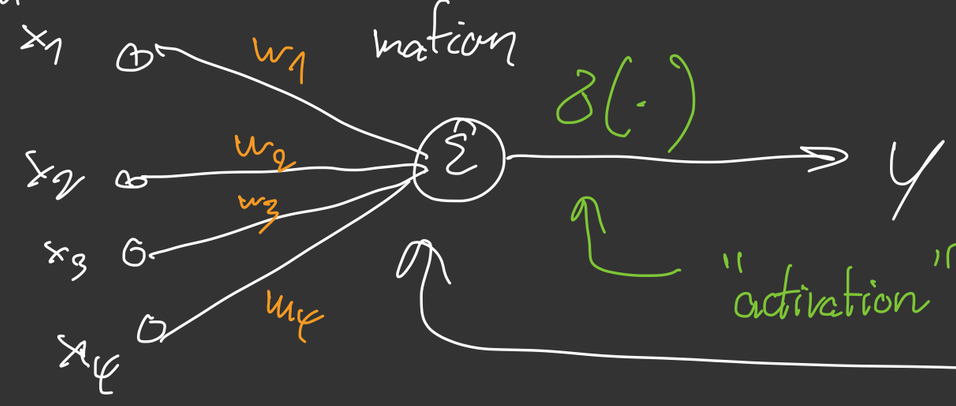
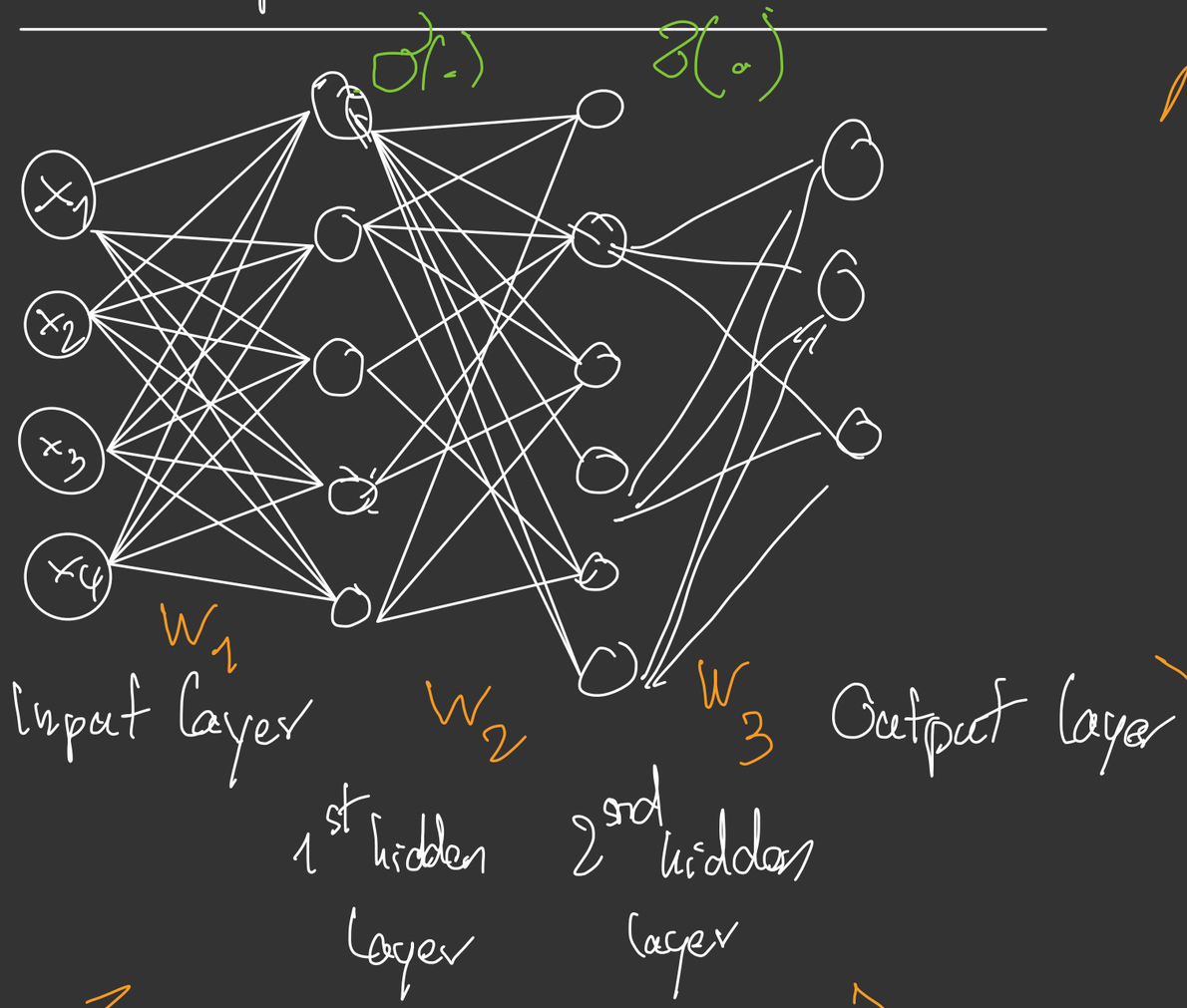
Artificial Neural Networks

(Selective) Life History: [McCulloch, Pitts '43]: Mathematical model for neurons in brain



- 1950's: Perceptron; 1st numerical scheme to use collection of) to do classification
- 1980's: "Deeper" networks, successful training via backpropagation (training algorithm)
- Since ~2007: ▷ Efficient training ($\hat{=}$ determination of weights (w_i) from training data set) becomes possible for deeper networks → regularization
  - ▷ Advances in computing technology: Use GPUs
  - ▷ Outperformance of traditional learning methods (such as support vector machines)

A template for neural networks



A typical hypothesis space $\mathcal{F}$ associated to an ANN:

$$\mathcal{F}_L = \{ h : X \to Y : h(x) = \delta(W_L(\sigma(W_{L-1}(\sigma(\dots \sigma(W_1(x))\dots)),$$

σ non-linear, acts componentwise

$W_\ell : \mathbb{R}^{N_{\ell-1}} \to \mathbb{R}^{N_\ell}$ (affine) Linear

$\forall \ell = 1, \dots, L \}$

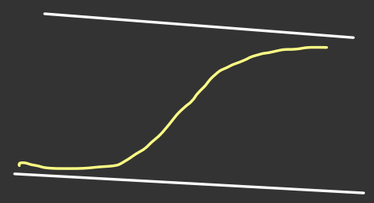
N_0 : dimension of input layer

N_i : —"— i-th hidden layer

N_L : —"— of output layer

Example:

- (Multiclass) Logistic Regression : 1-layer NN ($L=1$),



$\sigma(\cdot) = \mathrm{softmax}(\cdot)$

"sigmoidal"

- If $\downarrow$ with $L > 1$: Multilayer Perceptron

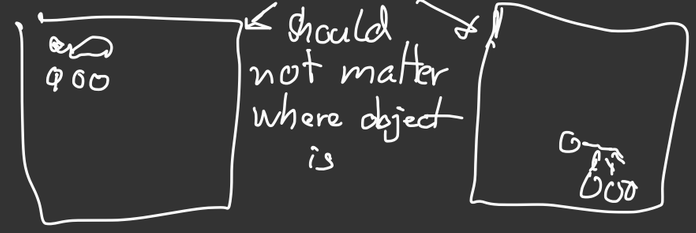
Thm: [Hornik '89, Cybenko '91]:

> Every measurable function can be approximated by a with at least $L=2$ layers (if wide enough).

Practical Q:

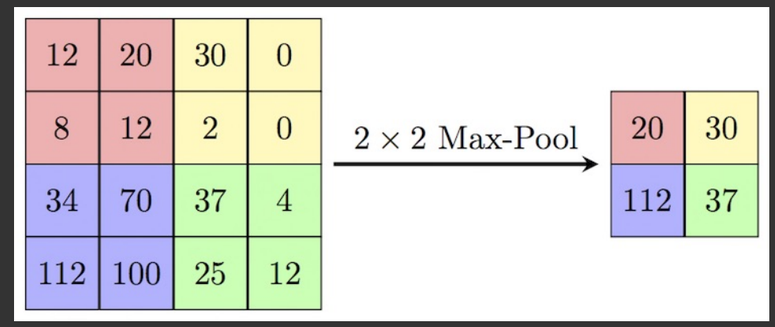
- How many layers L, how wide each layer?
- Choose which loss fcf?
- Which activation function?

- How to "train" the network (i.e., determine weights W_ℓ)
  - ↳ Some variant of stochastic gradient descent (SGD), implemented via backpropagation (i.e., a very smart implementation of the chain rule from calculus)



Modifications / Variants:

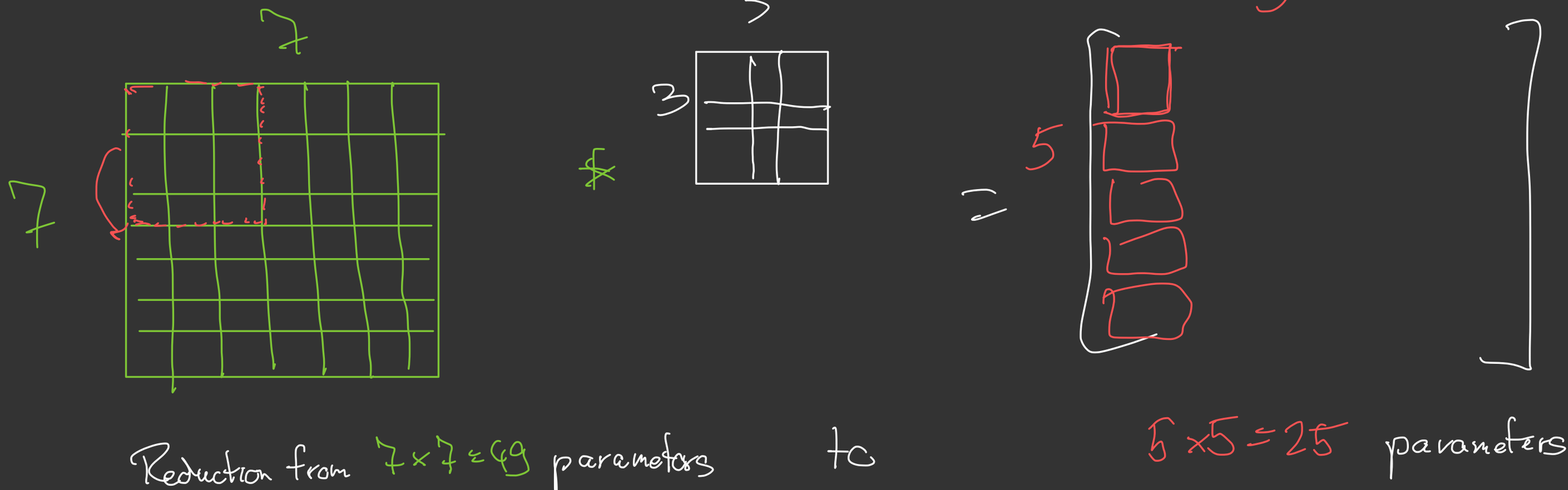
- Constrain weights to have certain properties: Convolutional Neural Network (CNN)
  good for imaging problems (enforces spatial invariance)
- Fewer connections: → often less overfitting
- (Max / Average) Pooling: Reduces nr. of parameters, avoids overfitting



- Dropout: Drops (at random) connections
  between layers in training phase
- Batch normalization: Normalizes input of a layer,
  → faster training, better generalization

Epoch: One pass through entire training data set.

Example : Convolutional 3×3 filter



Reduction from $7 \times 7 = 49$ parameters to $5 \times 5 = 25$ parameters