

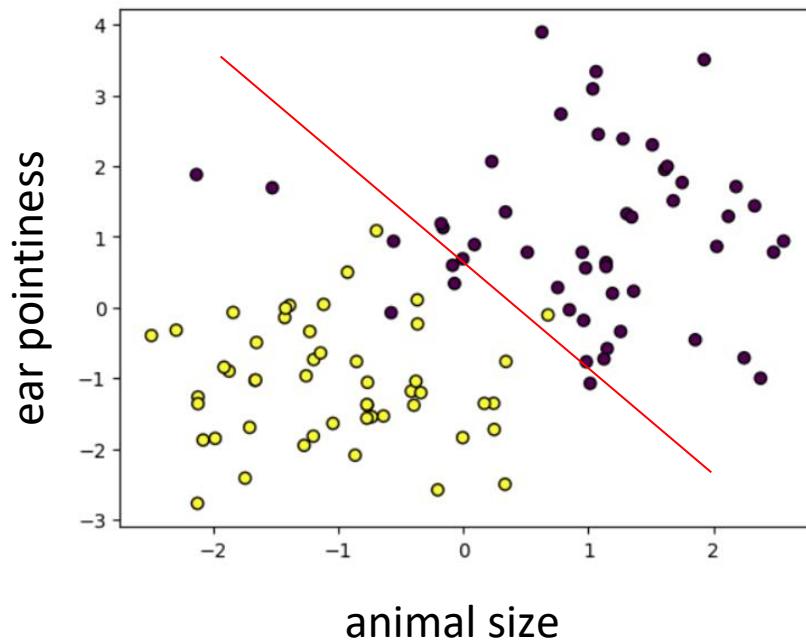
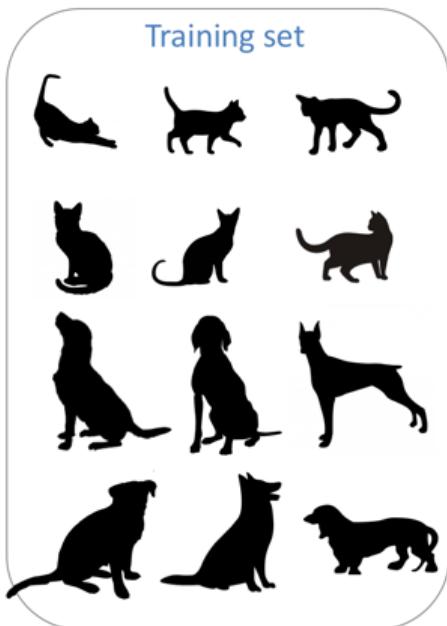
Tutorial 3: Supervised learning and feedforward neural networks

Plan

- Perceptron.
 - Multi-layer Perceptron.
 - Convolutional Neural Network
and digit recognition
 - Analogy between convolutional
networks and visual cortex

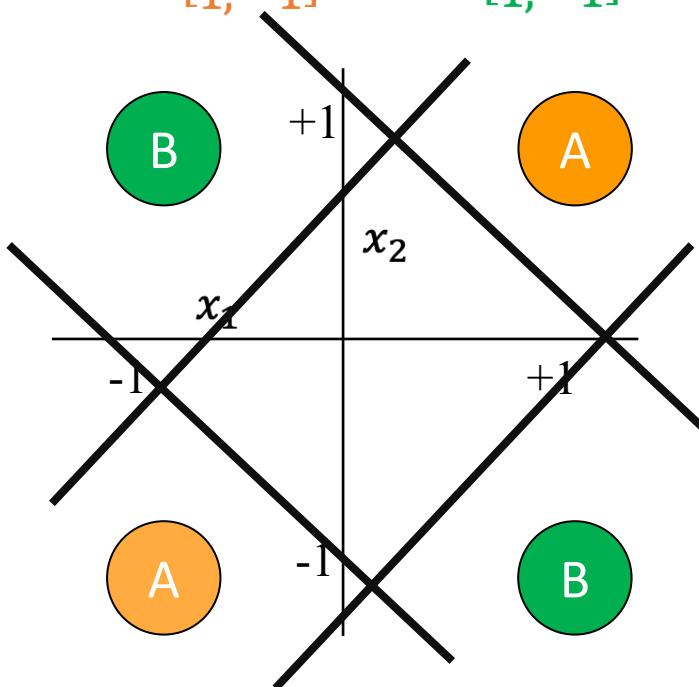
2 2 2 2 2 2 2 2 2
3 3 3 3 3 3 3 3 3
4 4 4 4 4 4 4 4 4
5 5 5 5 5 5 5 5 5
6 6 6 6 6 6 6 6 6
7 7 7 7 7 7 7 7 7

Linearly separable data

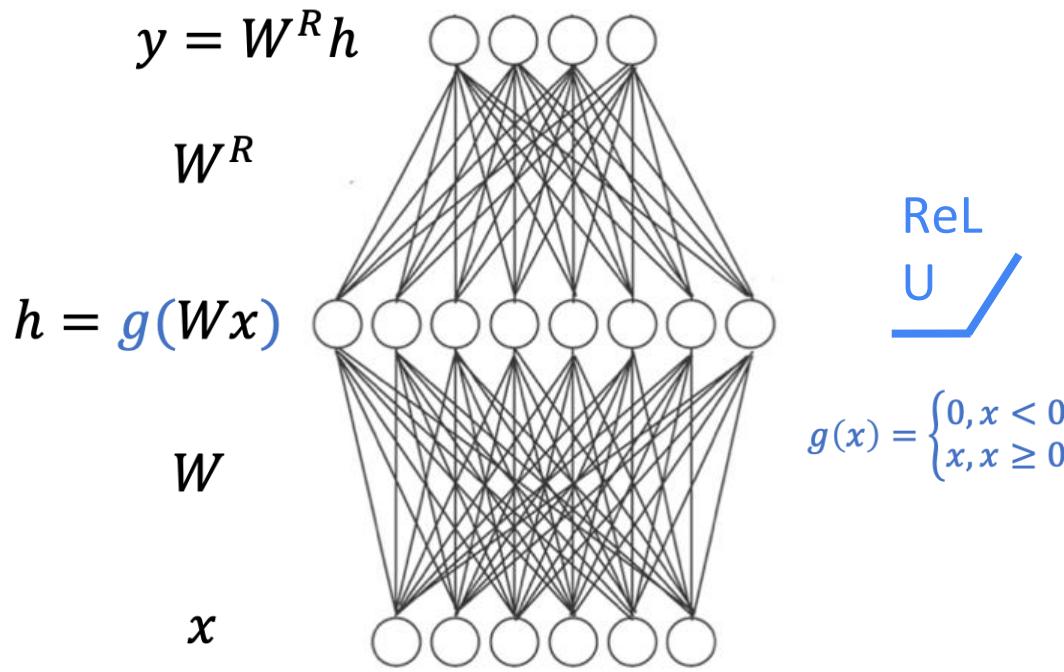


The XOR problem

$$x^A = \begin{bmatrix} 1, -1 \\ 1, -1 \end{bmatrix} \quad x^B = \begin{bmatrix} -1, 1 \\ 1, -1 \end{bmatrix}$$



Multi-layer perceptron



ReLU
U

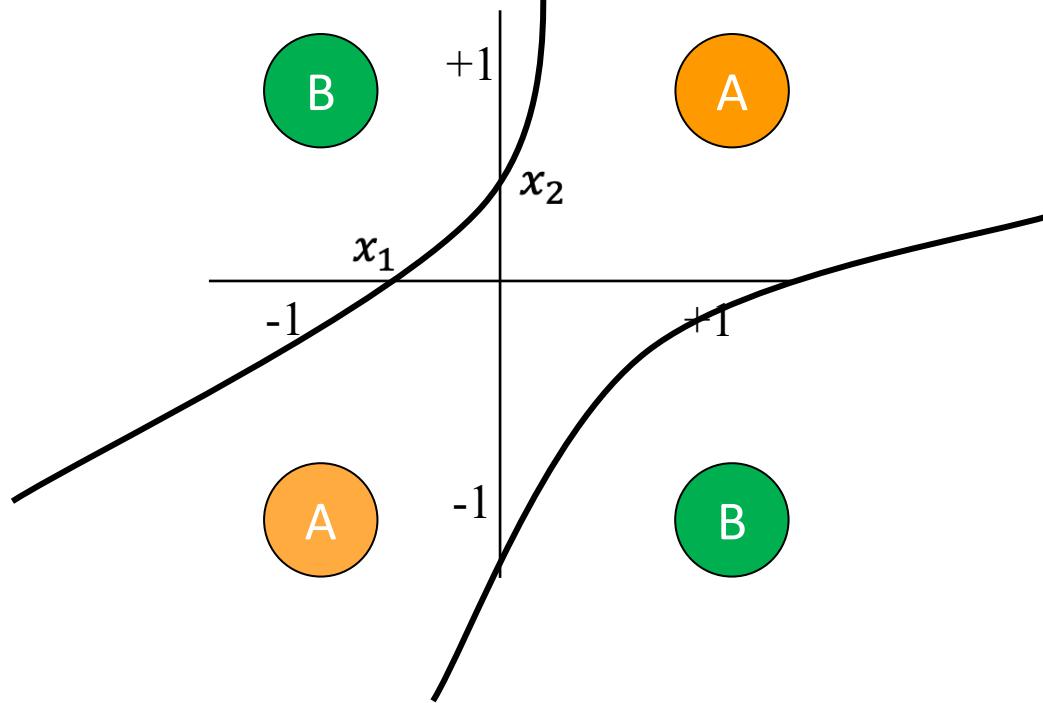
$$g(x) = \begin{cases} 0, & x < 0 \\ x, & x \geq 0 \end{cases}$$

deep **non**linear network

The XOR problem

$$x^A = \begin{bmatrix} 1, -1 \\ 1, -1 \end{bmatrix}$$

$$x^B = \begin{bmatrix} -1, 1 \\ 1, -1 \end{bmatrix}$$

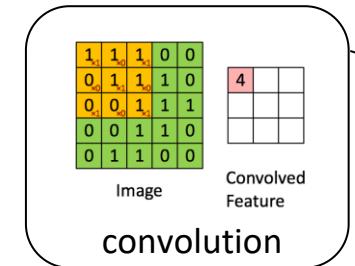
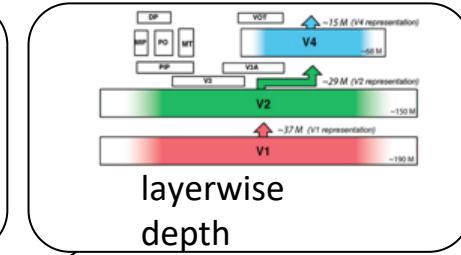
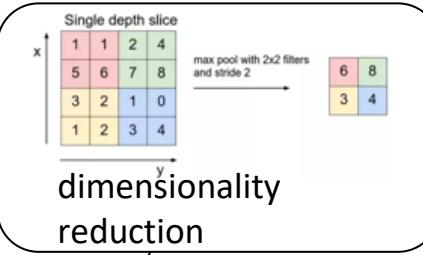




Convolutional neural networks

$$b_i = \frac{a_i}{(k + \alpha \cdot \sum a_j^2)^\beta}$$

divisive
normalisation



Conv_1
Convolution
(5 x 5) kernel
valid padding

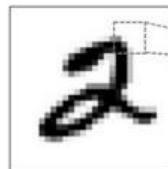
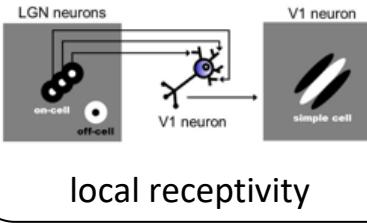
Max-Pooling
(2 x 2)

Conv_2
Convolution
(5 x 5) kernel
valid padding

Max-Pooling
(2 x 2)

fc_3
Fully-Connected
Neural Network
ReLU activation

fc_4
Fully-Connected
Neural Network



INPUT
(28 x 28 x 1)

n1 channels
(24 x 24 x n1)

n1 channels
(12 x 12 x n1)

n2 channels
(8 x 8 x n2)

n2 channels
(4 x 4 x n2)

0
1
2
⋮
9

OUTPUT

Flattened

(with
dropout)

PyTorch

```
import torch, torch.nn as nn

x = torch.tensor([[1.0], [2.0]])
y = torch.tensor([[3.0], [5.0]])

model = nn.Linear(1, 1)
```