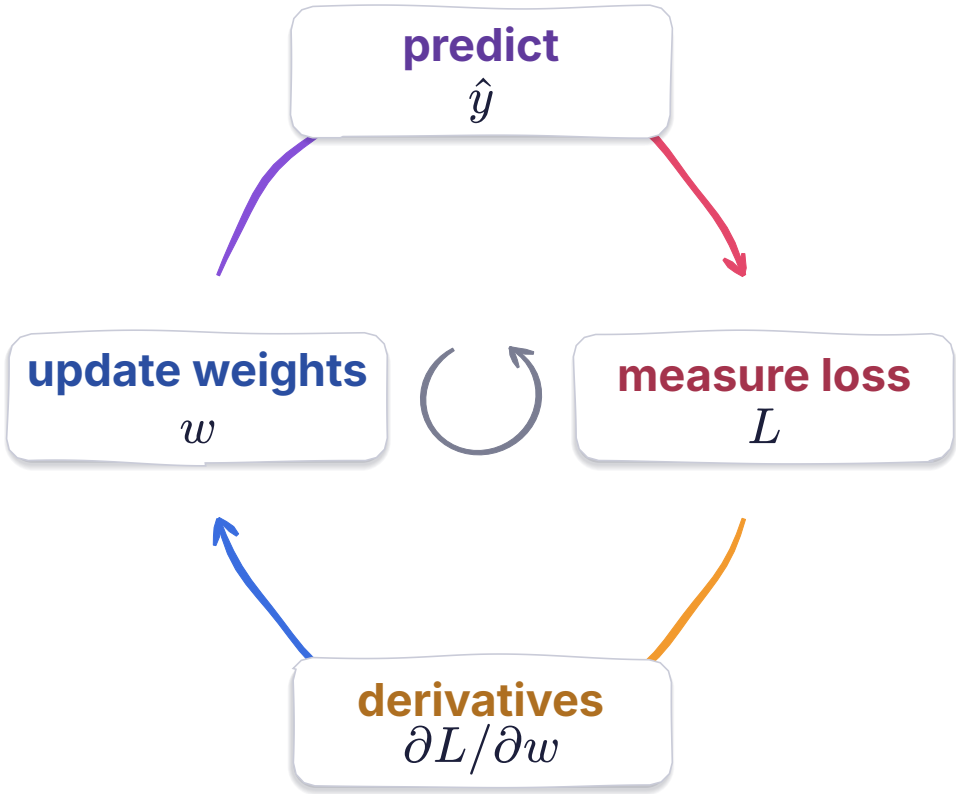
Repeat: predict, measure, adjust

backpropagation

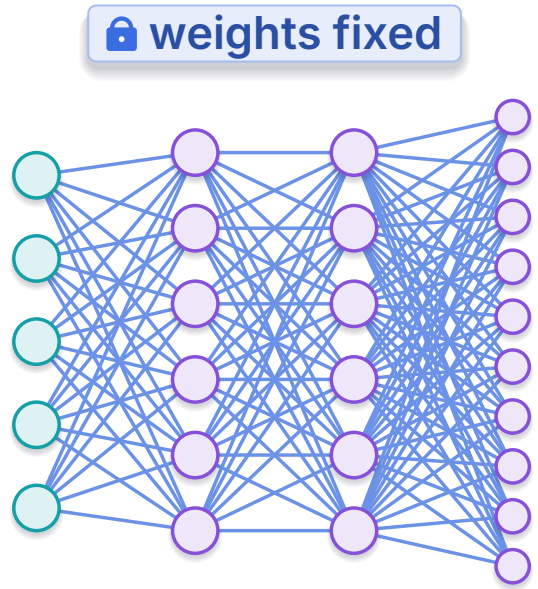
forward pass



backward pass



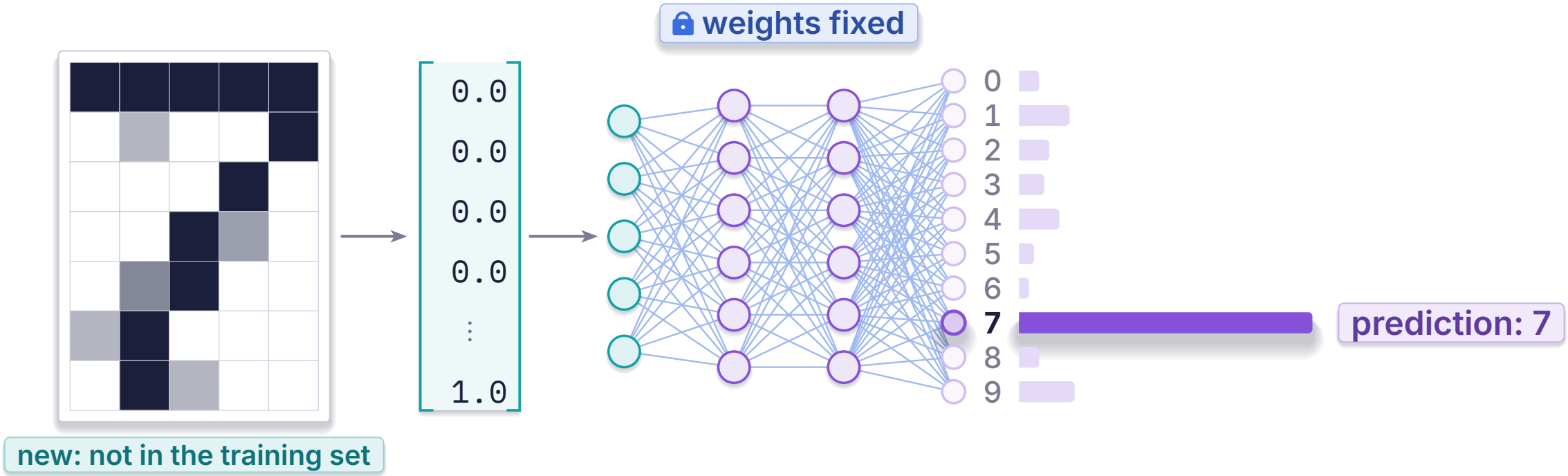
Inference: fixed weights, new inputs



making a prediction
does not change them

Inference: fixed weights, new inputs

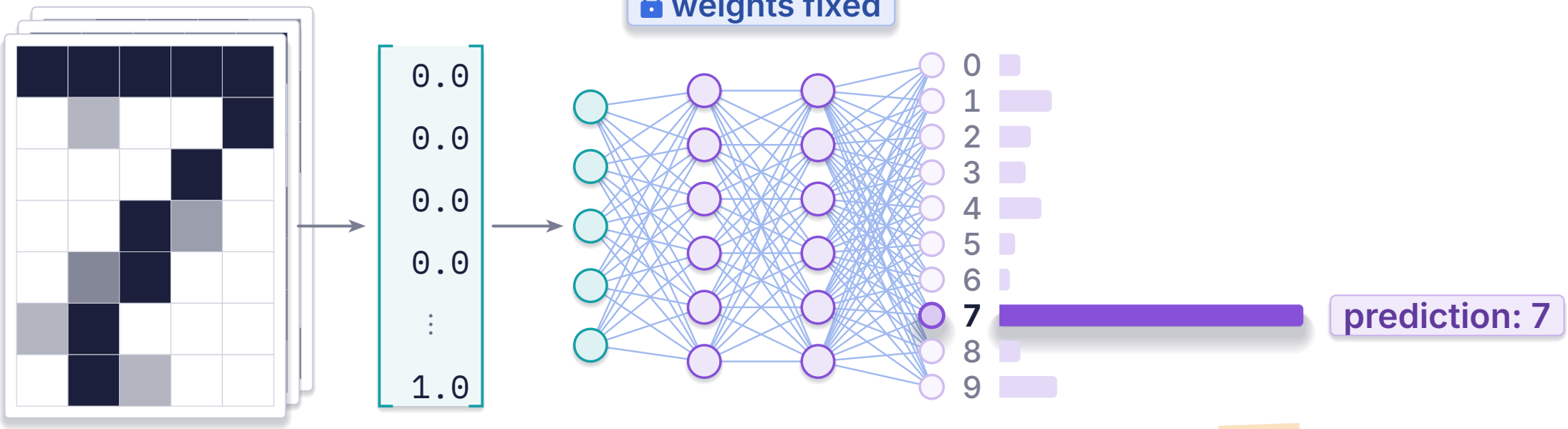
inference one forward pass



scores are illustrative

Inference: fixed weights, new inputs

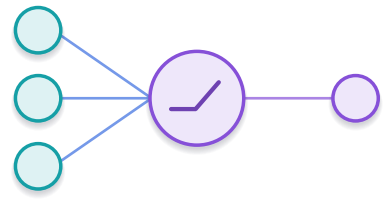
one fixed function:
the same weights,
image after image



A language model too: ordinary inference uses fixed weights. New information goes into its context, not its weights.

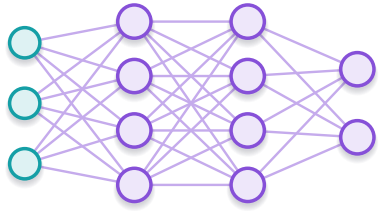
scores are illustrative

From one neuron to a trained network



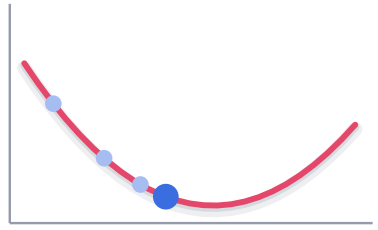
A neuron

A weighted sum and a bias, then a non-linear activation.



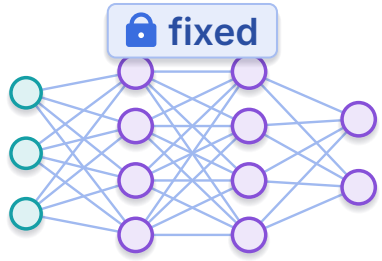
A network

Neurons in layers. Information flows one way: feed-forward.



Training

The loss measures the error. Backpropagation computes derivatives. Gradient descent updates the weights.



Inference

The learned weights stay fixed. One forward pass for each new input.

Language models use the same training ideas, in larger and more complex networks.