中国科学技术大学
University of Science and Technology of China

《人工智能数学原理与算法》
第 3 章：神经网络基础

3.1 前馈神经网络

连德富
liandefu@ustc.edu.cn

目录

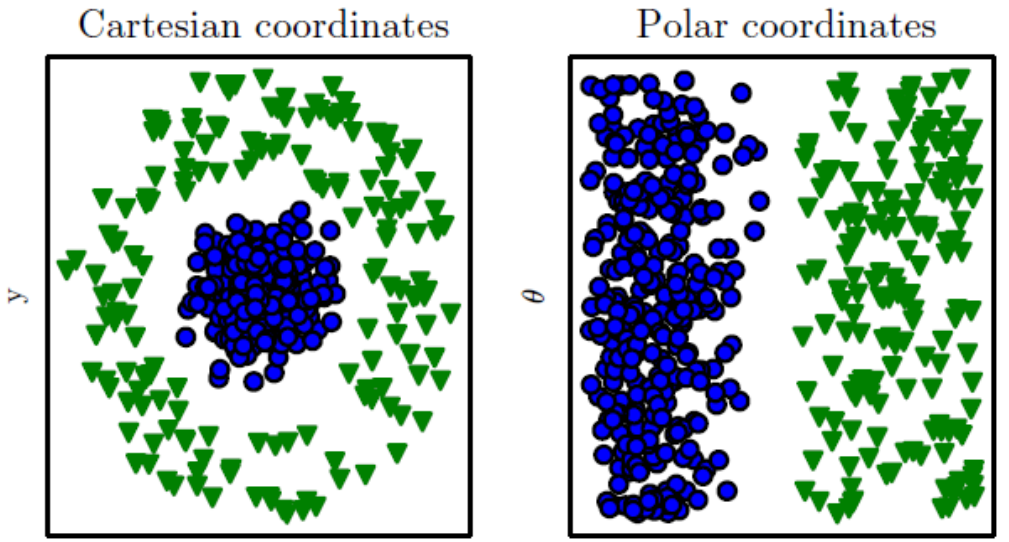
目录

神经网络动机

□ 问题：线性模型难以处理线性不可分情形

$y = \boldsymbol{w}^{\top}\boldsymbol{x} + b$

缺点：无法理解输入变量非线性关系



$\boldsymbol{x} = (x, y)$

$\phi(\boldsymbol{x}) = \big(r(x,y), \theta(x,y)\big)$

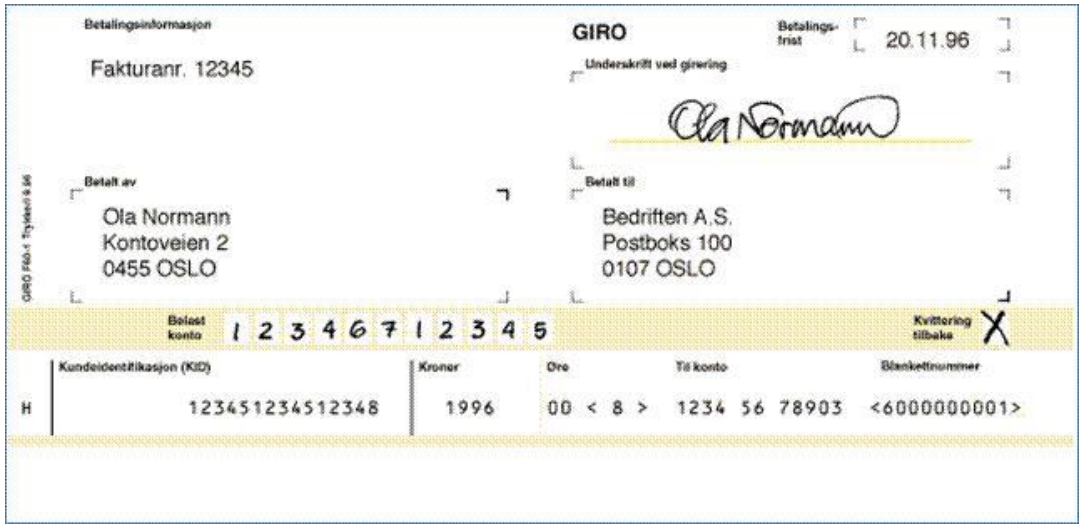
$y = \boldsymbol{w}^{\top}\phi(\boldsymbol{x}) + b$

线性模型用在一个非线性变换后的输入 $\phi(\boldsymbol{x})$

ϕ的选择：

- 核技巧：隐式定义 ϕ 。如RBF 核定义的无限维的ϕ
- 特征工程：根据经验手动设计ϕ
- 神经网络学习 ϕ，即$y = f(\boldsymbol{x}; \theta, \boldsymbol{w}) = \phi(\boldsymbol{x}; \theta)^{T}\boldsymbol{w}$

图像分类任务示例：手写数字识别

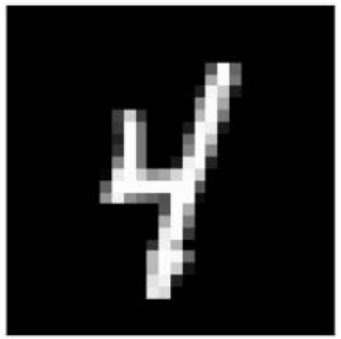


□MNIST (handwritten digits) 数据集

6万训练样本和1万测试样本

神经网络动机

- **问题：线性模型泛化能力较弱**

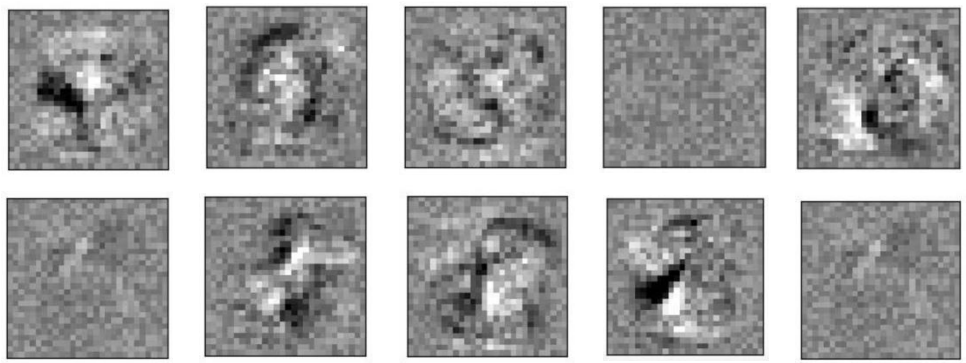


手写数字识别
图片分类

$y = Wx$

数字：4

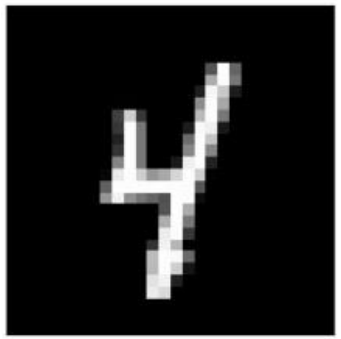
线性模型只能为每个类学习到一个模板



W参数矩阵对每个类的可视化结果，线性模型难以学习到每类数字的关键特征。

神经网络动机

- 问题：线性模型泛化能力较弱

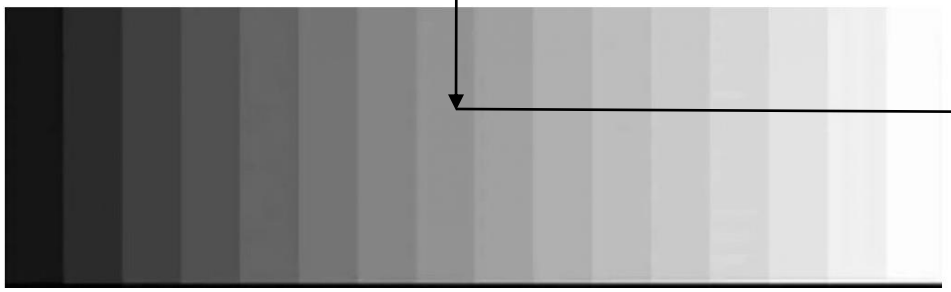


图片分类

图像特征
HoG:方向梯度直方图
SIFT:尺度不变特征变换
EoG:边缘方向直方图

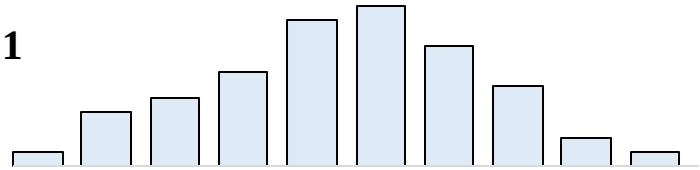
数字：4

$y = Wx$



灰度谱图

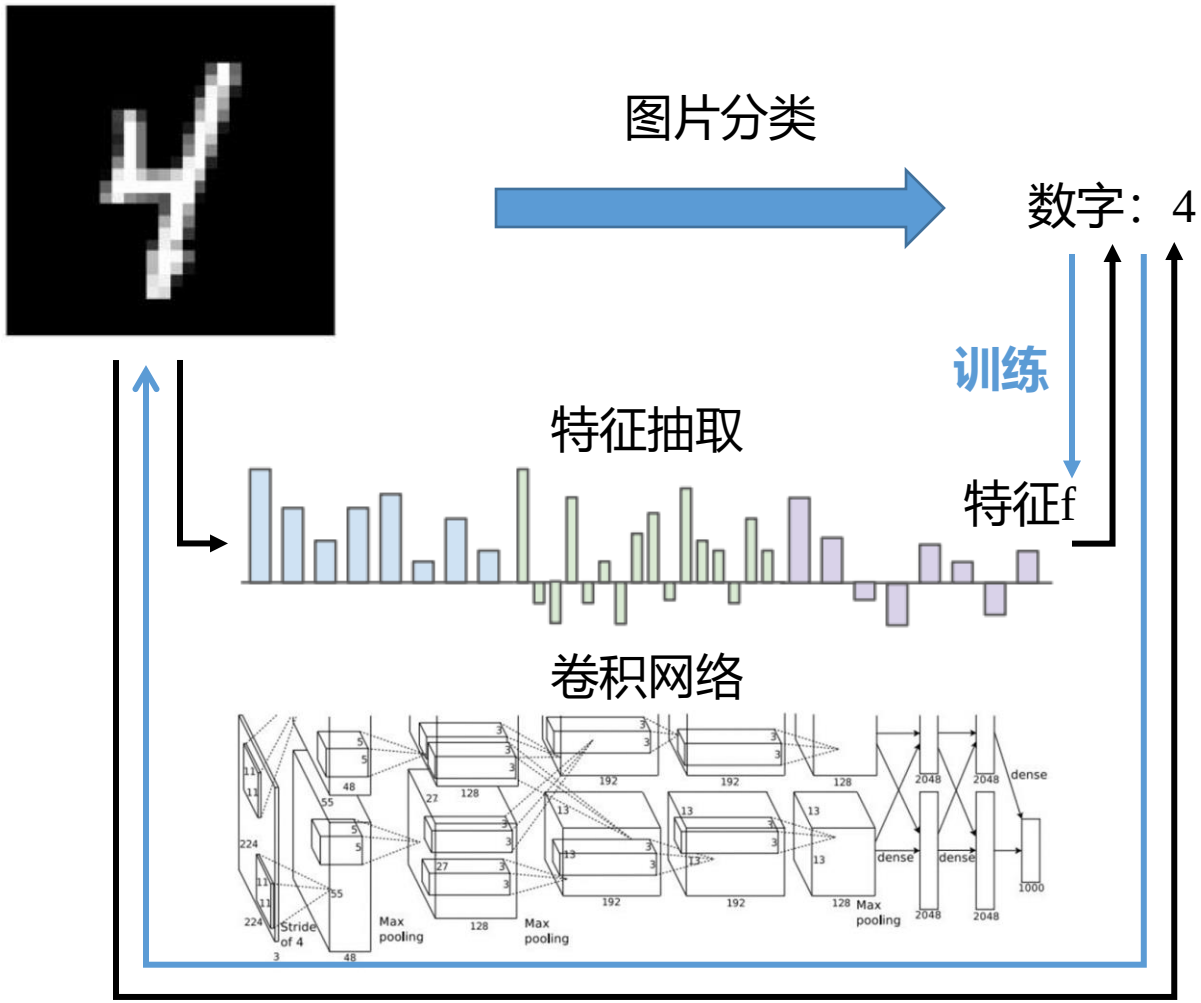
+1



灰度直方图

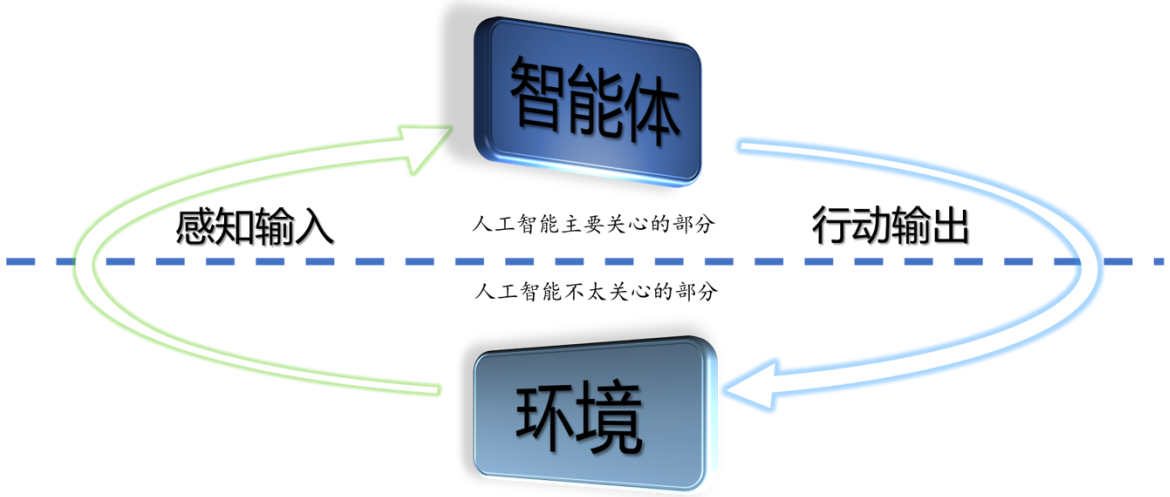
神经网络动机

- 问题：线性模型泛化能力较弱



人工智能：从智能的外延到智能体

- 每一种智能行为X都对应着一种人工X智能，行为X与环境需要进行交互



	人脸识别	对话问答	围棋象棋	机器翻译	数学证明	……
input	人脸	问题	棋盘状态	语言1句子	题目	……
output	ID	回答	下一步落子	语言2句子	答案	……
feedback	正确与否	正确与否	输赢（多步）	正确与否	正确与否（单/多步）	……

深度学习的数学描述

$f_{\phi_l}(x)$为非线性函数，不一定连续

$$\hat{y} = g_\theta(\boldsymbol{x})$$

浅层学习

$$\hat{y} = g_\theta\left(f_{\phi_1}(\boldsymbol{x})\right)$$

$$\hat{y} = g_\theta\left(f_{\phi_2}\left(f_{\phi_1}(\boldsymbol{x})\right)\right)$$

$$\vdots$$

深度学习

$$\hat{y} = g_\theta\left(f_{\phi_{n-1}}\left(\cdots f_{\phi_3}\left(f_{\phi_2}\left(f_{\phi_1}(\boldsymbol{x})\right)\right)\right)\right)$$

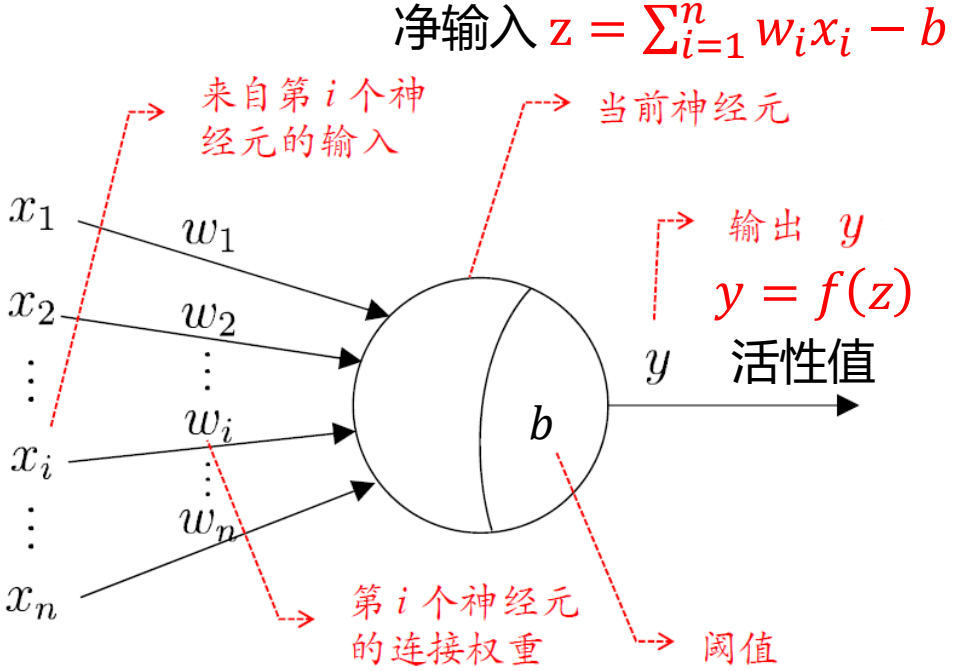
神经元模型—如何表示 $f_{\phi_l}(x)$

- M-P 神经元模型 [McCulloch and Pitts, 1943]

输入：来自其他n个神经云传递过来的输入信号

处理：输入信号通过带权重的连接进行传递, 神经元接受到总输入值将与神经元的阈值进行比较

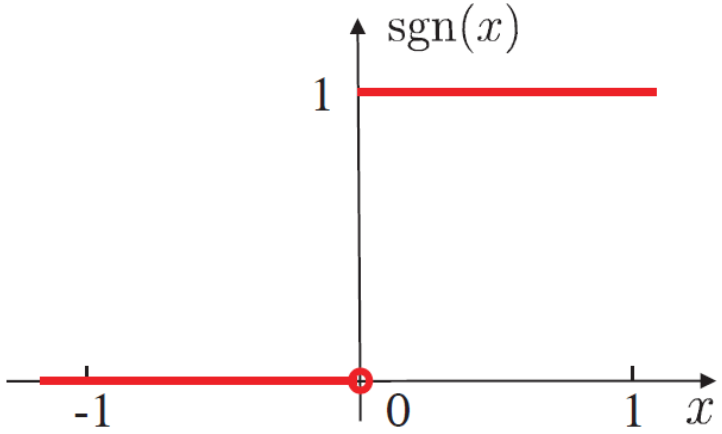
输出：通过激活函数的处理以得到输出



$$y = f(\boldsymbol{w}^{\mathsf{T}}\boldsymbol{x} + b)$$

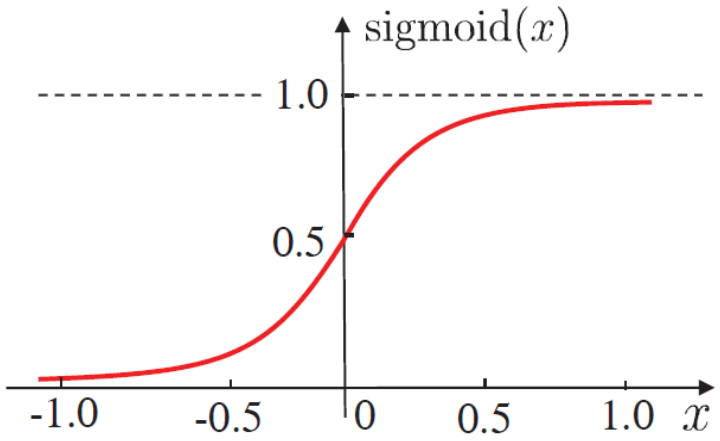
神经元模型—激活函数

口阶跃函数和Sigmoid函数



$$\text{sgn}(x) = \begin{cases} 1, & \text{if } x \geq 0; \\ 0, & \text{if } x < 0. \end{cases}$$

(a) 阶跃函数



$$\text{sigmoid}(x) = \frac{1}{1+e^{-x}}$$

(b) Sigmoid 函数

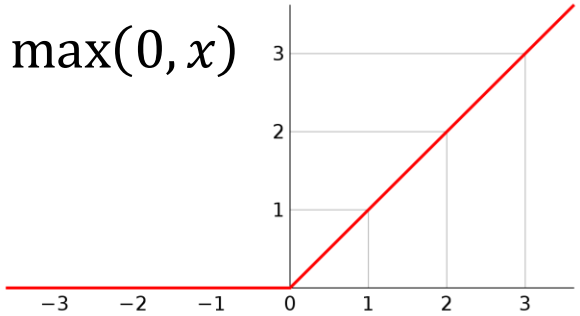
理想激活函数是阶跃函数
0表示抑制神经元；1表示激活神经元

阶跃函数具有不连续、不光滑等不好的性质, 常用的是 Sigmoid 函数

神经元模型—常用激活函数

□ReLU函数

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

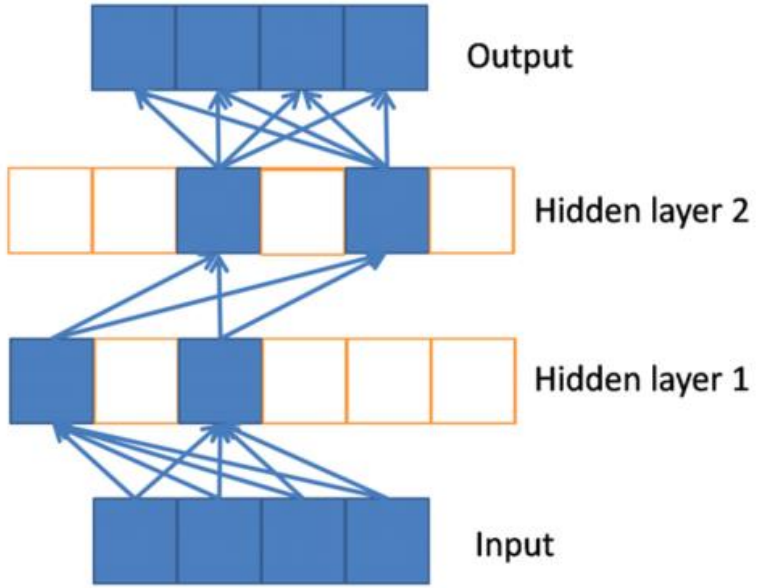


大多数情况下默认选择

人脑在同一时刻只有1%-4%的
神经元处于活跃状态

- 计算上更加高效
- 生物学合理性
  - 单侧抑制、宽兴奋边界
  - 具有很好的稀疏性
- 在一定程度上缓解梯度消失问题

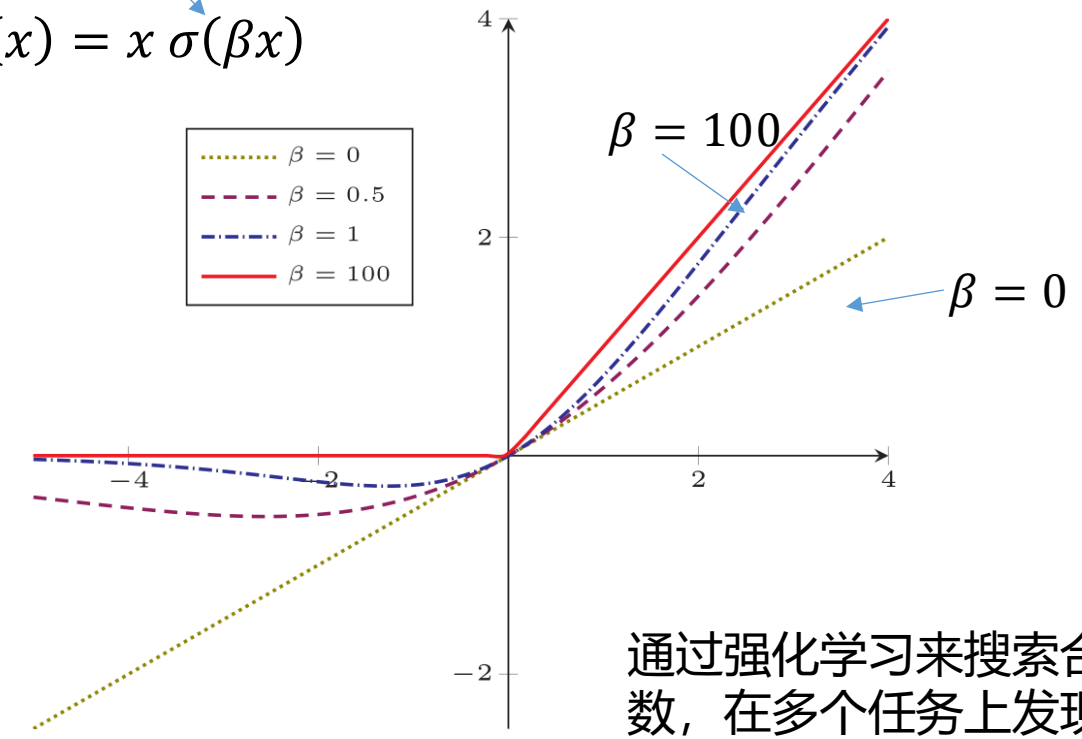
大约50%神经元处于激活状态



神经元模型—常用激活函数

□Swish函数：自门控激活函数

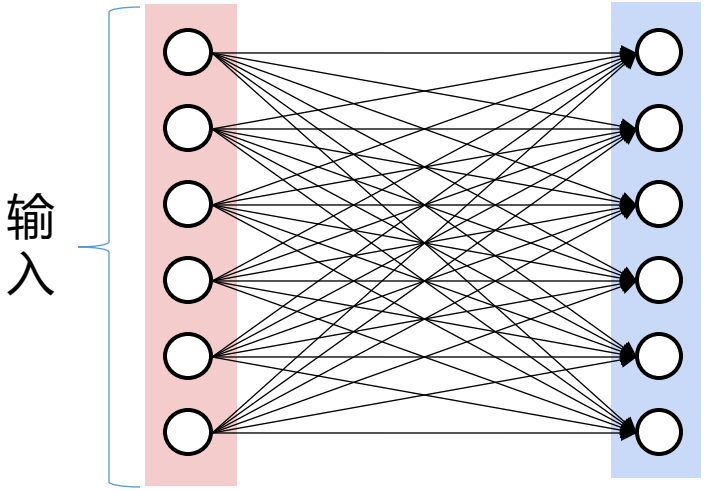
$$swish(x) = x\,\sigma(\beta x)$$



通过强化学习来搜索合适的激活函数，在多个任务上发现Swish函数具有更好的性能

Ramachandran, P., Zoph, B., & Le, Q. V. (2017). Searching for activation functions. *arXiv preprint arXiv:1710.05941*.

神经元模型—常用激活函数



$$y = f(\boldsymbol{w}_1^{\mathrm{T}}\boldsymbol{x} + b_1)$$
$$y = f(\boldsymbol{w}_2^{\mathrm{T}}\boldsymbol{x} + b_2)$$
$$y = f(\boldsymbol{w}_3^{\mathrm{T}}\boldsymbol{x} + b_3)$$
$$y = f(\boldsymbol{w}_4^{\mathrm{T}}\boldsymbol{x} + b_4)$$
$$y = f(\boldsymbol{w}_5^{\mathrm{T}}\boldsymbol{x} + b_5)$$
$$y = f(\boldsymbol{w}_6^{\mathrm{T}}\boldsymbol{x} + b_6)$$

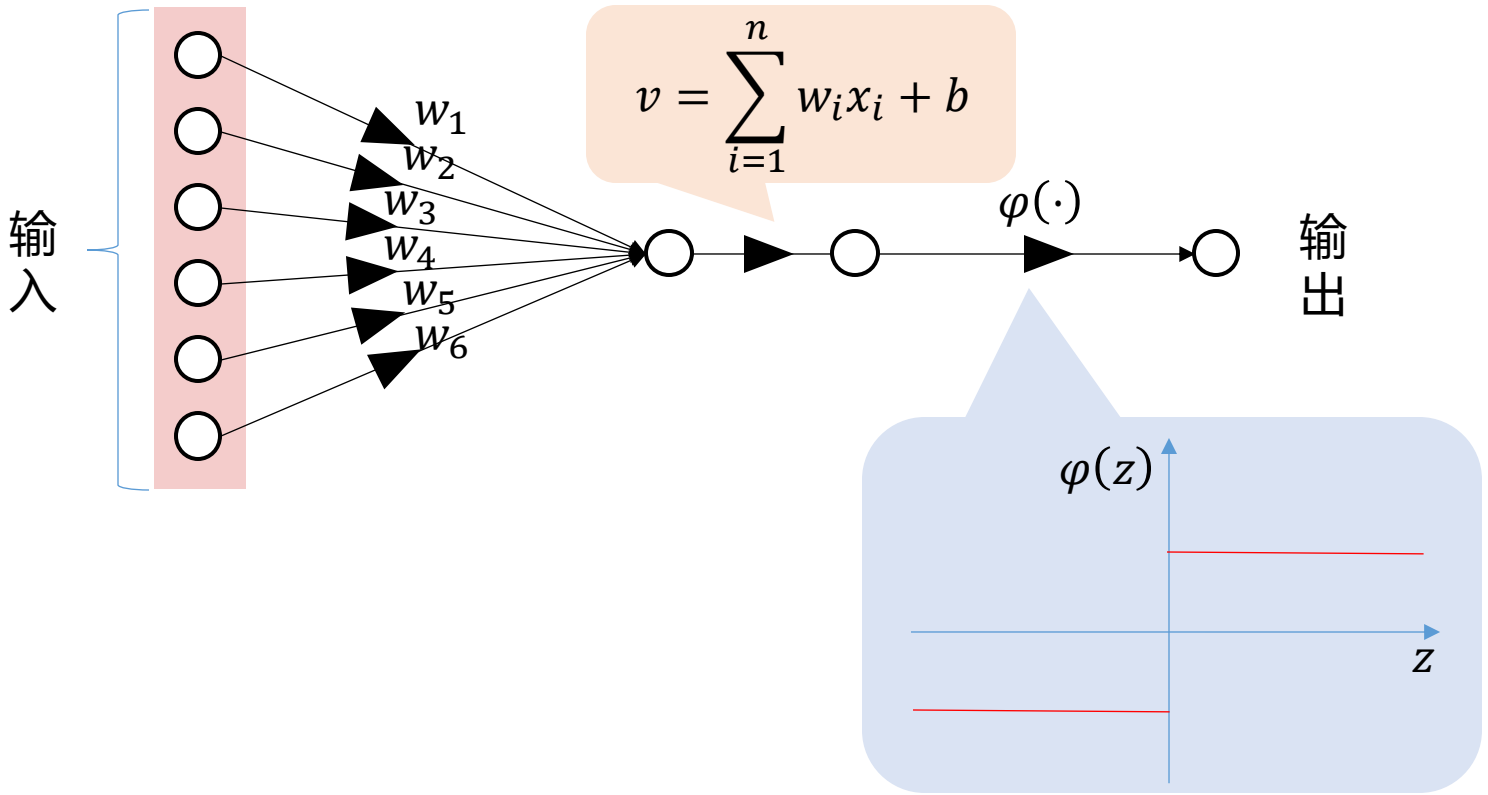
$$y = f(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b})$$

$$\boldsymbol{W} = \begin{bmatrix} \boldsymbol{w}_1^{\mathrm{T}} \\ \boldsymbol{w}_2^{\mathrm{T}} \\ \boldsymbol{w}_3^{\mathrm{T}} \\ \boldsymbol{w}_4^{\mathrm{T}} \\ \boldsymbol{w}_5^{\mathrm{T}} \\ \boldsymbol{w}_6^{\mathrm{T}} \end{bmatrix} \qquad \boldsymbol{b} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \end{bmatrix}$$

目录

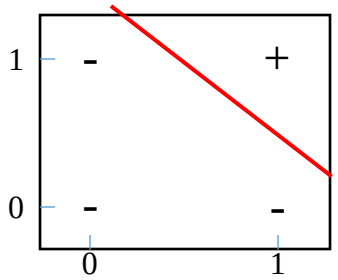
感知机

口感知机由两层神经元组成，输入层接受外界输入信号传递给输出层，输出层是M-P神经元（阈值逻辑单元）

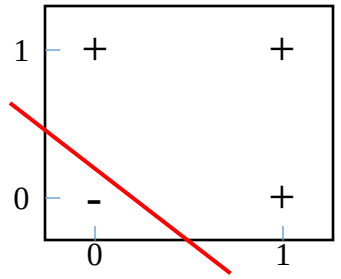


感知机学习

- 与、或、非问题是线性可分的，因此感知机学习过程能够求得适当的权值向量

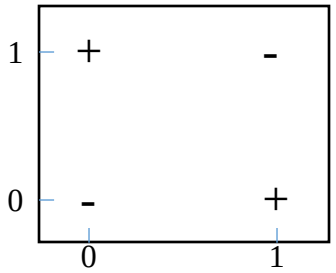


与



或

- 异或问题不是线性可分的，感知机学习不能求得合适解



异或 $x_1 \oplus x_2$

对于非线性可分问题，如何求解？ 多层感知机

神经网络

口神经网络的定义

“神经网络是由具有适应性的简单单元组成的广泛并行互联的网络，它的组织能够模拟生物神经系统对真实世界物体所作出的反应”

[Kohonen, 1988]

口主要三个特性

- 信息表示是分布式的
- 记忆和知识是存储在单元之间的连接上的
- 通过逐渐改变单元之间的连接强度来学习新知识

网络结构

□神经网络设计的另一个关键点是确定它的结构

- 具有多少单元，以及这些单元应该如何连接。

□大多数神经网络被组织成层的单元组

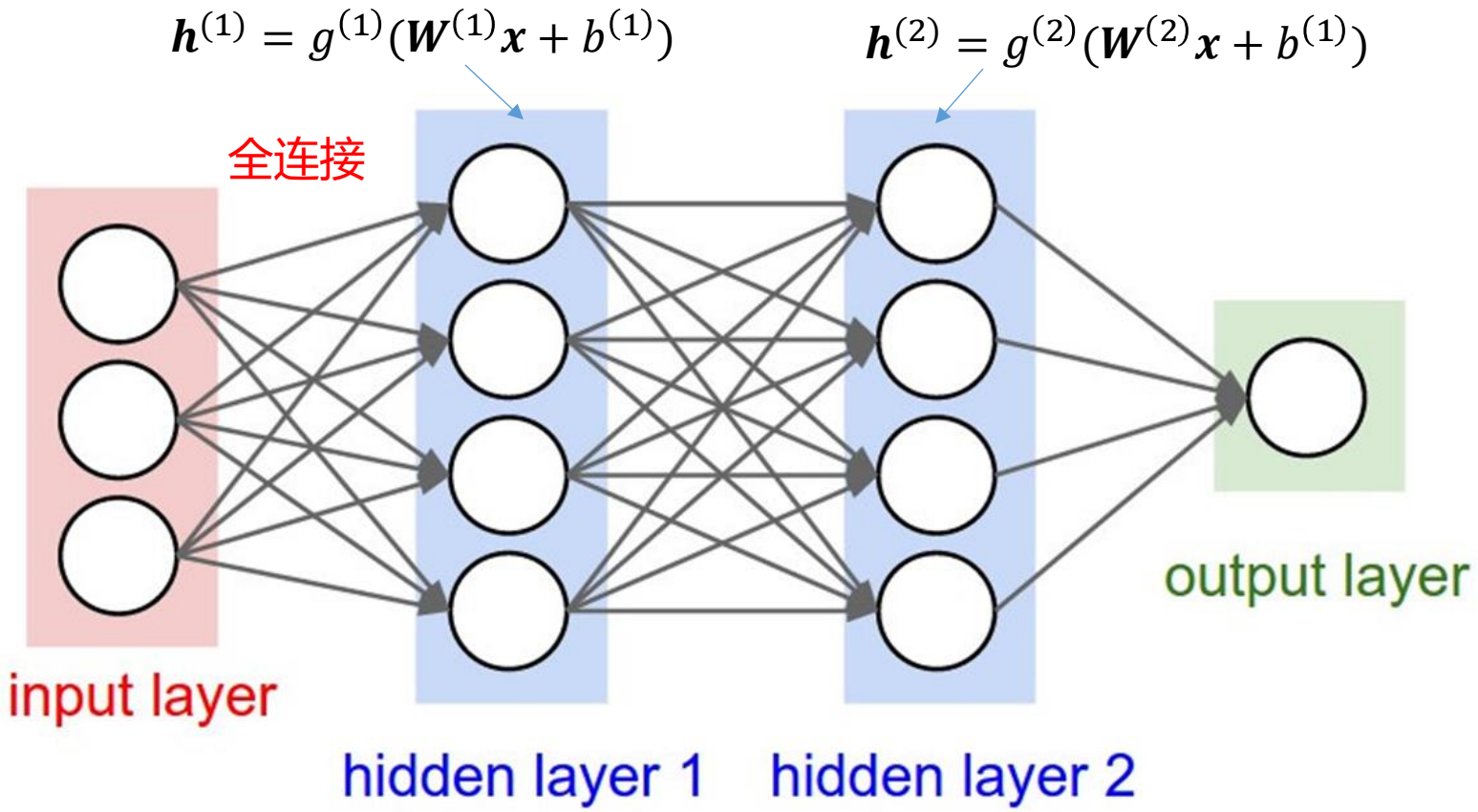
□大多数神经网络架构将这些层布置成链式结构，其中每一层都是前一层的函数

□神经网络设计在于选择网络的深度和每一层的宽度

更深层网络通常能在每一层使用更少的单元数和更少的参数，并且有更强的泛化能力。但是通常也更难以优化

多层感知机

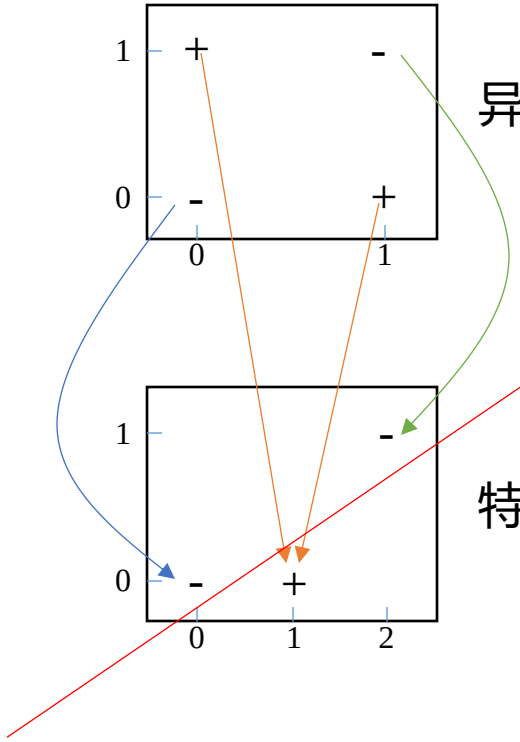
- 输出层与输入层之间的一层神经元，被称之为**隐层或隐含层**，隐含层和输出层神经元都是具有激活函数的功能神经元



多层感知机

能解决异或问题的两层感知机示例

- $\boldsymbol{h} = \max\left(0, \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} \boldsymbol{x} + \begin{bmatrix} 0 \\ -1 \end{bmatrix}\right)$
- $y = [1, -2]\,\boldsymbol{h} - 0.5$



异或 $x_1 \oplus x_2$

$\boldsymbol{h}$ 将原始数据映射入特征空间

y 特征空间中的线性分类器

特征空间

目录

万能近似定理 Universal Approximation Theorem

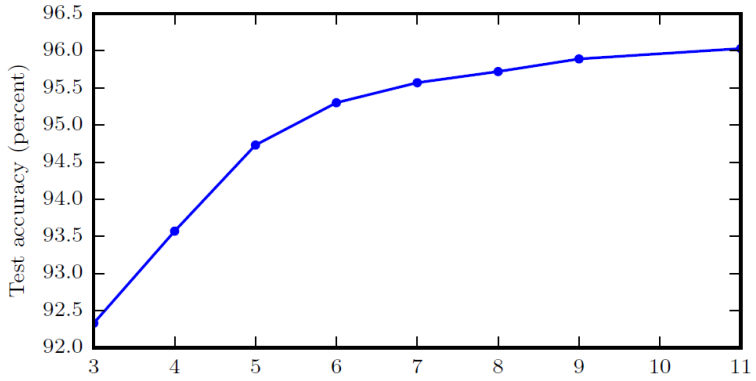
一个前馈神经网络如果具有线性输出层和至少一层具有任何一种“挤压”性质的激活函数（例如logistic sigmoid激活函数）的隐藏层，只要给予网络足够数量的隐藏单元，它可以以任意的精度来近似任何从一个有限维空间到另一个有限维空间的大多数常见函数[1]。

万能近似定理也已经被证明对于更广泛类别的激活函数，比如ReLU也是适用的

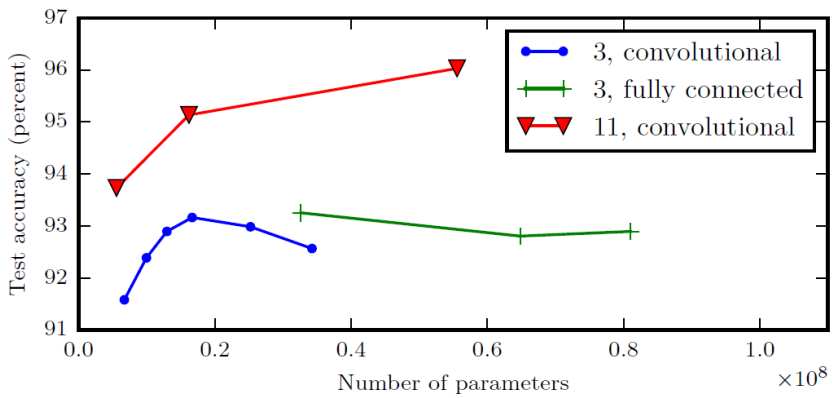
[1]: “大多数常见函数”指 Borel 可测函数，包括了在$\mathbb{R}^n$的有限闭集上的连续函数，涵盖绝大多数常见函数。

万能近似定理 Universal Approximation Theorem

- 浅层神经网络需要与输入维度呈**指数级增长的隐藏单元数量**$O(\exp(d))$ **才能有效逼近目标函数，这可能导致模型难以学习和泛化**
- 通过**增加网络深度**，可以减少所需的神经元数量，从而提升模型的表达能力，并降低泛化误差。
- 某些函数族在网络深度**超过某个阈值** d**时，可以以更紧凑逼近目标函数，而如果深度受限于**d**或更小**，则需要远超深层网络的单元数才能达到相同的逼近效果。
- 相较于浅层网络，深层网络在表达复杂函数时更加高效，能够以较少的参数实现更强的逼近能力。



地址号码转录测试集上的性能随着深度的增加而不断增加



模型尺寸增加并不能产生相同的效果

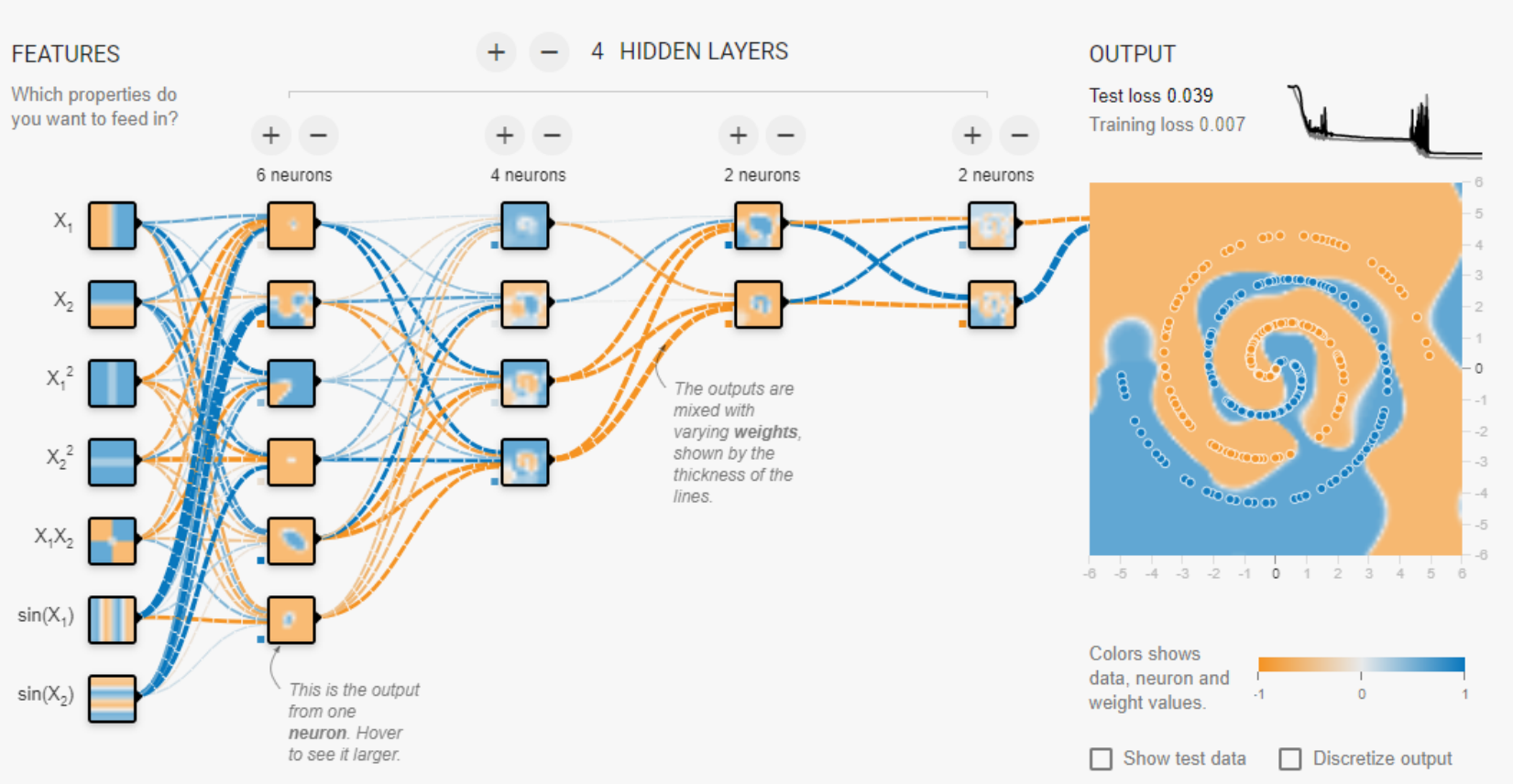
万能近似定理 Universal Approximation Theorem

口万能近似定理只说明神经网络表达能力强大到可以近似任意一个的连续函数，却并没有给出如何找到这样的神经网络，以及是否是最优的

- 神经网络非线性，其学习是非凸优化问题，优化算法可能找不到近似函数的参数值
- 过于强大的表达能力，训练算法可能由于过拟合而选择了错误的函数

神经网络动机

□ 演示：多层前馈神经网络处理线性不可分数据



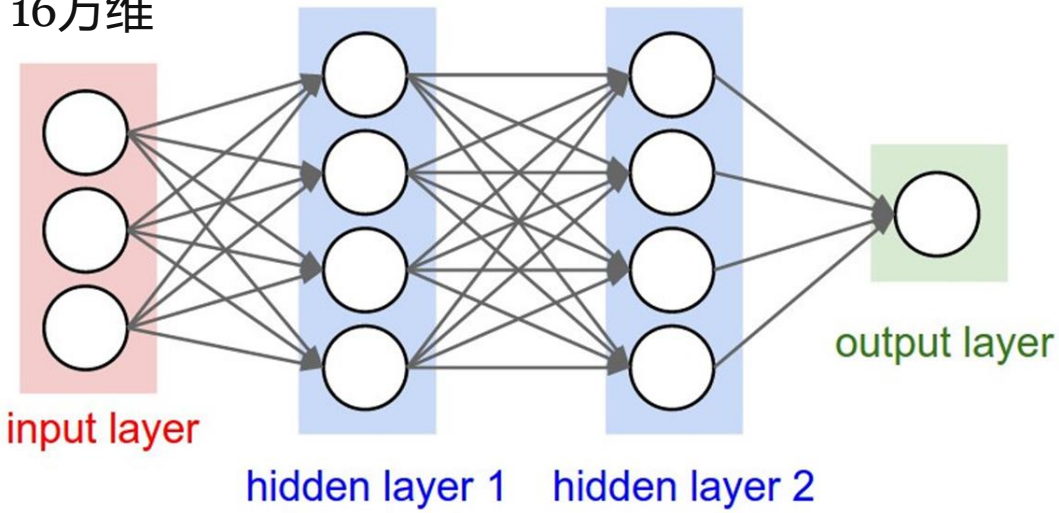
http://playground.tensorflow.org/

目录

全连接网络的问题—参数量可能巨大

一张 400 x 400 的图片，一共包含了 16 万个像素点

16万维



$W^{(l)}$参数很多

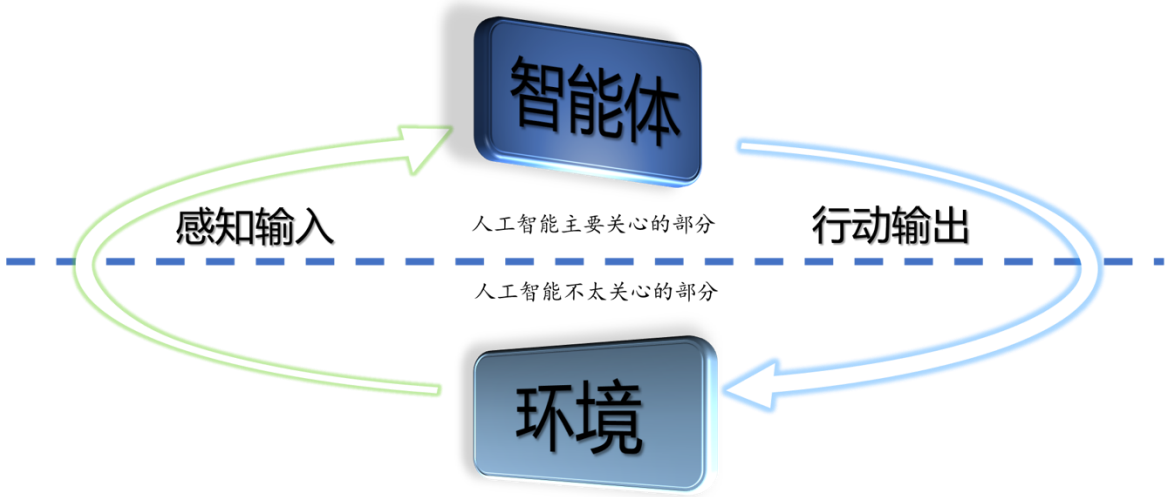
M_l	第 l 层神经元的个数
$f_l(\cdot)$	第 l 层神经元的激活函数
$\boldsymbol{W}^{(l)} \in \mathbb{R}^{M_l \times M_{l-1}}$	第 $l-1$ 层到第 l 层的权重矩阵
$\boldsymbol{b}^{(l)} \in \mathbb{R}^{M_l}$	第 $l-1$ 层到第 l 层的偏置
$\boldsymbol{z}^{(l)} \in \mathbb{R}^{M_l}$	第 l 层神经元的净输入（净活性值）
$\boldsymbol{a}^{(l)} \in \mathbb{R}^{M_l}$	第 l 层神经元的输出（活性值）

习题：在如图所示的四层全连接神经网络中，输入层维度为 160,000，第一个隐藏层维度为 40,000，第二个隐藏层维度为 2,000，输出层维度为 100。请计算该神经网络的参数总量。

目录

人工智能：从智能的外延到智能体

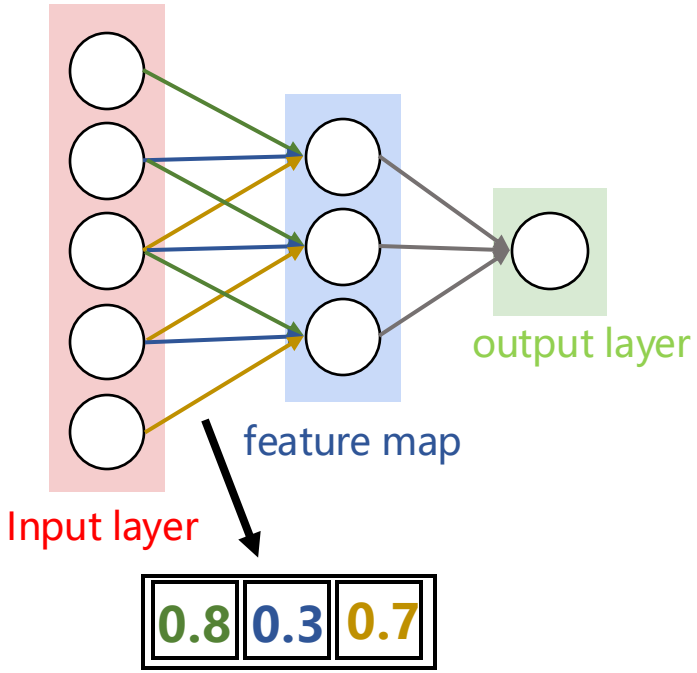
- 每一种智能行为X都对应着一种人工X智能，行为X与环境需要进行交互

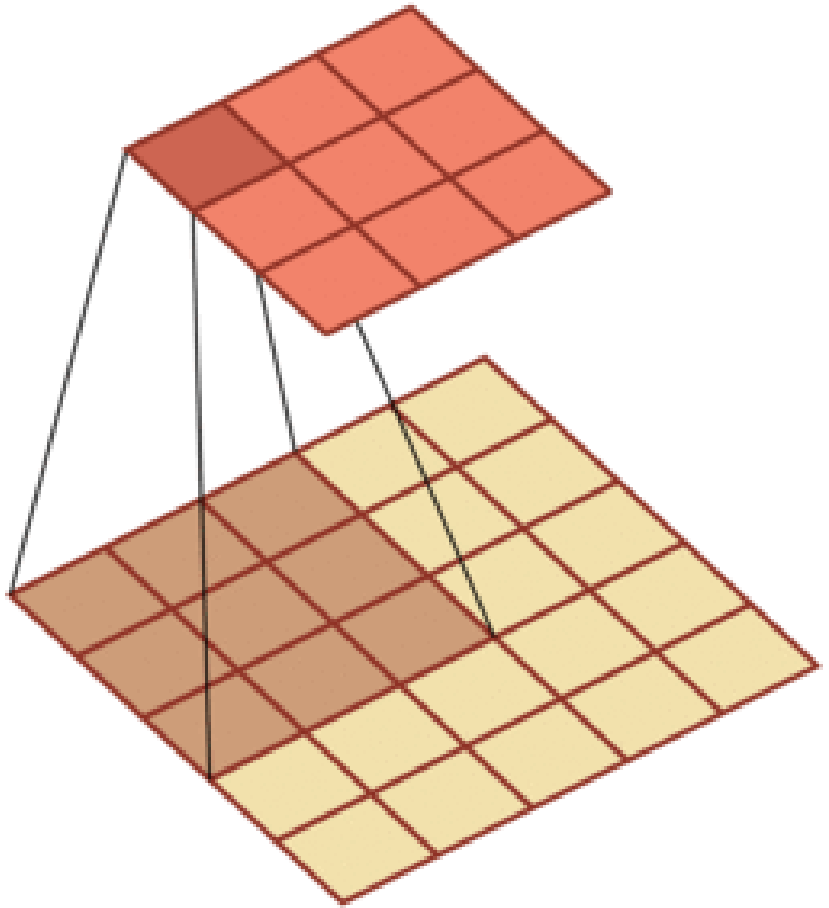


	人脸识别	对话问答	围棋象棋	机器翻译	数学证明	……
input	人脸	问题	棋盘状态	语言1句子	题目	……
output	ID	回答	下一步落子	语言2句子	答案	……
feedback	正确与否	正确与否	输赢（多步）	正确与否	正确与否（单/多步）	……

共享参数的前馈网络——卷积

卷积操作示例（一维、二维）



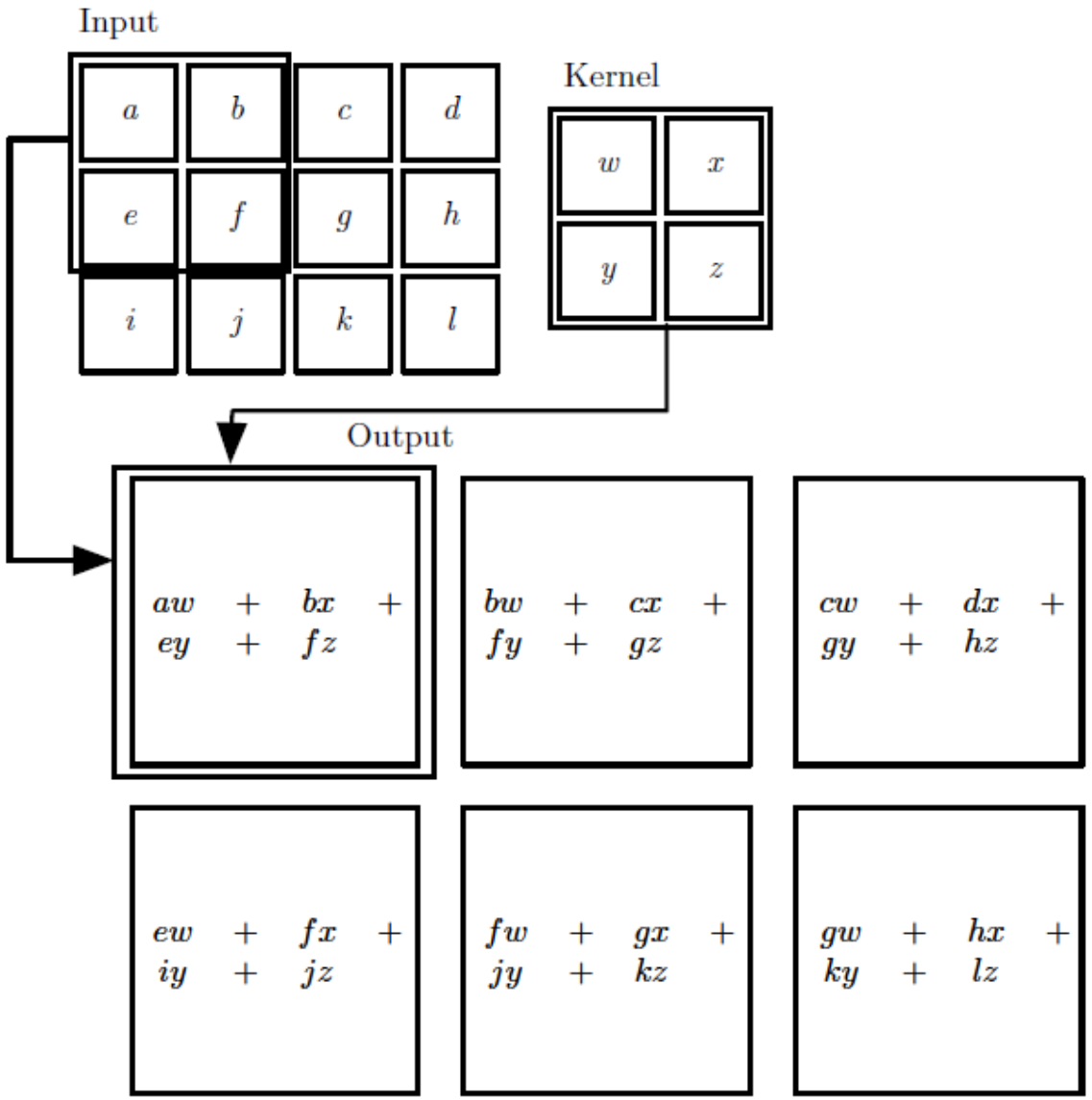


一维卷积核 对应 **参数矩阵** $w^{(l)}$

卷积操作可共享卷积核参数

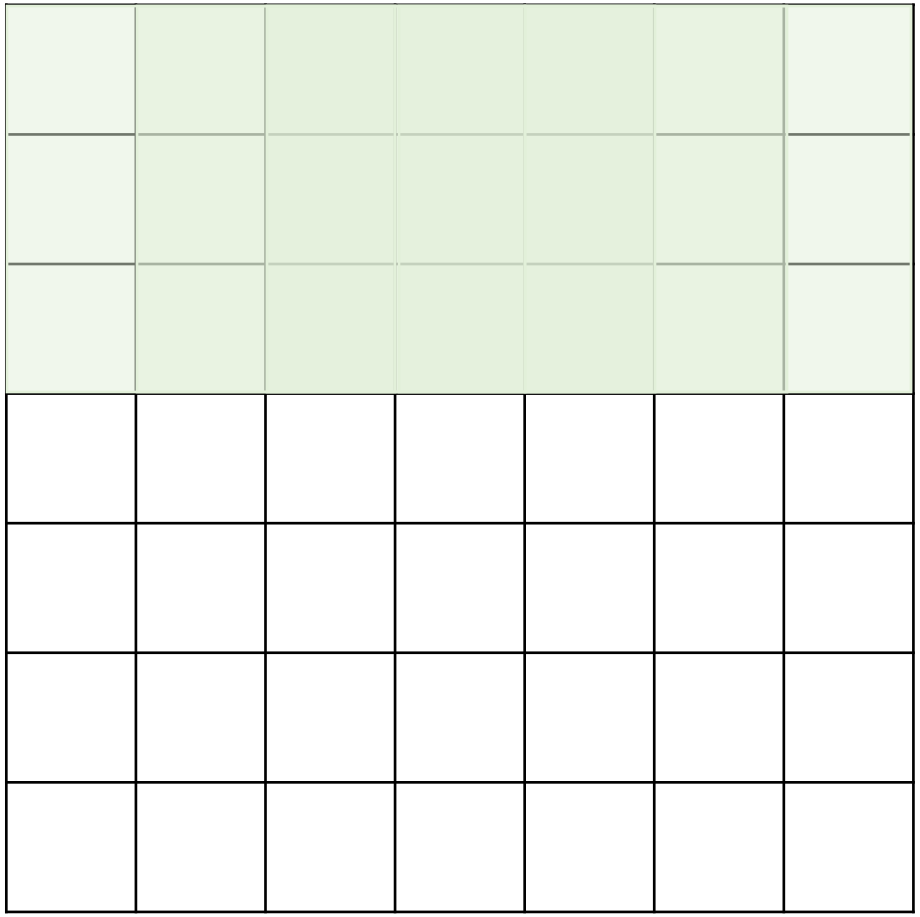
全连接层参数量： 5*3 = 15
卷积层参数量： 3

卷积



$$y_{ij} = \sum_{u=1}^{U} \sum_{v=1}^{V} w_{uv} x_{i+u+1,j+v+1}$$

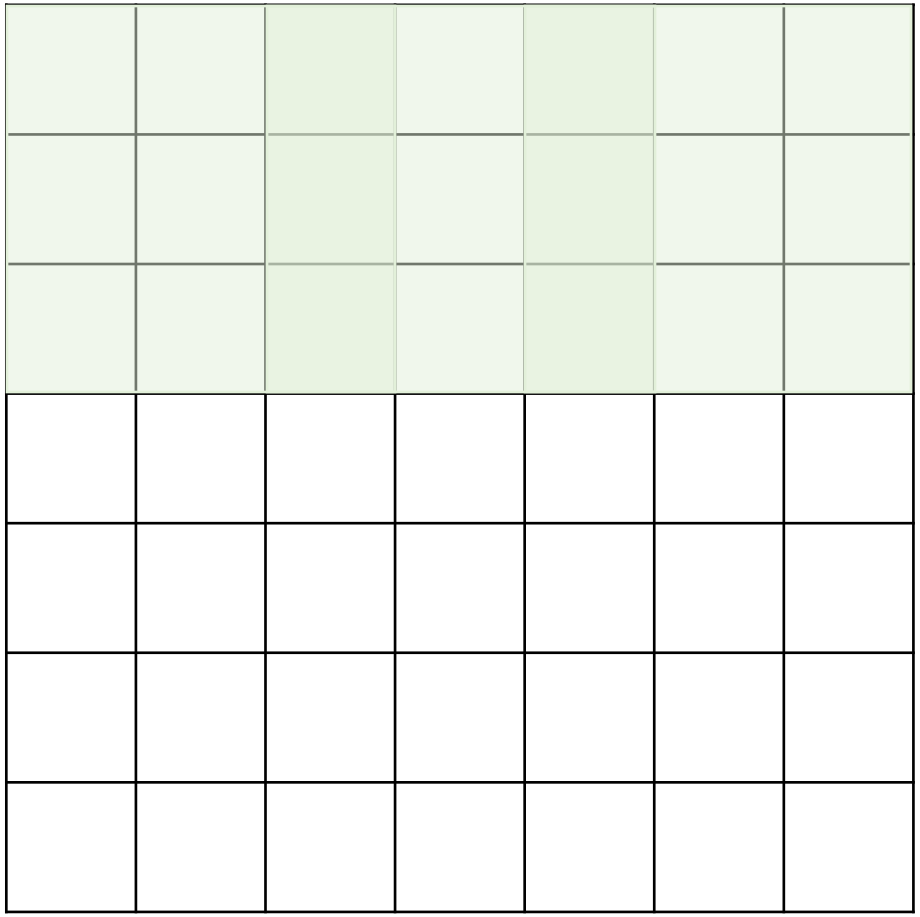
卷积



7x7的输入
3x3的核

5x5的输出

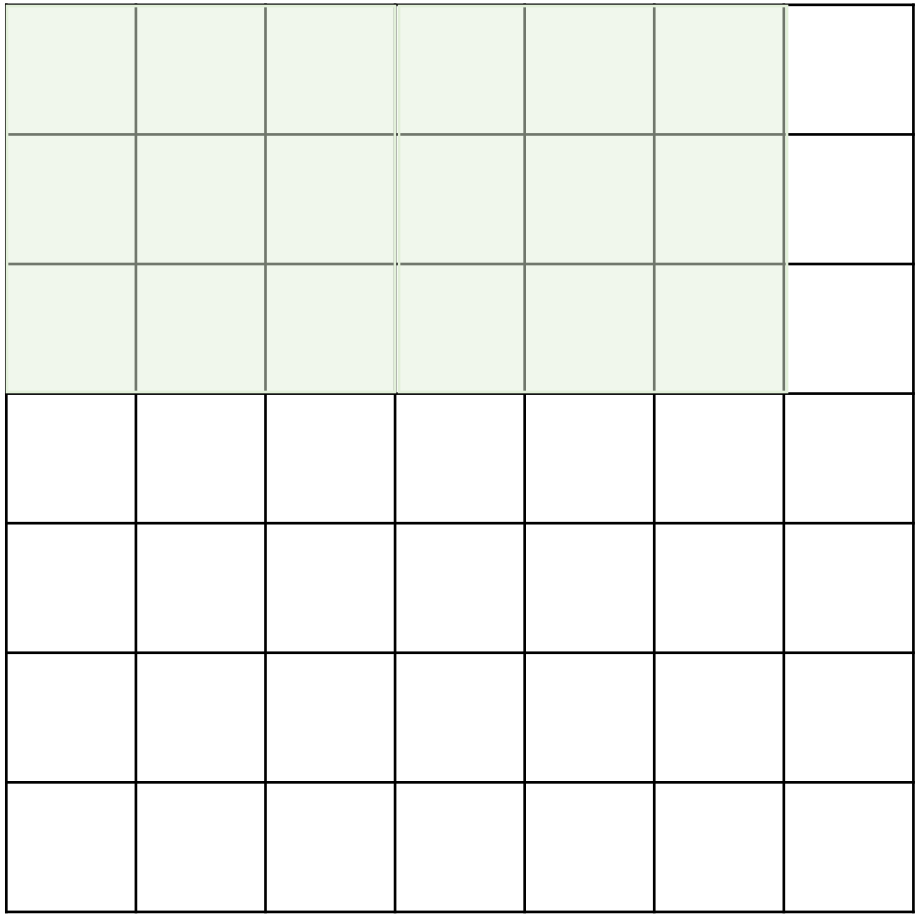
卷积



7x7的输入
3x3的核
步幅为2

3x3的输出

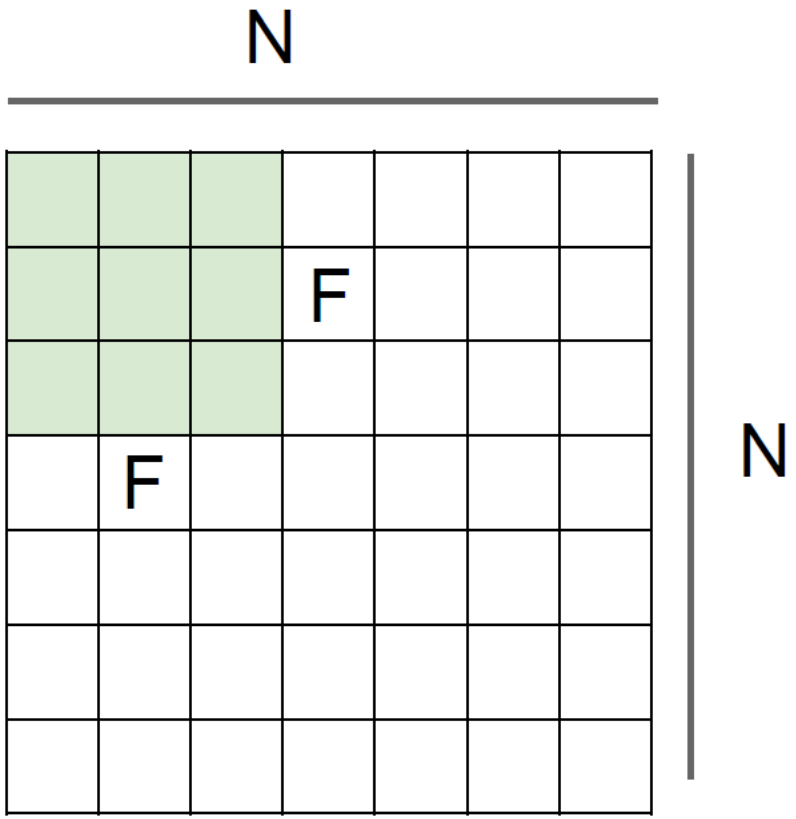
卷积



7x7的输入
3x3的核
步幅为3

不匹配

卷积



口输出大小：(N-F)/S+1

- S为步幅大小

口例：N=7, F=3

- 步幅为1，(7-3)/1+1=5
- 步幅为2，(7-3)/2+1=3
- 步幅为3，(7-3)/3+1=2.3

卷积的直观理解

原始图像

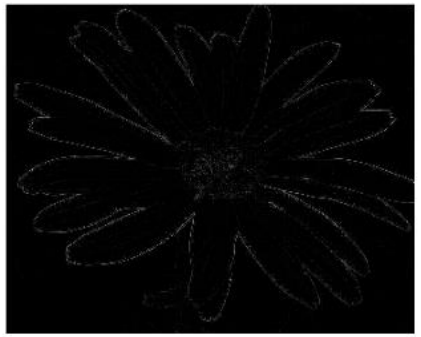
$\otimes$

$\frac{1}{16}$	$\frac{1}{8}$	$\frac{1}{16}$
$\frac{1}{8}$	$\frac{1}{4}$	$\frac{1}{8}$
$\frac{1}{16}$	$\frac{1}{8}$	$\frac{1}{16}$

$=$

0	1	0
1	-4	1
0	1	0

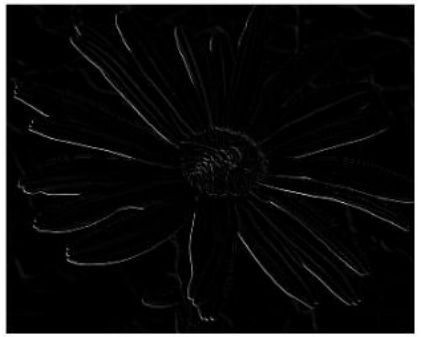
$=$



0	1	1
-1	0	1
-1	-1	0

滤波器

$=$



输出特征映射

卷积的直观理解

原始图像

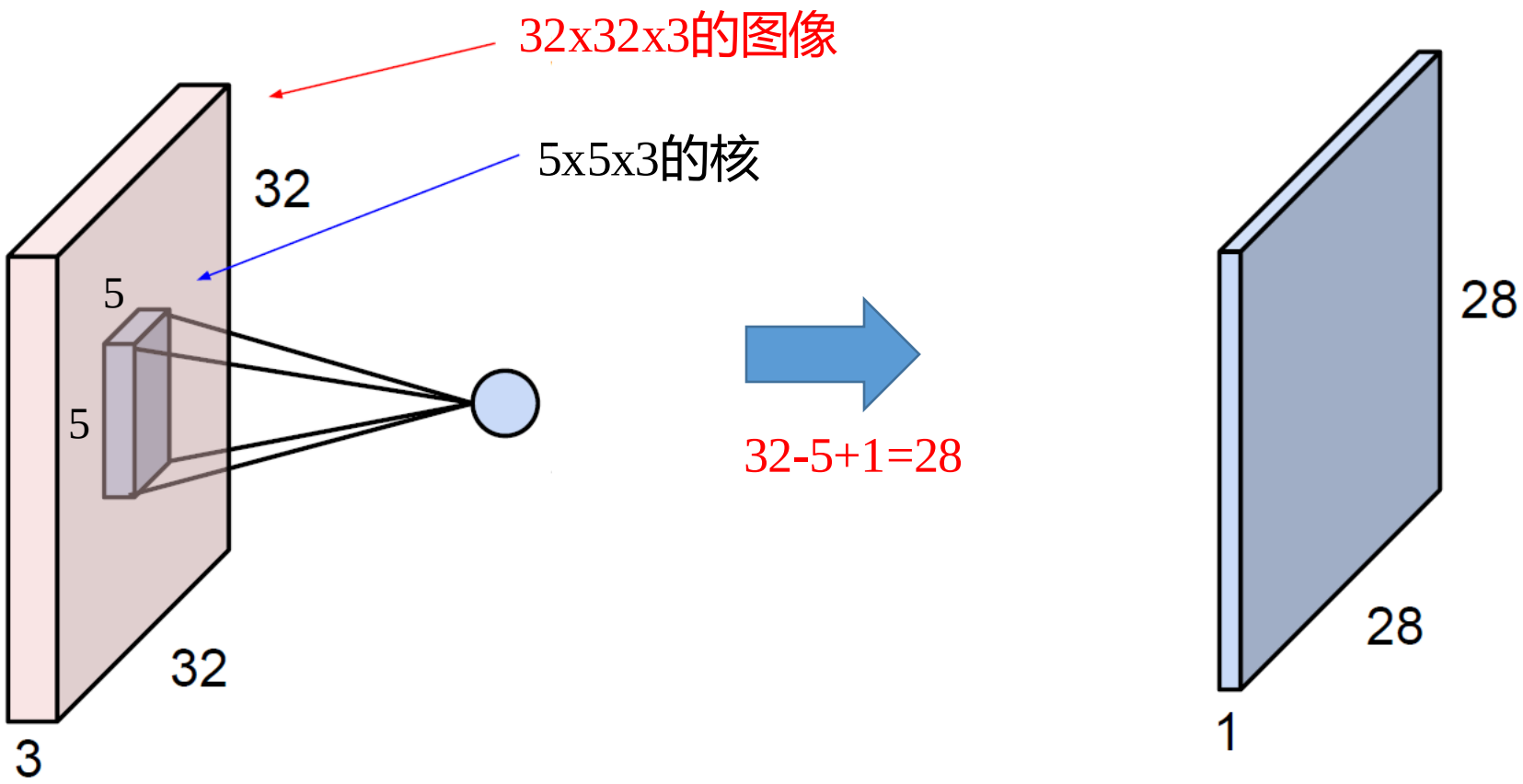
低通滤波

图像的高频区域变成低频，即色彩变化剧烈的区域变得平滑，也就是出现模糊效果

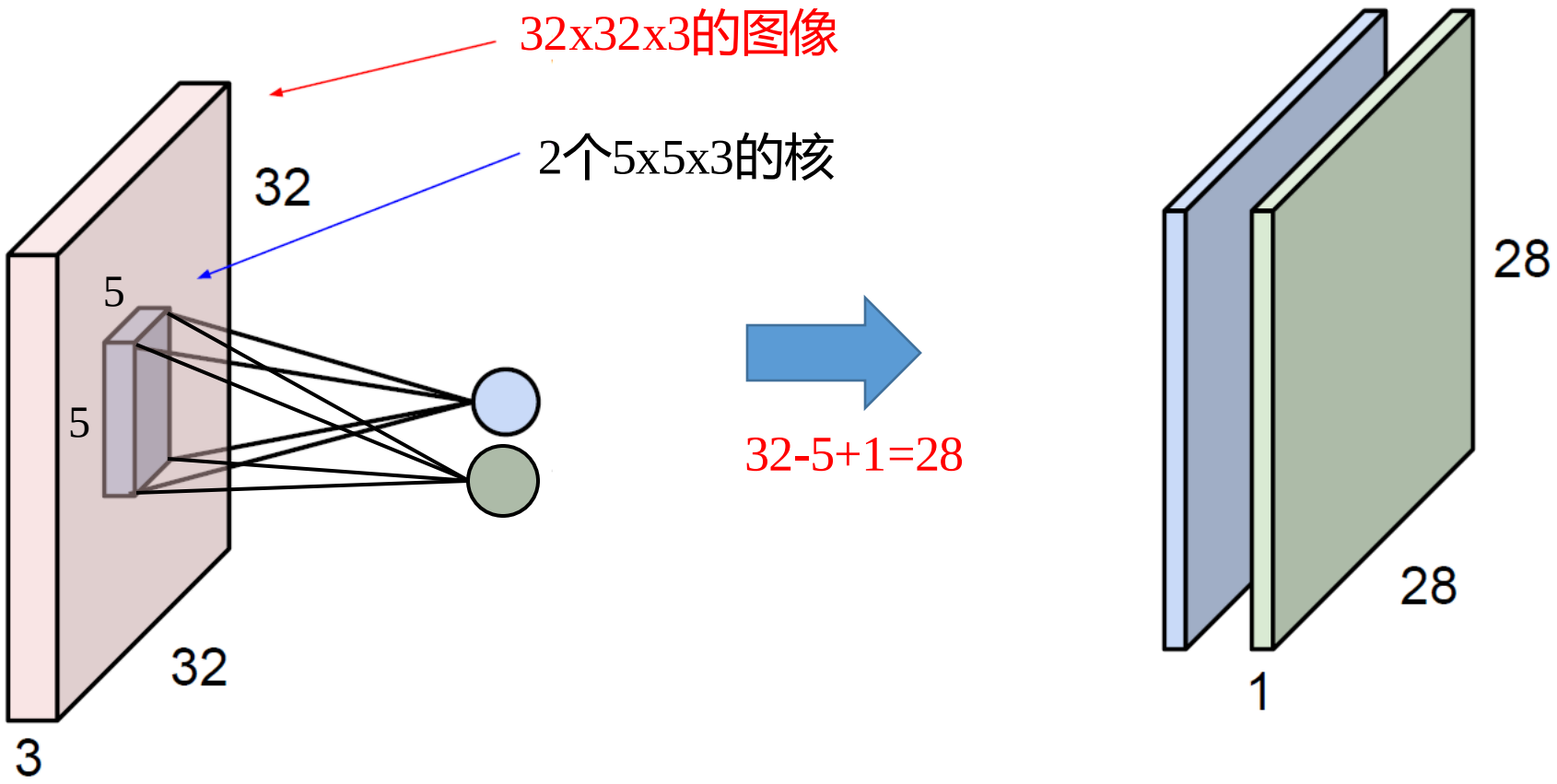
高通滤波

只保留那些变化最快速最剧烈的区域，也就是图像里面的物体边缘

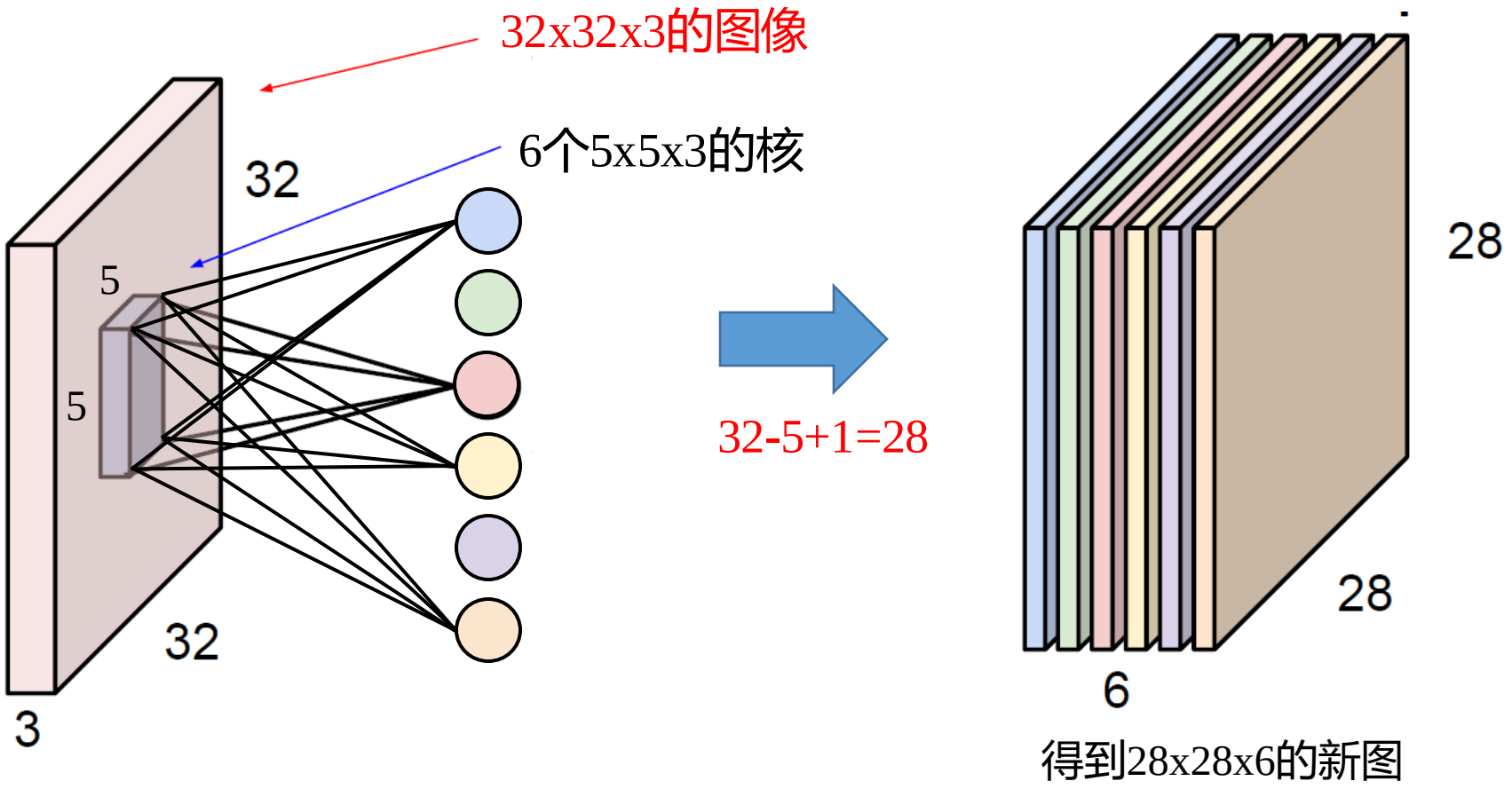
卷积



卷积

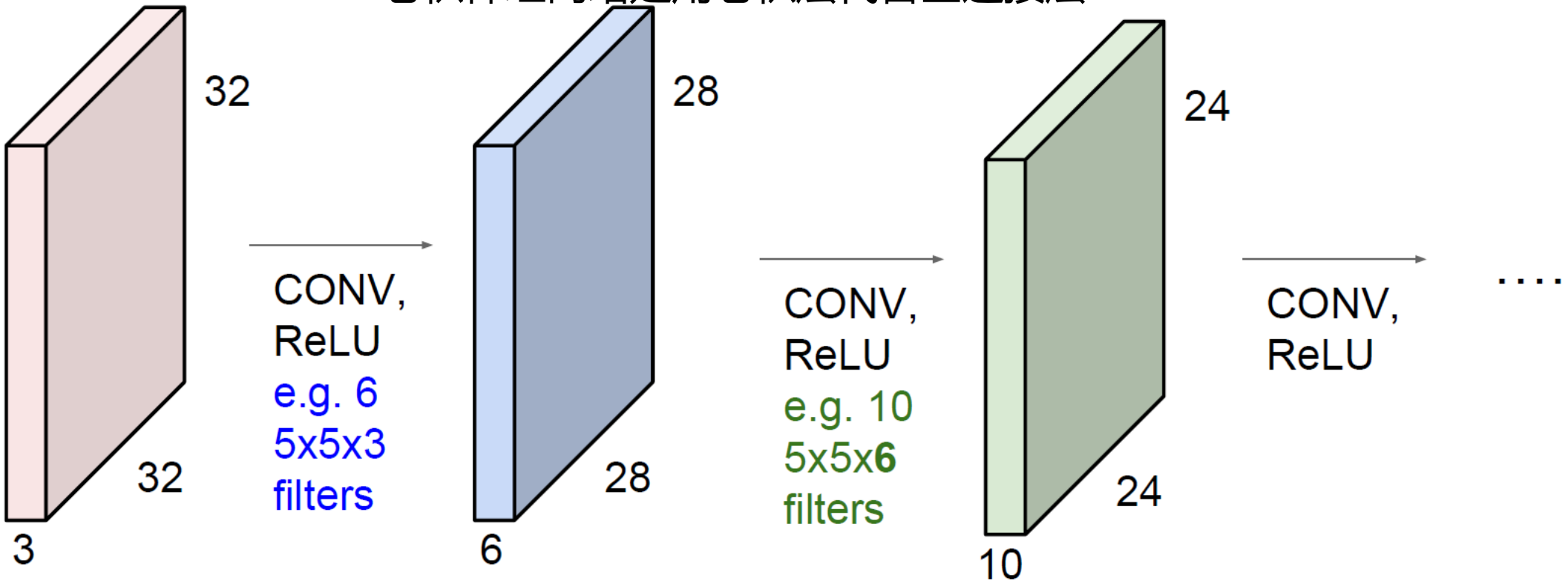


卷积



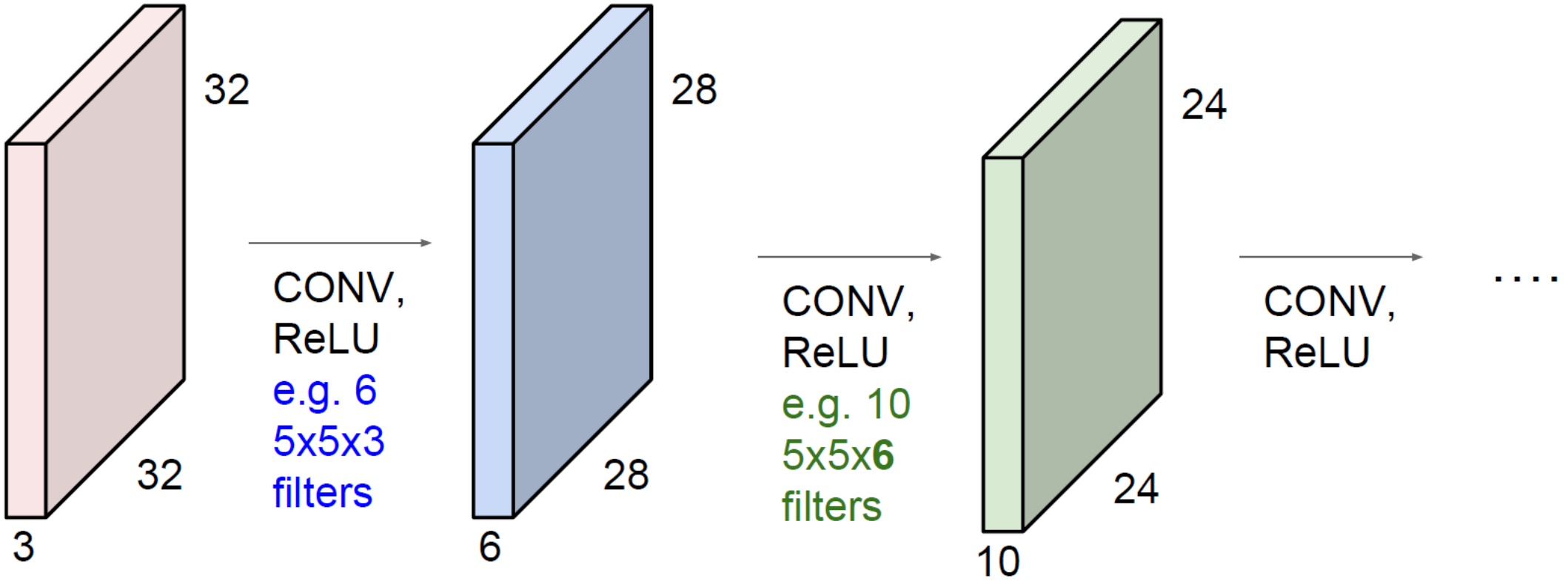
多层卷积

卷积神经网络是用卷积层代替全连接层

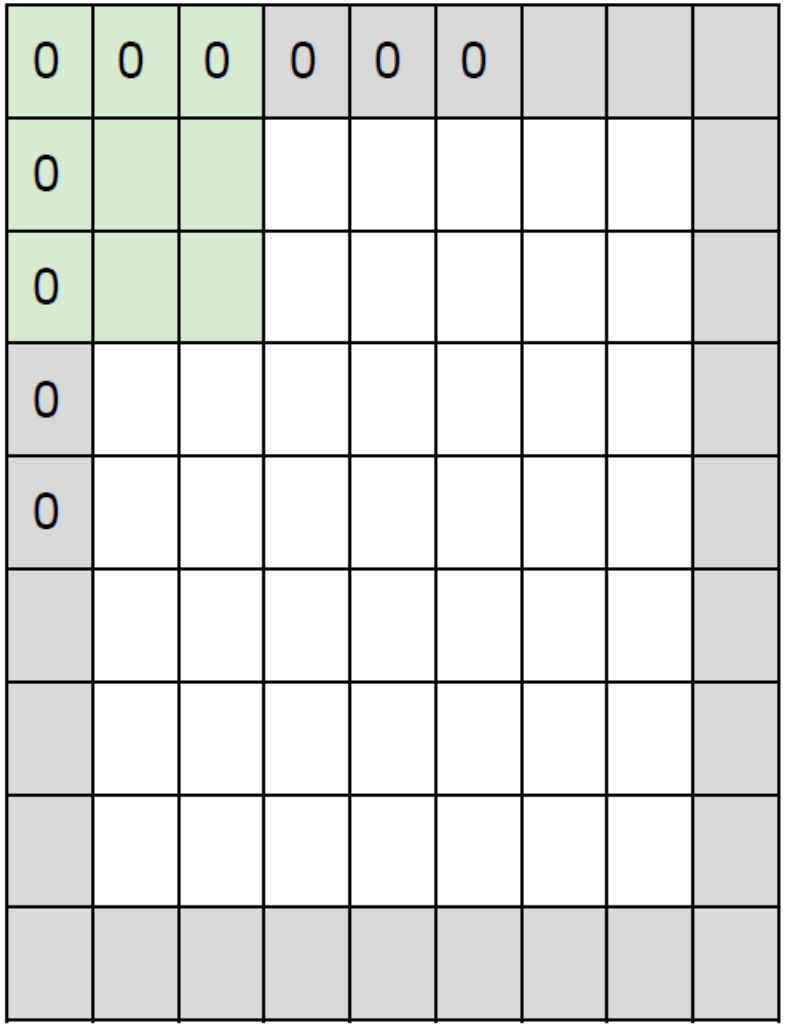


多层卷积

32x32 的输入，每次通过 5x5 的核卷积，大小都会发生改变!
(32 -> 28 -> 24 ...)。**大小改变太快是不好的，实际中的性能很差**

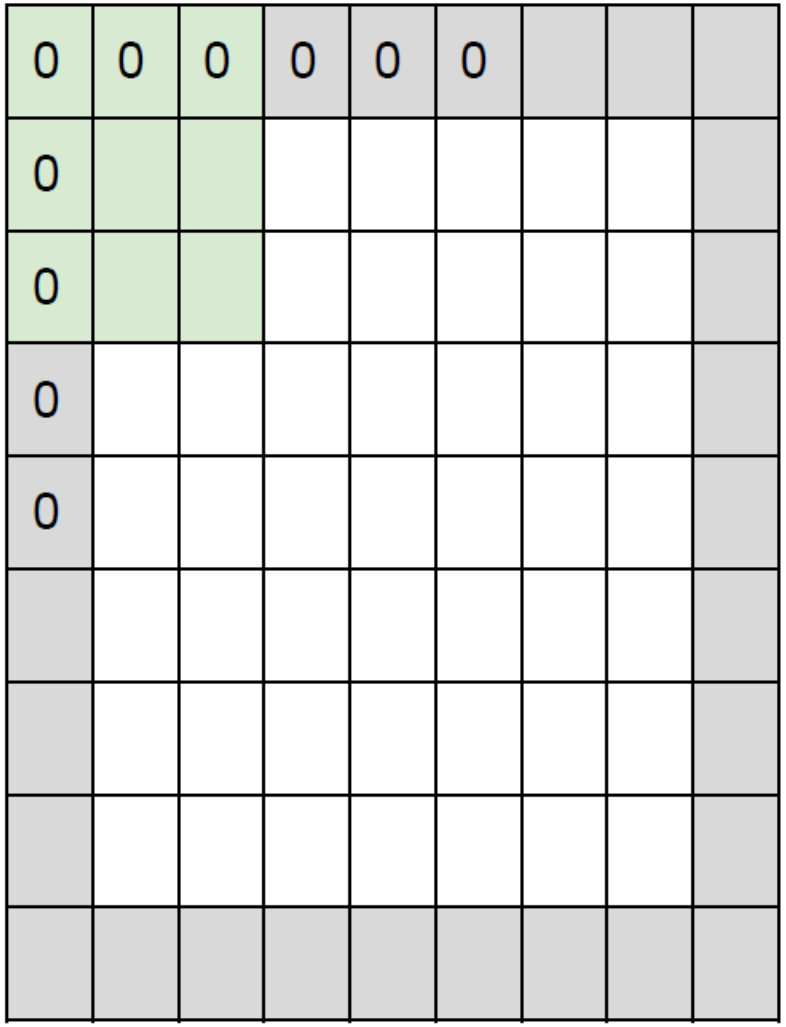


填充（PADDING）



- 在输入周边填0，以解决特征图大小改变太快问题
- 例子1
  - 7x7的输入，3x3的核，步幅为1
  - 填一个像素的边界，输入实际上是9x9
  - 输出大小为：(9-3)/1+1 = 7
  - 输出7x7，与输入相同大小
- 例子2
  - 7x7的输入，3x3的核，步幅为3
  - 填一个像素边界
  - 输出大小为：(9-3)/3+1=3

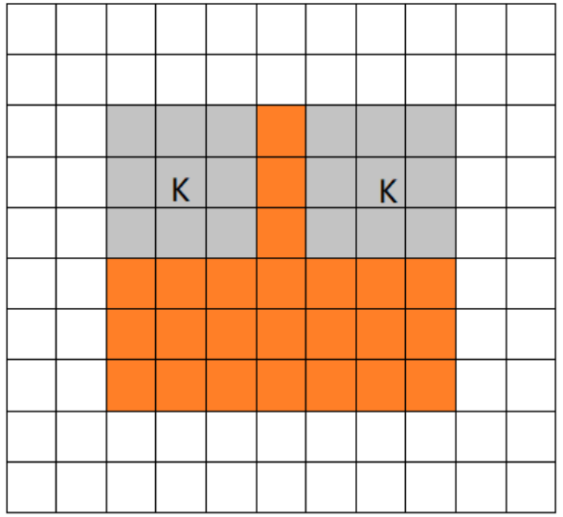
填充（PADDING）



- 填0时，经常要求保持输出大小与输入大小一样
- 步幅为1，FxF的核，填(F-1)/2的0
- 步幅为S，FxF的核，填
  - $\frac{(S-1)N+F-S}{2}$的0

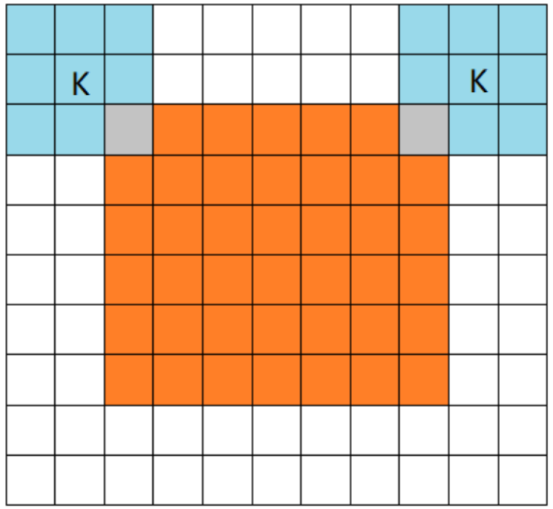
卷积模式

有效(valid)卷积
窄卷积



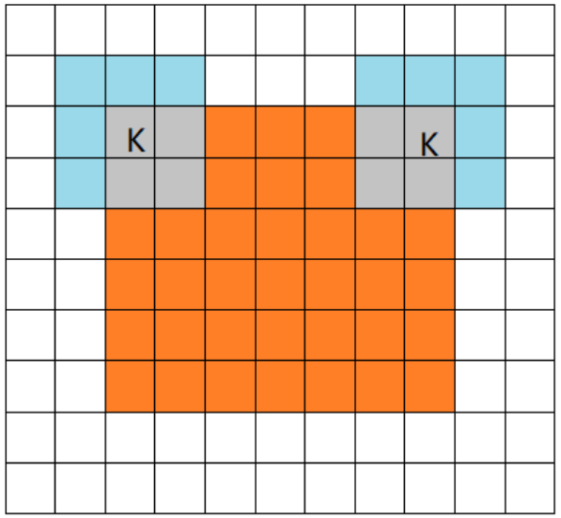
卷积核只在特征图内卷积

全(full)卷积
宽卷积



从卷积核和特征图刚相交
开始卷积

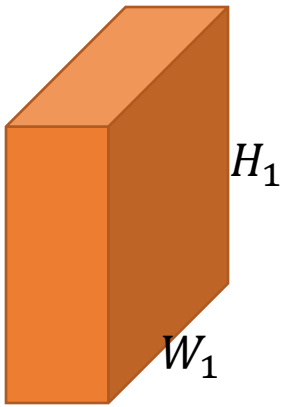
相同(same)卷积
等宽卷积



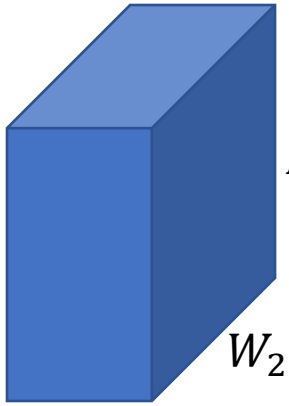
特征图的大小保持不变

通常零填充的最优数量（对于测试集的分类正确率）处于“有效卷积” 和“相同卷积”
之间的某个位置

卷积层的信息概括



核数K
核大小F
步幅S
补0数P



$H_2 = (H_1 + 2P - F)/S + 1$

$W_2 = (W_1 + 2P - F)/S + 1$

$D_2 = K$

常用设置：K为2的幂，32,64,128,512

- F=3, S=1, P=1
- F=5, S=1, P=2
- F=1, S=1, P=0

参数个数?

$F \times F \times D_1 \times K$个权重
K个偏置项

Pytorch中的卷积层

CLASS `torch.nn.Conv2d(in_channels, out_channels, kernel_size, stride=1, padding=0, dilation=1, groups=1, bias=True, padding_mode='zeros')` [SOURCE]

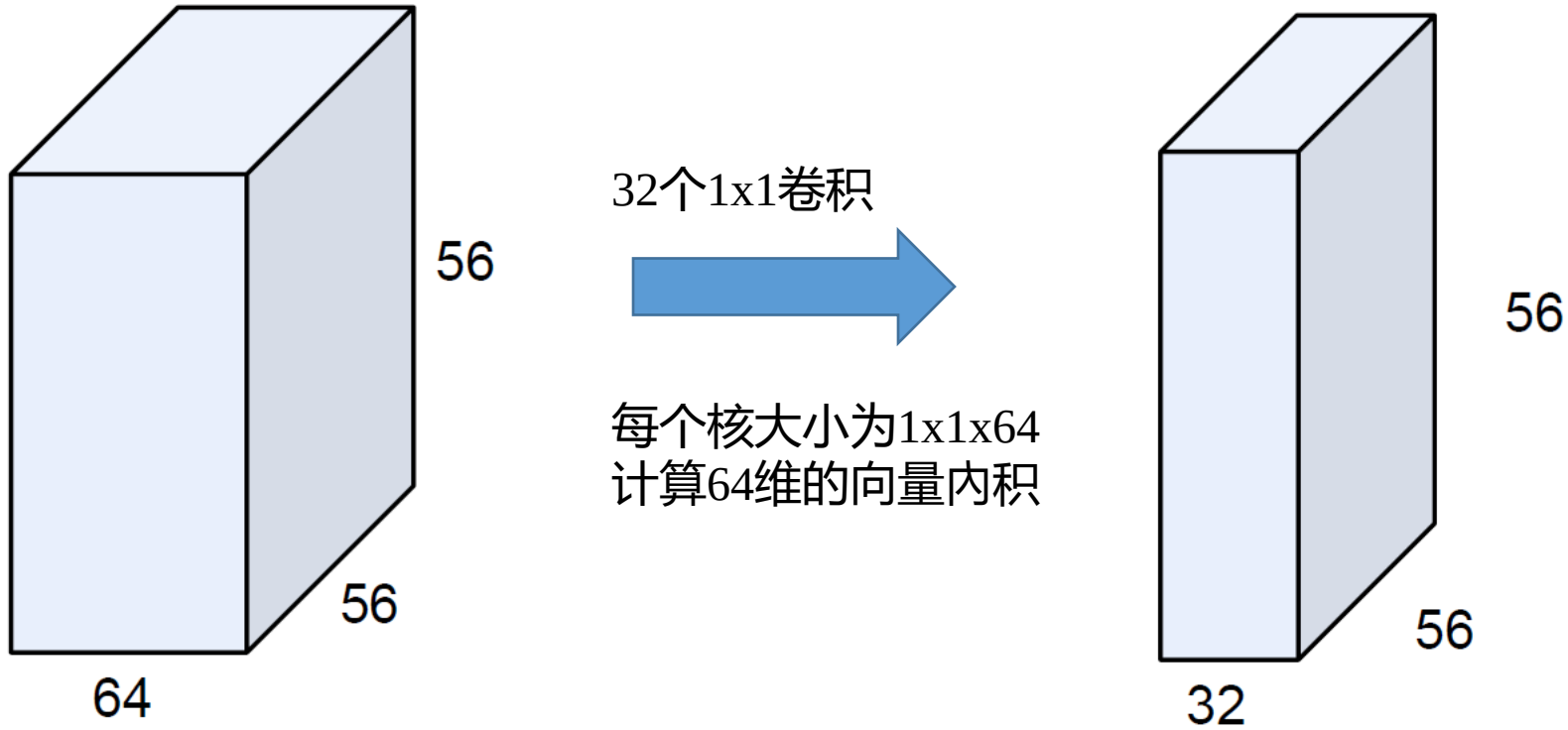
Parameters

- **in_channels** (*int*) – Number of channels in the input image
- **out_channels** (*int*) – Number of channels produced by the convolution
- **kernel_size** (*int or tuple*) – Size of the convolving kernel
- **stride** (*int or tuple, optional*) – Stride of the convolution. Default: 1
- **padding** (*int or tuple, optional*) – Zero-padding added to both sides of the input. Default: 0
- **padding_mode** (*string, optional*) – `'zeros'`, `'reflect'`, `'replicate'` or `'circular'`. Default: `'zeros'`
- **dilation** (*int or tuple, optional*) – Spacing between kernel elements. Default: 1
- **groups** (*int, optional*) – Number of blocked connections from input channels to output channels. Default: 1
- **bias** (*bool, optional*) – If `True`, adds a learnable bias to the output. Default: `True`

```
>>> # With square kernels and equal stride
>>> m = nn.Conv2d(16, 33, 3, stride=2)
>>> # non-square kernels and unequal stride and with padding
>>> m = nn.Conv2d(16, 33, (3, 5), stride=(2, 1), padding=(4, 2))
>>> # non-square kernels and unequal stride and with padding and dilation
>>> m = nn.Conv2d(16, 33, (3, 5), stride=(2, 1), padding=(4, 2), dilation=(3, 1))
>>> input = torch.randn(20, 16, 50, 100)
>>> output = m(input)
```

卷积

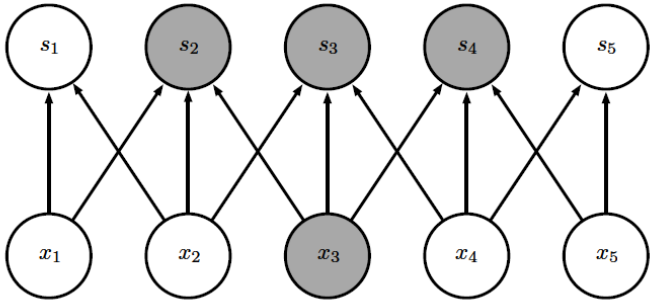
- 1x1的卷积也有意义，可以方便一些运算

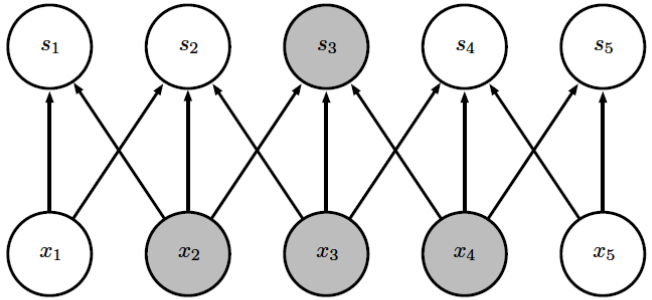


卷积的动机—稀疏链接

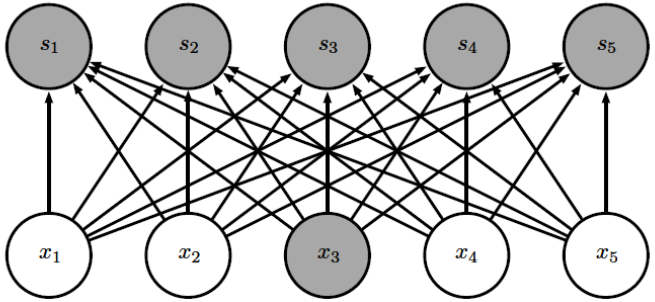
s_3的接受域（receptive field）

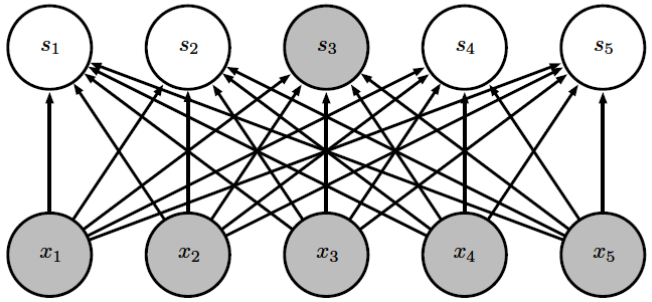
卷积
K=3



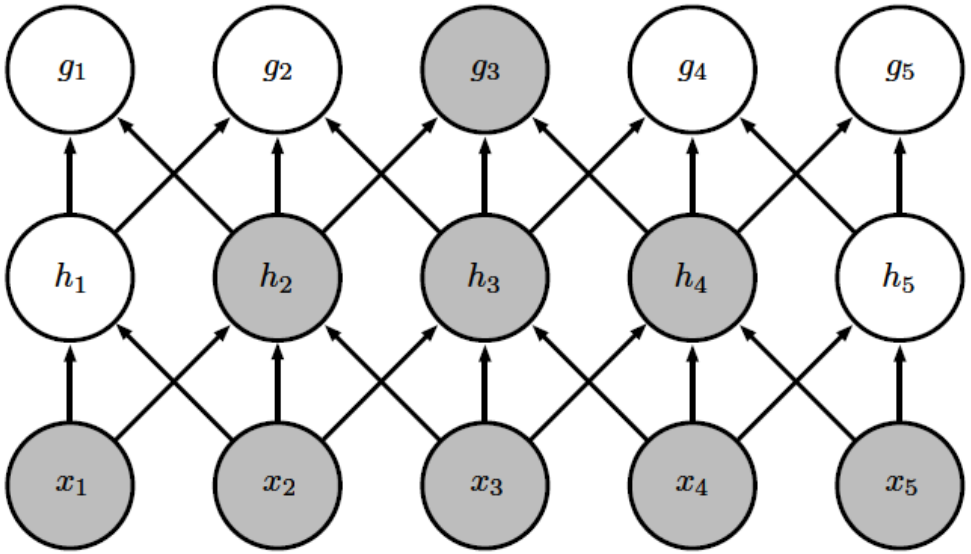


全连接





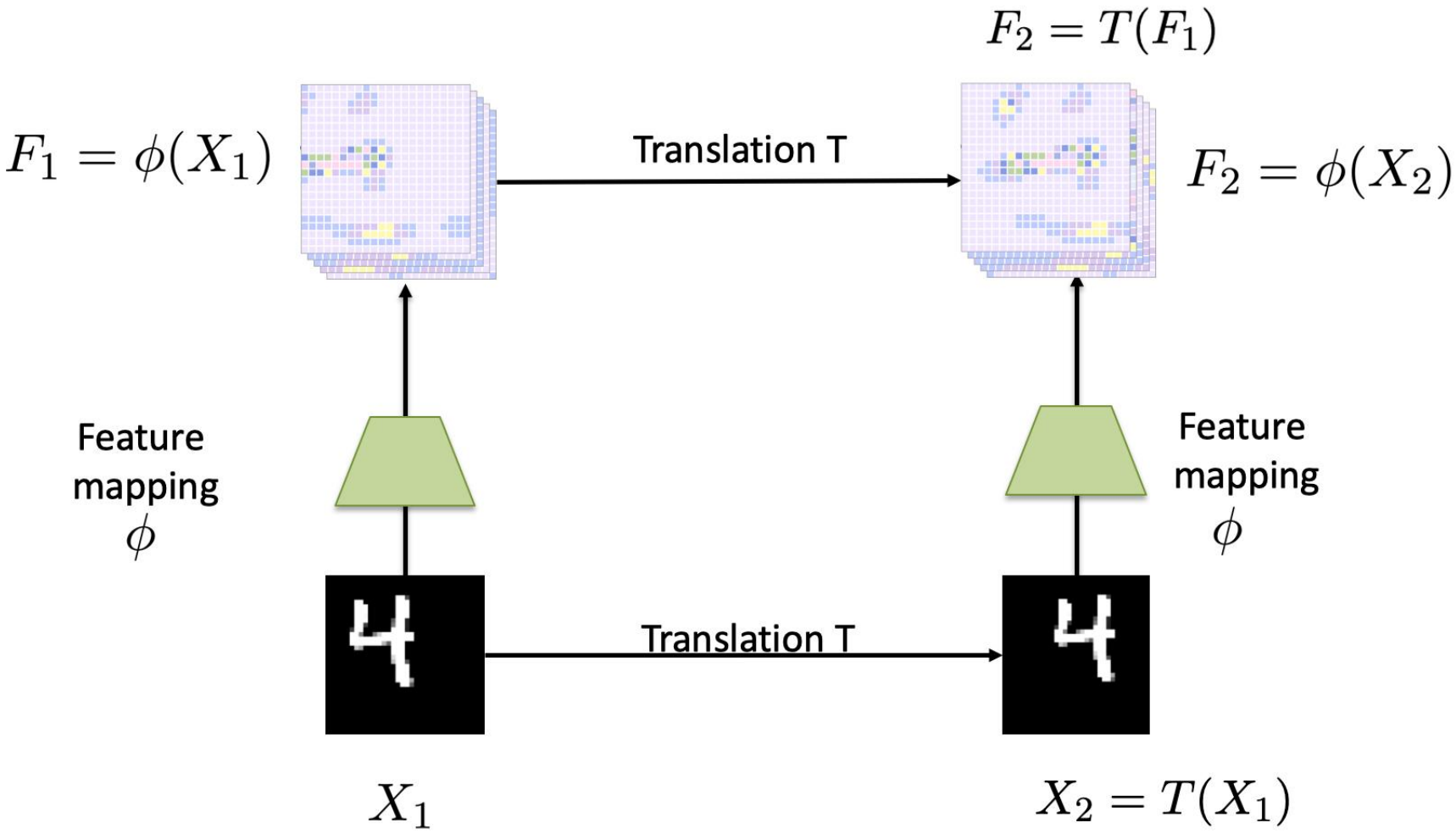
卷积的动机—稀疏链接



更深的层中的单元可以间接地连接到全部或者大部分输入图像

卷积的动机—平移不变性

口卷积具有平移同变性

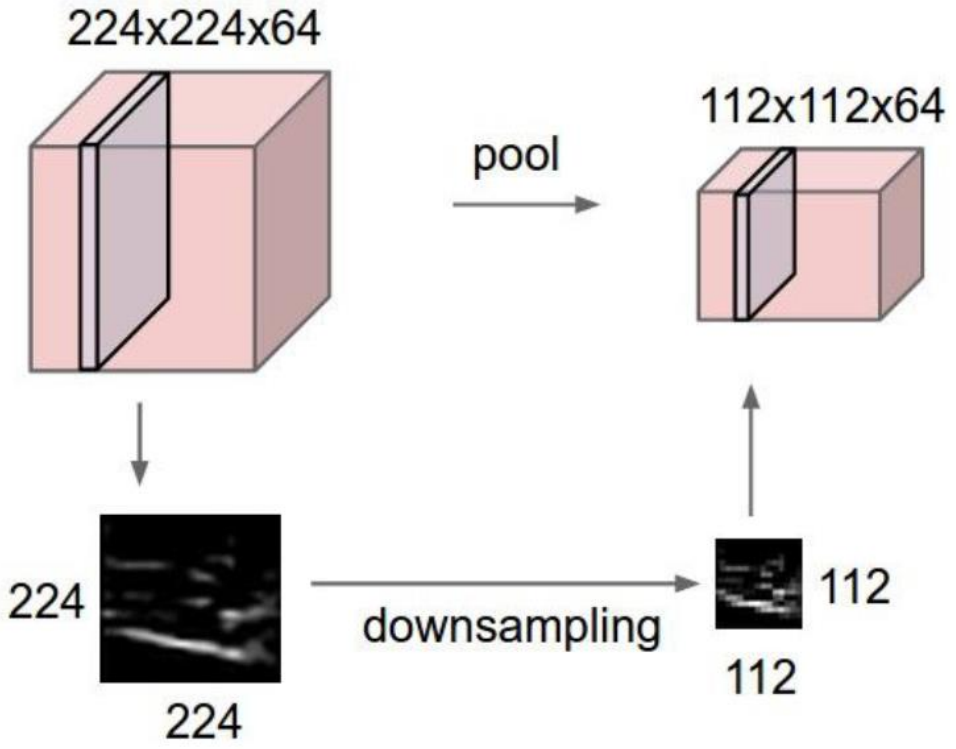


池化

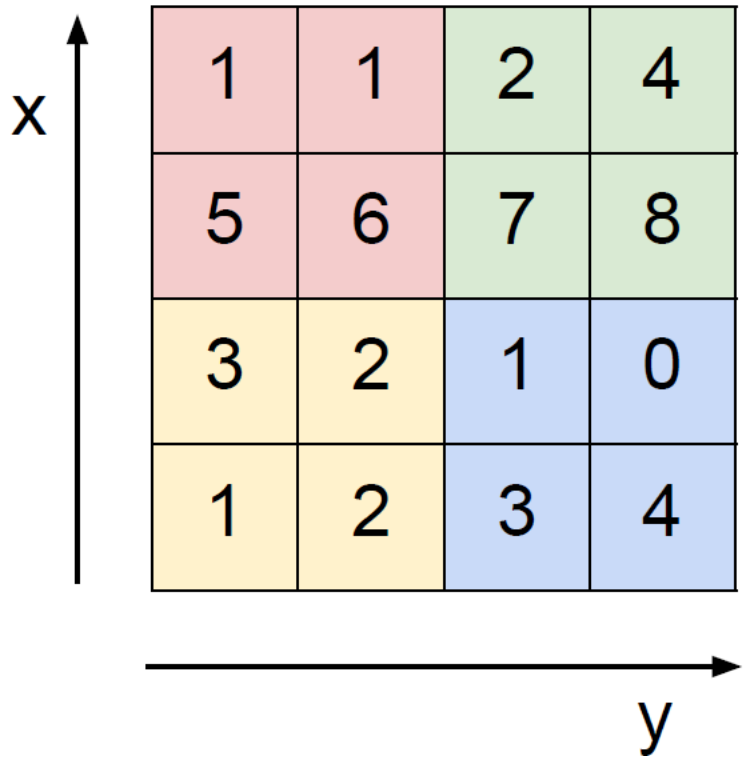
口动机：卷积神经网络中特征图尺寸大。池化操作能减小卷积神经网络中特征图的尺寸

口池化函数使用某一位置的相邻输出的总体统计特征来代替网络在该位置的输出，使得特征图尺寸更小且更易于管理

- 最大池化（max pooling）函数给出相邻矩形区域内的最大值
- 其他常用的池化函数包括相邻矩形区域内的平均值、L_2范数以及基于据中心像素距离的加权平均函数



最大池化



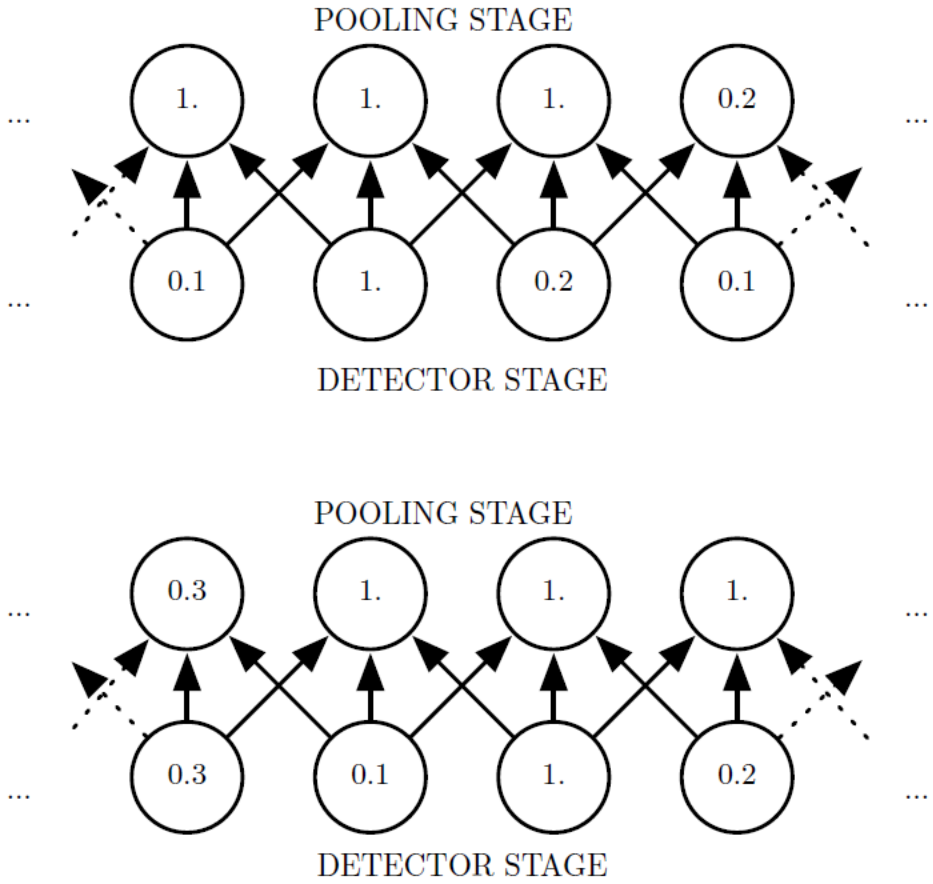
2x2的核，步幅为2

6	8
3	4

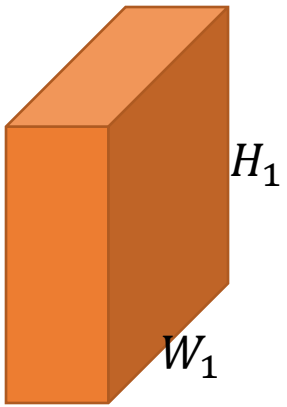
池化对局部平移的稳定性

口对局部平移更加稳定

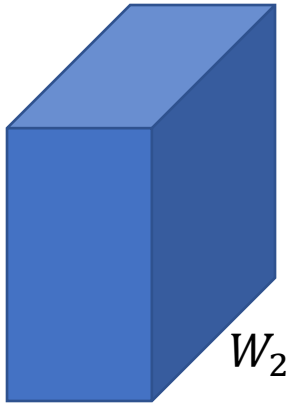
输入右移一位，输入全变，但是输出只有一半的值发生变化



池化层的信息概括



核大小F
步幅S



$H_2 = (H_1 - F)/S + 1$

$W_2 = (W_1 - F)/S + 1$

$D_2 = D_1$

常用设置：

- F=2, S=2
- F=3, S=2

Pytorch中的池化层

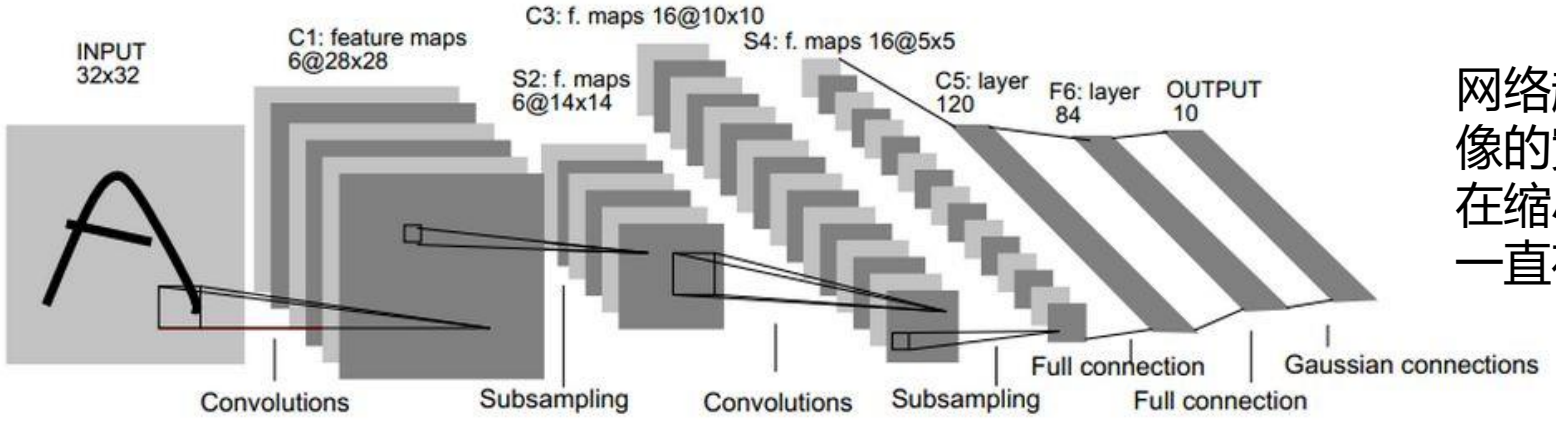
CLASS `torch.nn.MaxPool2d(kernel_size, stride=None, padding=0, dilation=1, return_indices=False, ceil_mode=False)` [SOURCE]

Parameters

- **kernel_size** – the size of the window to take a max over
- **stride** – the stride of the window. Default value is `kernel_size`
- **padding** – implicit zero padding to be added on both sides
- **dilation** – a parameter that controls the stride of elements in the window
- **return_indices** – if `True`, will return the max indices along with the outputs. Useful for `torch.nn.MaxUnpool2d` later
- **ceil_mode** – when True, will use *ceil* instead of *floor* to compute the output shape

```
>>> # pool of square window of size=3, stride=2
>>> m = nn.MaxPool2d(3, stride=2)
>>> # pool of non-square window
>>> m = nn.MaxPool2d((3, 2), stride=(2, 1))
>>> input = torch.randn(20, 16, 50, 32)
>>> output = m(input)
```

典型的卷积网络—LeNet-5



网络越来越深，图像的宽度和高度都在缩小，信道数量一直在增加

5x5 s=1 p=0	2x2 s=1 p=0	5x5 s=1 p=0	2x2 s=1 p=0

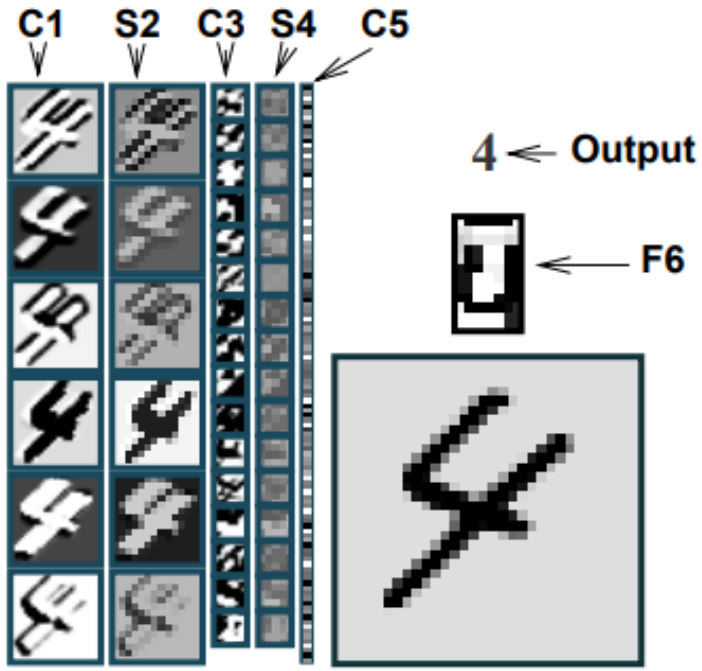
- 每个卷积层含：卷积、池化、非线性激活
- 使用卷积提取空间特征
- 降采样的平均池化层
- 双曲正切或S型的激活函数
- MLP作为最后的分类器

层（layer）	激活后维度	参数个数
Input	(32,32,1)	0
CONV1(f=5,s=1)	(28,28,6)	(5*5+1)*6=156
POOL1	(14,14,6)	0
CONV2(f=5,s=1)	(10,10,16)	(5*5*6+1)*16=2416
POOL2	(5,5,16)	0
FC3	(120,1)	120*(400+1)=48120
FC4	(84,1)	84*(120+1)=10164
Softmax	(10,1)	10*(84+1)=850

图像分类任务示例：手写数字识别

□MNIST (handwritten digits) 数据集

6万训练样本和1万测试样本

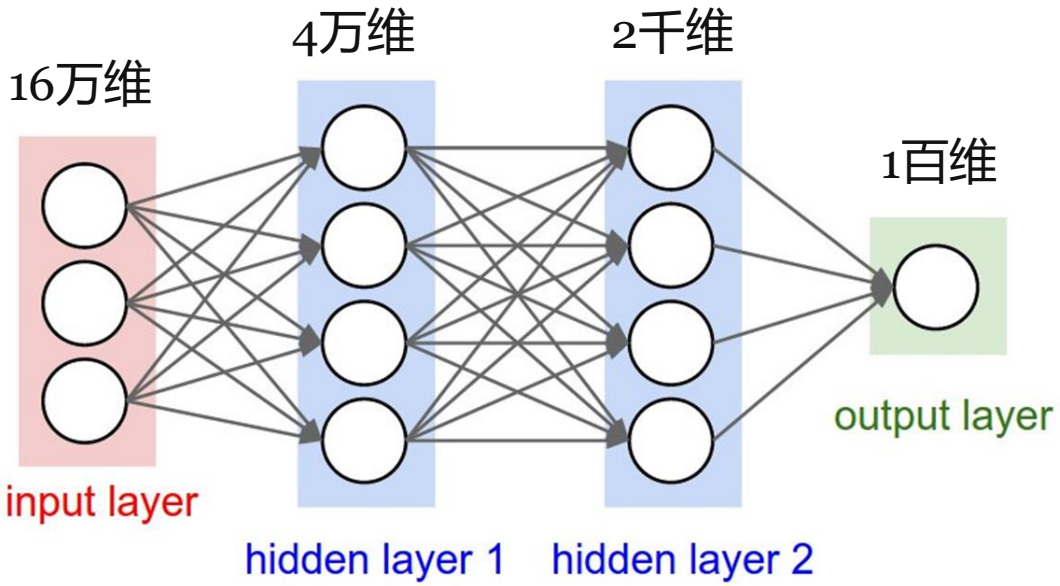


错误案例

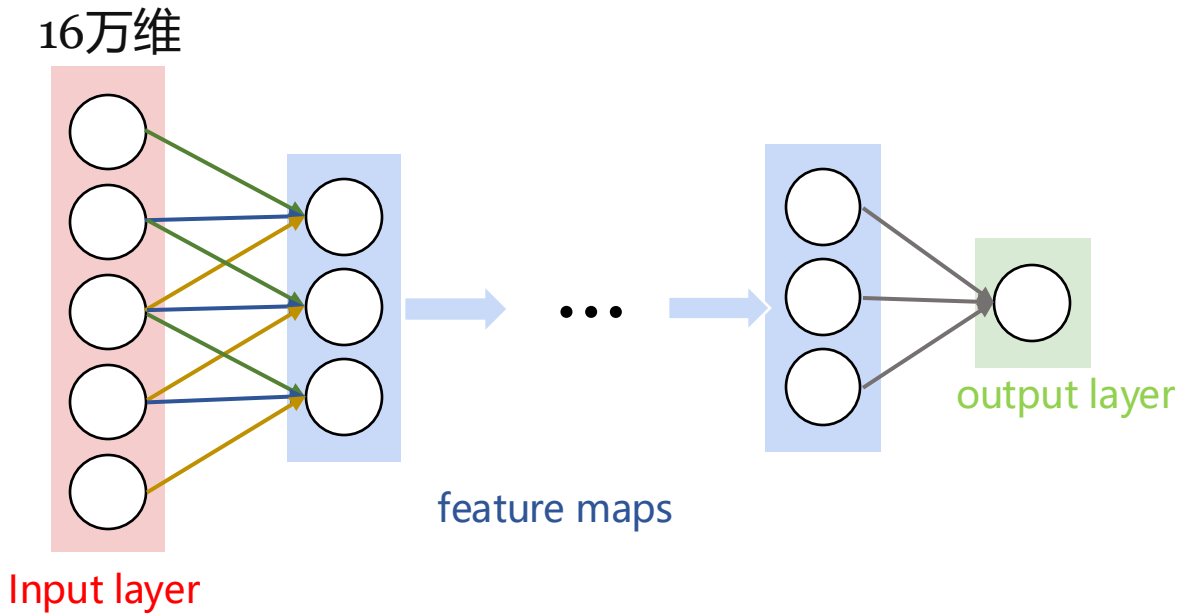
目录

作业习题

1. 在如下左图所示的四层全连接神经网络中，输入层维度为 160,000，第一个隐藏层维度为 40,000，第二个隐藏层维度为 2,000，输出层维度为 100。请计算该神经网络的参数总量。
2. 在如下右图所示的卷积神经网络中，输入层维度为 160,000，第一个卷积层使用 3 个 5*5 的卷积核，使用max pooling 将特征图尺寸降为 40,000, 第二个卷积层使用 5 个 4*4*3 的卷积核，紧接着使用 max pooling 将特征图尺寸降为 2,000，第三个卷积层使用 5 个 4*4*5 的卷积核，使用 max pooling 将特征图尺寸降为 400， 在卷积过程中始终使用padding保持输入和输出维度一致，最后一层使用全连接层将维度展平后降为 100。请计算该神经网络的参数总量。



全连接神经网络



卷积神经网络