数字媒体技术基础

Meng Yang

www.smartllv.com

SUN YAT-SEN University

机器智能与先进计算教育部重点实验室

智能视觉语言学习研究组

Course Outline

- 第11章 智能新媒体信息表示基础
- 11.1 深度神经网络基础
  - 11.1.1 全连接神经网络
  - 11.1.2 卷积神经网络
  - 11.1.3 自编码网络
  - 11.1.4 生成对抗网络
- 11.2 智能新媒体的信息表示学习
  - 11.2.1 图像预训练学习
  - 11.2.2 自然语言预训练学习

目 录

第一部分

11.1 深度神经网络基础

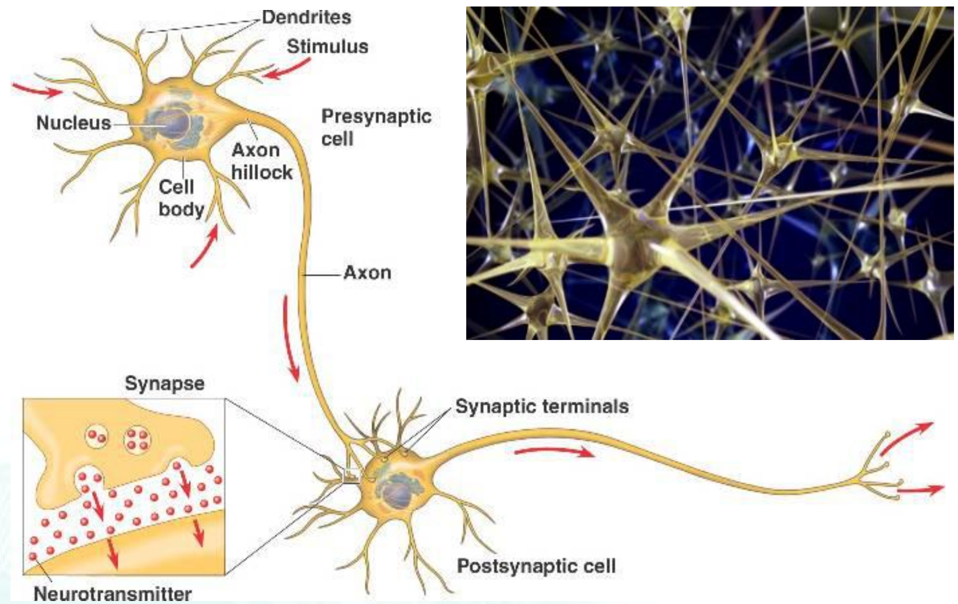
人脑神经网络

- 一些在童年时期因癫痫接受了大脑半球切除术的病人，只剩一半的大脑是否能正常发挥功能？

问题？

11.1 深度神经网络基础

引言



2019年，加州理工学院开展的一项新研究发现，一些在童年时期因癫痫接受了大脑半球切除术的病人，只剩一半的大脑仍然能正常发挥功能

实验过程

对实验组和对照组分别进行脑功能磁共振成像（fMRI）测试，并考察了大脑中负责视觉、运动、情绪和认知能力等日常功能的神经网络

实验结果

切除了一半大脑的受试者的脑功能与正常人无异，且这些患者大脑中各神经网络之间的联系和交流甚至比普通人更强，表明大脑似乎能够自行弥补缺失部分的功能

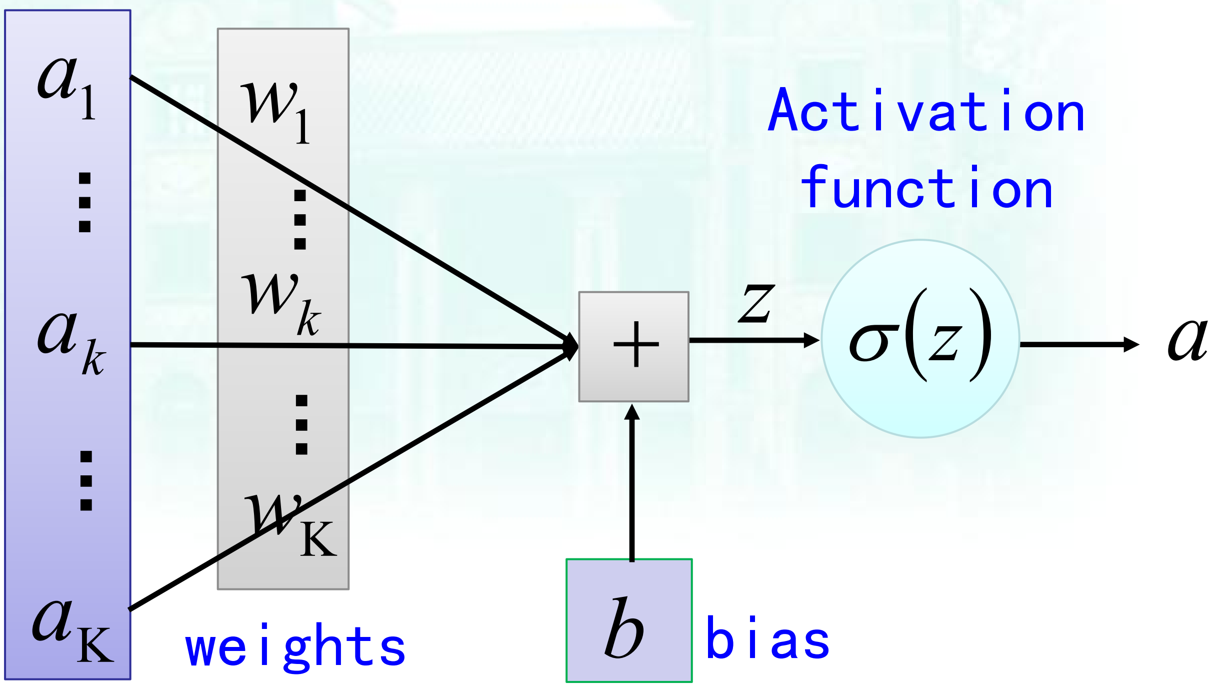
实验结论

大脑的“弹性”很强，会不断生成新的神经网络、在脑细胞之间建立起新的联系

Neural Network

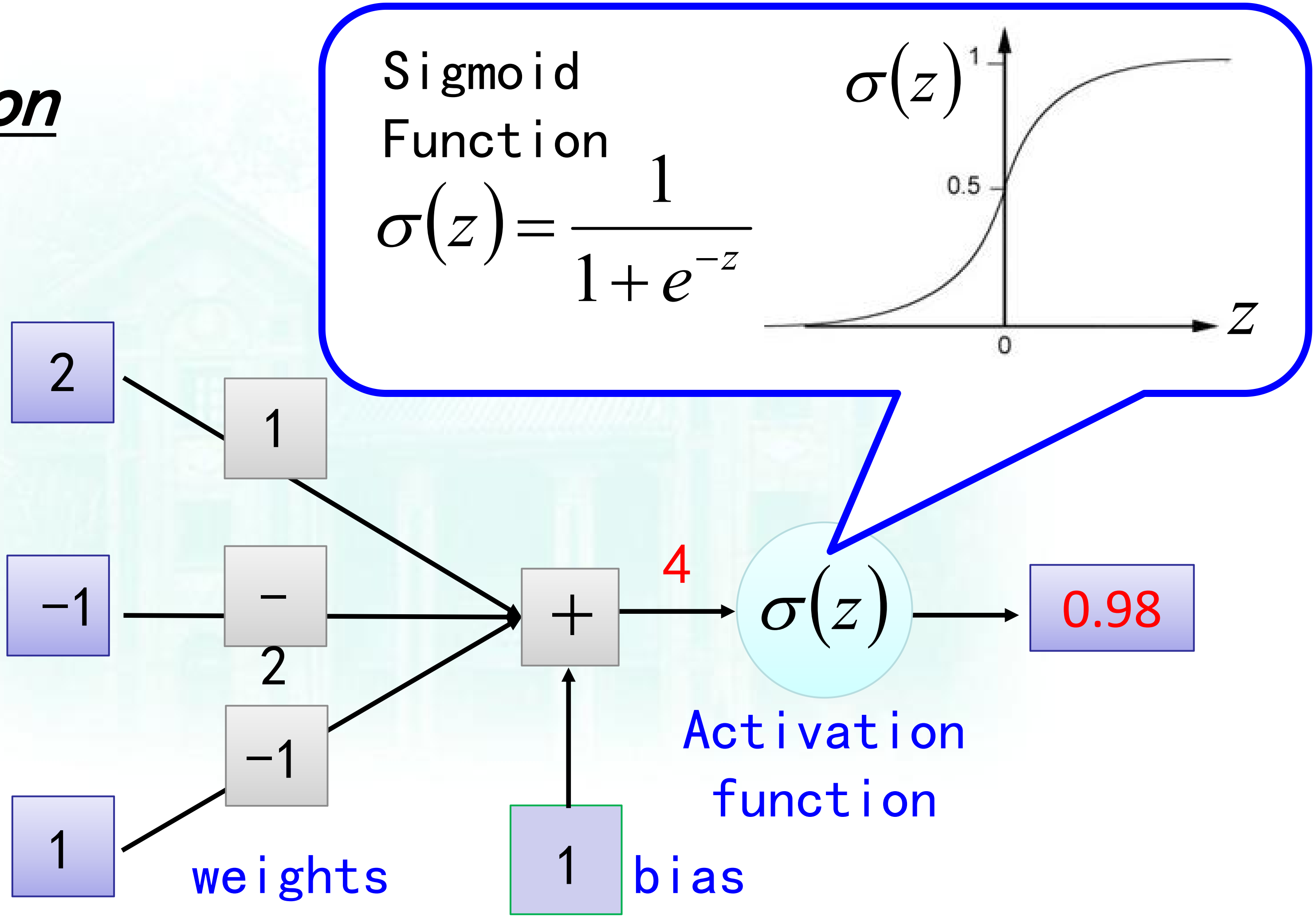
Neuron

$$z = a_1 w_1 + \cdots + a_k w_k + \cdots + a_K w_K + b$$



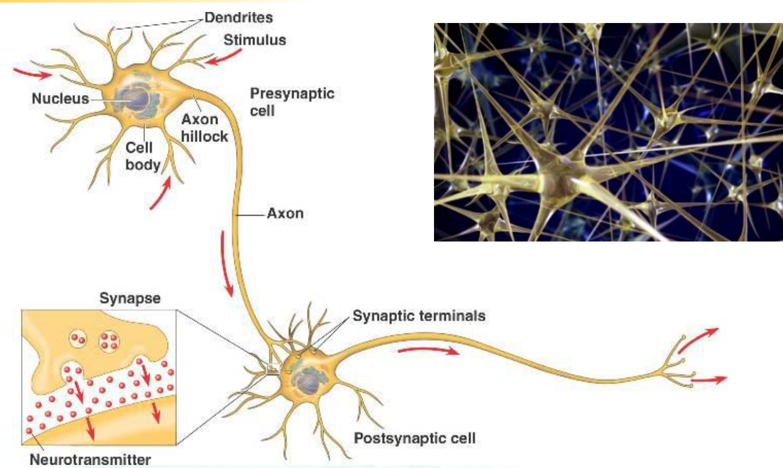
Neural Network

Neuron

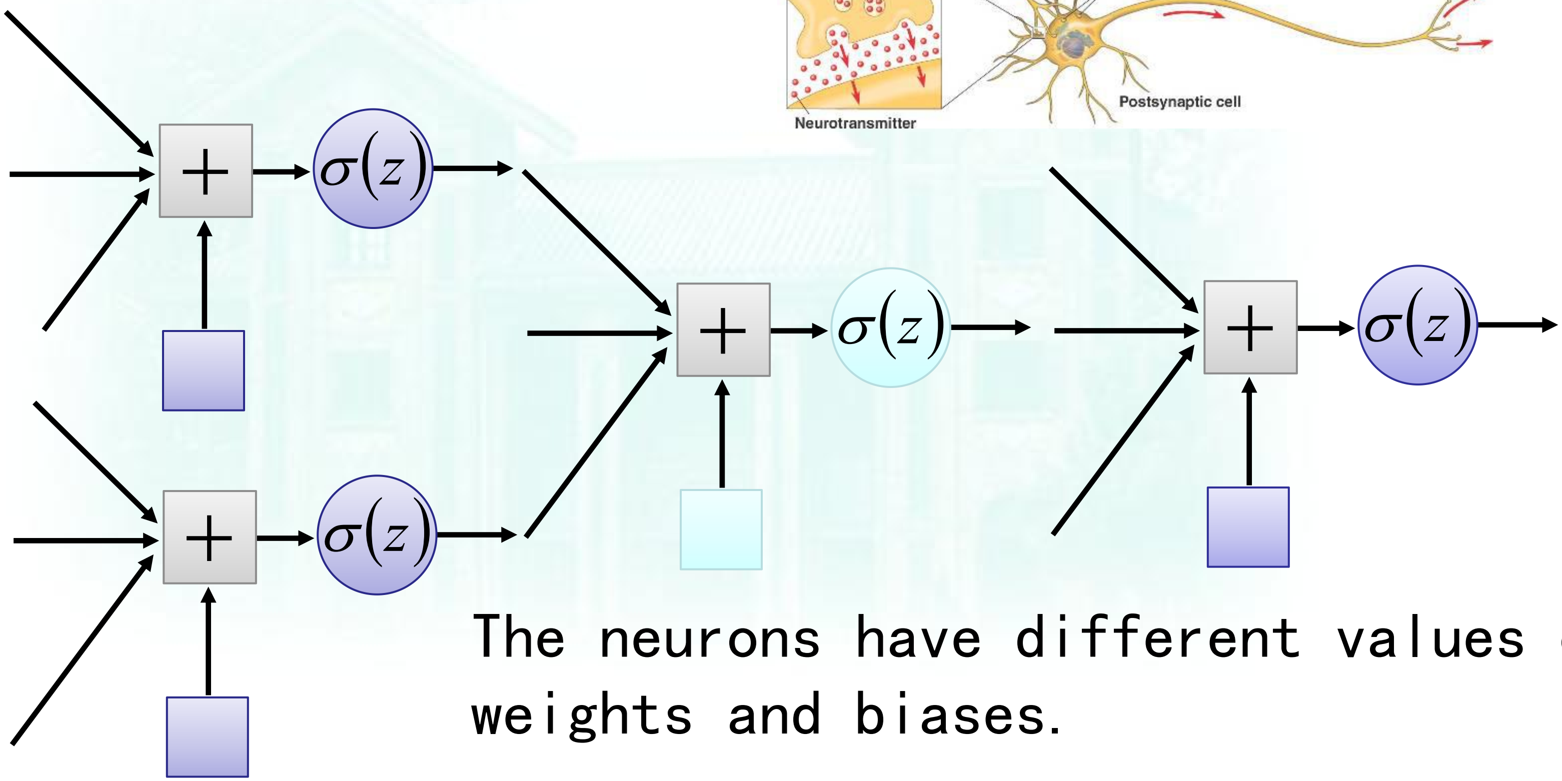


11.1.1 全连接神经网络

Neural Network



Different connections lead to different network structures

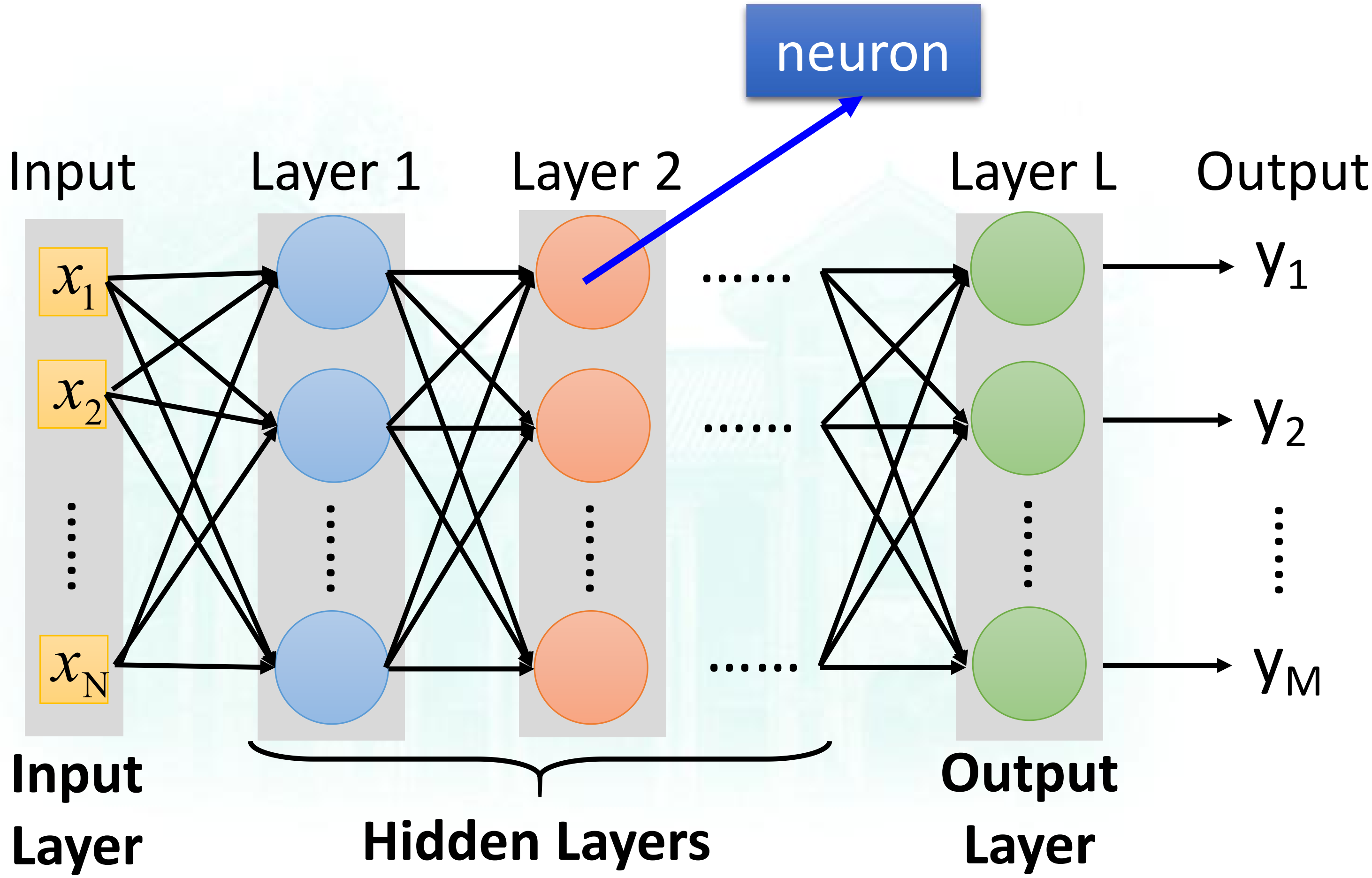


The neurons have different values of weights and biases.

Weights and biases are network parameters θ

11.1.1 全连接神经网络

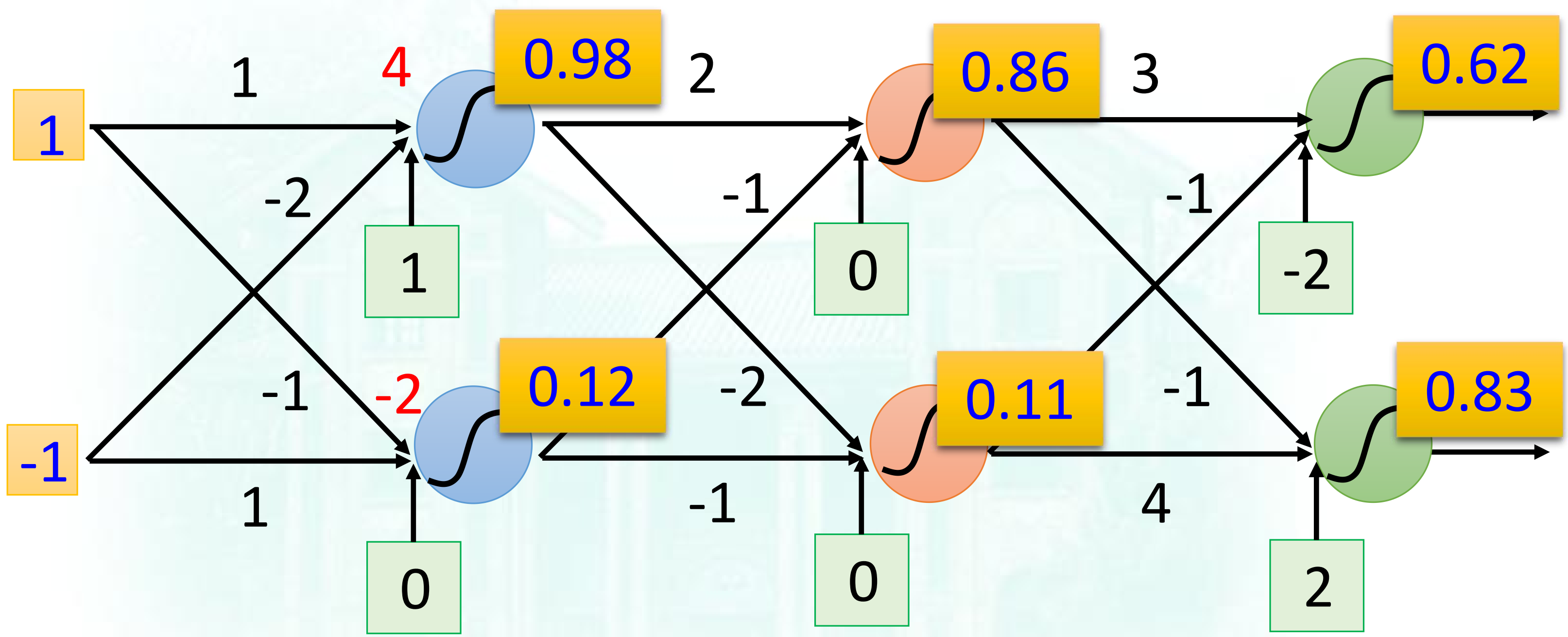
Fully Connect Feedforward Network



Deep means many hidden layers

11.1.1 全连接神经网络

Fully Connect Feedforward Network



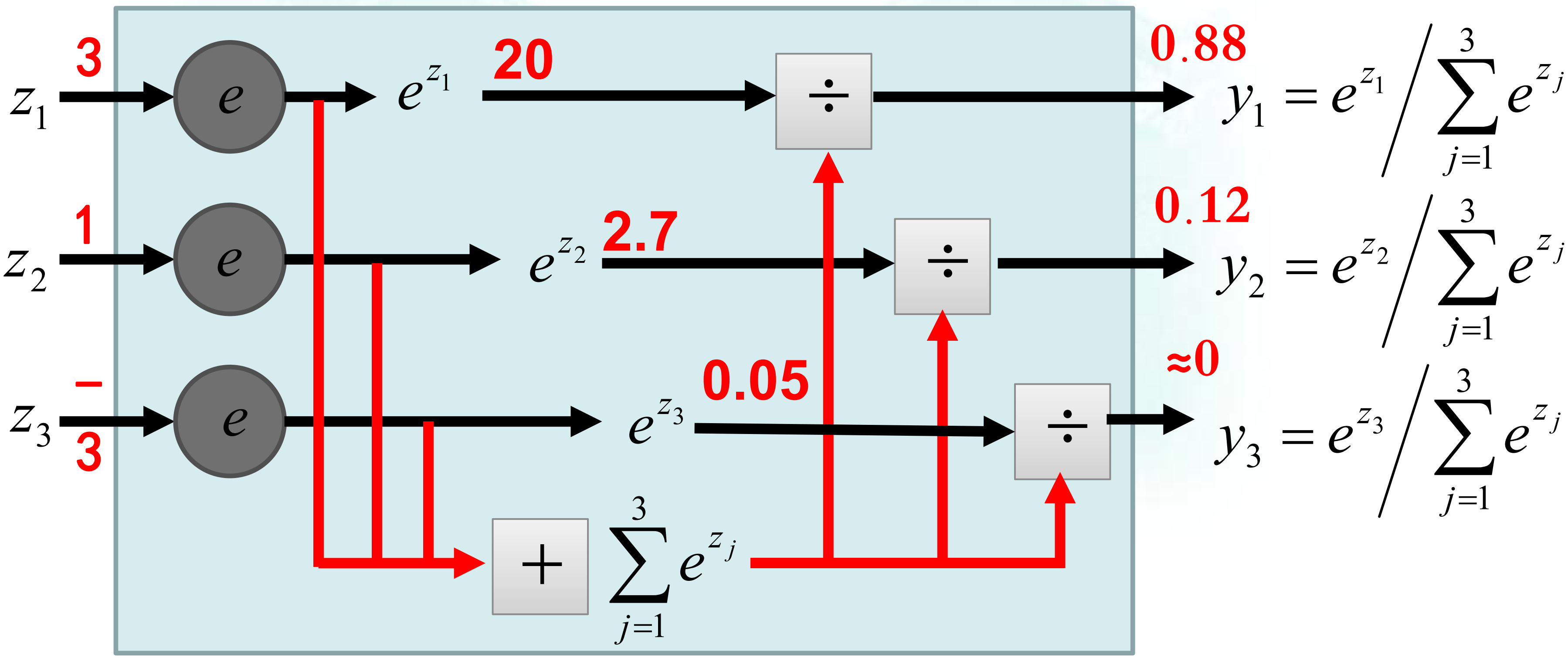
11.1.1 全连接神经网络

Output Layer

- Softmax layer as the output layer

Probability:
- $1 > y_i > 0$
- $\sum_i y_i = 1$

Softmax Layer



人机大战

- 举例：一些人工智能机器与人类竞赛的例子？

问题？

11.1 深度神经网络基础

AlphaGo　人工智能发展史上的里程碑事件

AlphaGo
Google DeepMind团队
人工智能围棋程序

4:1

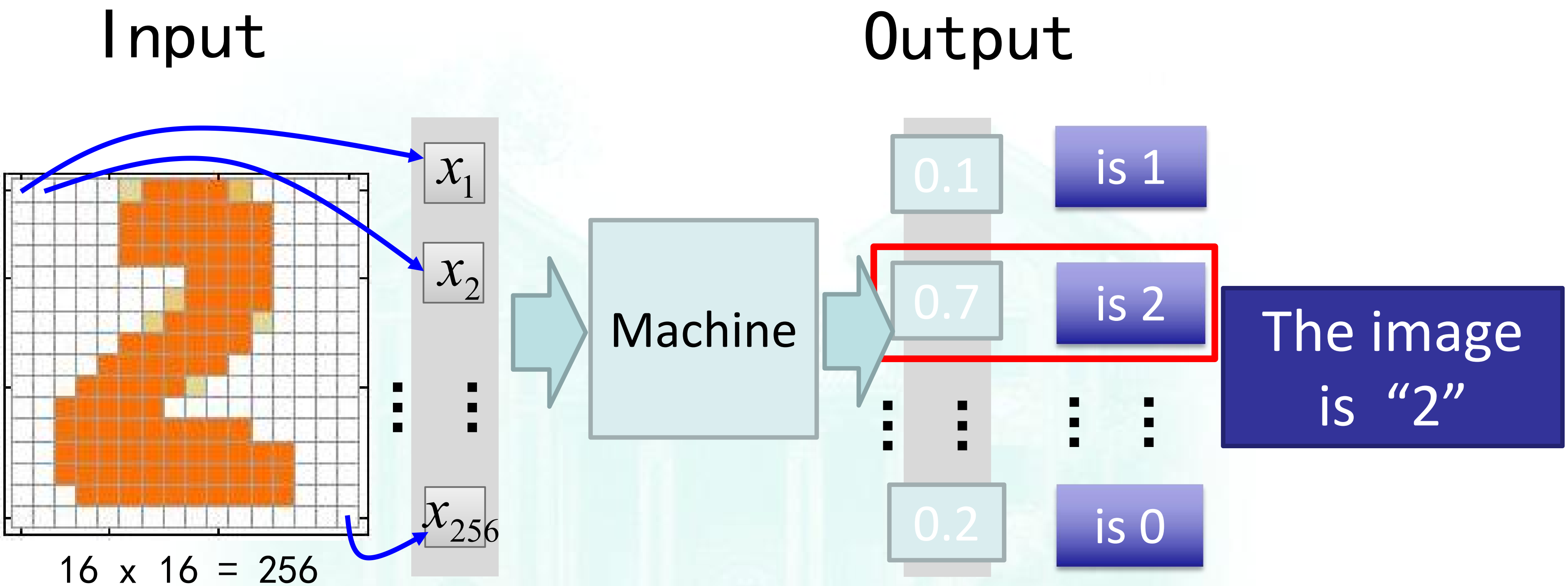
2016年3月

李世石
人类围棋顶尖高手
生涯14次世界冠军
18次国际赛冠军

荣登《Nature》封面

11.1.1 全连接神经网络

Example



Ink → 1

No ink → 0

What is needed is a function……

Each dimension represents the confidence of a digit.

11.1.1 全连接神经网络

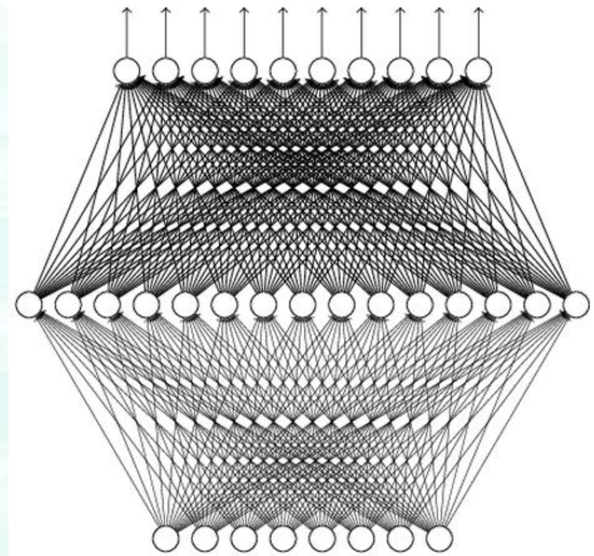
Why Deep?

Any continuous function f

$$f : R^N \rightarrow R^M$$

Can be realized by a network
with one hidden layer

(given **enough** hidden neurons)



Reference for the reason:
http://neuralnetworksanddeeplearning.com/chap4.html

11.1.1 全连接神经网络

Why Deep?

问题？

Logic circuits

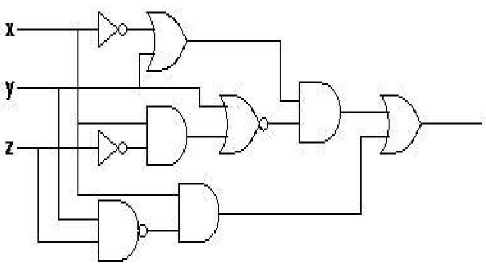
- Logic circuits consists of **gates**
- **A two layers of logic gates** can represent **any Boolean function.**
- Using multiple layers of logic gates to build some functions are much simpler

→ less gates needed

Neural network

- Neural network consists of **neurons**
- **A hidden layer network** can represent **any continuous function.**
- Using multiple layers of neurons to represent some functions are much simpler

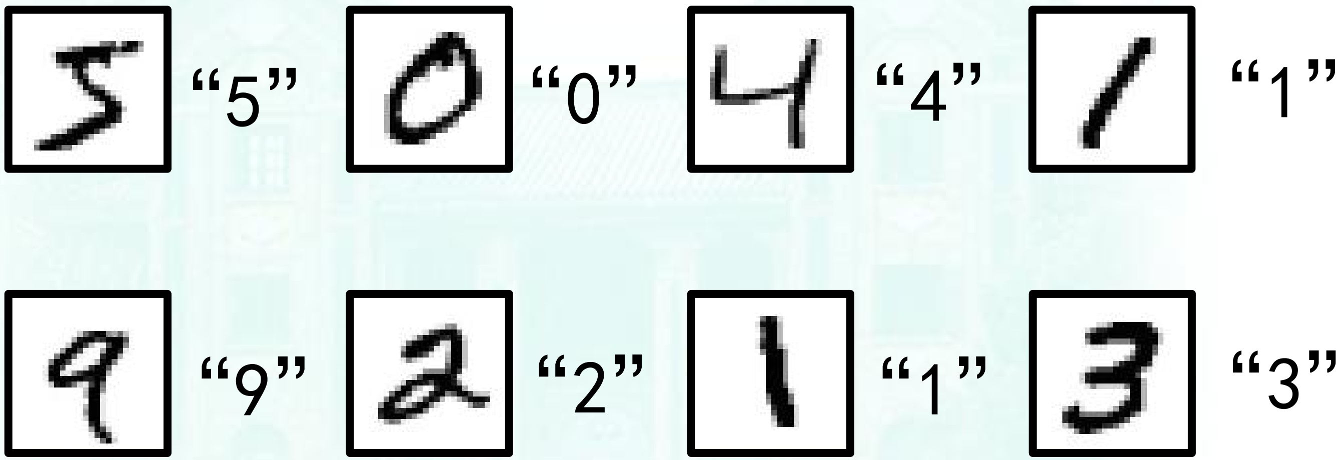
→ less parameters



More reason:
https://www.youtube.com/watch?v=XsC9byQkUH8&list=PLJV_el3uVTsPy9oCRY30oBPNLCo89yu49&index=13

Training Data

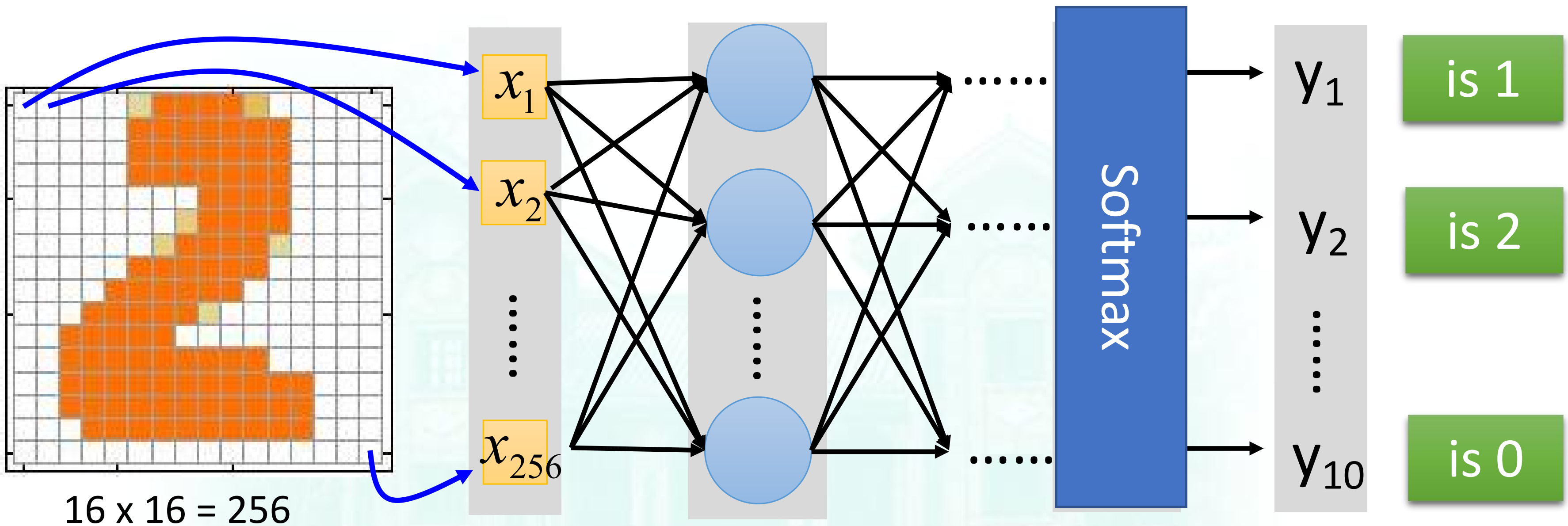
- Preparing training data: images and their labels



The learning target is defined on the training data.

11.1.1 全连接神经网络

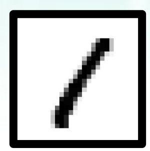
Learning Target

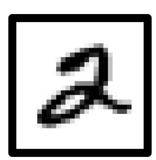


Ink → 1

No ink → 0

The learning target is ……

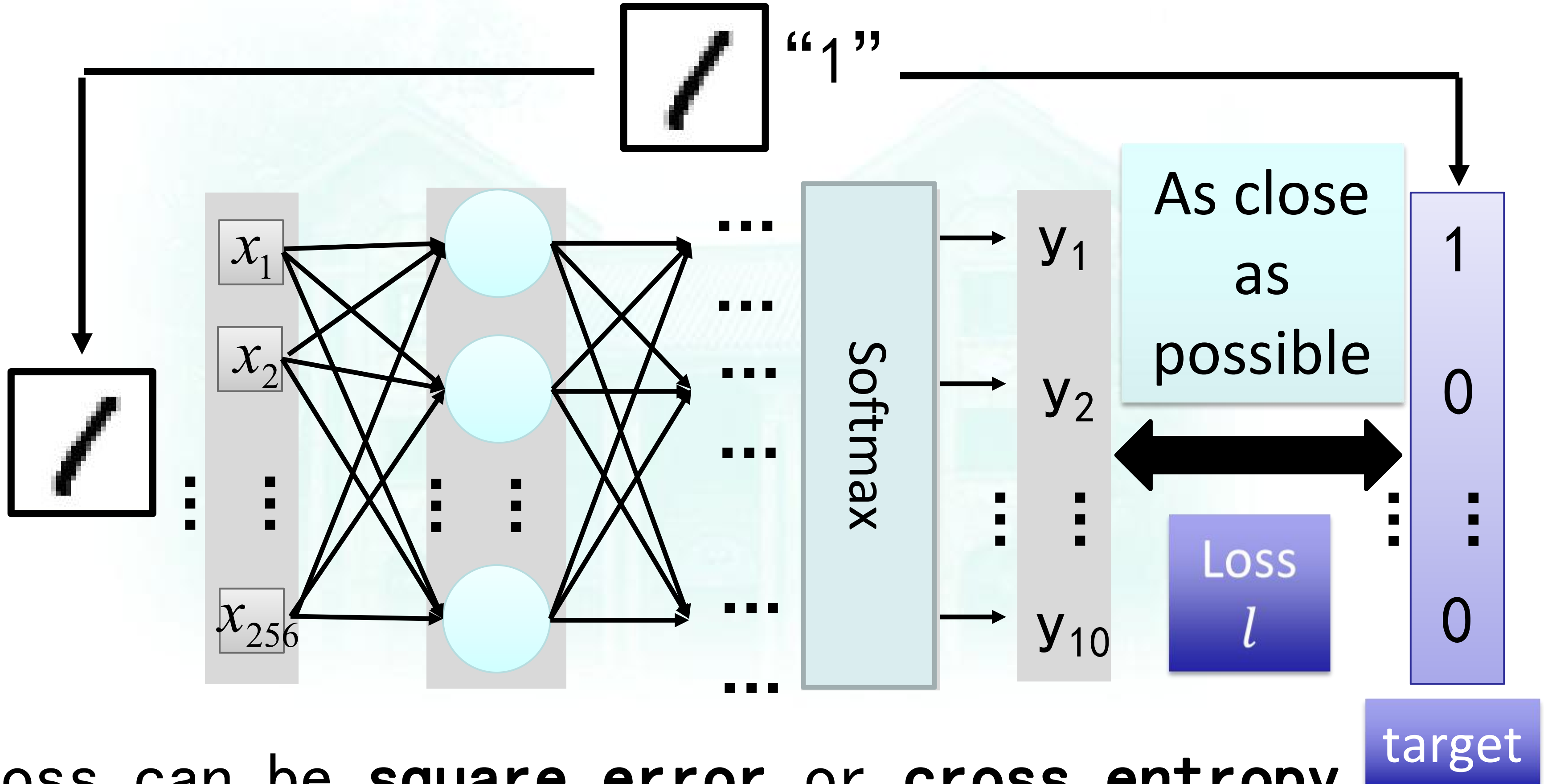
Input:  y_1 has the maximum value

Input:  y_2 has the maximum value

11.1.1 全连接神经网络

Loss

Given a set of parameters, a good function should make the loss of all examples as small as possible

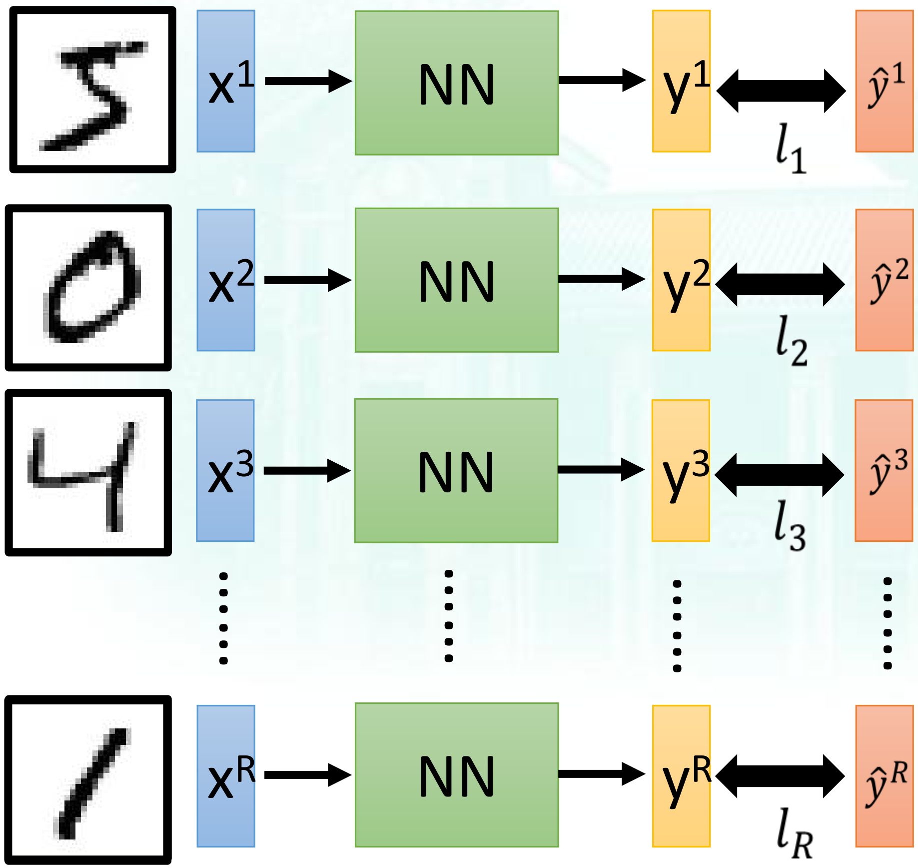


Loss can be **square error** or **cross entropy** between the network output and target

11.1.1 全连接神经网络

Total Loss

For all training data ...



Total Loss: $$L = \sum_{r=1}^{R} l_r$$

As small as possible

Find ***a function*** that minimizes total loss L

Find ***the network parameters θ^**** that minimize total loss L

11.1.1 全连接神经网络

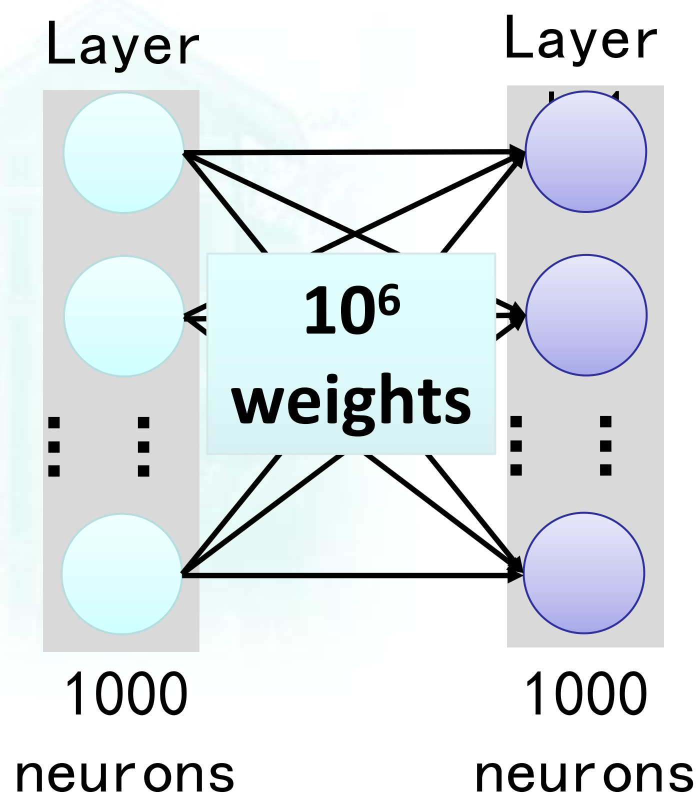
How to pick the best function

Find *network parameters θ^** that minimize total loss L

Enumerate all possible values

Network parameters $\theta = \{w_1, w_2, w_3, \cdots, b_1, b_2, b_3, \cdots\}$

Millions of parameters



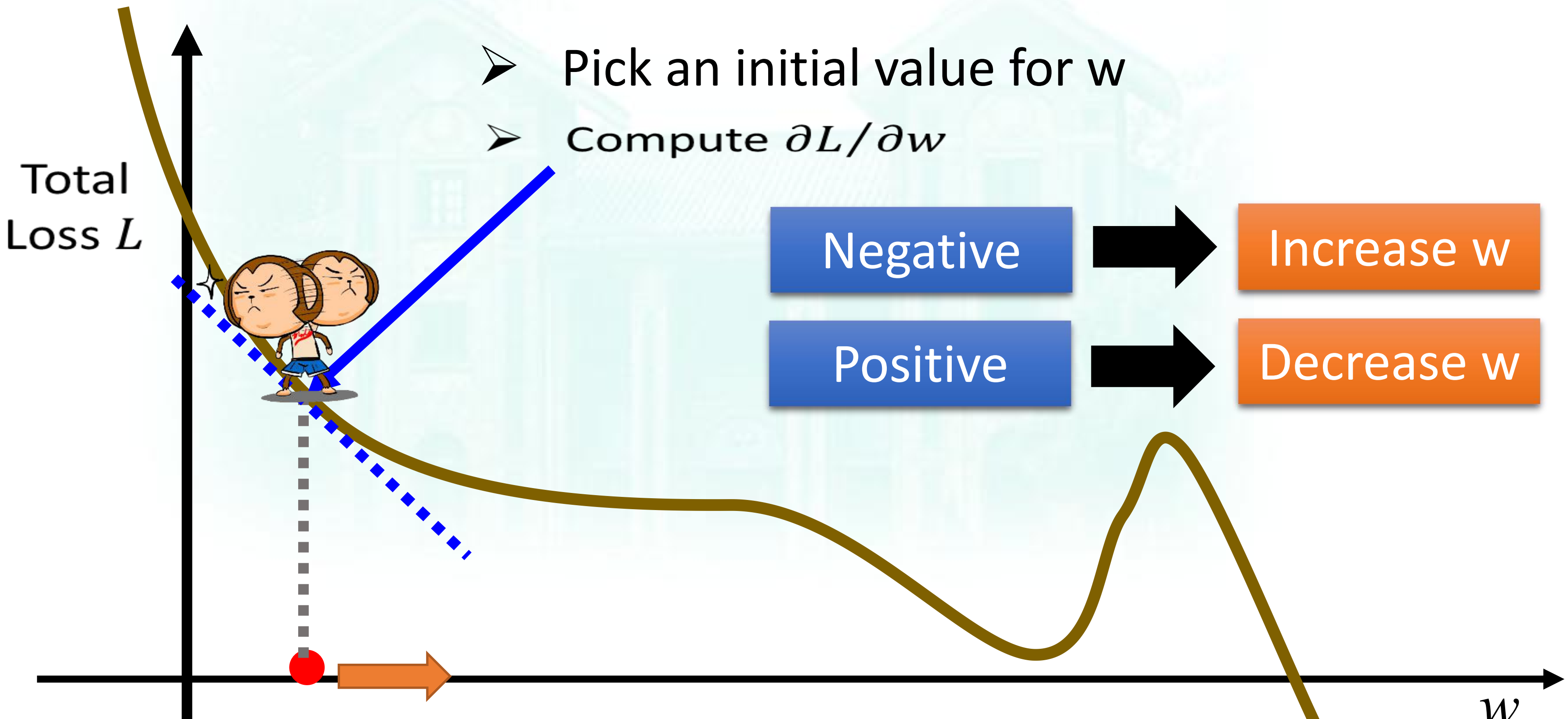
问题？

11.1.1 全连接神经网络

Gradient Descent

Network parameters $\theta = \{w_1, w_2, \cdots, b_1, b_2, \cdots\}$

Find ***network parameters*** θ^* that minimize total loss L



http://chico386.pixnet.net/album/photo/171572850

11.1.1 全连接神经网络

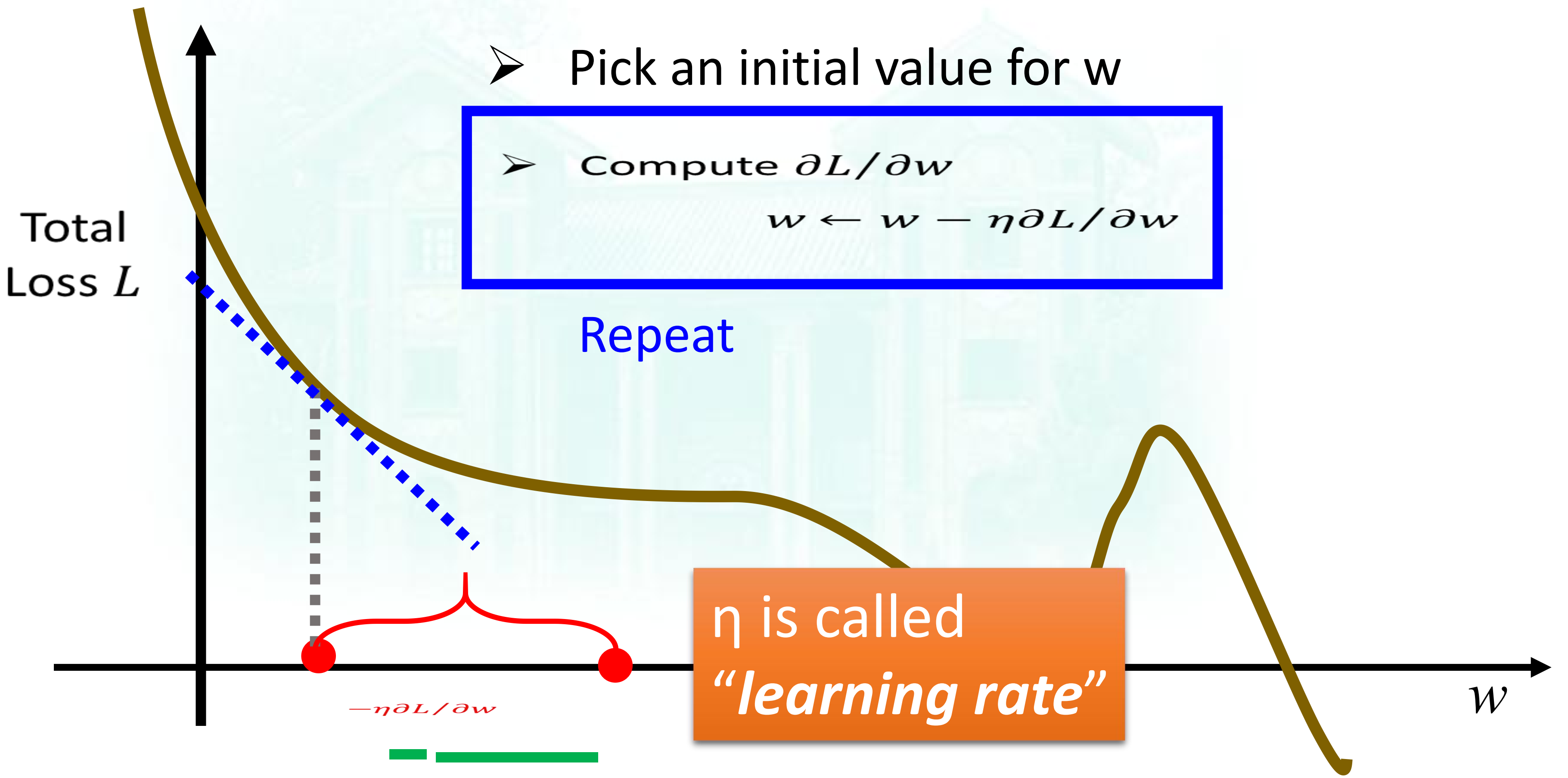
Gradient Descent

Network parameters $\theta = \{w_1, w_2, \cdots, b_1, b_2, \cdots\}$

Find ***network parameters*** $\boldsymbol{\theta^*}$ that minimize total loss L

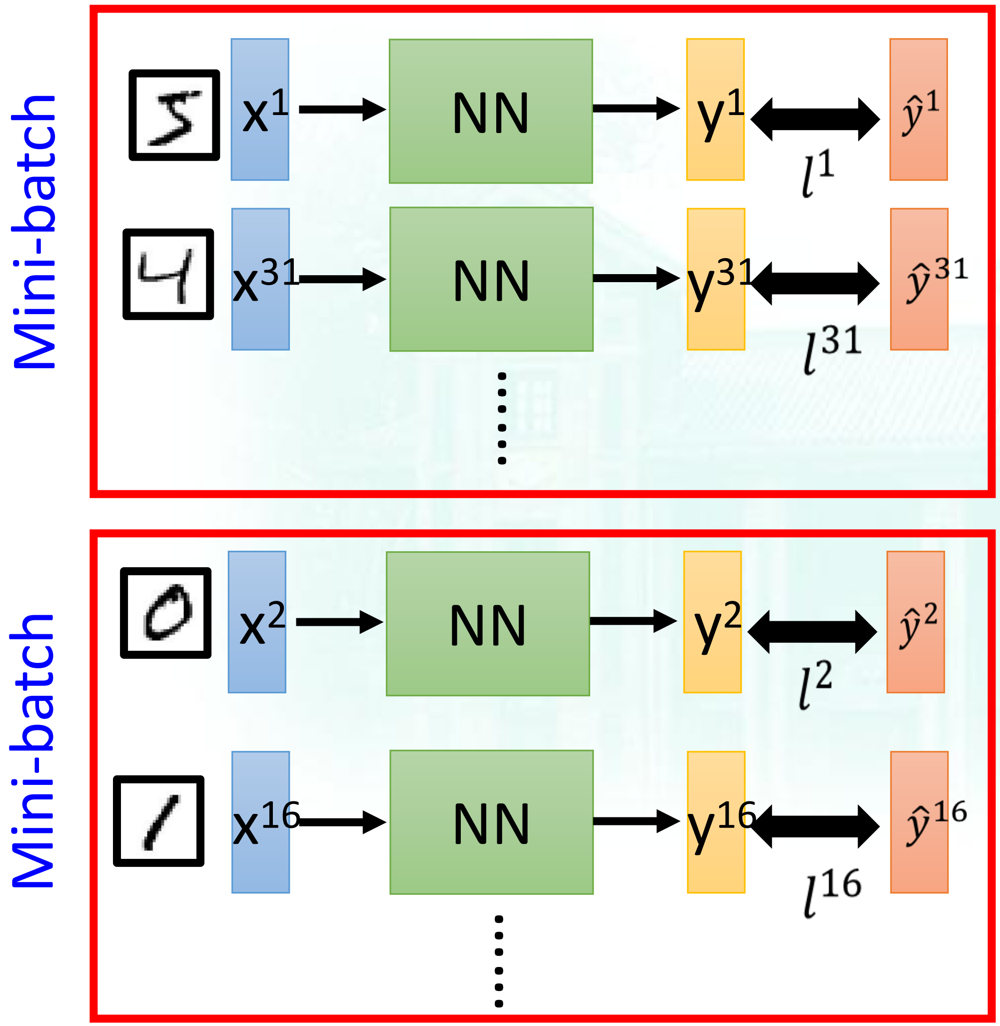
- Pick an initial value for w
  - Compute $\partial L / \partial w$
    $$w \leftarrow w - \eta \partial L / \partial w$$

Repeat



11.1.1 全连接神经网络

Mini-batch



We do not really minimize total loss!

- Randomly initialize network parameters
- Pick the 1st batch

$$L' = l^1 + l^{31} + \cdots$$

Update parameters once

- Pick the 2nd batch

$$L'' = l^2 + l^{16} + \cdots$$

Update parameters once

⋮

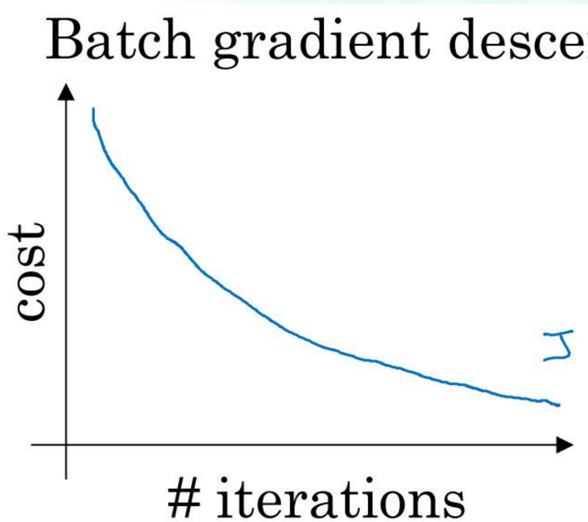
- Until all mini-batches have been picked

one epoch

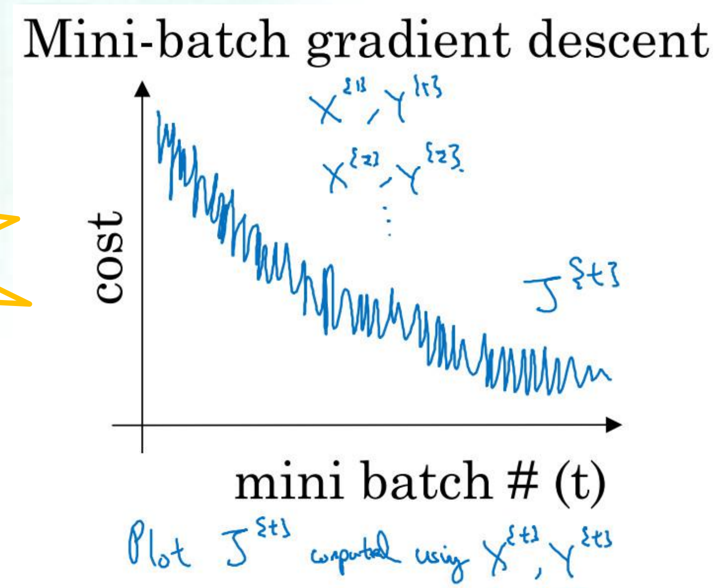
Repeat the above process

为什么使用mini-batch?

- 使用 batch 梯度下降法时，每次迭代你都需要历遍整个训练集
- 每看一个数据就算一下损失函数，然后求梯度更新参数，这个称为随机梯度下降
- 使用 mini-batch 梯度下降法，如果你作出成本函数在整个过程中的图，则并不是每次迭代都是下降的



问题？



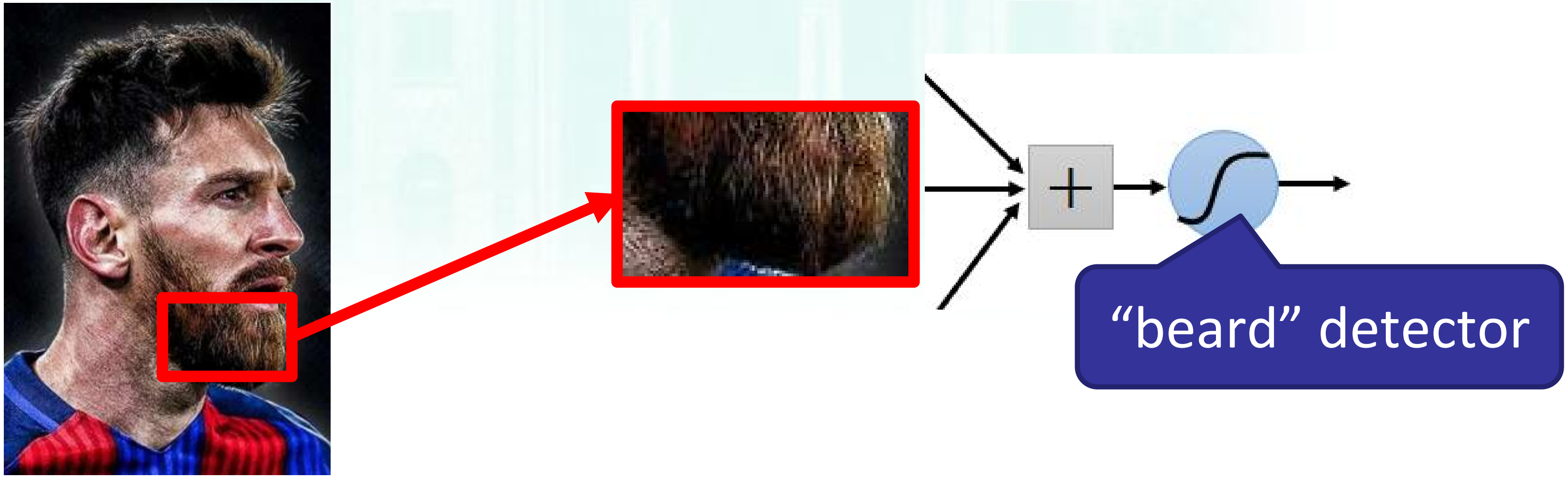
11.1.2 卷积神经网络

Why CNN?

- Some patterns are much smaller than the whole image

A neuron does not have to see the whole image to discover the pattern.

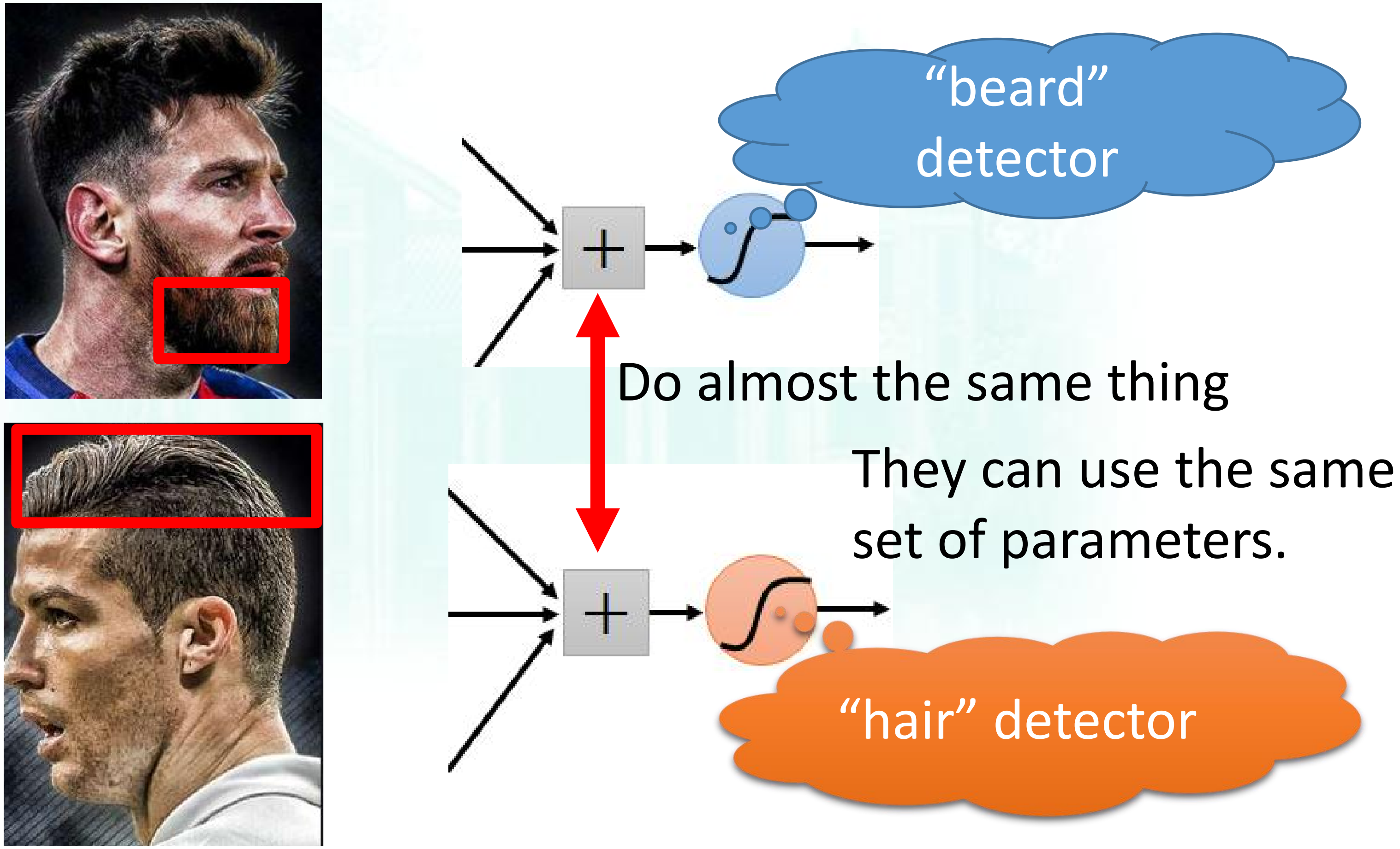
Connecting to small region with less parameters



11.1.2 卷积神经网络

Why CNN?

- The same patterns appear in different regions.



11.1.2 卷积神经网络

Why CNN?

- Subsampling the pixels will not change the object

Cristiano Ronaldo

Cristiano Ronaldo

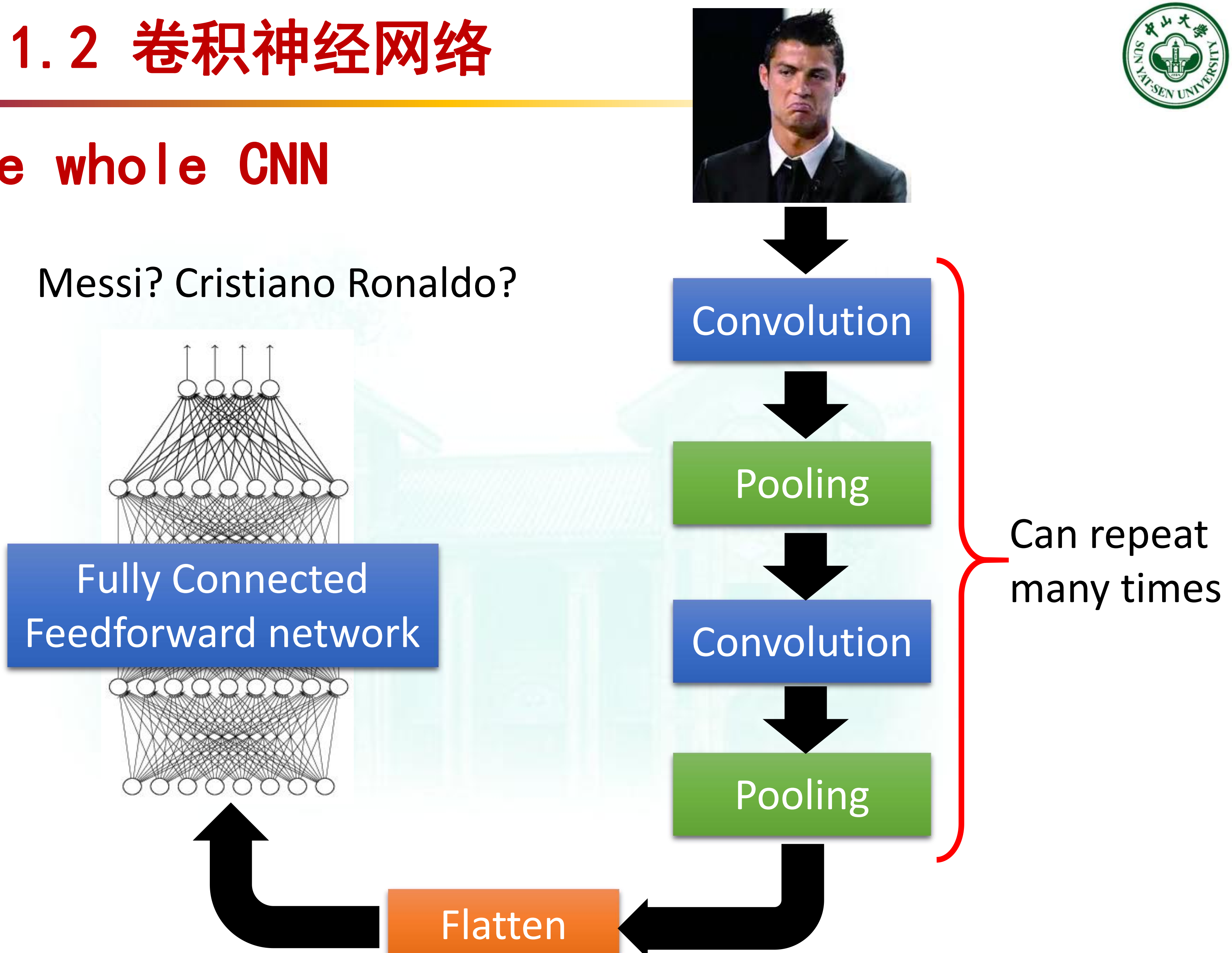
subsampling

We can subsample the pixels to make image smaller

→ Less parameters for the network to process the image

11.1.2 卷积神经网络

The whole CNN



11.1.2 卷积神经网络

The whole CNN

Property 1

- Some patterns are much smaller than the whole image

Property 2

- The same patterns appear in different regions.

Property 3

- Subsampling the pixels will not change the object

Convolution

Max Pooling

Convolution

Max Pooling

Can repeat many times

Flatten

Convolution

Those are the network parameters to be learned.

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

6 x 6 image

1	-1	-1
-1	1	-1
-1	-1	1

Filter 1

Matrix

-1	1	-1
-1	1	-1
-1	1	-1

Filter 2

Matrix

⋮ ⋮

Property 1

Each filter detects a small pattern (3 x 3).

11.1.2 卷积神经网络

Convolution

1	-1	-1
-1	1	-1
-1	-1	1

Filter 1

stride=1

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

6 x 6 image

3	-1	-3	-1
-3	1	0	-3
-3	-3	0	1
3	-2	-2	-1

Property 2

11.1.2 卷积神经网络

Convolution

-1	1	-1
-1	1	-1
-1	1	-1

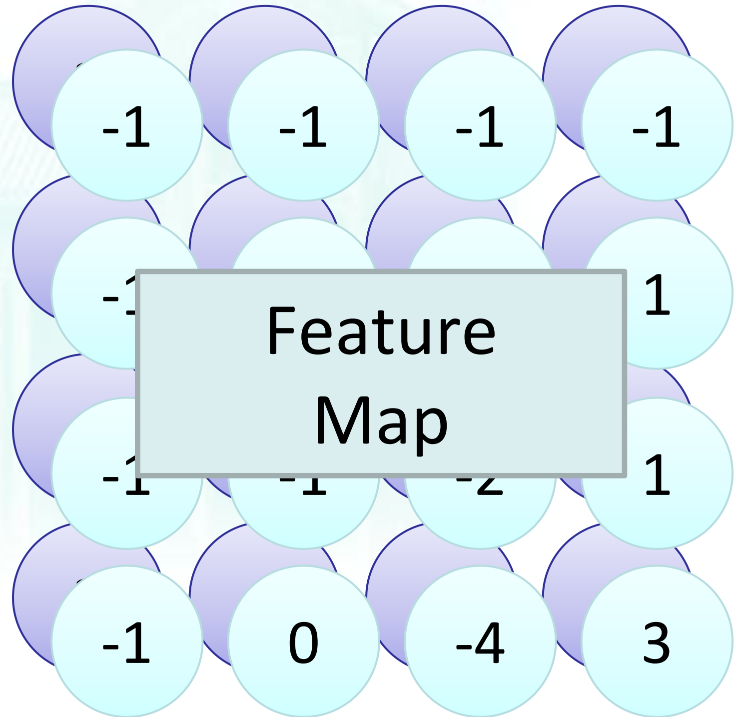
Filter 2

stride=1

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

6 x 6 image

Do the same process for every filter



4 x 4 image

11.1.2 卷积神经网络

Colorful image

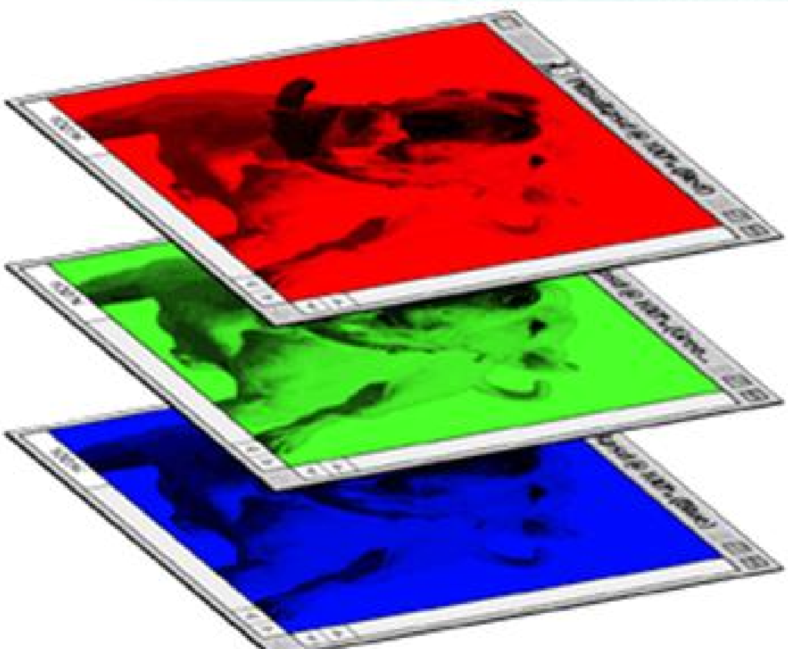
Filter 1

1	-1	-1
-1	1	-1
-1	-1	1

Filter 2

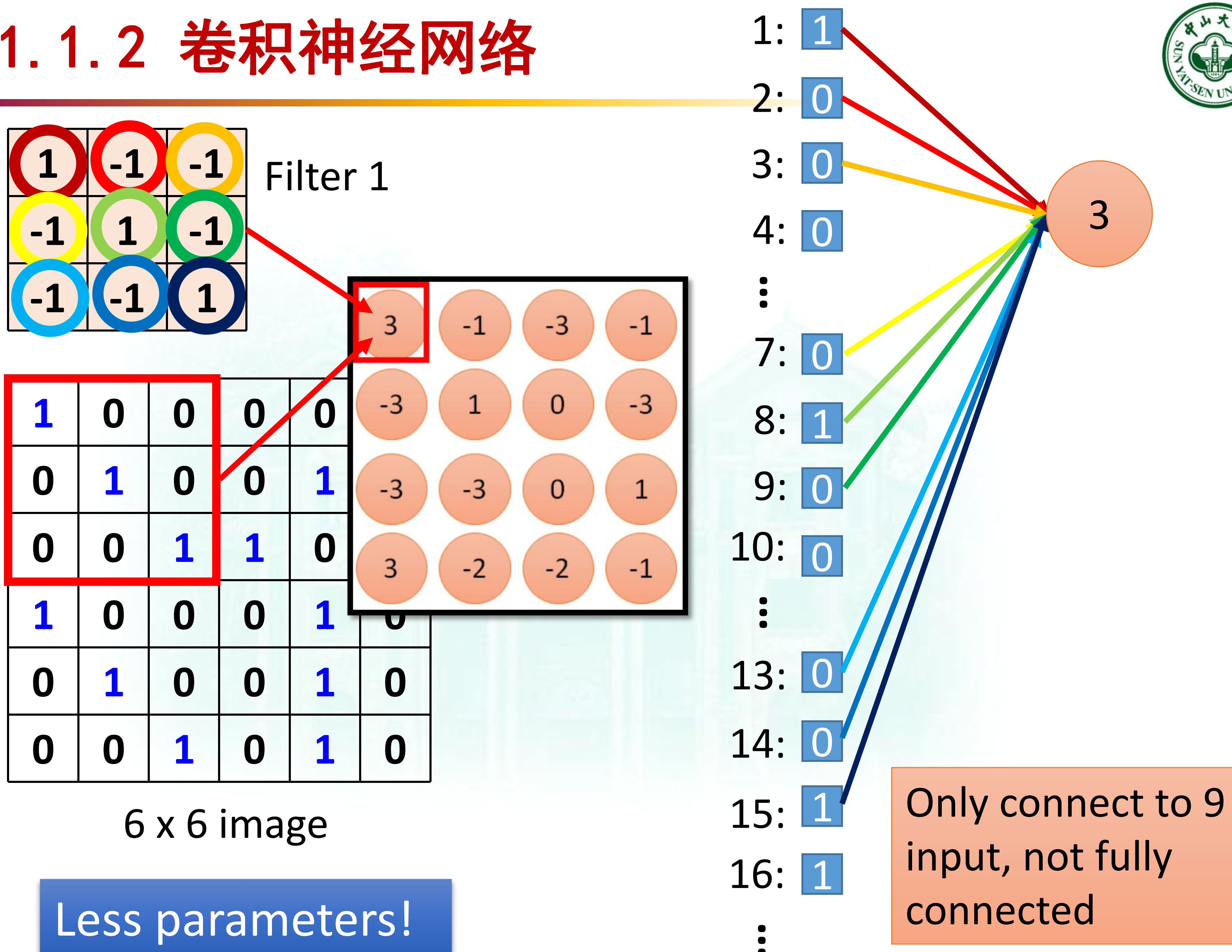
-1	1	-1
-1	1	-1
-1	1	-1

Colorful image



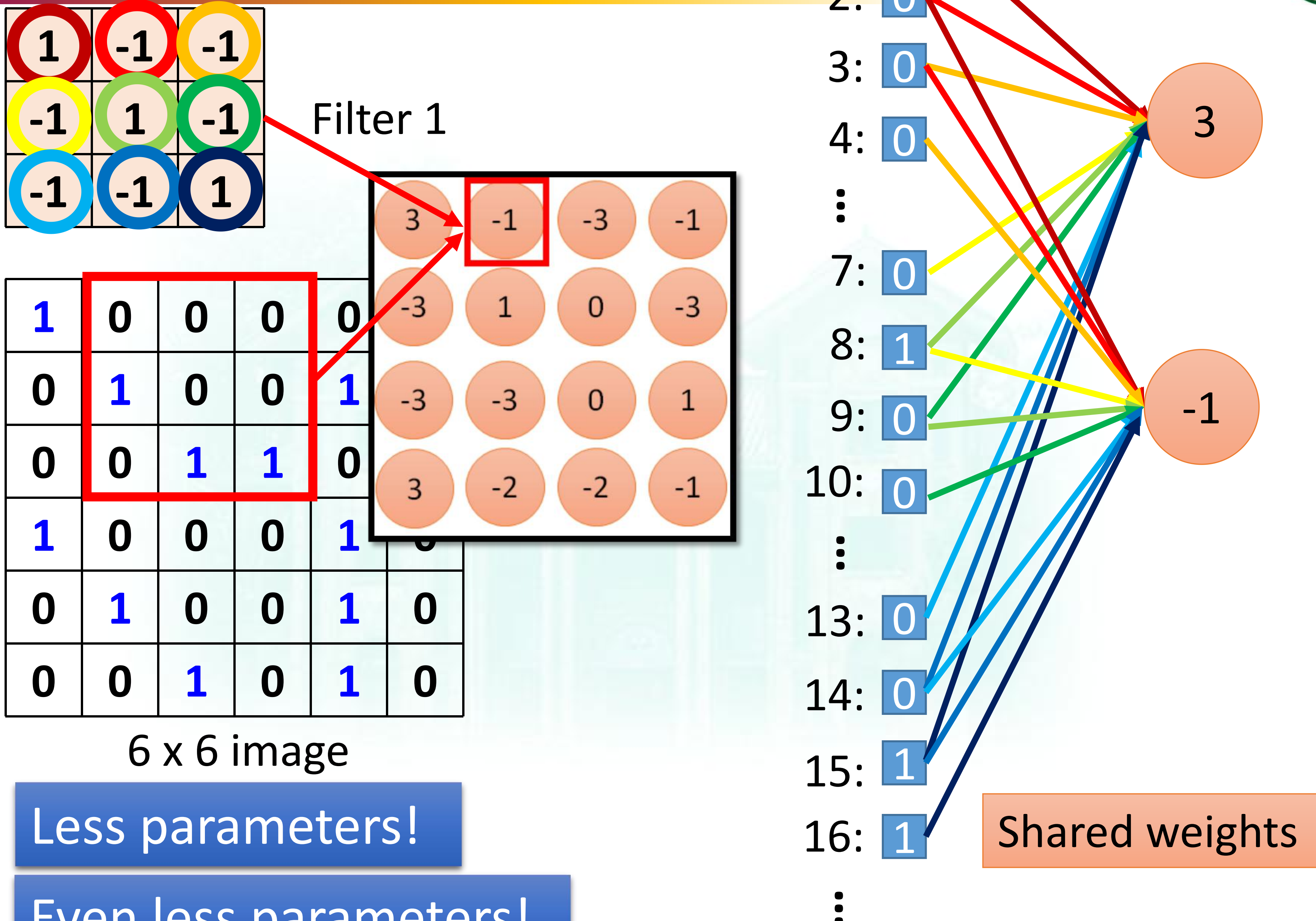
1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

11.1.2 卷积神经网络



1 -1 -1
-1 1 -1
-1 -1 1
Filter 1
1 0 0 0 0
0 1 0 0 1
0 0 1 1 0
1 0 0 0 1
0 1 0 0 1 0
0 0 1 0 1 0
6 x 6 image
3 -1 -3 -1
-3 1 0 -3
-3 -3 0 1
3 -2 -2 -1
1: 1
2: 0
3: 0
4: 0
7: 0
8: 1
9: 0
10: 0
13: 0
14: 0
15: 1
16: 1
3
Only connect to 9 input, not fully connected
Less parameters!

11.1.2 卷积神经网络



Less parameters!

Even less parameters!

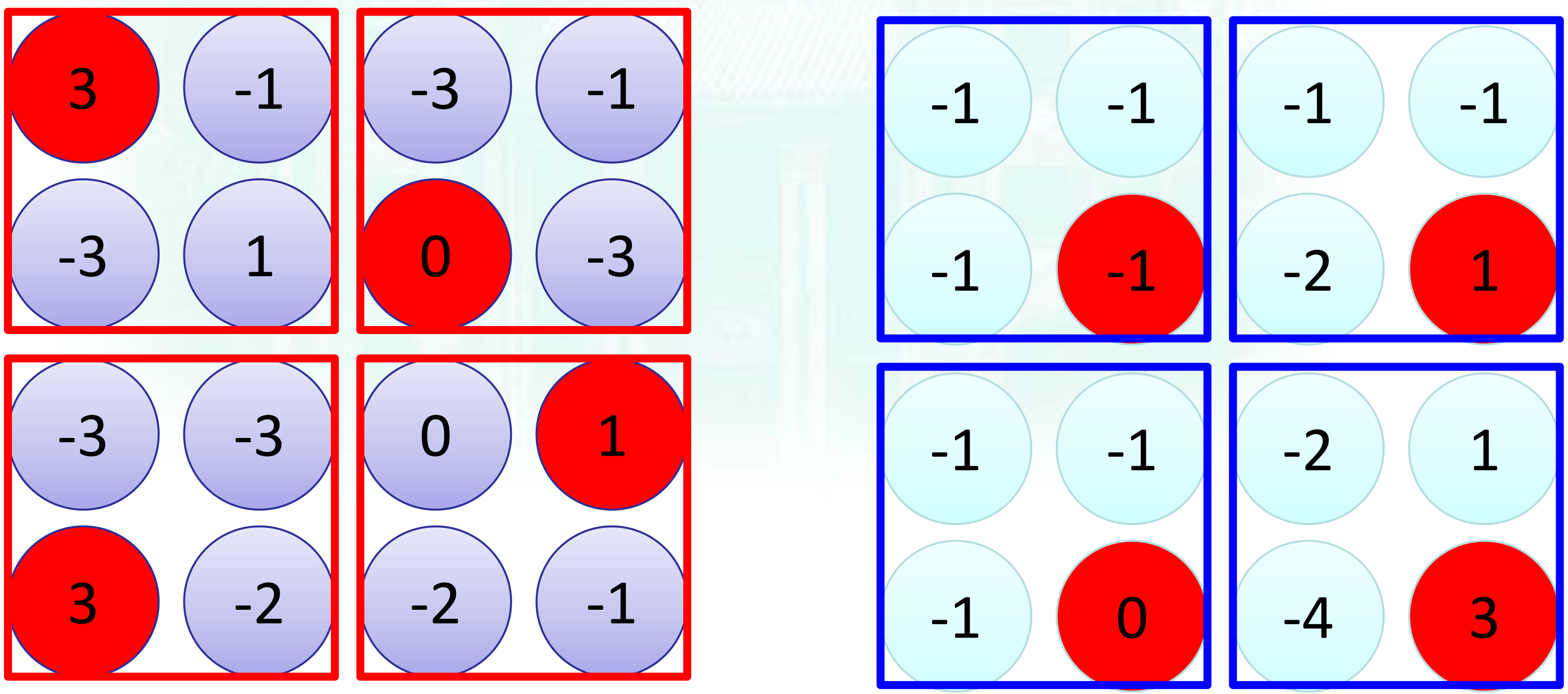
Pooling（Max）

1	-1	-1
-1	1	-1
-1	-1	1

Filter 1

-1	1	-1
-1	1	-1
-1	1	-1

Filter 2

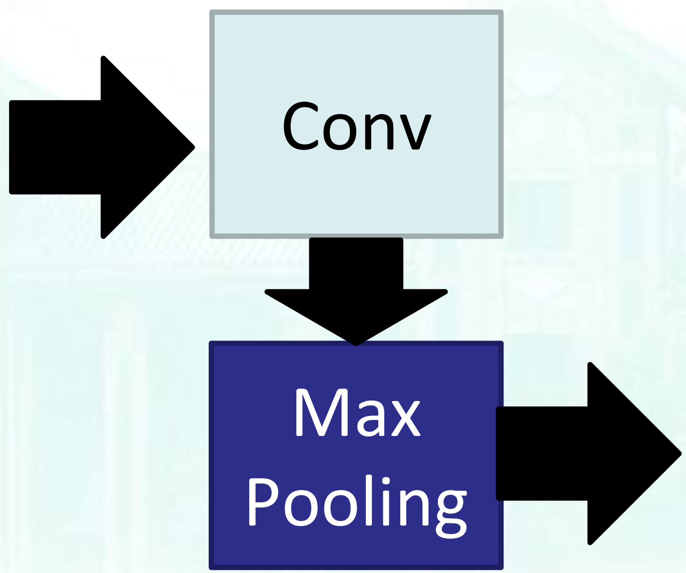


11.1.2 卷积神经网络

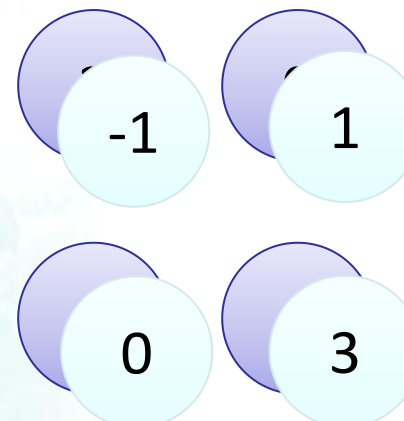
Pooling（Max）

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

6 x 6 image



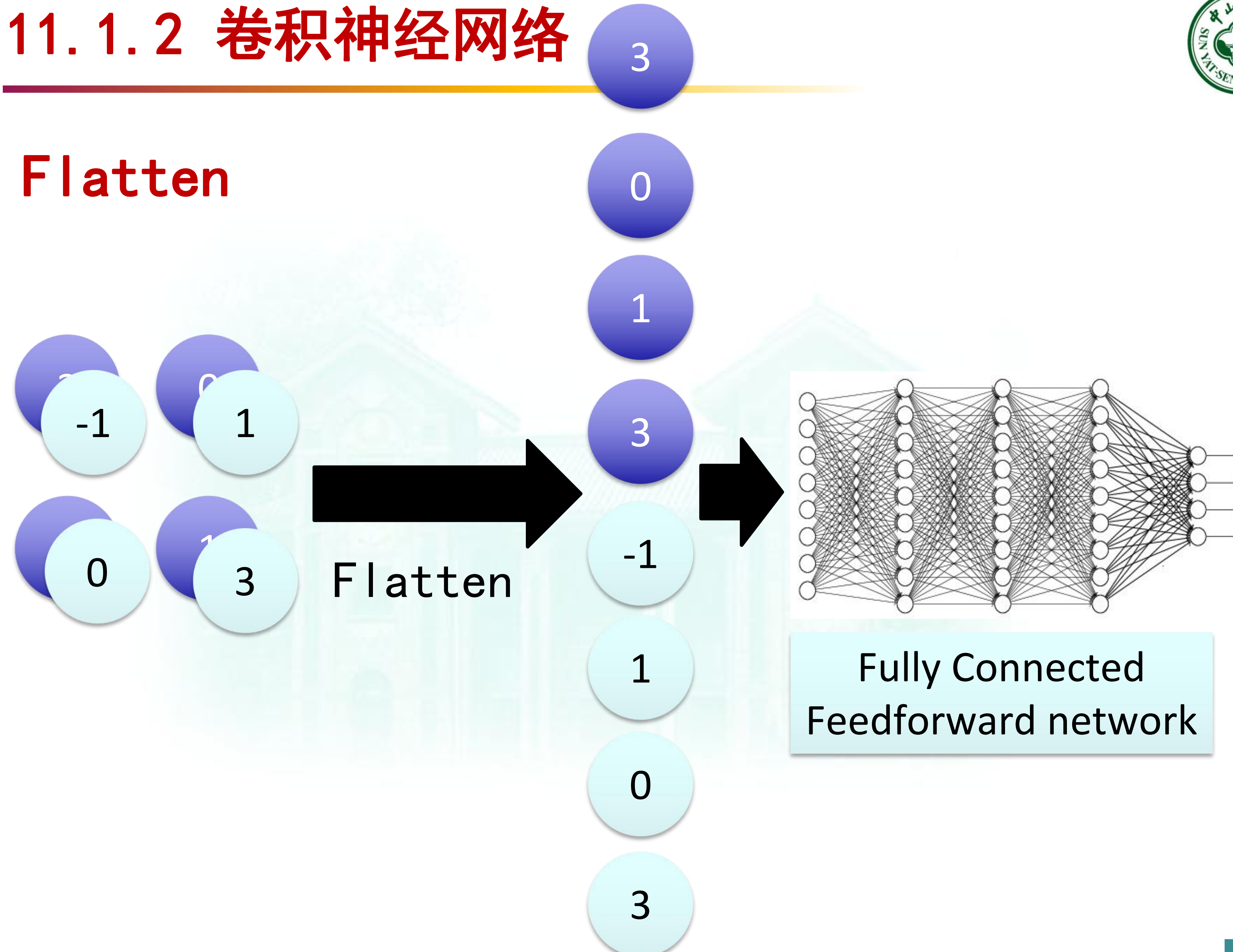
New image but smaller



2 x 2 image

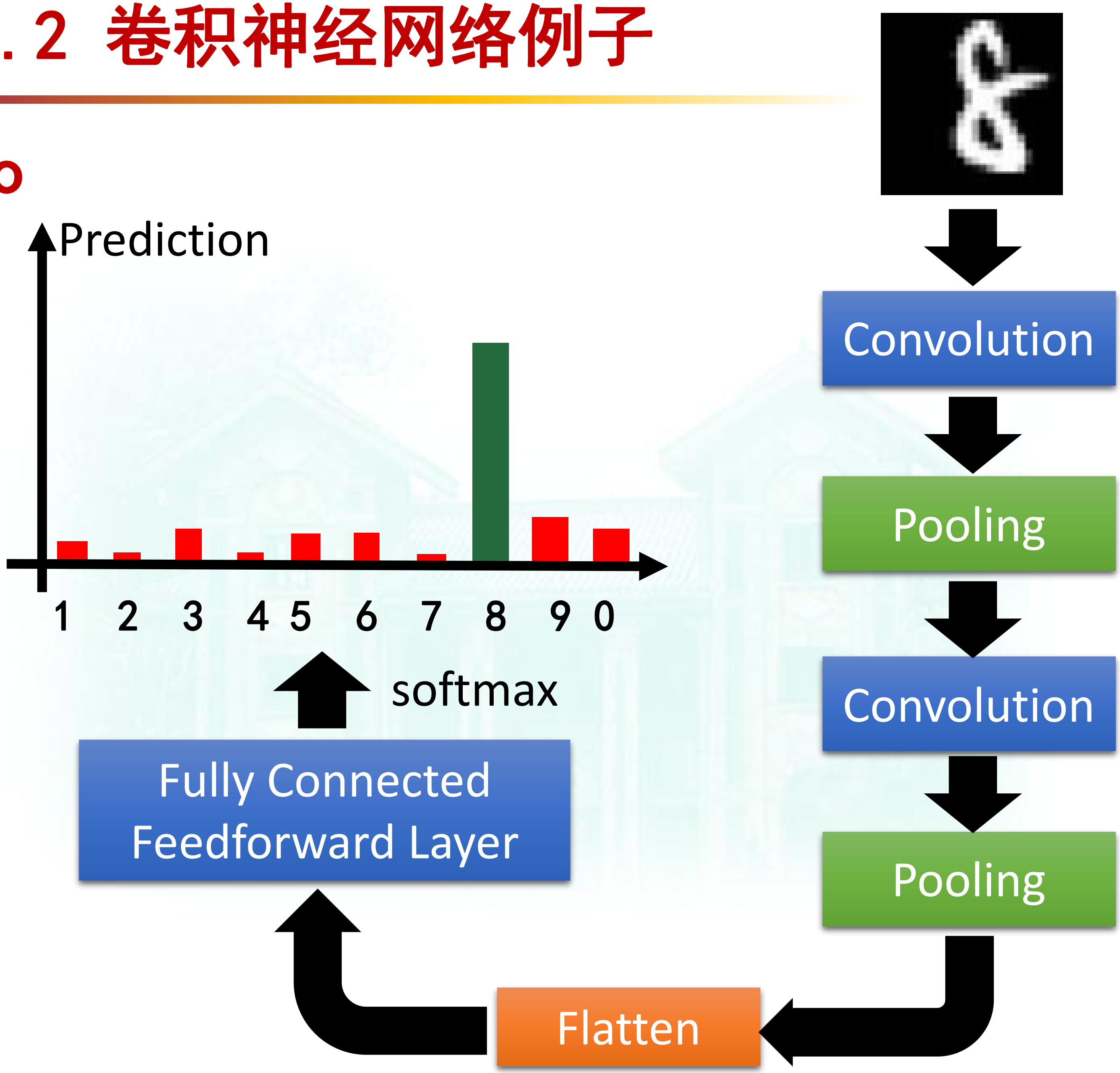
Each filter is a channel

Flatten


-1
1
0
3
Flatten
3
0
1
3
-1
1
0
3
Fully Connected
Feedforward network

11.1.2 卷积神经网络例子

Demo

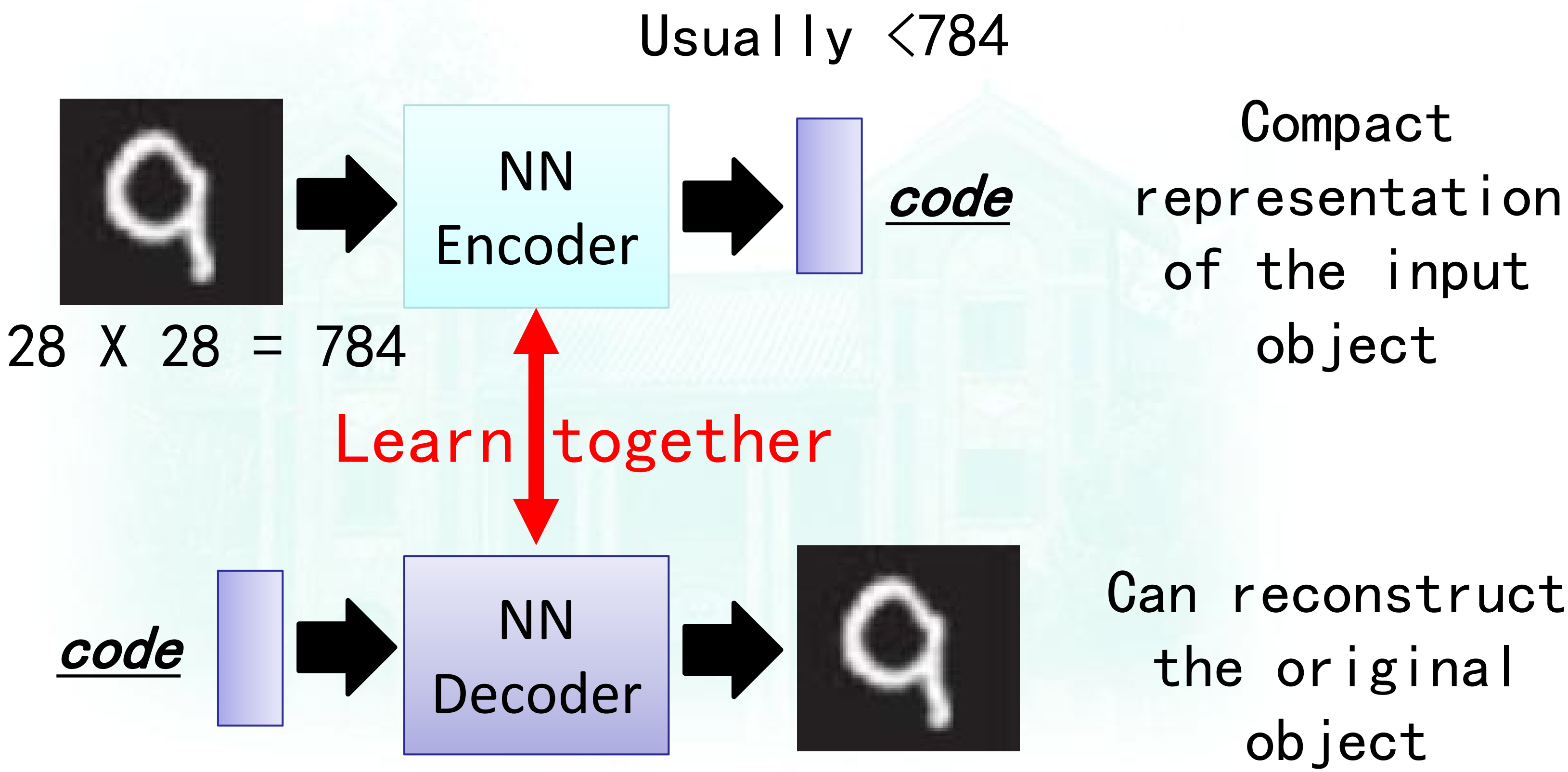


https://cs.stanford.edu/people/karpathy/convnetjs/demo/mnist.html

11.1.3 自编码网络

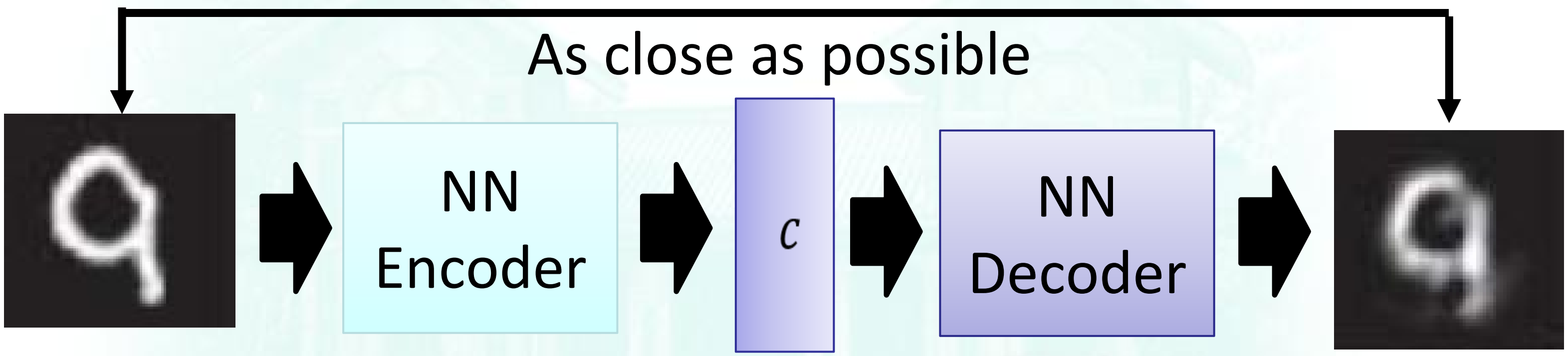
Encoder & Decoder

问题？



11.1.3 自编码网络

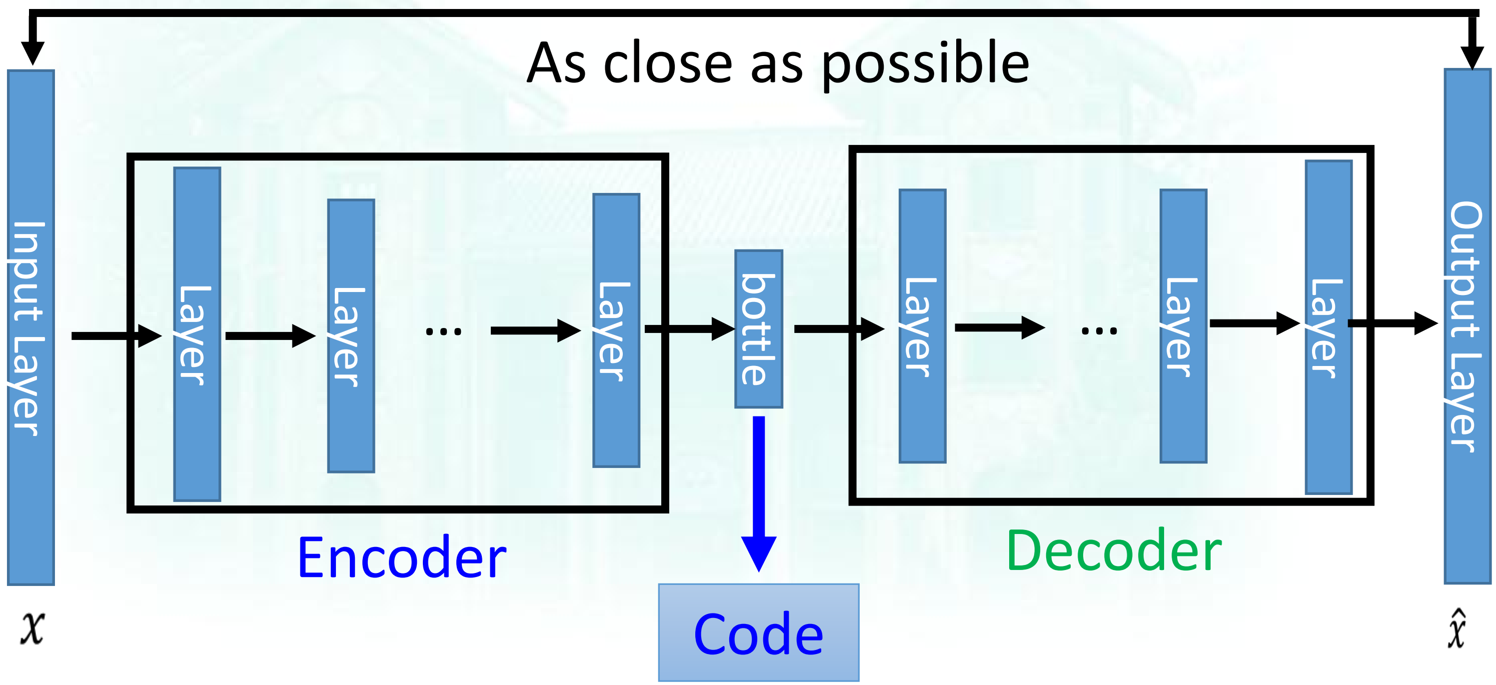
Auto-encoder



11.1.3 自编码网络

Deep Auto-encoder

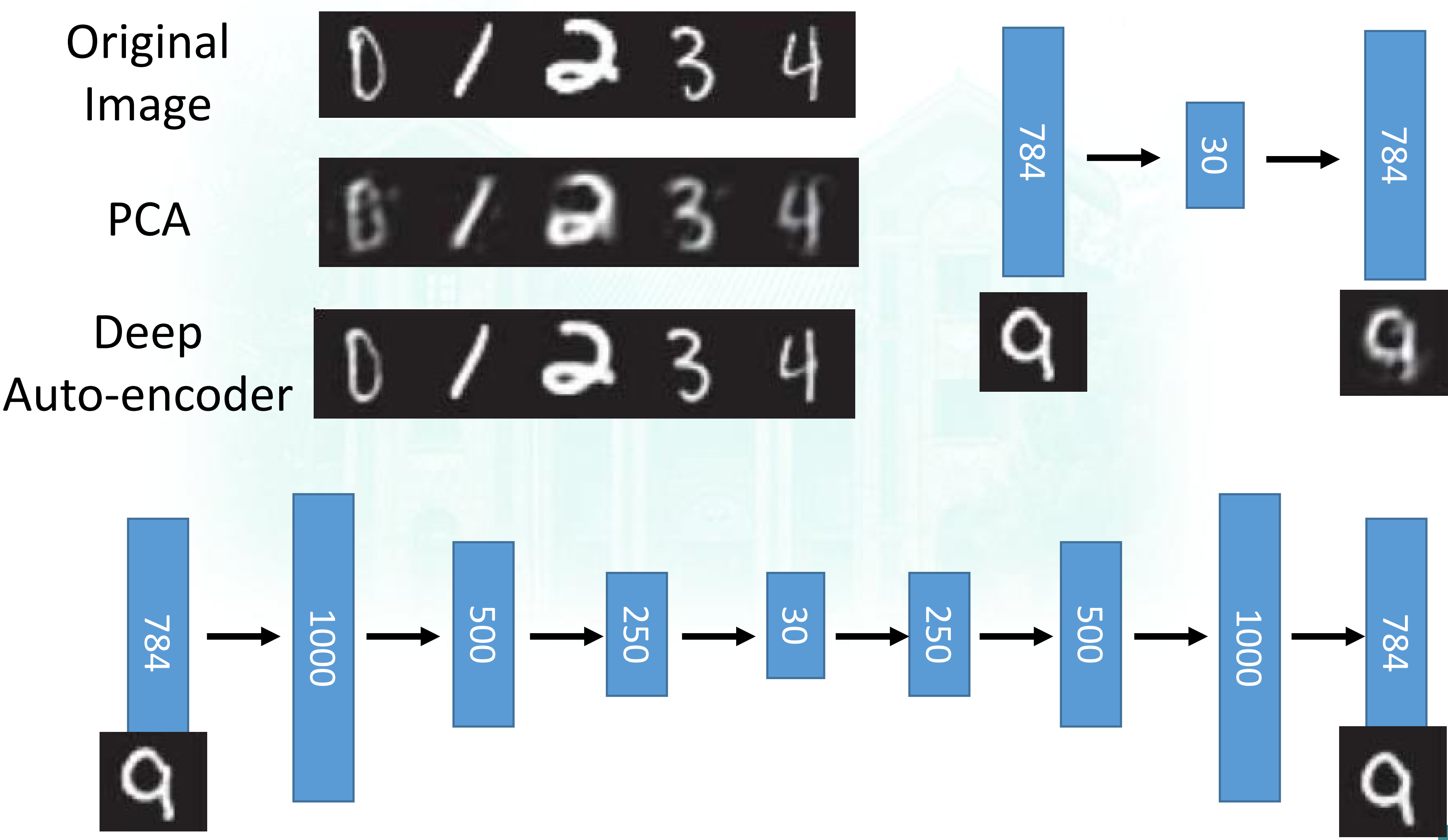
- NN encoder + NN decoder = a deep network



Reference: Hinton, Geoffrey E., and Ruslan R. Salakhutdinov. "Reducing the dimensionality of data with neural networks." *Science* 313.5786 (2006): 504-507

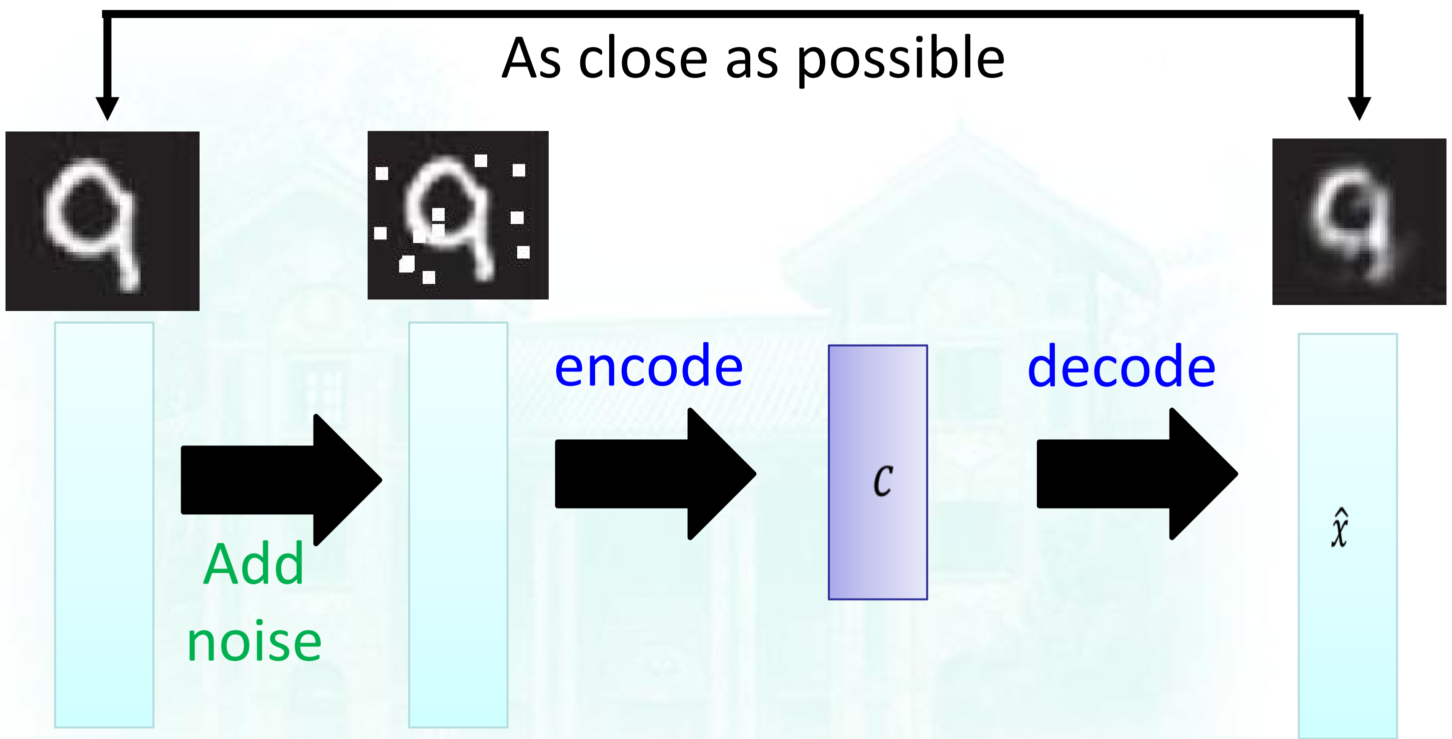
11.1.3 自编码网络

Deep Auto-encoder



11.1.3 自编码网络

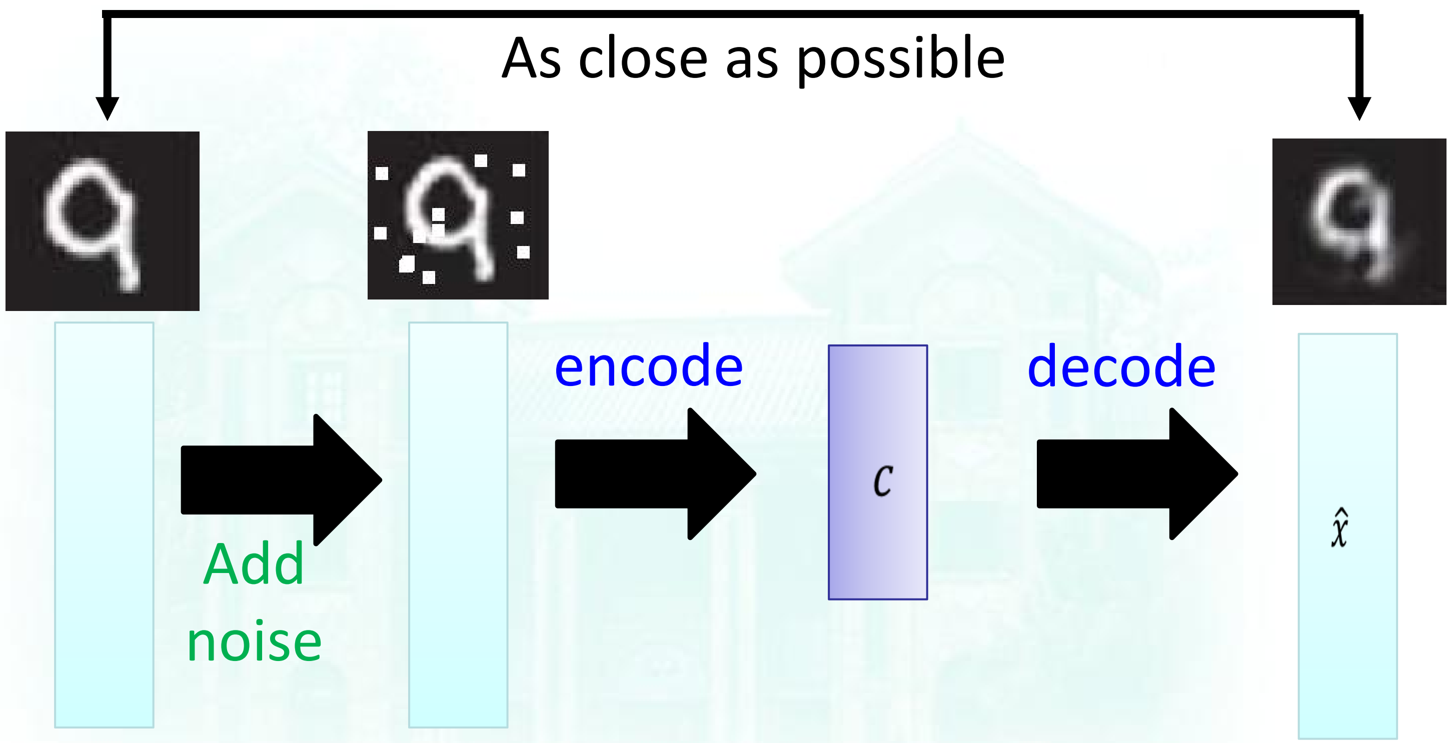
De-noising auto-encoder



Vincent, Pascal, et al. "Extracting and composing robust features with denoising autoencoders." *ICML*, 2008.

11.1.3 自编码网络

De-noising auto-encoder



Vincent, Pascal, et al. "Extracting and composing robust features with denoising autoencoders." *ICML,* 2008.