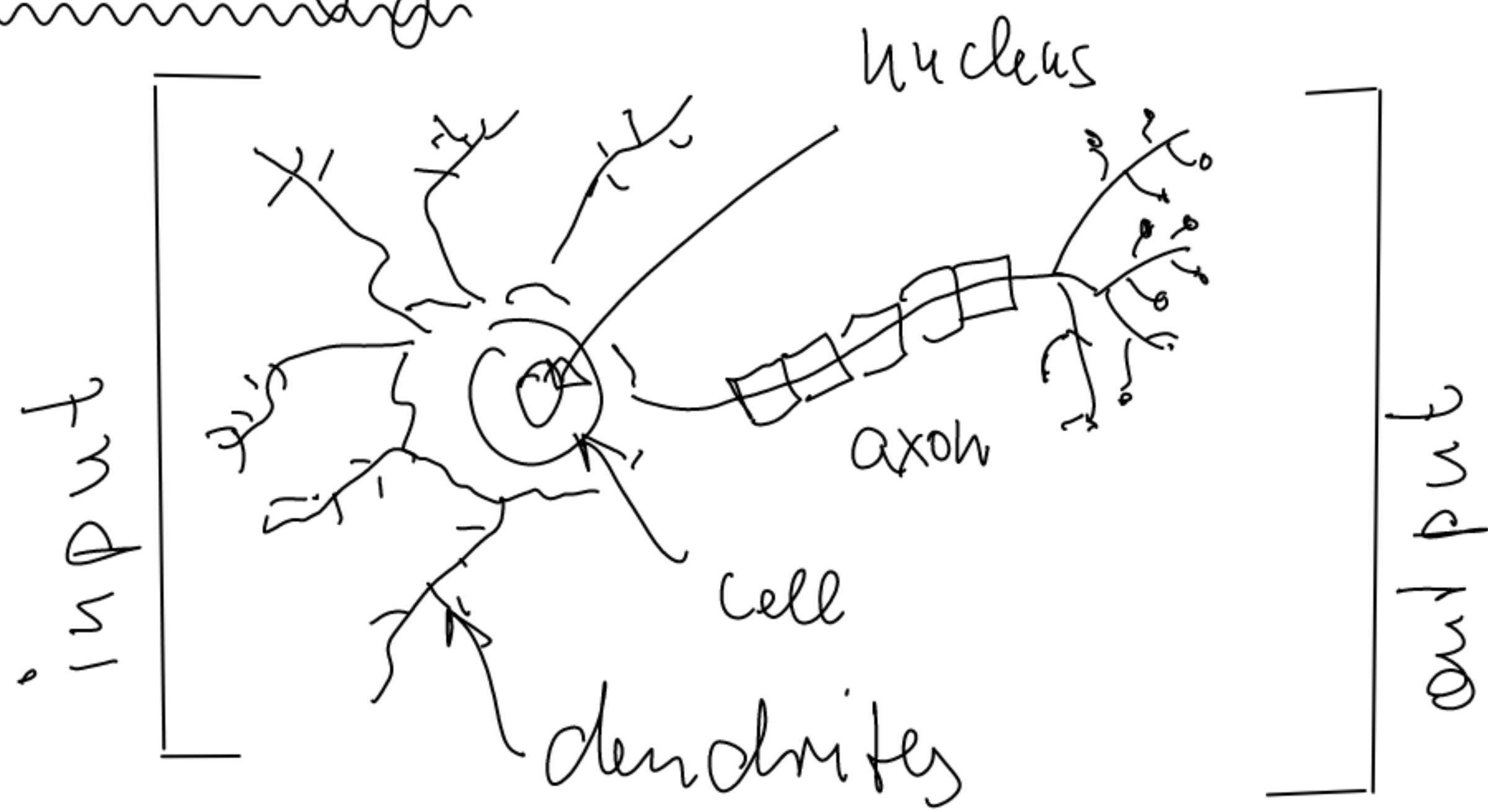# 15. Neural networks
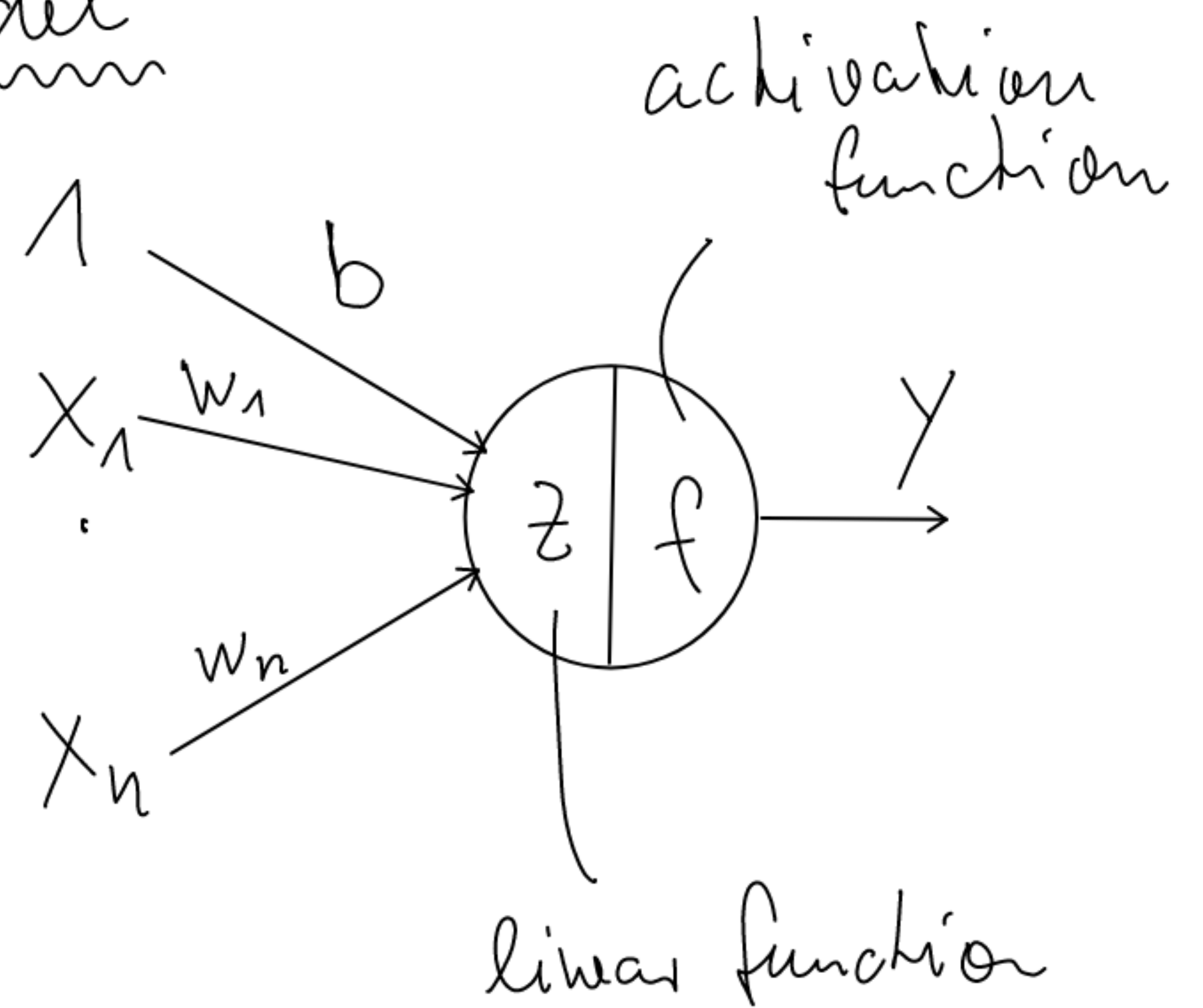
Simulate how our brain works on computer.
composed out of 10^{10} neurons, each connected to
$\sim 10^4$ others

15.1. Neuron
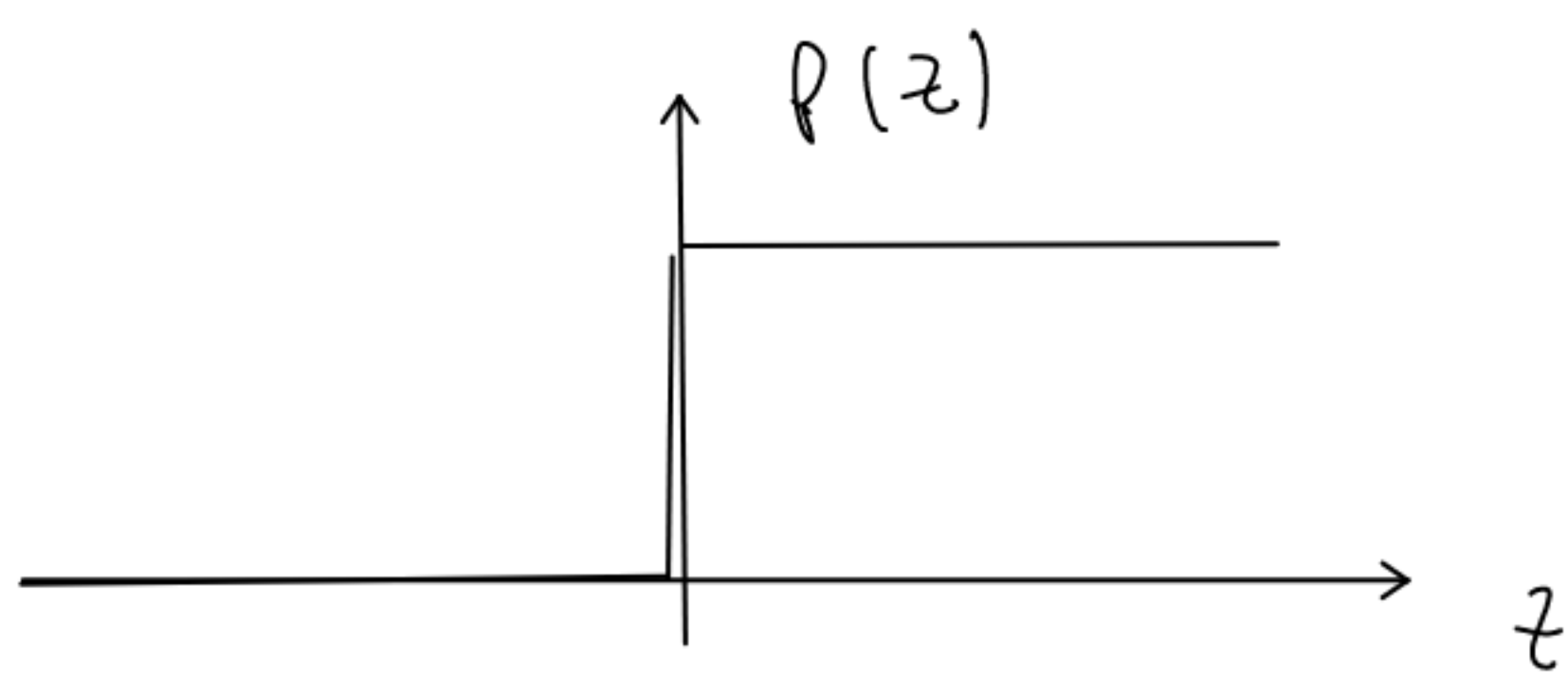
biology

nucleus

axon

cell

dendrites

input

output

model

activation function

1

b

X_1 W_1

z f y

W_n

X_n

linear function

for sufficient input neuros fire (binary operation)

this is simulated by:

weights   input   bias

1) input function $z(X_1,...,X_n) = W_1 X_1 + ... + W_n X_n + b$

2) activation function $y = f(z)$, examples are

step function

$\tanh(z) = (e^z - e^{-z})/(e^z + e^{-z})$
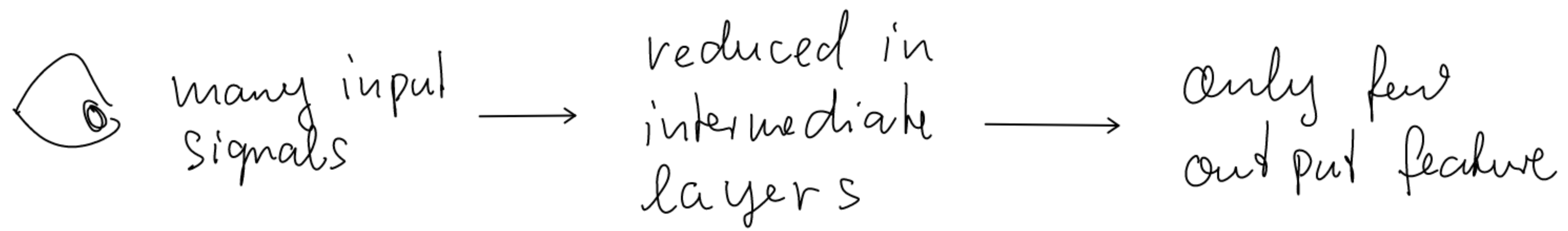
$f(z)$

z

$f(z)$

z
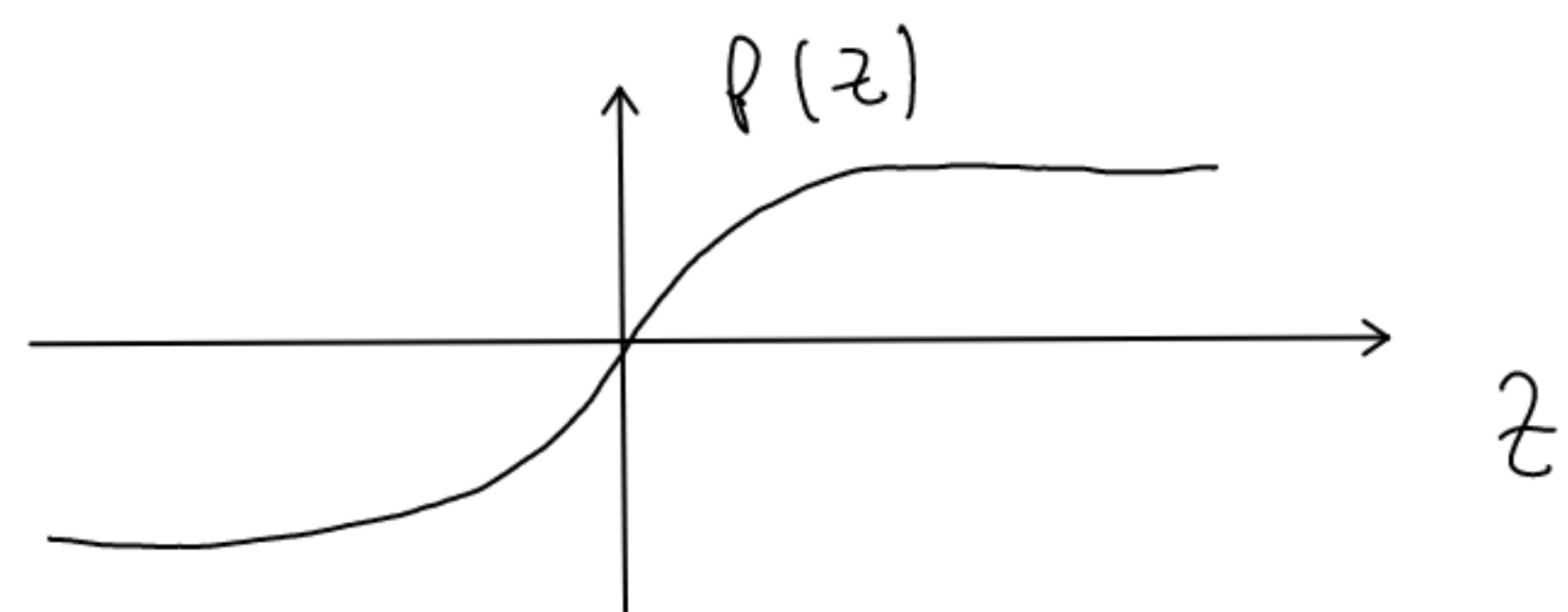
but there are many more

15.2. Network architecture

Question: How to connect the neurons?
Convolutional Neural Network CNN

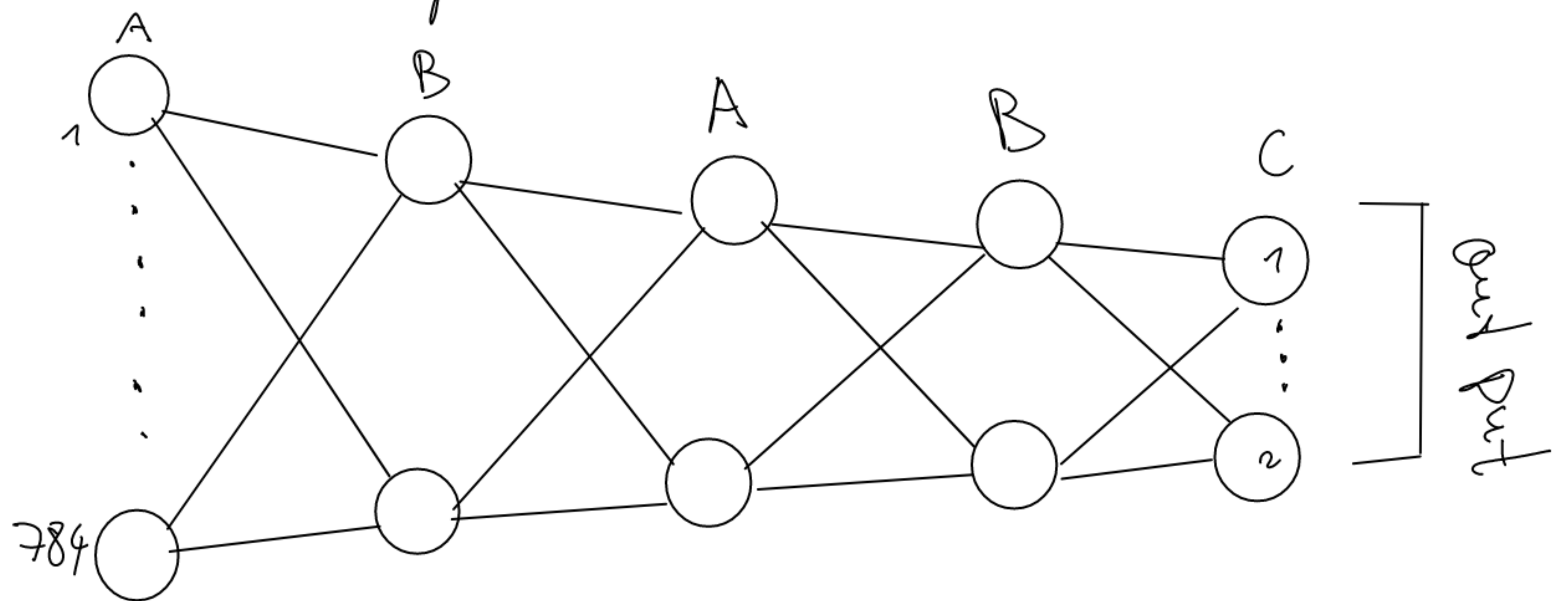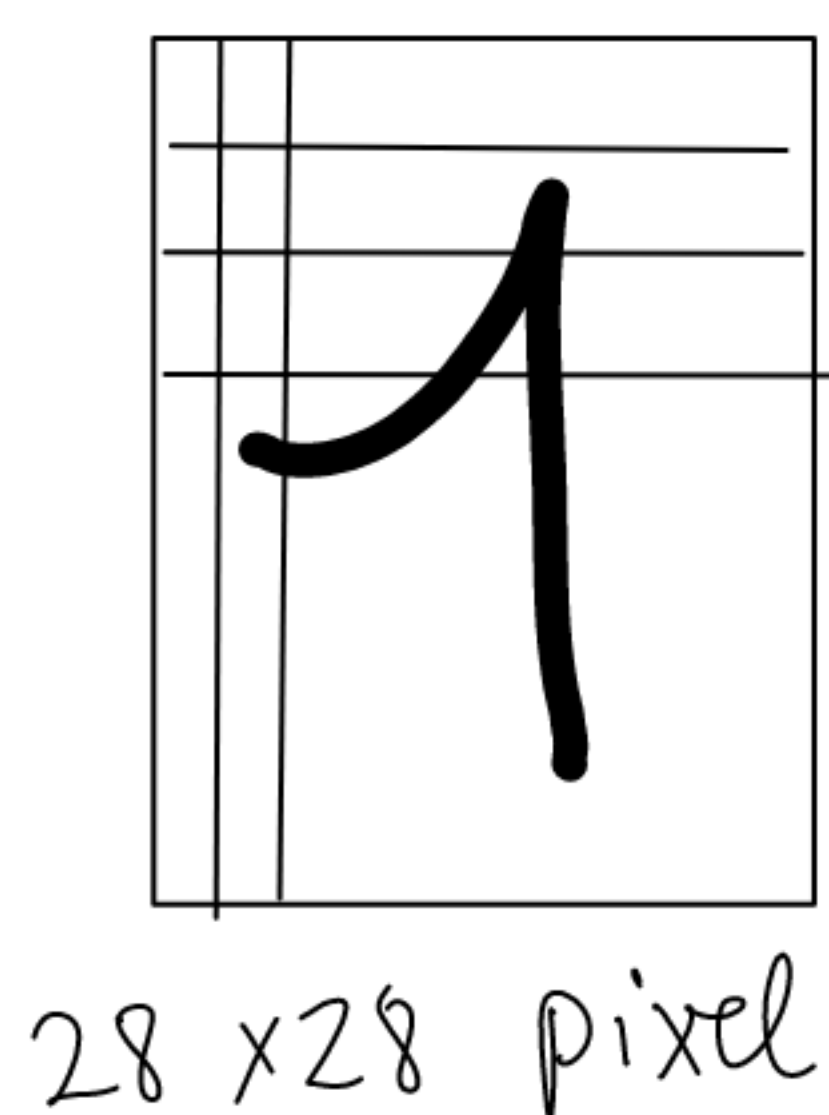- inspired by the visual cortex
  → good for pattern recognition
- hirachical structure

many input signals ⟶ reduced in intermediate layers ⟶ only few output feature

- only "forward connections to neighbors

<u>example</u> number recognition

28 x28 pixel   784

A) 2d <u>convolution</u> layer (3,3)

$$z_{ij} = \sum_{m=1}^{3} \sum_{n=1}^{3} x_{i+m, j+n} \cdot w_{mn} \quad \sim 9 \text{ weights (size of kernel)}$$
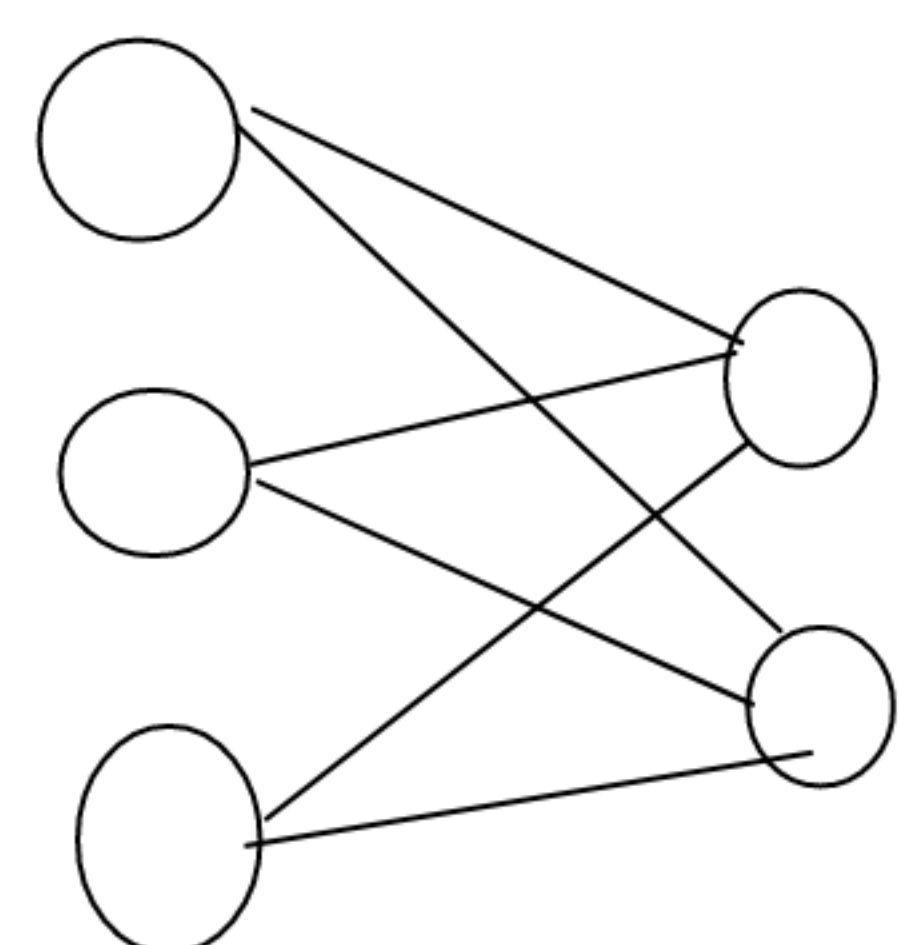
activation function $relu(x) = max(0, x)$

B) Max <u>pooling</u> layer (2,2)
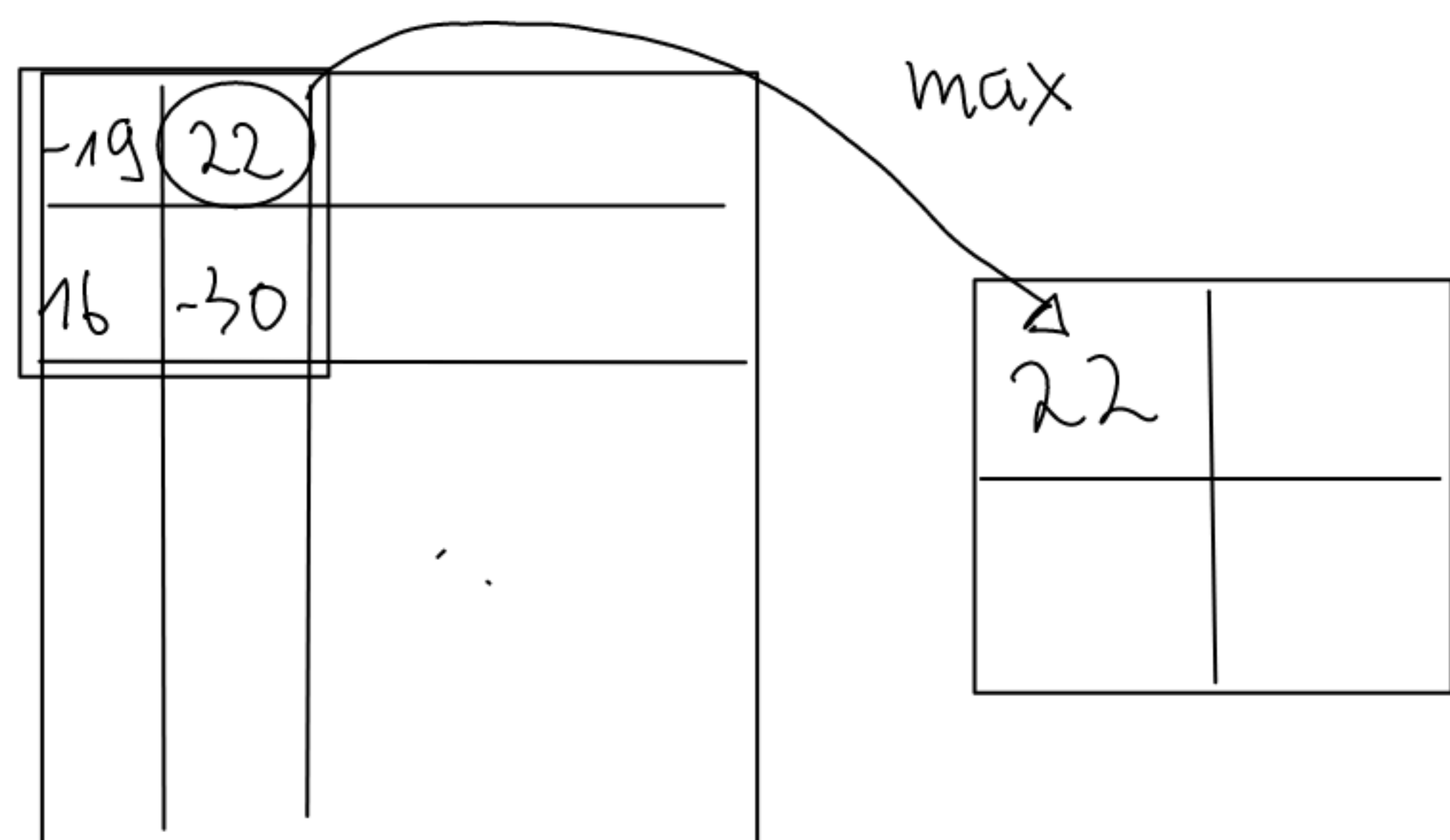
max   could also average or interpolat

22

c) dense layer is fully connected

15.3. Training

Question: How to fix all the weights of the neurons?

1) We need data to "learn" from.

MNIST data set: $6 \cdot 10^4$ training } images
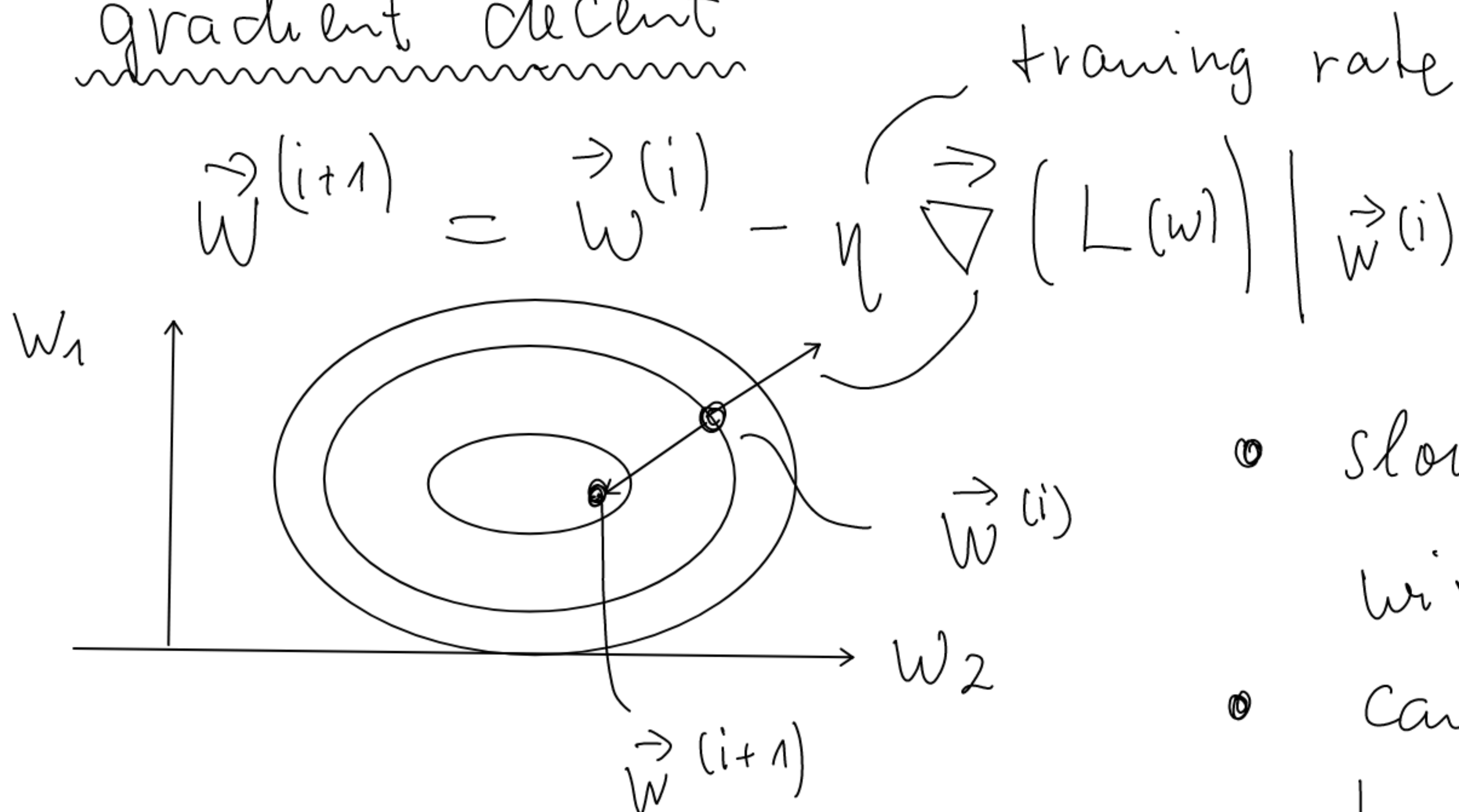10^4 testing }

28x28 black white ⇝ anti-aliased → gray scale

2) loss function indicates how well the network performs, i.e. Cross Entropy Loss

$$L = - \sum_{K=1}^{10} y_K \cdot \log(p_K)$$

normalized output $\left(\sum_{K=1}^{10} y_K = 1 \right)$

probability of feature in data set

3) optimization algorithm, to tune the weights such that the loss function is minimized, i.e. gradient decent

$$\vec{W}^{(i+1)} = \vec{W}^{(i)} - \eta \vec{\nabla}(L(w)) \big|_{\vec{W}^{(i)}}$$

training rate

W_1

$\vec{W}^{(i)}$

W_2

$\vec{W}^{(i+1)}$

- slow for functions with long "valleys"
- can't distinguish local vs. global min.

better option Adam Optimizer

complete training algorithm:

1. initialize weights randomly
2. train on a batch of n samples ⎫
3. check accuracy on test data ⎭ repeat for m epochs

15.4. Libraries

- Pytorch
- tensor flow ← we use this one