



首都师范大学

尚学尚师 求实求新

深度学习应用与工程实践

4. 卷积神经网络

4. Convolutional Neural Networks

李冰

Bing Li

Tenure-track Associate Professor

Academy of Multidisciplinary Studies

Capital Normal University



回顾

- 从传统机器学习到深度学习
 - 传统机器学习方法
 - 为什么深度学习必要
 - 深度学习种类
 - 深度学习的应用
- Numpy和Pytorch
 - 安装使用—Anaconda
 - 入门---数据类型、基本计算



上节课内容

深度学习应用

工程实践

计算机视觉 (CV)

MLP

CNN

UNet

YoLo

MobileNet

模型设计

自然语言处理(NLP)

RNN

LSTM/GRU

- 介绍卷积神经网络 (Convolutional Neural Network, CNN)
- CNN架构的基本构成模块
- CNN设计原则

生成式模型

GAN

Diffusion

GPT

部署

新增内容

教材和软件框架

1. 动手学深度学习pytorch

作者: 阿斯顿·张 (Aston Zhang) / 李沐 (Mu Li) / [美] 扎卡里·C. 立顿 (Zachary C. Lipton) / [德] 亚历山大·J. 斯莫拉 (Alexander J. Smola)

2. 深度学习 (2016),

作者: [美] 伊恩·古德费洛 / [加] 约书亚·本吉奥 / [加] 亚伦·库维尔

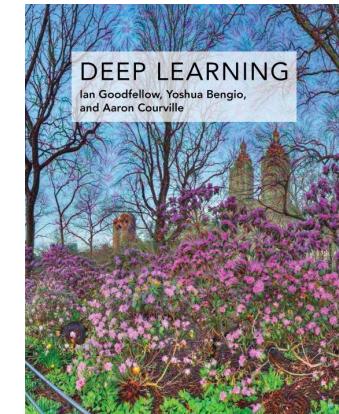
3. 深度学习入门-基于Python的理论与实现,

作者: [日] 斋藤康毅

4. 机器学习

作者: 周志华, 清华大学出版社, 2016.

- 推荐下载使用Pytorch (<https://pytorch.org/>)

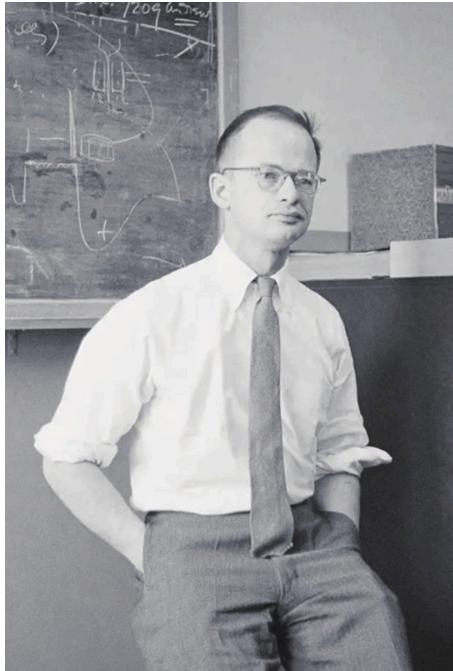


深度学习模型

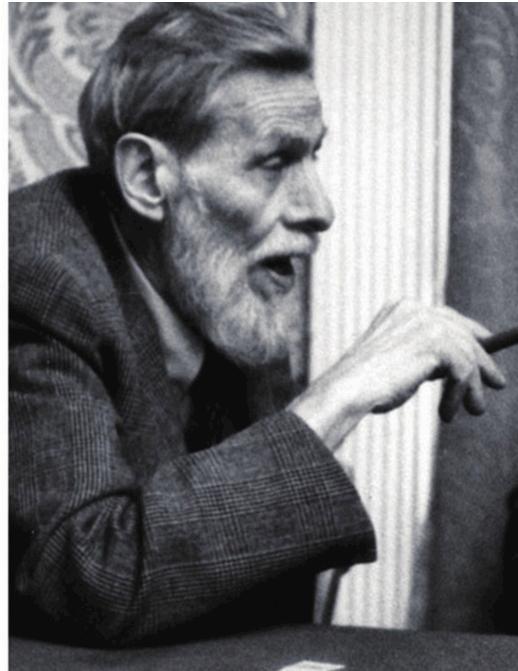
- 历史
- 定义/模型结构
- 学习算法

Perceptron 感知机

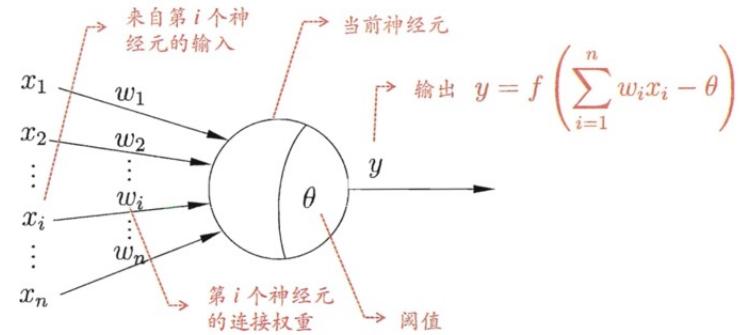
- 1943, 沃伦·麦卡洛克和沃尔特·皮茨创造了一种神经网络的计算模型。



沃尔特·皮茨
(1923-1969)

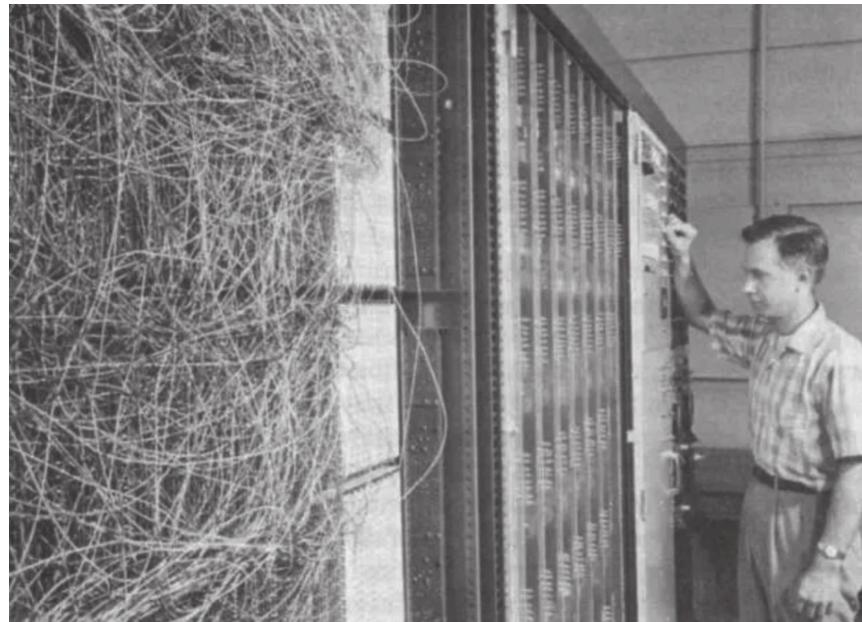


沃伦·麦卡洛克
(1923-1969)



Perceptron 感知机

- 1943, 沃伦·麦卡洛克和沃尔特·皮茨创造了一种神经网络的计算模型。
- 1957年, 弗兰克·罗森布拉特发明的二元线性分类器, 在IBM 704机上完成了感知机的仿真。
- 1959后, 弗兰克·罗森布拉特成功实现了能够识别一些英文字母、基于感知机的神经计算机——Mark1.



Perceptron 感知机

- 1943, 沃伦·麦卡洛克和沃尔特·皮茨创造了一种神经网络的计算模型。
- 1957年, 弗兰克·罗森布拉特发明的二元线性分类器, 在IBM 704机上完成了感知机的仿真。
- 1959后, 弗兰克·罗森布拉特成功实现了能够识别一些英文字母、基于感知机的神经计算机——Mark1。
- 1969年, 马文·明斯基和西摩尔·派普特指出了感知机不能解决XOR等线性不可分问题。
- 20世纪80年代, 人们认识到多层感知机及反向传播算法的能力, 人工神经网络领域发展才有所恢复。
- 1998之后的研究表明感知机除了二元分类, 也能应用在较复杂、被称为structured learning类型的任务上 (Collins, 2002), 和大规模机器学习问题上 (McDonald, Hall and Mann, 2011)。

感知器是最简单形式的前馈式人工神经网络。

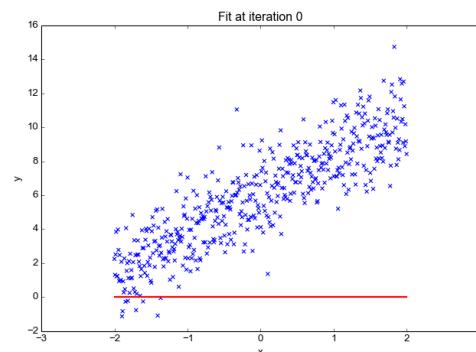
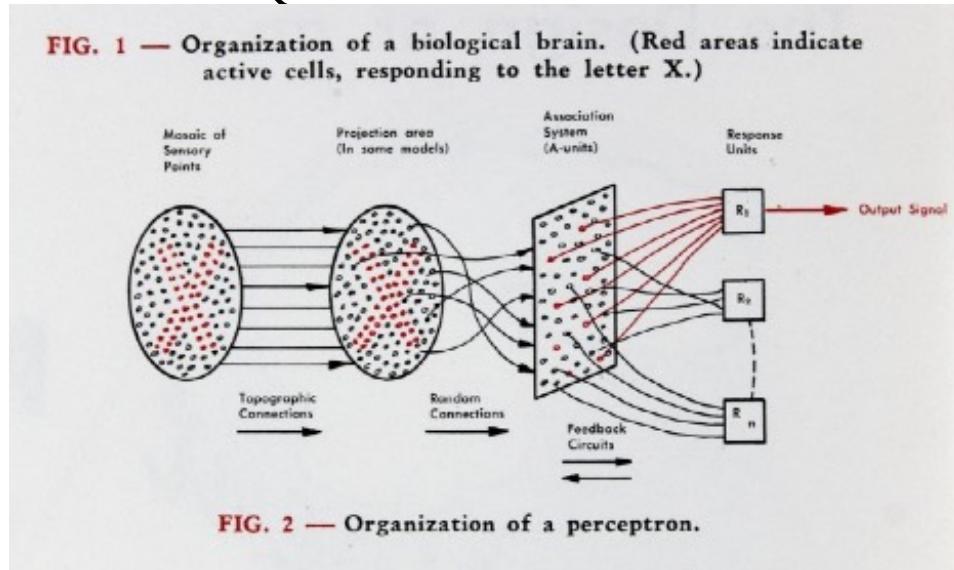


Perceptron

- 感知机的定义：

- $f(x) = \begin{cases} 1 & \text{if } w^T x + b > 0 \\ 0 & \text{else} \end{cases}$

w : 权重
 b : 偏置，一个常数
 $w^T x$: 点积

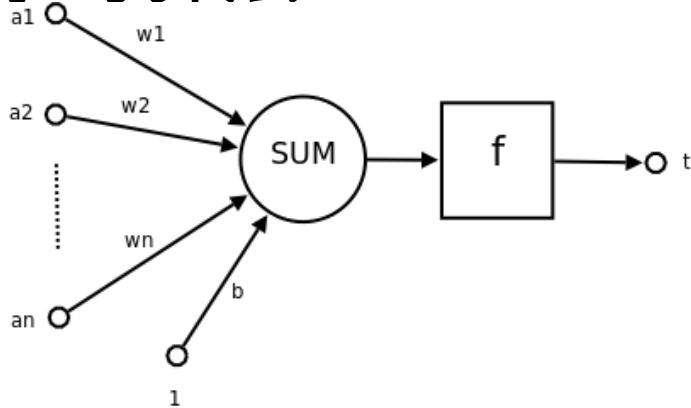


机

$w^T x + b = 0$ 确定一个超平面，在超平面一侧，输出值为1，在超平面另一侧，输出为0，将输入分为两类。

Perceptron

• 学习算法



x_j 表示n维输入向量中的第j项
 w_j 表示权重向量中的第j项
 $f(x)$ 表示神经元接受输入 x 产生的输出
 α 是个常数，符合 $0 < \alpha \leq 1$
假定 $b = 0$

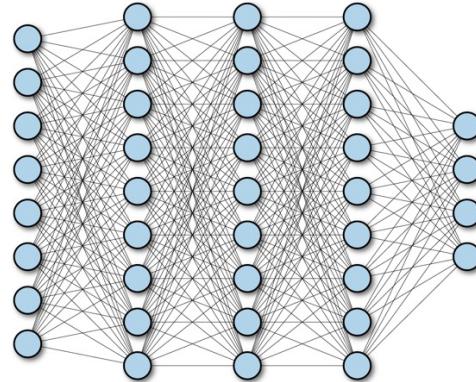
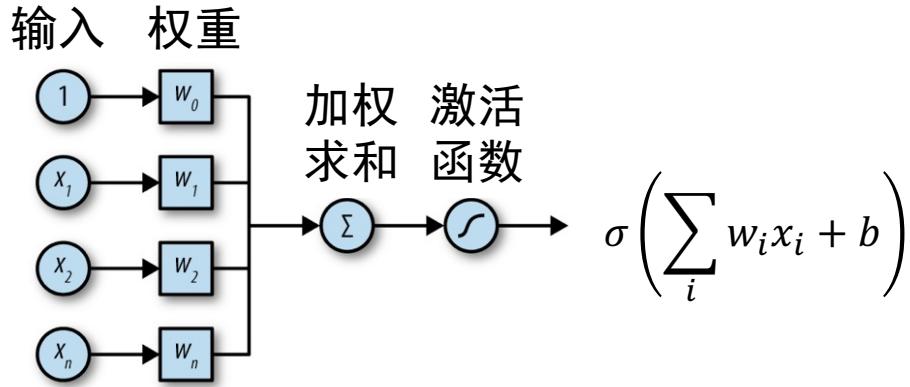
通过对所有训练实例进行多次的迭代进行更新完成学习

训练集 $D_m = \{(x(1), y(1)), (x(2), y(2)) \dots (x(m), y(m))\}$, 对其中的每个 (x, y) 对,

$$w_j := w_j + \alpha(y - f(x))x_j, \quad (j = 1, \dots, n)$$

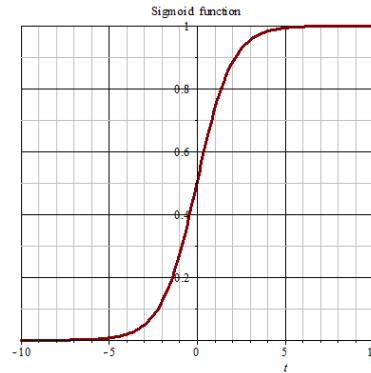
如果训练集是线性分隔的，那么感知器算法可以在有限次迭代后收敛，否则不保证收敛。

MLP(Multilayer Perceptron)



- 激活函数：
 - 可微，S函数，如逻辑函数

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

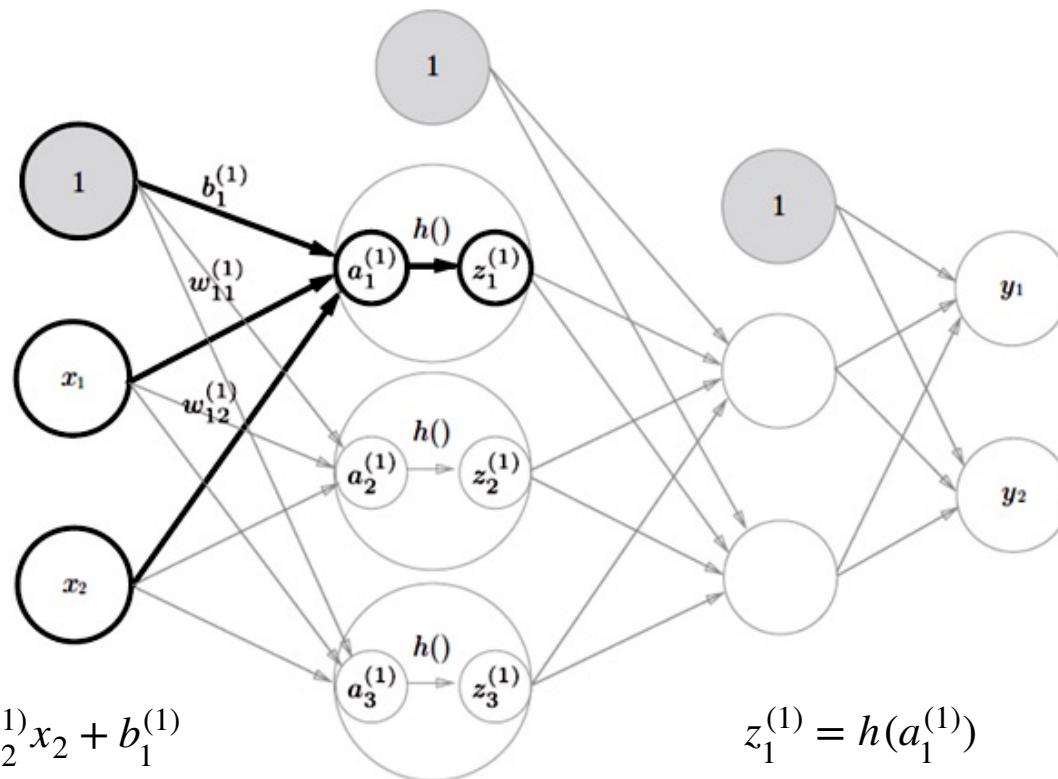


- 层数
 - 一个输入层和输出层，一个或者多个隐藏层
 - 多层之间完全互连
- 学习算法：反向传播学习算法
- 克服感知机线性不可分的问题。

前向传播

前向传播

输入层 中间层 输出层



$$a_1^{(1)} = w_{11}^{(1)}x_1 + w_{12}^{(1)}x_2 + b_1^{(1)}$$

$$a_2^{(1)} = w_{21}^{(1)}x_1 + w_{22}^{(1)}x_2 + b_1^{(1)}$$

$$a_3^{(1)} = w_{31}^{(1)}x_1 + w_{32}^{(1)}x_2 + b_3^{(1)}$$

$$z_1^{(1)} = h(a_1^{(1)})$$

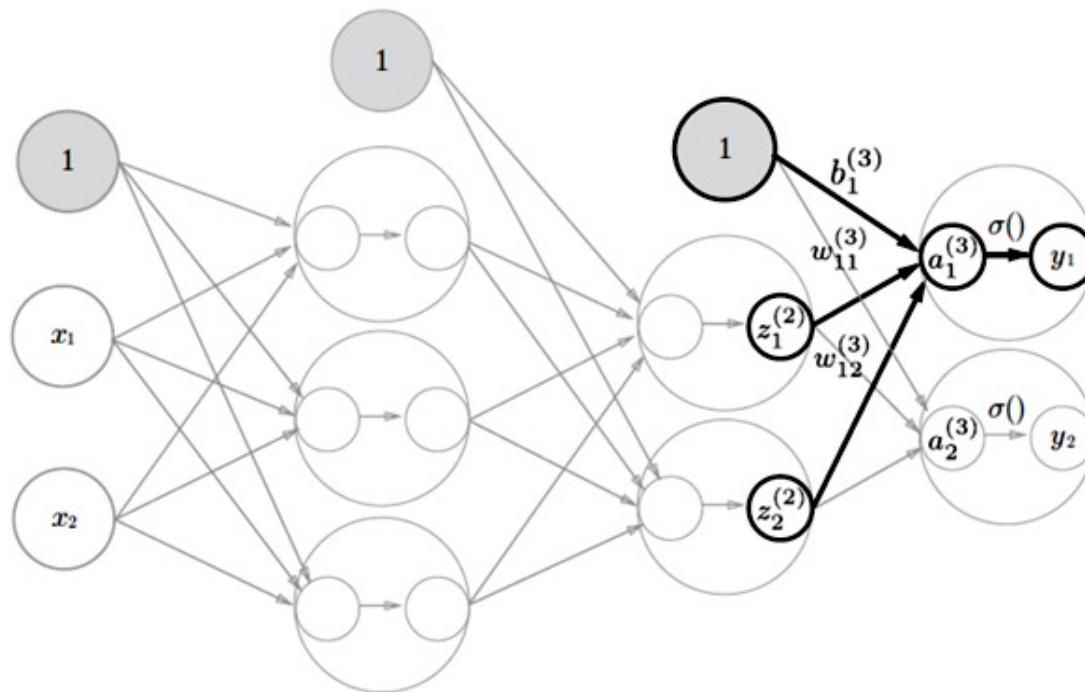
$$z_2^{(1)} = h(a_2^{(1)})$$

$$z_3^{(1)} = h(a_3^{(1)})$$

前向传播

前向传播

输入层 中间层 输出层

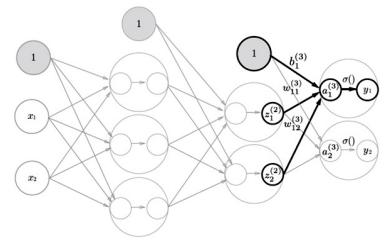


$$a_1^{(3)} = w_{11}^{(3)} z_1^{(2)} + w_{12}^{(3)} z_2^{(2)} + b_1^{(3)}$$

$$a_2^{(3)} = w_{21}^{(3)} z_1^{(2)} + w_{22}^{(3)} z_2^{(2)} + b_2^{(3)}$$

$$y_1 = \sigma(z_1^{(3)})$$

$$y_2 = \sigma(z_2^{(3)})$$



反向传播

第 j 个输出节点的误差 $d_j - y_j$, 其中 d 是目标值

反向传播的目标是最小均方差

$$E = \frac{\sum_j (d_j - y_j)^2}{n}$$

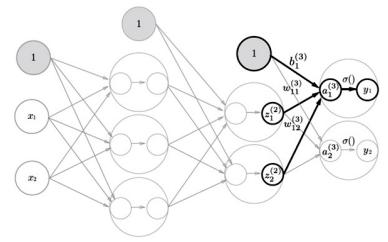
$$\partial w_{jk} = \frac{\partial E}{\partial w_{jk}} = \frac{\partial E}{\partial a_k} \frac{\partial a_k}{\partial w_{jk}}$$

$$\frac{\partial E}{\partial a_k} = \frac{\partial E}{\partial y_k} \frac{\partial y_k}{\partial a_k}$$

$$\begin{aligned} \frac{\partial a_k}{\partial w_{jk}} &= \frac{\partial \sum_q^m (z_q w_{qk} + b_k)}{\partial w_{jk}} \\ &= z_j \end{aligned}$$

$$\begin{aligned} a_1^{(3)} &= w_{11}^{(3)} z_1^{(2)} + w_{12}^{(3)} z_2^{(2)} + b_1^{(3)} \\ a_2^{(3)} &= w_{21}^{(3)} z_1^{(2)} + w_{22}^{(3)} z_2^{(2)} + b_2^{(3)} \end{aligned}$$

$$\begin{aligned} y_1 &= \sigma(z_1^{(3)}) \\ y_2 &= \sigma(z_2^{(3)}) \end{aligned}$$



反向传播

第 j 个输出节点的误差 $d_j - y_j$, 其中 d 是目标值

反向传播的目标是最小均方差

$$E = \frac{\sum_j (d_j - y_j)^2}{n}$$

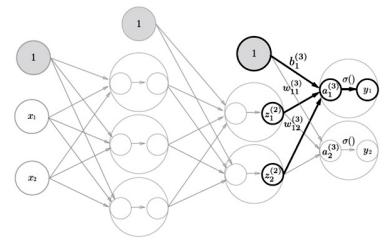
$$\partial w_{jk} = \frac{\partial E}{\partial w_{jk}} = \frac{\partial E}{\partial a_k} \frac{\partial a_k}{\partial w_{jk}}$$

$$\frac{\partial E}{\partial a_k} = \frac{\partial E}{\partial y_k} \frac{\partial y_k}{\partial a_k}$$

$$\begin{aligned} \frac{\partial a_k}{\partial w_{jk}} &= \frac{\partial \sum_q^m (z_q w_{qk} + b_k)}{\partial w_{jk}} \\ &= z_j \end{aligned}$$

比如该网络中，输出层 a_1 的偏导

$$\begin{aligned} \frac{\partial a_1}{\partial w_{11}} &= \frac{\partial (z_1 w_{11} + z_2 w_{12} + b_1)}{\partial w_{11}} \\ &= z_1 \end{aligned}$$



反向传播

第 j 个输出节点的误差 $d_j - y_j$, 其中 d 是目标值

反向传播的目标是最小均方差 $E = \frac{\sum_j (d_j - y_j)^2}{n}$

$$\partial w_{jk} = \frac{\partial E}{\partial w_{jk}} = \frac{\partial E}{\partial a_k} \frac{\partial a_k}{\partial w_{jk}}$$

$$\frac{\partial E}{\partial a_k} = \frac{\partial E}{\partial y_k} \frac{\partial y_k}{\partial a_k}$$

$$\frac{\partial a_k}{\partial w_{jk}} = \frac{\partial \sum_q^m (z_q w_{qk} + b_k)}{\partial w_{jk}} = z_j$$

写作下面的形式

$$\delta_k = \frac{\partial E}{\partial a_k} = \frac{\partial E}{\partial y_k} \frac{\partial y_k}{\partial a_k}$$

$$\partial w_{jk} = \delta_k z_j$$

权重梯度可以用 z 和 δ 的乘积表示

$$a_1^{(3)} = w_{11}^{(3)} z_1^{(2)} + w_{12}^{(3)} z_2^{(2)} + b_1^{(3)}$$

$$a_2^{(3)} = w_{21}^{(3)} z_1^{(2)} + w_{22}^{(3)} z_2^{(2)} + b_2^{(3)}$$

$$y_1 = \sigma(z_1^{(3)})$$

$$y_2 = \sigma(z_2^{(3)})$$

反向传播

输出层偏置梯度

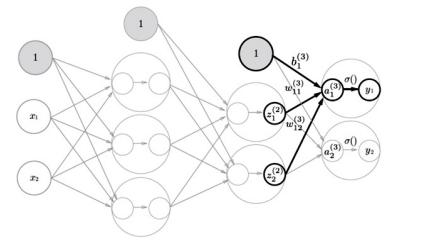
$$\frac{\partial b_k}{\partial b_k} = \frac{\partial E}{\partial b_k} = \frac{\partial E}{\partial a_k} \frac{\partial a_k}{\partial b_k}$$

$$\frac{\partial E}{\partial a_k} = \frac{\partial E}{\partial y_k} \frac{\partial y_k}{\partial a_k}$$

$$\delta_k = \frac{\partial E}{\partial a_k} = \frac{\partial E}{\partial y_k} \frac{\partial y_k}{\partial a_k}$$

$$\frac{\partial b_k}{\partial b_k} = \delta_k$$

偏置梯度与 δ 相同



$$a_1^{(3)} = w_{11}^{(3)}z_1^{(2)} + w_{12}^{(3)}z_2^{(2)} + b_1^{(3)}$$

$$a_2^{(3)} = w_{21}^{(3)}z_1^{(2)} + w_{22}^{(3)}z_2^{(2)} + b_2^{(3)}$$

$$y_1 = \sigma(z_1^{(3)})$$

$$y_2 = \sigma(z_2^{(3)})$$

$$\begin{aligned}\frac{\partial a_k}{\partial b_k} &= \frac{\partial \sum_q^m (z_q w_{qk} + b_k)}{\partial b_k} \\ &= 1\end{aligned}$$

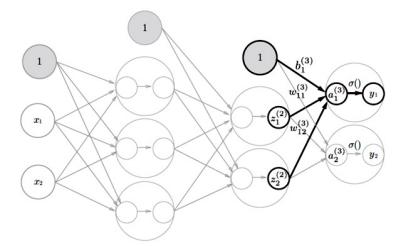
反向传播

输出层输入梯度

$$\partial z_j = \frac{\partial E}{\partial z_j} = \sum_{r=1}^n \frac{\partial E}{\partial a_r} \frac{\partial a_r}{\partial z_j}$$

$$\delta_r = \frac{\partial E}{\partial a_r} = \frac{\partial E}{\partial y_r} \frac{\partial y_r}{\partial a_r}$$

$$\partial z_j = \frac{\partial E}{\partial z_j} = \sum_{r=1}^n \delta_r w_{jr}$$



$$a_1^{(3)} = w_{11}^{(3)}z_1^{(2)} + w_{12}^{(3)}z_2^{(2)} + b_1^{(3)}$$

$$a_2^{(3)} = w_{21}^{(3)}z_1^{(2)} + w_{22}^{(3)}z_2^{(2)} + b_2^{(3)}$$

$$y_1 = \sigma(z_1^{(3)})$$

$$y_2 = \sigma(z_2^{(3)})$$

$$\begin{aligned} \frac{\partial a_r}{\partial z_j} &= \frac{\partial \sum_q^m (z_q w_{qr} + b_r)}{\partial z_j} \\ &= w_{jr} \end{aligned}$$

反向传播

• 输出层的梯度总结

$$\delta_k = \frac{\partial E}{\partial a_k} = \frac{\partial E}{\partial y_k} \frac{\partial y_k}{\partial a_k}$$

激活误差

$$\partial w_{jk} = \delta_k z_j$$

权重梯度

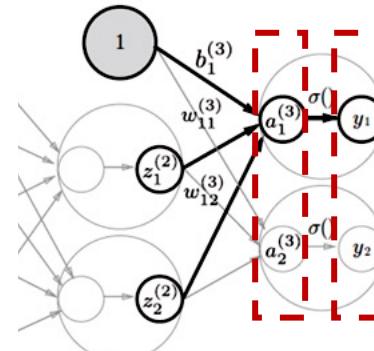
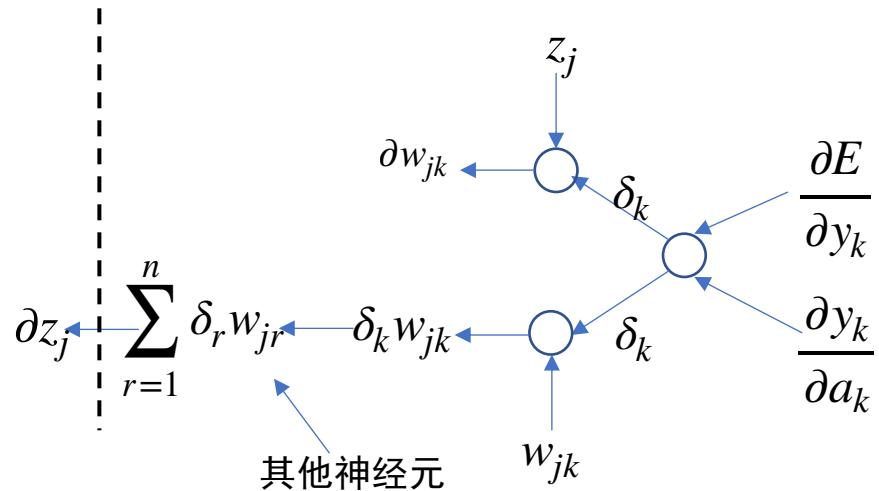
$$\partial b_k = \delta_k$$

偏置梯度

$$\partial z_j = \frac{\partial E}{\partial z_j} = \sum_{r=1}^n \delta_r w_{jr}$$

输入梯度

激活误差与对应权重的乘累加



反向传播

- 中间层输入梯度

$$\partial z_i = \frac{\partial E}{\partial z_i} = \sum_{r=1}^m \frac{\partial E}{\partial a_r} \frac{\partial a_r}{\partial z_i}$$
$$\frac{\partial E}{\partial a_j} = \frac{\partial E}{\partial z_j} \frac{\partial z_j}{\partial a_j}$$
$$\delta_j = \frac{\partial E}{\partial a_j} = \frac{\partial E}{\partial z_j} \frac{\partial z_j}{\partial a_j}$$
$$\partial z_i = \sum_{r=1}^n \delta_r w_{ir}$$
$$\frac{\partial a_r}{\partial z_i} = \frac{\partial \sum_q^m (z_q w_{qr} + b_r)}{\partial z_i} = w_{ir}$$

反向传播

• 中间层梯度总结

$$\delta_j = \frac{\partial E}{\partial a_j} = \partial z_j \frac{\partial z_j}{\partial a_j} \quad \text{激活误差}$$

$$\partial w_{ij} = \delta_j z_i \quad \text{权重梯度}$$

$$\partial b_j = \delta_j \quad \text{偏置梯度}$$

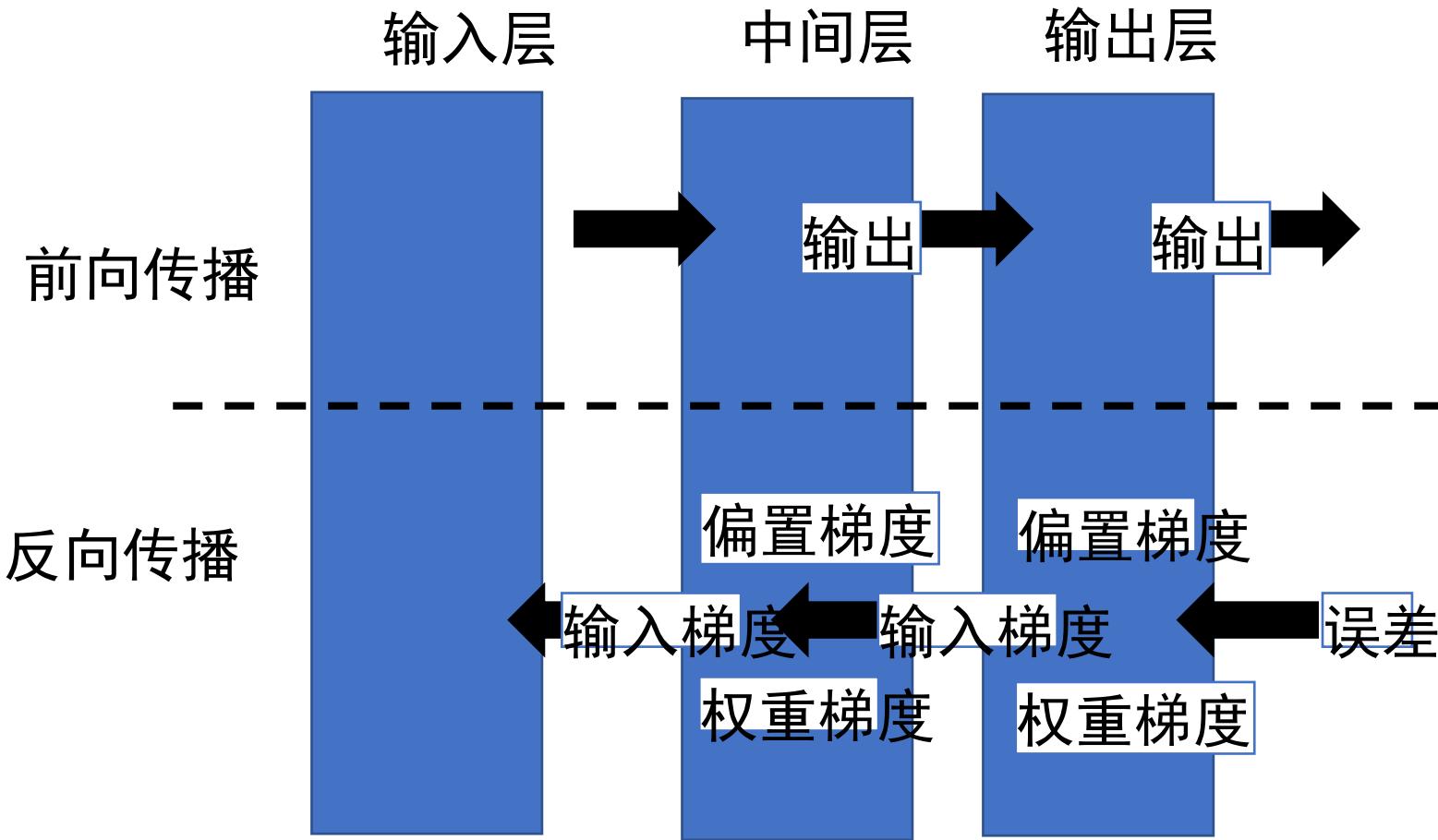
$$\partial z_i = \sum_{r=1}^n \delta_r w_{ir} \quad \text{输入梯度}$$

激活误差与对应权重的乘累加

无论输入层还是中间层，梯度计算都是相同的公式。
无论是增加多少层，梯度都可以这样计算。



反向传播



CNN 开始之前

传统的图像识别模型应用特征提取的方法
用来压缩特征维度

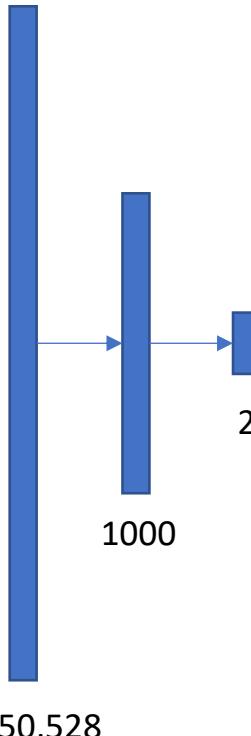
- 光学字符识别OCR (Optical character recognition)
- 主成分分析PCA (Principle component analysis)
- 径向基函数RBF (Radial basis function)
- 启发式方法HOS (Heuristic over segmentation)
- ...

但这些方法忽视输入特征中的拓扑
和区域相关性信息



CNN 开始之前

- 一个 $224 \times 224 \times 3$ 图像变为 1D 数组 (1×150528).
- 对于一个全连接网络，隐藏层有 1000 个神经元：



全部神经元数目是=

$$(150528+1) * 1000 + (1000+1) * 2 = 150M$$

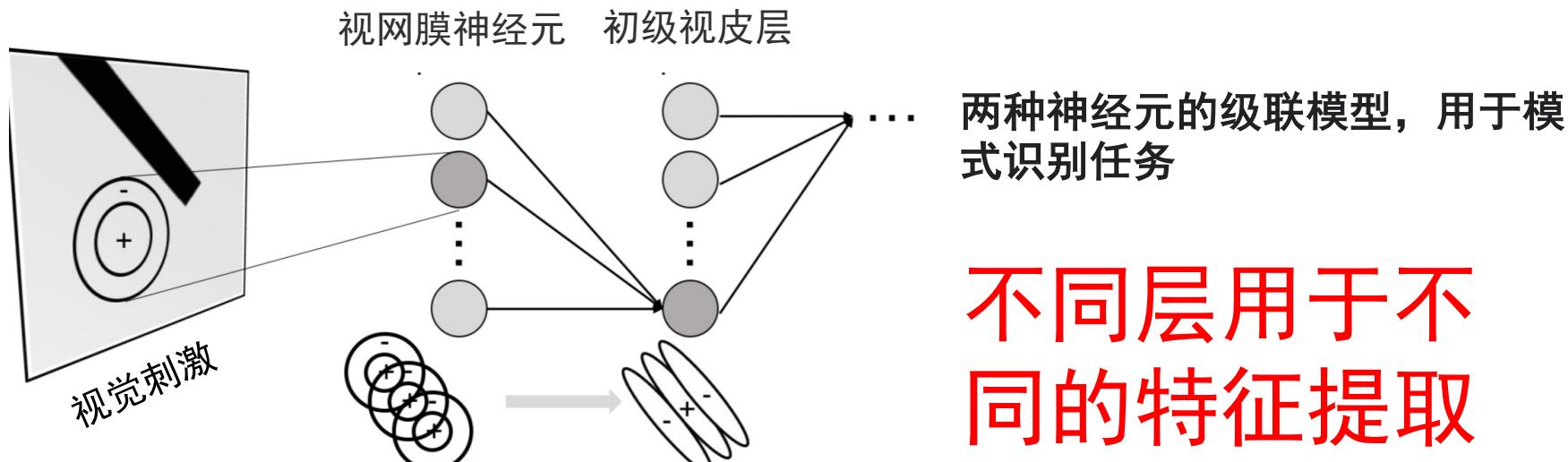
把图像展开成 1D 向量也会丢掉图片中像素点的空间信息.

CNN 开始之前

层次化结构

1968年，D.H. Hubel et al. 确定了大脑中两种基本神经元：简单神经元和复杂神经元；后来又找到了超复杂神经元。

- 简单神经元：对特定位置移动的定向边缘做出响应，光线朝向。
- 复杂神经元：对光线朝向和移动做出响应。
- 超复杂神经元：对移动、方向和长度、有端点的移动做出响应。



CNN 开始之前

感受野（Receptive Field）：搜集一个区域的信息

- 休波尔和威塞尔发现猫和猴的视觉皮层包含不同神经元，它们对视野的小部分区域做出响应。
- 单个神经元仅在视野限定的区域（称为感受野）中对刺激做出反应。

视觉皮层提取并收集区域信息。

- 不同神经元的感受野部分重叠，这样覆盖了整个视野。

神经元检索到的信息可能重叠。

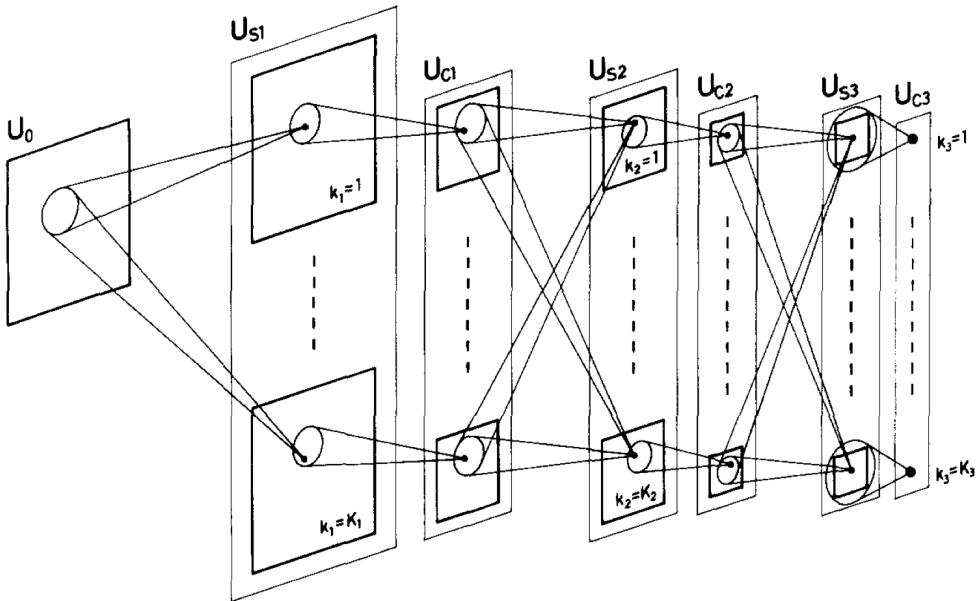
局部连接

不同神经元

D.H. Hubel等博士1962年发表的一篇文献：“Receptive fields, binocular interaction and functional architecture in the cat's visual cortex ”

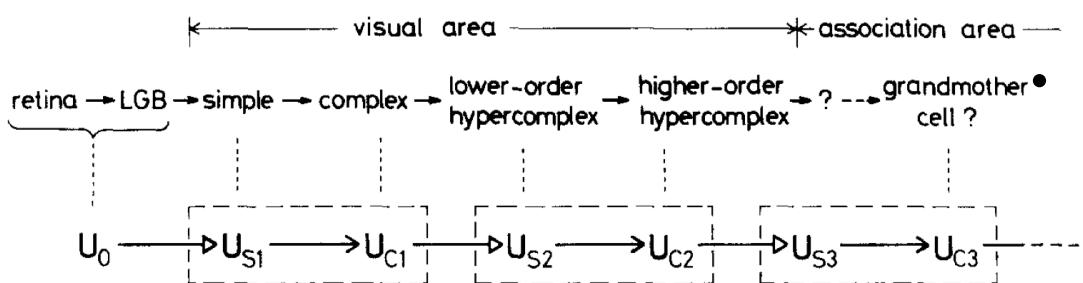


CNN 开始之前



S->C: 转换不变层;

C->S: 可修改的可训练层以提取特征



Neocognitron新认知机

- 福岛邦彦
- "Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position." (1980)
- S-C-S-C-S-C 结构
 - 简单单元(S): 有可训练的参数
 - 复杂单元(C): 执行下采样 (down-sampling) 来提取信息并不被更新

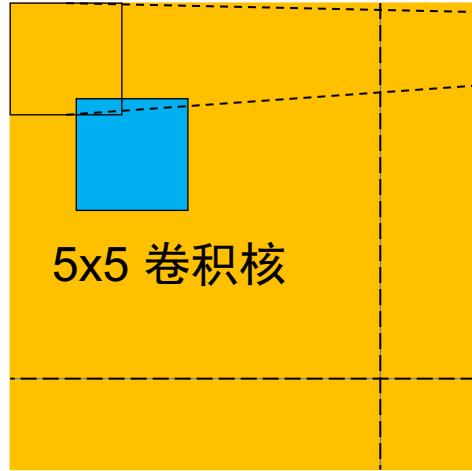
早期的卷积神经网络结构

→ modifiable synapses
→ unmodifiable synapses

卷积

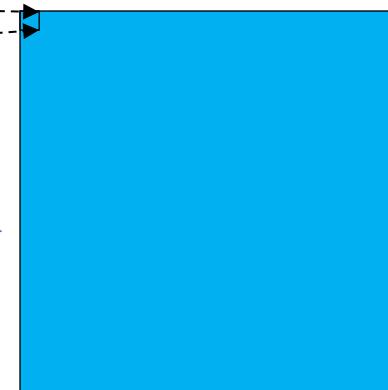
- 假设输入图通道数是1，大小是 32×32 ，在上面施加一个蓝色卷积核。
- 卷积在输入特征图上滑动计算。
- 每一步，卷积核和它连接的那一小块输入区域做点积计算。

局部连接



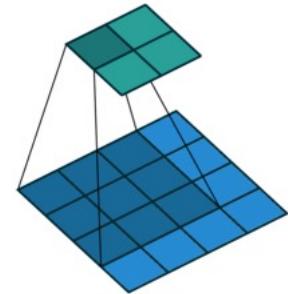
32x32 输入特征图

在2D 平面滑动
卷积窗口



28x28 输入特征图

卷积核大小 $K=5$



卷积

- 图像有三个通道 (对应R,G,B)
- 把蓝色 卷积核拓展为3D卷积核 延伸到输入特征图的深度 (Depth=3).
- 卷积核在图像上空间移动，并计算卷积核与对应输入特征图的点积，得到输出特征图.

不同神经元



Original image (RGB)



R channel



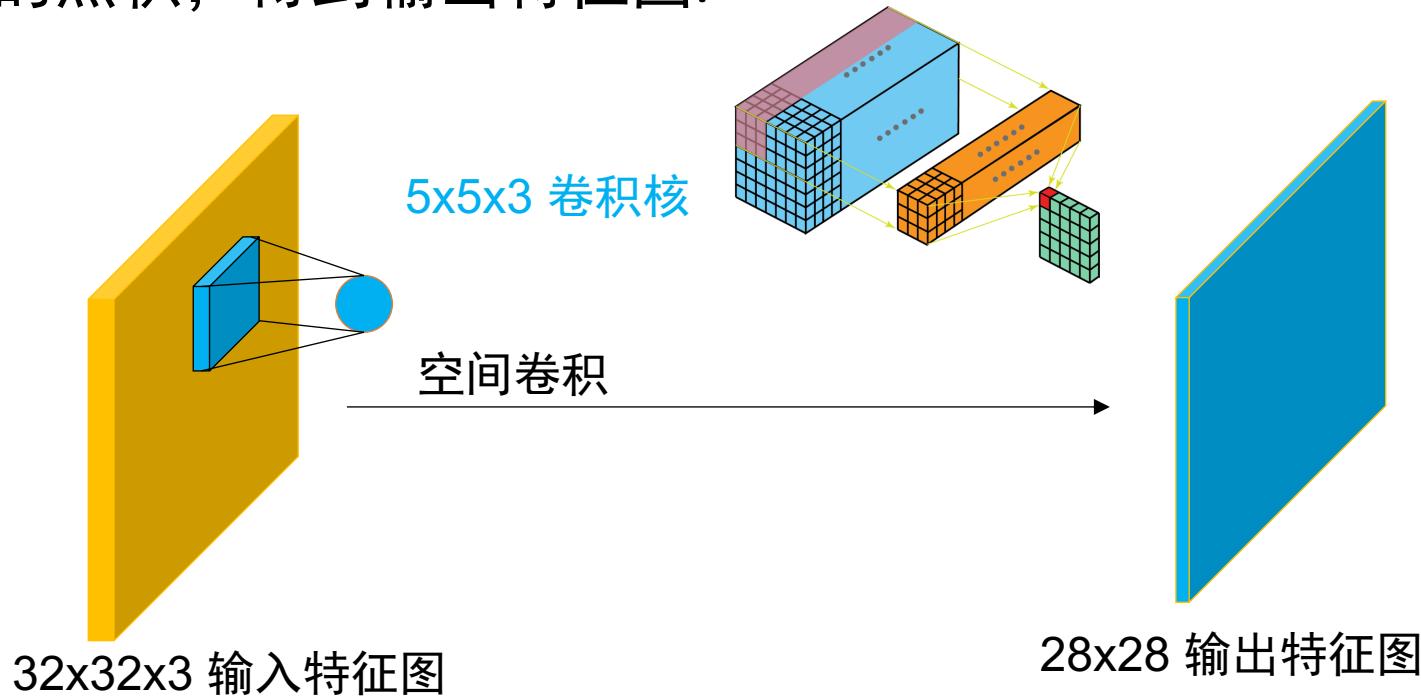
G channel



B channel

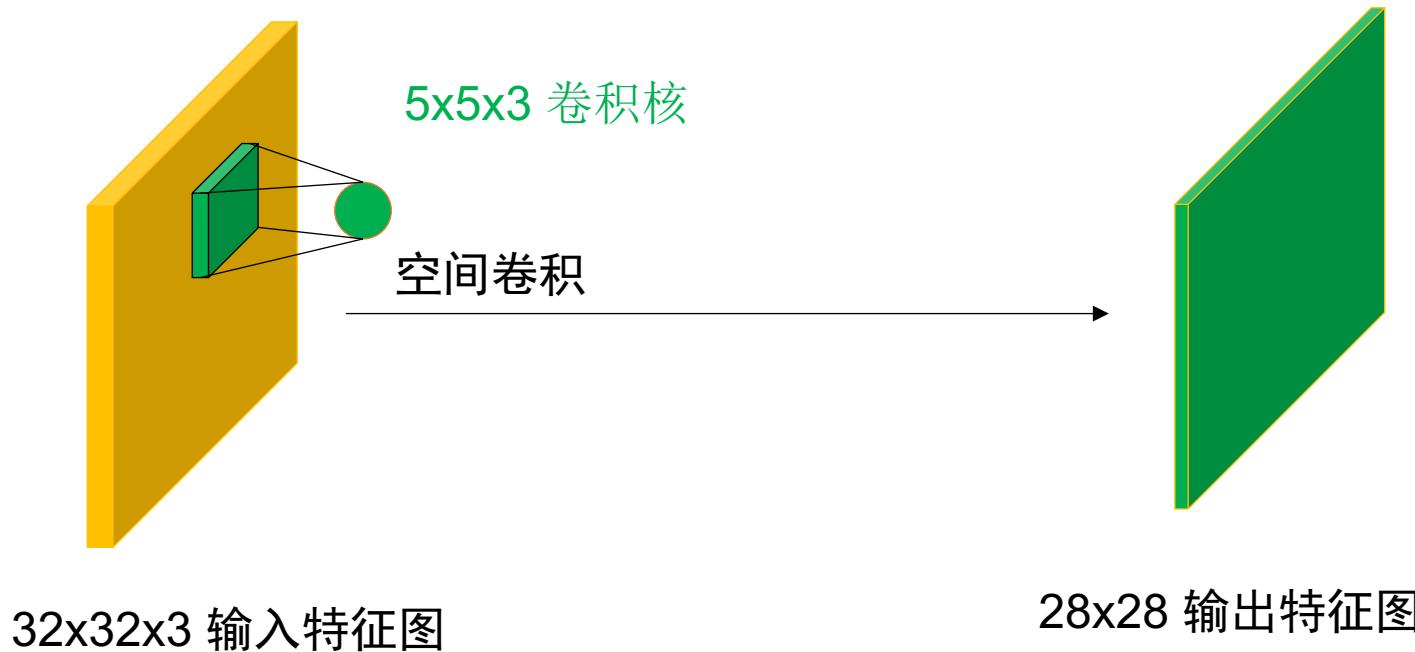
卷积

- 图像有三个通道 (对应R,G,B)
- 把蓝色 卷积核拓展为3D卷积核 延伸到输入特征图的深度 (Depth=3).
- 卷积核在图像上空间移动，并计算卷积核与对应输入特征图的点积，得到输出特征图.



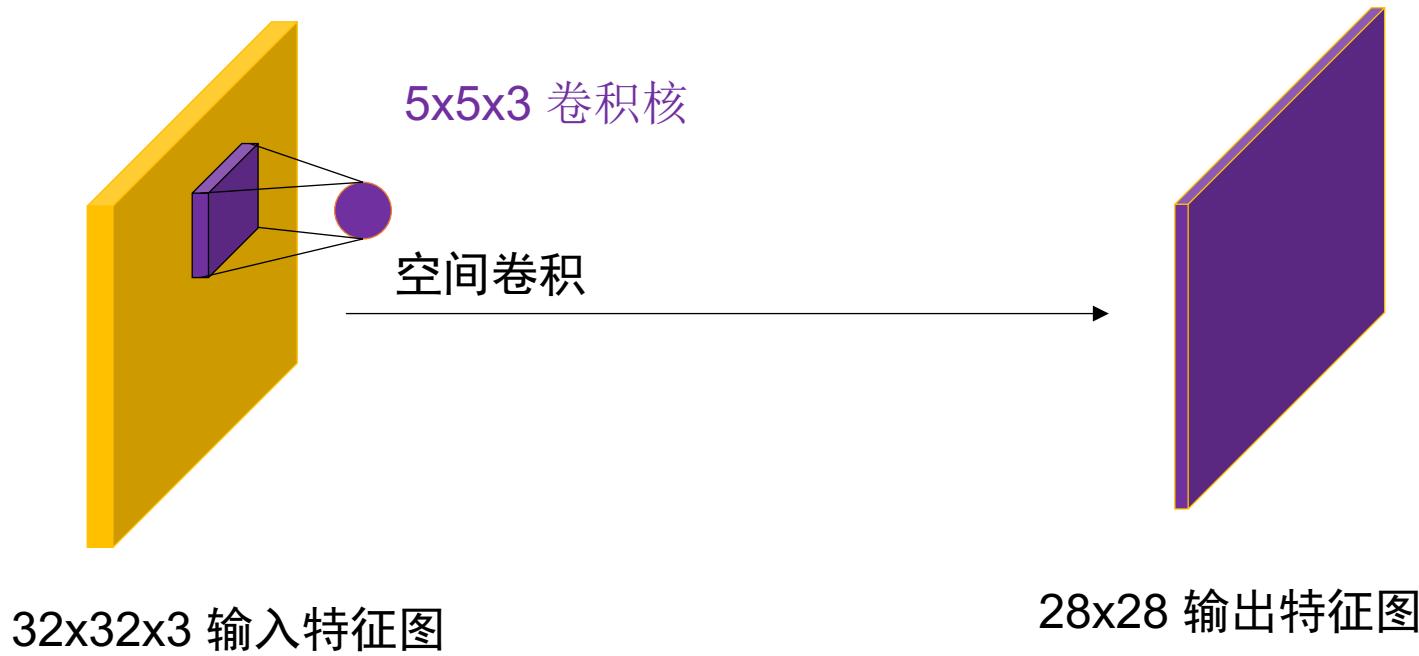
卷积

- 另一个 三维 绿色 卷积核在输入特征图上计算卷积.
- 移动这个绿色卷积核， 得到一个输出特征图.



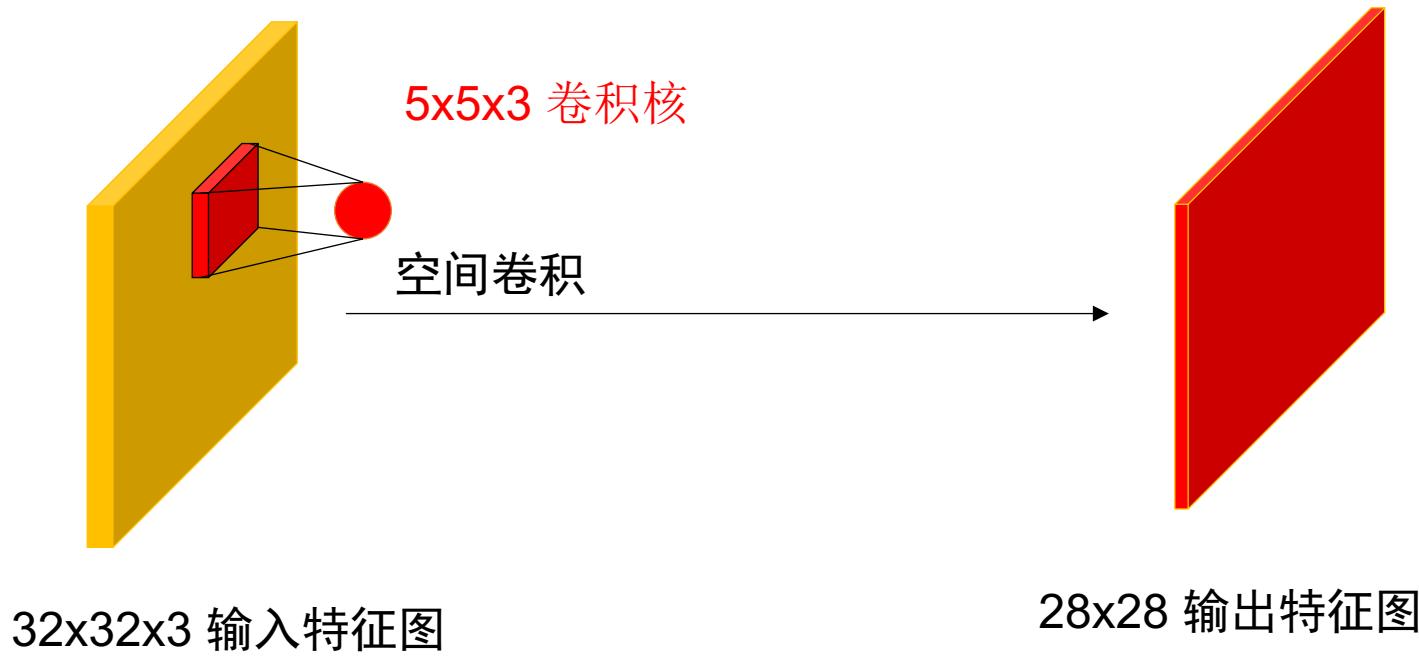
卷积

- 另一个 3-D 紫色 卷积核在输入特征图上计算卷积.
- 移动这个紫色卷积核， 得到一个输出特征图.



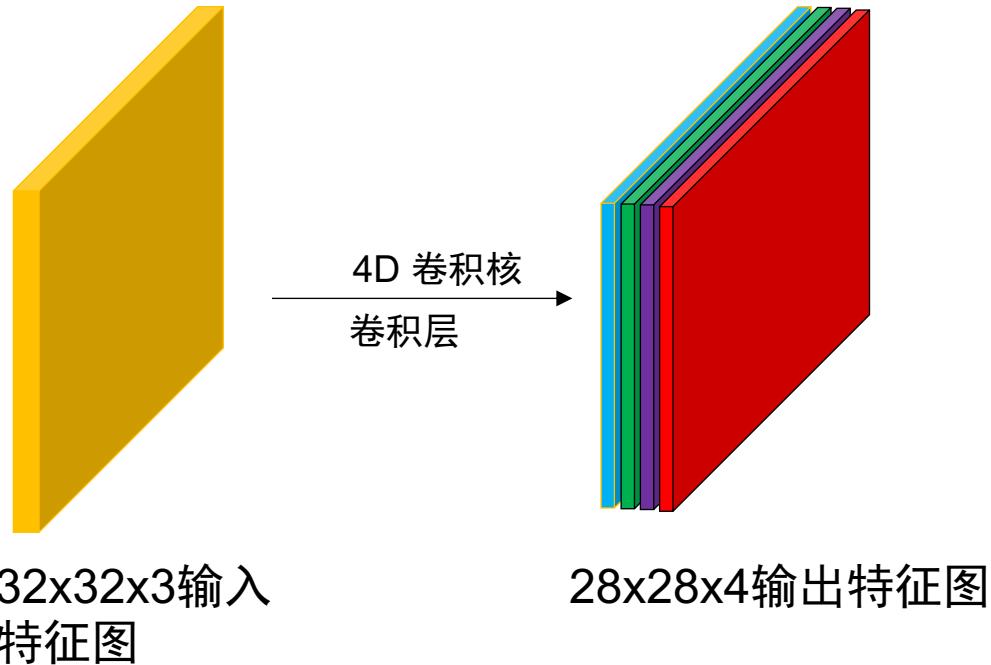
卷积

- 另一个 3-D 红色 卷积核在输入特征图上计算卷积.
- 移动这个红色卷积核， 得到一个输出特征图.



卷积

- 这4个 3-D 卷积核堆叠在一起成了一个 4-D 卷积核，形状大小是 $5 \times 5 \times 3 \times 4$.
- 这些输出特征图也串联在一起形成一个 3-D 输出特征图，形状为 $28 \times 28 \times 4$.



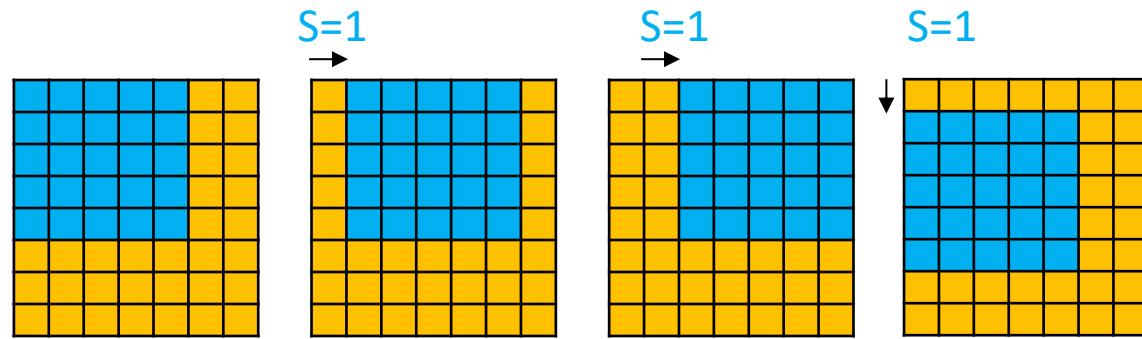
卷积核形状：
 $(H, W, I, O) \leftrightarrow (5, 5, 3, 4)$

H: 卷积核高度
W: 卷积核宽度
I: 卷积操作输入的通道数
O: 卷及操作输出的通道数

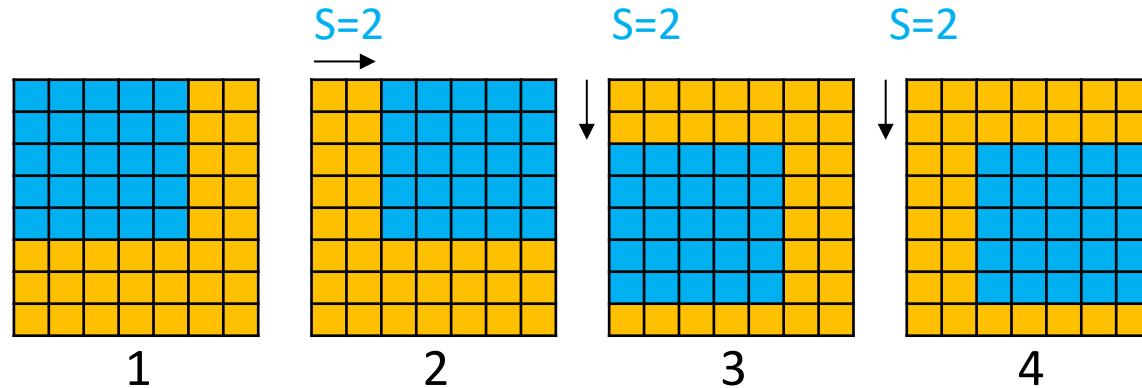
卷积核：步长

- 步长控制的是卷积核在输入特征图上的滑动步数。
- 用 S 表示

5x5 卷积
 $S=1$

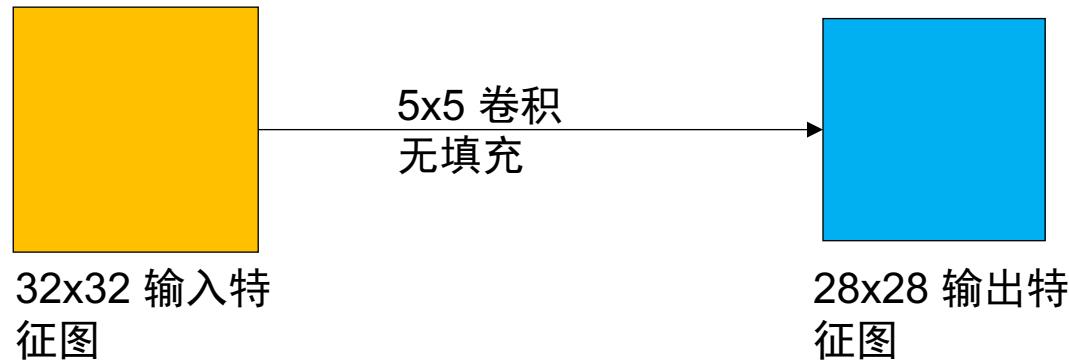


5x5 卷积
 $S=2$



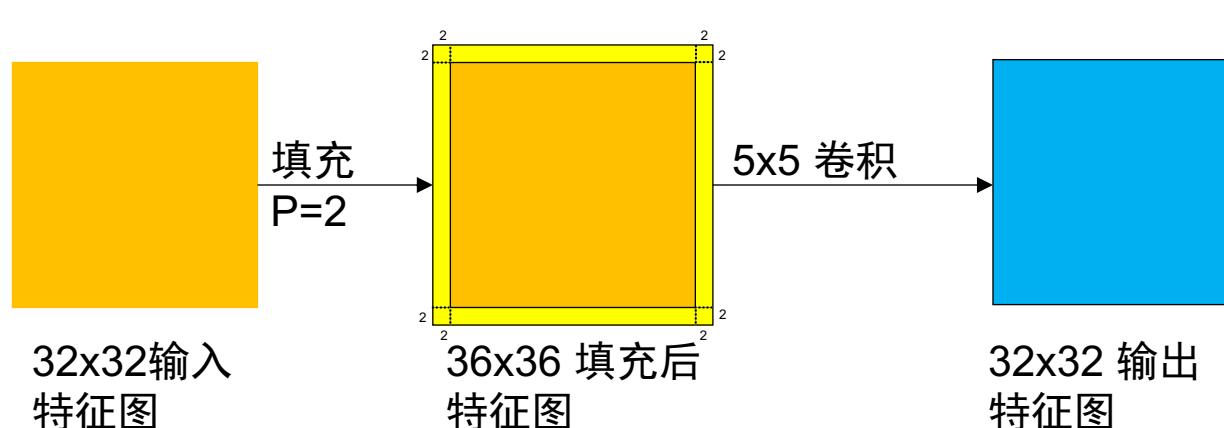
卷积层：填充 Padding

- 填充 避免卷积前后特征图尺寸变化.



零填充是最常用的特征图填充方法.

用 P 表示在四周填充的边长.



对 步长 $S=1$, 要保持特征图大小不变, 填充值大小为

$$P = \left\lfloor \frac{K - 1}{2} \right\rfloor$$

K 通常是奇数

卷积层：形状规则

- 输入特征图形状是 $H_1 \times W_1 \times C_1$.
- 卷积层的配置为：
 - 卷积核数目：C
 - 卷积核大小：K
 - 卷积步长：S
 - 边填充值：P
- 输出特征图形状是 $H_2 \times W_2 \times C_2$, 那么

$$W_2 = \left\lfloor \frac{W_1 - K + 2P}{S} \right\rfloor + 1$$

$$H_2 = \left\lfloor \frac{H_1 - K + 2P}{S} \right\rfloor + 1$$
$$C_2 = C$$

卷积层：形状规则

- 输入特征图形状是 $H_1 \times W_1 \times C_1$.

- 卷积层的配置为：

- 卷积核数目：C
- 卷积核大小：K
- 卷积步长：S
- 边填充值：P

- 输出特征图形状是 $H_2 \times W_2 \times C_2$, 那么

卷积核：我们要从输入中学习多少不同的东西。

步长：步长越长，输出特征图尺寸越小

零填充：控制输出特征图的大小

$$W_2 = \left\lfloor \frac{W_1 - K + 2P}{S} \right\rfloor + 1$$

$$H_2 = \left\lfloor \frac{H_1 - K + 2P}{S} \right\rfloor + 1$$
$$C_2 = C$$

激活函数

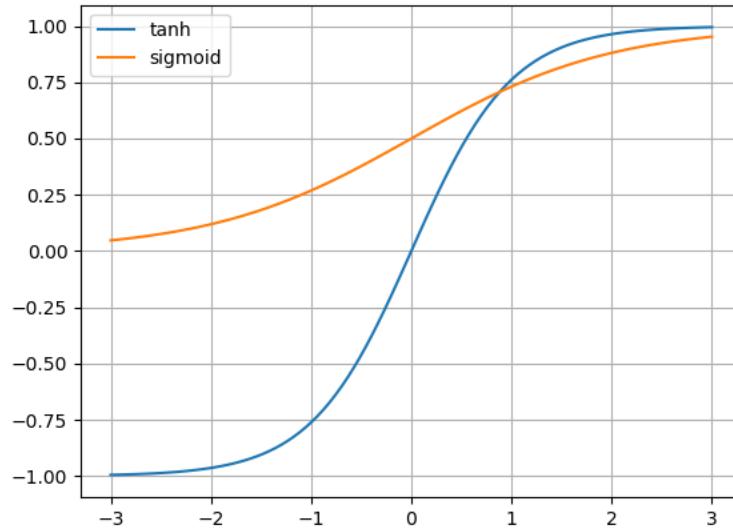
- 卷积层紧跟一个激活函数，实现输入到输出的非线性变化。

双曲正切(tanh)

$$\tanh(z) = \frac{e^{2z} - 1}{e^{2z} + 1}$$

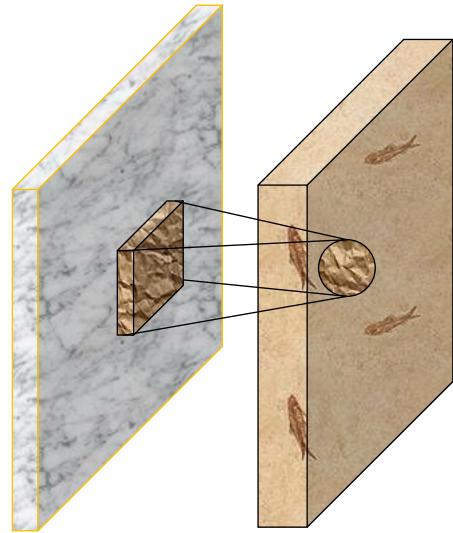
Softmax

$$\text{softmax}(z) = \frac{e^{z_i}}{\sum_{j=1}^k e^{z_j}}$$



关于激活函数更多的内容将在下一节介绍

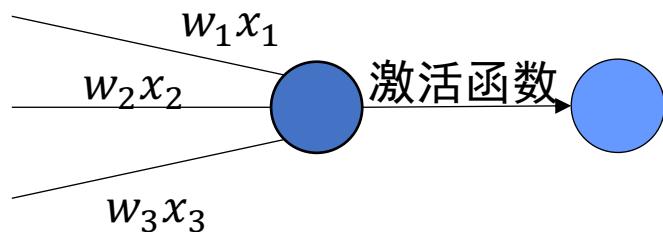
权重共享



32x32x3

28x28x4

- 每个输出神经元只和部分输入神经元用卷积核相连接；
- 这意味着卷积层的权重（卷积核）在输出的神经元之间是共享的。
- 每张输出特征图里的神经元共享相同的卷积核。
- 卷积操作有304个权重参数。
- 如果是全连接层，我们需要9.63M权重参数。
- 所以，卷积操作大幅减少了权重参数量



$$304 = 5 \times 5 \times 3 \times 4 \text{ (卷积核权重)} + 4 \text{ (偏差)}$$

$$9.63M = 32 \times 32 \times 3 \times 28 \times 28 \times 4 \text{ (卷积核权重)} + 4 \text{ (偏差)}$$

Pooling 池化/汇聚

- 池化：下采样 输入特征来提取特征

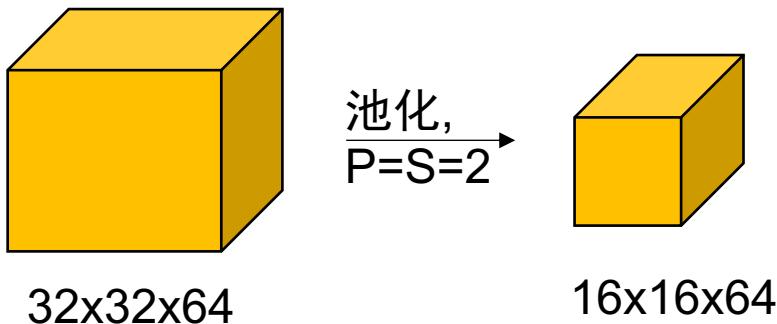
池化尺寸大小 (P)

根据池化规则在指定区域下采样特征.

步长 (S)

池化区域滑动的步数

通常，池化层里池化尺寸大小与步长大小相同

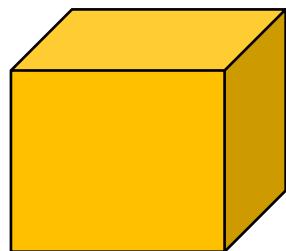


Note:

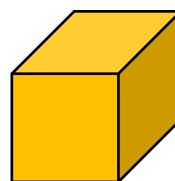
池化的执行不在通道上执行

最大池化

- 最大池化: 选择池化区域下最大值



池化,
 $P=S=2$



32x32x64

16x16x64

池化尺寸大小 (P)

根据池化规则在指定区域下采样特征.

步长 (S)

池化区域滑动的步数

通常, 池化层里池化尺寸大小与步长大小相同

1	3	2	4	5	-1
7	-6	-3	-2	-6	-3
2	0	9	3	8	6
0	1	2	4	-1	-3
-2	0	1	1	-4	-1
-3	4	0	-1	-5	-2

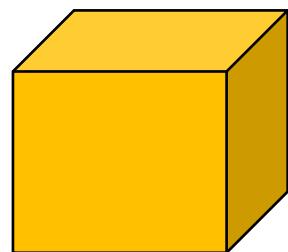
最大池化
 $P=2, S=2$

7	4	5
2	9	8
4	1	-1

池化是沿着特征尺寸的长/宽执行, 不会改变深度

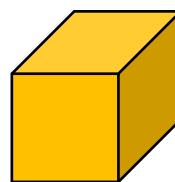
平均池化

- 平均池化：平均当前区域下的特征值。



32x32x64

池化,
 $P=S=2$



16x16x64

与最大池化类似，把求
最大值改成求平均值。

1	3	2	4	5	-1
7	-6	-3	-2	-6	-3
2	0	9	3	8	6
0	1	2	4	-1	-3
-2	0	1	1	-4	-1
-3	4	0	-1	-5	-2

$$((-2)+(-3)+0+4)/4=-0.25$$

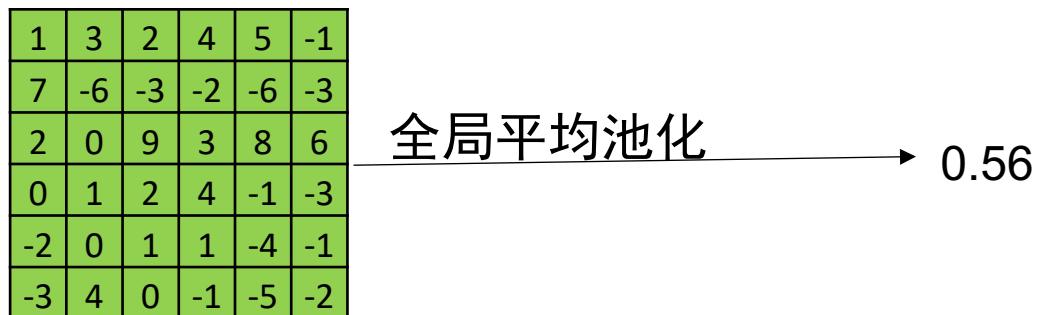
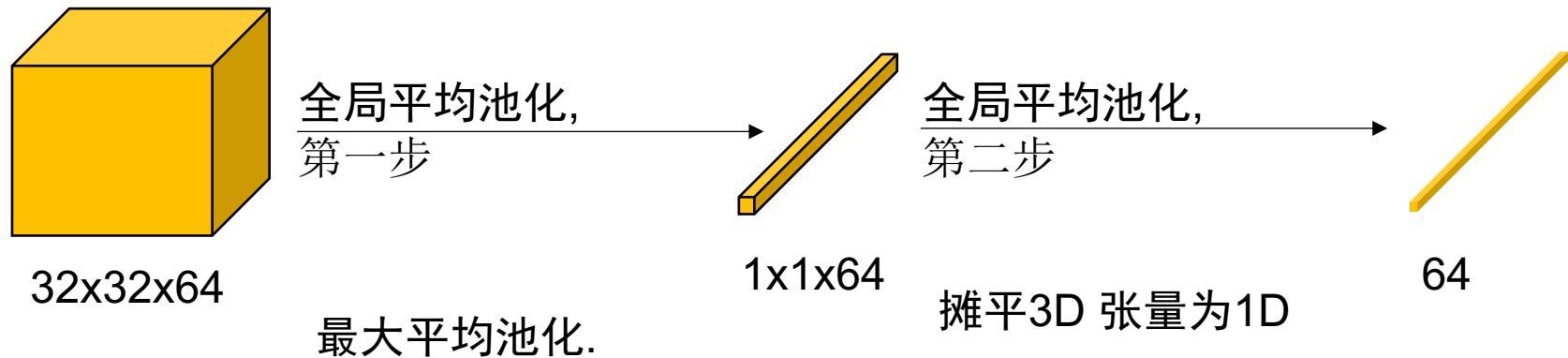
平均池化
 $P=2, S=2$

1.25	0.25	-1.25
0.75	4.50	2.50
-0.25	0.25	-3.00

平均池化会需要额外的计算

全局平均池化

- 池化所有的空间维度的元素，得到一个摊平的张量。



现在， 我们已经有了

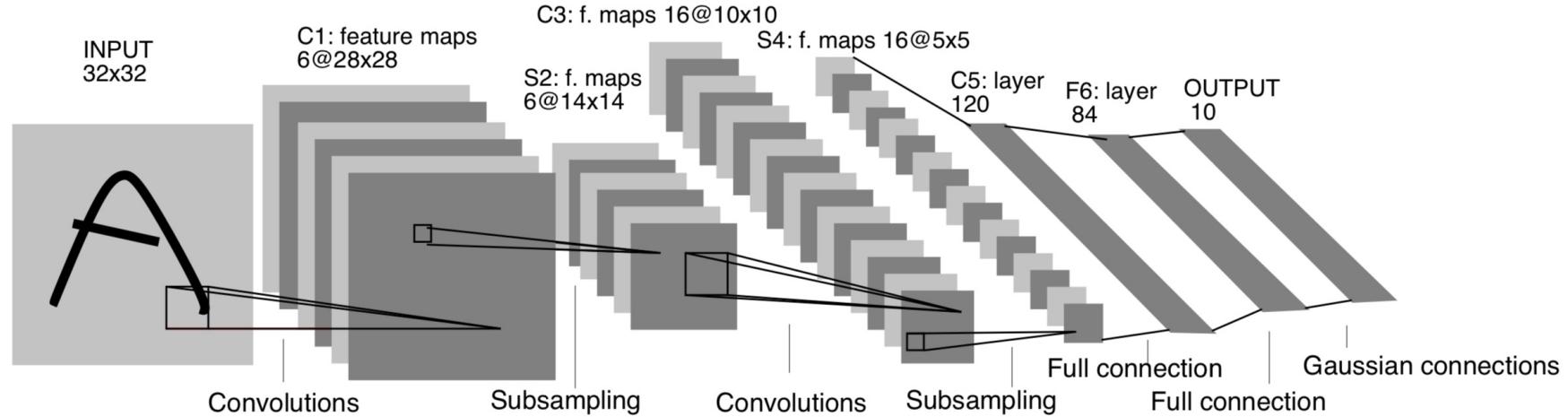
- 卷积
 - 局部连接，参数量减少
 - 不同的特征

- 池化
 - 平移不变性

- 如何形成网络？

LeNet-5

- 例子: LeNet-5 用来MNIST 手写字符分类
10, 000个示例的测试集, 仅有82个识别错误



LeNet 结构:

卷积-最大池化-卷积-最大池化-全连接-全连接-分类

CONV-POOL-CONV-POOL-FC-FC-SOFTMAX

LeNet-5

• LeNet-5 结构

Q: 第一层卷积的输出大小?

INPUT	32x32
CONV,6	5x5 卷积核, 无填充, 双曲正切激活
POOL	2x2 步长 & 池化大小
CONV,1 6	5x5 卷积核, 无填充, 双曲正切激活
POOL	2x2 步长 & 池化大小
FC,120	双曲正切激活
FC,84	双曲正切激活
FC,10	Softmax 激活

LeNet-5

• LeNet-5 结构

INPUT	32x32
CONV,6	5x5 卷积核, 无填充, 双曲正切激活
POOL	2x2 步长 & 池化大小
CONV,1 6	5x5 卷积核, 无填充, 双曲正切激活
POOL	2x2 步长 & 池化大小
FC,120	双曲正切激活
FC,84	双曲正切激活
FC,10	Softmax 激活

Q: 第一层卷积的输入大小?

A:

特征图的大小: $32-5+1=28$

深度: 6

输出大小: 28x28x6

LeNet-5

• LeNet-5 结构

INPUT	32x32
CONV,6	5x5 卷积核, 无填充, 双曲正切激活
POOL	2x2 步长 & 池化大小
CONV,1 6	5x5 卷积核, 无填充, 双曲正切激活
POOL	2x2 步长 & 池化大小
FC,120	双曲正切激活
FC,84	双曲正切激活
FC,10	Softmax 激活

Q: 第一层卷积的输入大小?

A:

28x28x6

Q: 第一层池化后的输出大小?
?

LeNet-5

• LeNet-5 结构

INPUT	32x32
CONV,6	5x5 卷积核, 无填充, 双曲正切激活
POOL	2x2 步长 & 池化大小
CONV,1 6	5x5 卷积核, 无填充, 双曲正切激活
POOL	2x2 步长 & 池化大小
FC,120	双曲正切激活
FC,84	双曲正切激活
FC,10	Softmax 激活

Q: 第一层卷积输入大小?

A:

28x28x6

Q: 第一层池化后的输出大小

?

A:

池化只是在长、宽应用，深度不变

长度/宽度: $28/2=14$

输出大小: 14x14x6

LeNet-5

• LeNet-5 结构

INPUT	32x32
CONV,6	5x5 卷积核, 无填充, 双曲正切激活
POOL	2x2 步长 & 池化大小
CONV,1 6	5x5 卷积核, 无填充, 双曲正切激活
POOL	2x2 步长 & 池化大小
FC,120	双曲正切激活
FC,84	双曲正切激活
FC,10	Softmax 激活

Q: 第一层卷积输入大小?

A:

28x28x6

Q: 第一层池化后的输出大小

?

A:

14x14x6

Q: 第一个全连接层参数数量

?

LeNet-5

• LeNet-5 结构

INPUT	32x32
CONV,6	5x5 卷积核, 无填充, 双曲正切激活
POOL	2x2 步长 & 池化大小
CONV,1 6	5x5 卷积核, 无填充, 双曲正切激活
POOL	2x2 步长 & 池化大小
FC,120	双曲正切激活
FC,84	双曲正切激活
FC,10	Softmax 激活

Q: 第一层卷积输入大小?

A:

28x28x6

Q: 第一层池化后的输出大小

?

A:

14x14x6

Q: 第一个全连接层参数数量

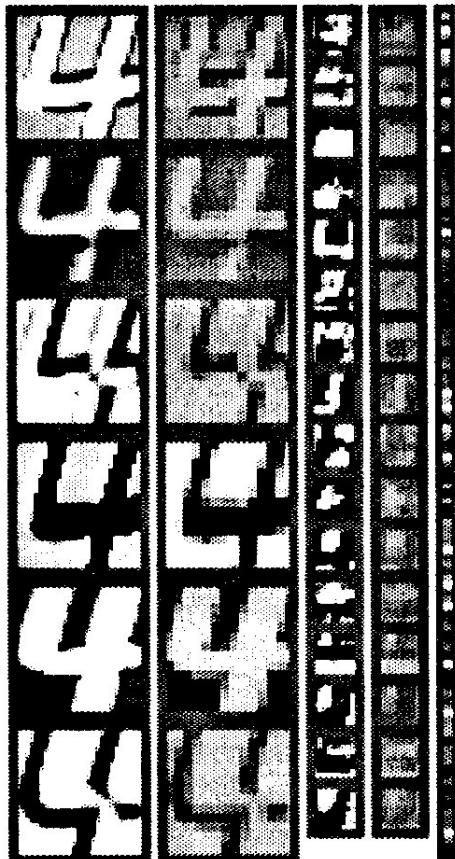
?

第二层池化的输出大小是 $(14-5+1)/2=5$.

深度是 16

所以第一层全连接层的参数总数是
 $5 \times 5 \times 16 \times 120 + 120 = 48120$

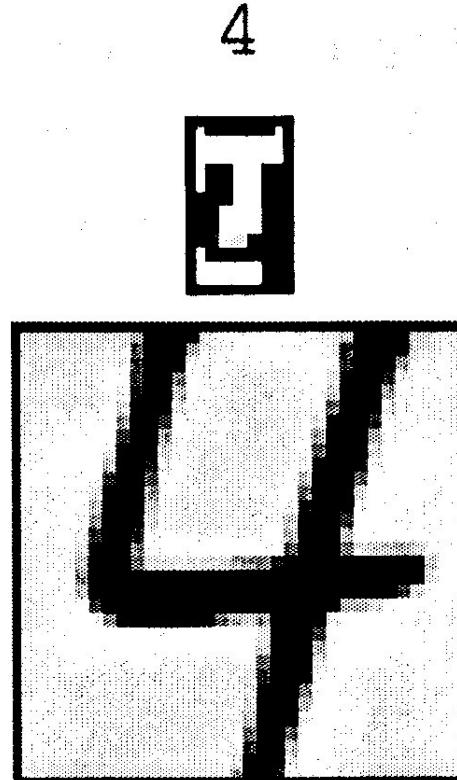
LeNet-5: 识别结果



C1

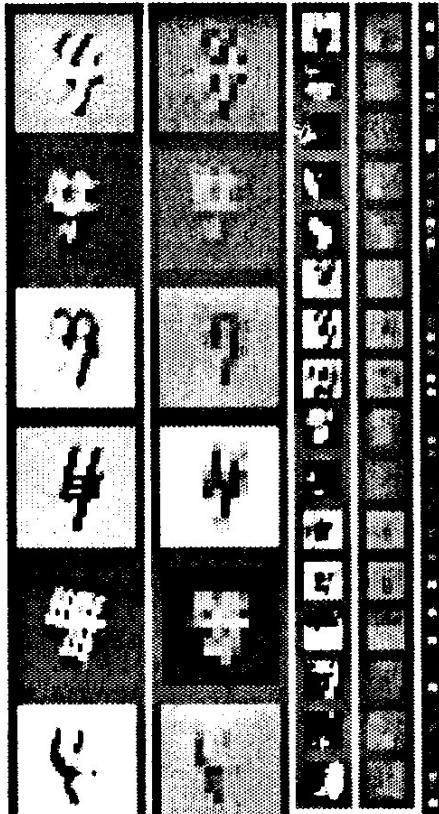
C3

C5



输入

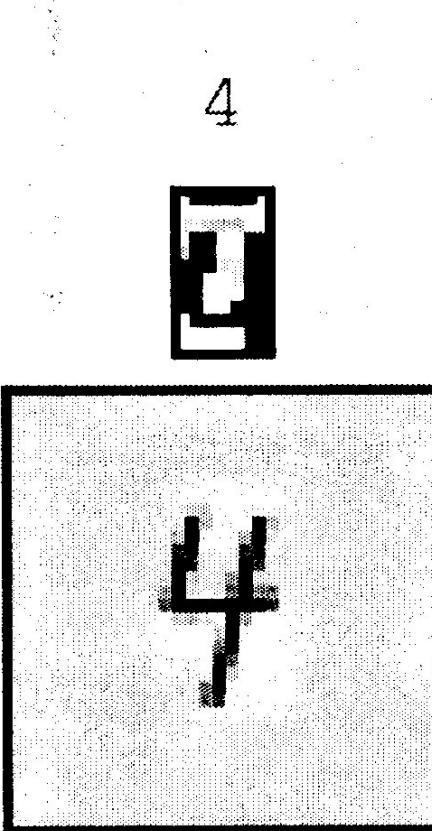
LeNet-5: 识别结果



C1

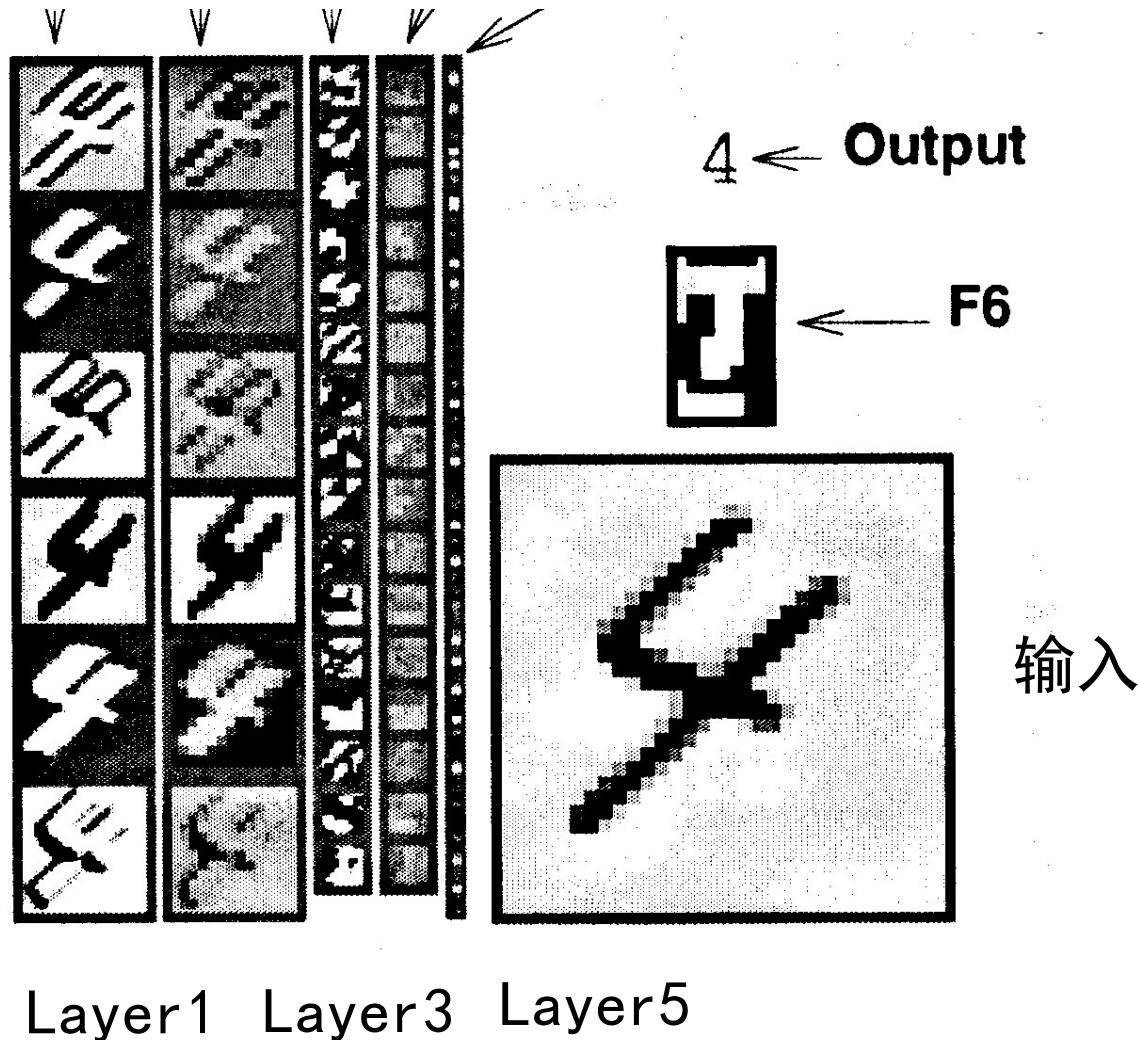
C3

C5

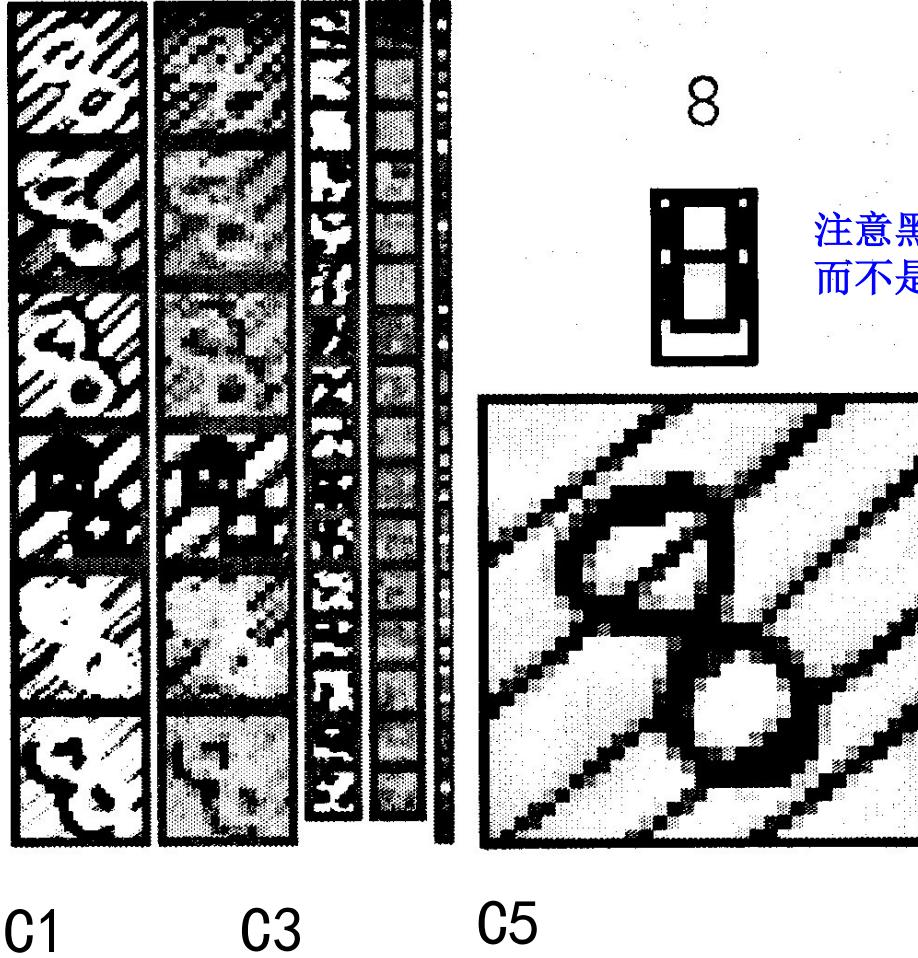


输入

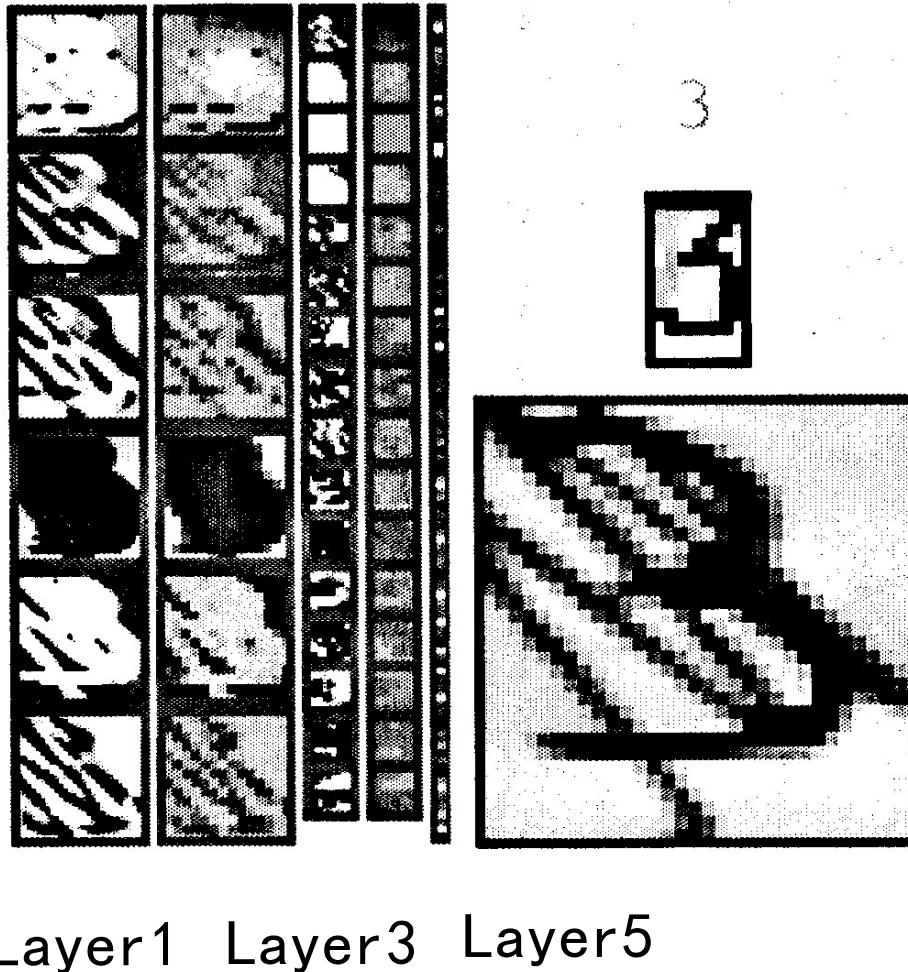
LeNet-5: 识别结果



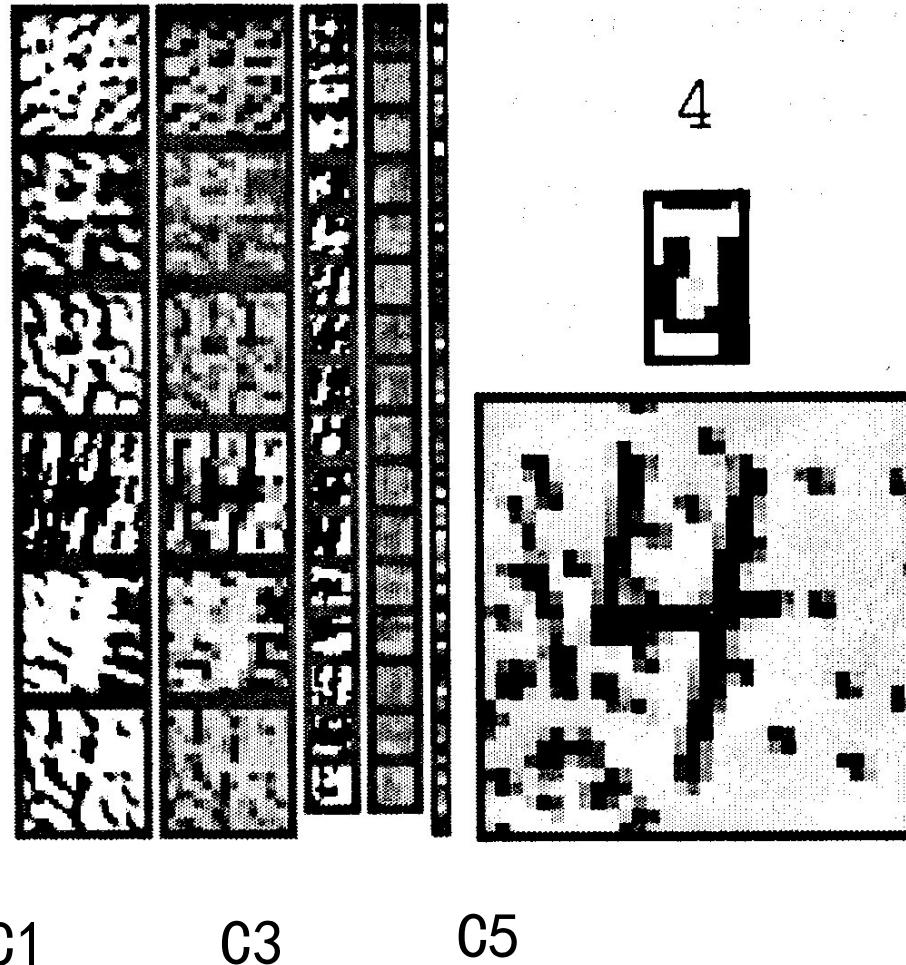
LeNet-5: 识别结果



LeNet-5: 识别结果

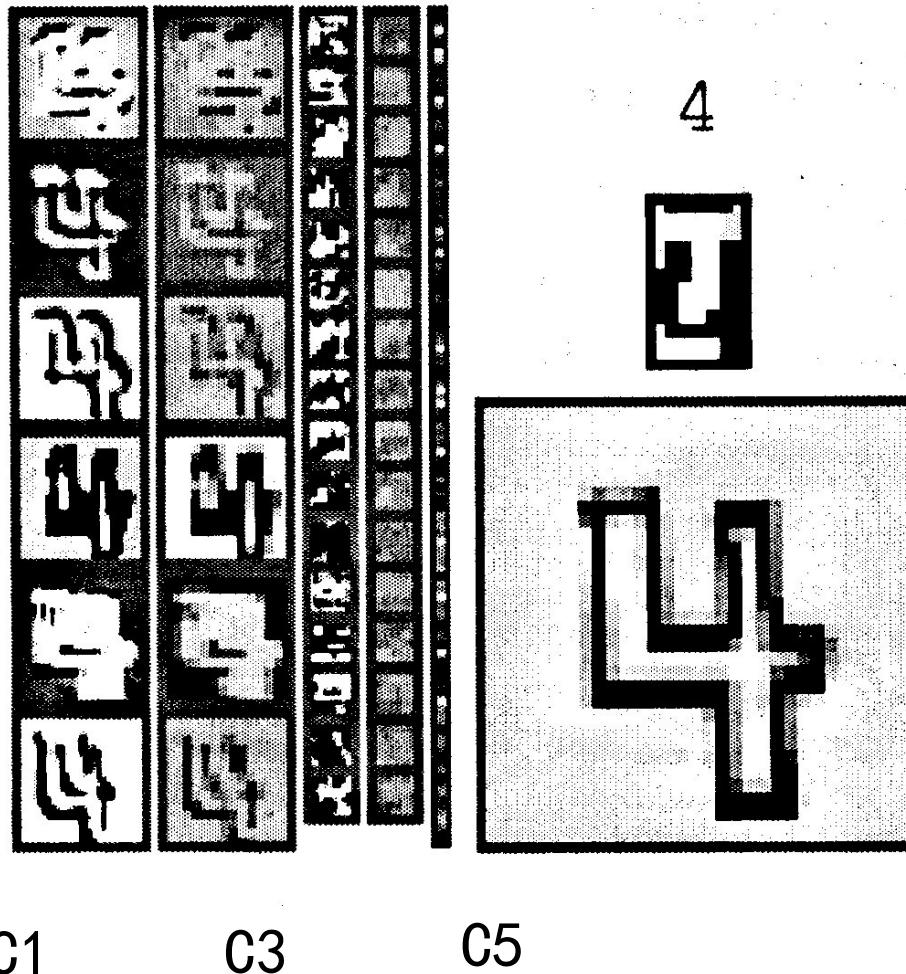


LeNet-5: 识别结果



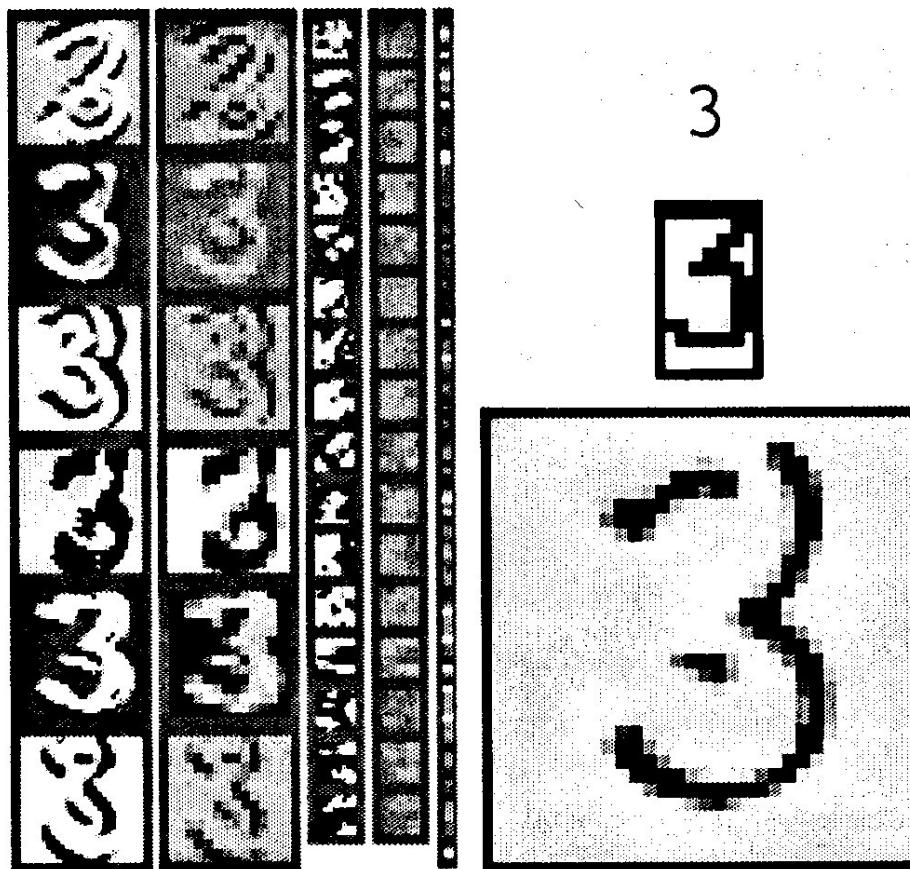
输入
背景噪声

LeNet-5: 识别结果



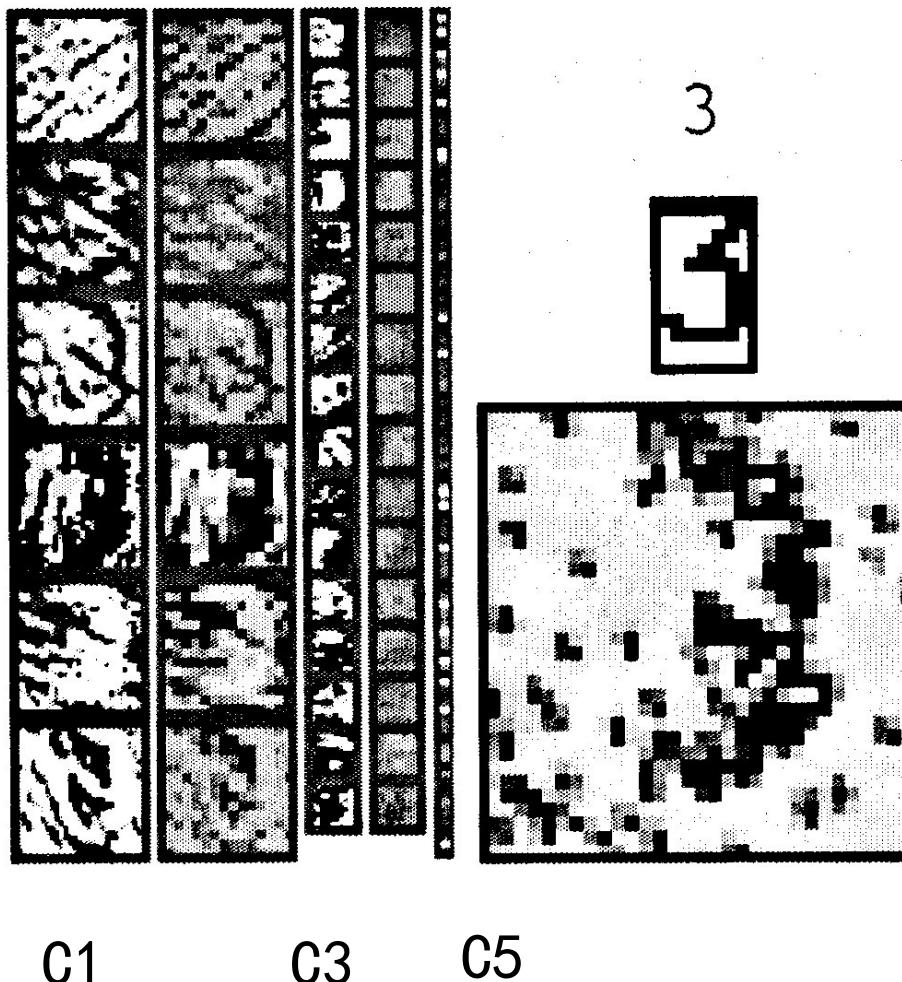
输入
特殊形状输入

LeNet-5: 识别结果



Layer1 Layer3 Layer5

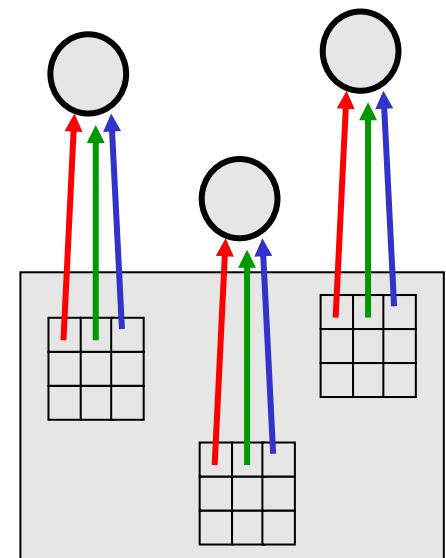
LeNet-5: 识别结果



卷积

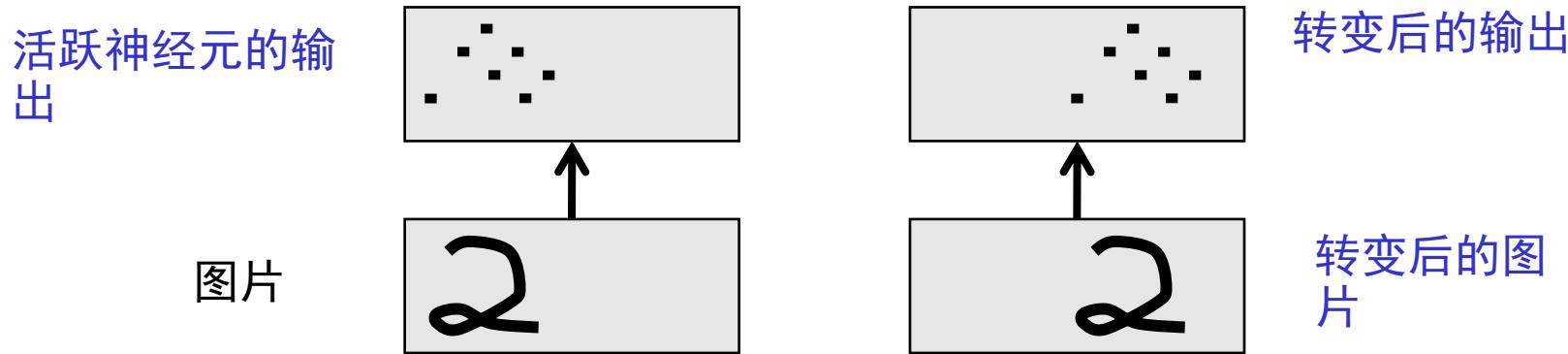
- 在不同位置都用了相同的卷积核(滤波器)
 - 在不同的位置和方向上进行重复的特征提取 (计算量大)
 - 相同的卷积核极大地减少了要学习的参数的数量.
- 用了不同类型的卷积核
 - 用于提取不同的特征.
 - 图片中的每一块能表达出不同的信息。

相同颜色的连接表示同样的权重



卷积核重复的意义

- 等变化的行为 (Equivariant activities) : 重复的特征并不会使神经元的行为不变，而是使神经元的行为发生等同的变化。



- 知识的不变性 (knowledge invariant) : 如果在训练过程中某个特征在某些位置有用，则该特征的卷积核将在测试过程中的所有位置可用。因为变换不会对其产生影响。

池化层的输出

- 通过对相邻的相同卷积核的输出求平均/Max，得到单个输出传给下一个层，可以在每个层获得近似的平移不变性。
 - 这样减少了下一层的输入，我们可以有更多不同的特征图
 - 最大池化的效果会更好。
- **问题：** 经过多个层的池化后，我们丢失了有关事物精确位置的信息。
 - 这使得在更高层的时候，无法对空间关系进行精确识别

LeNet的82个错误



请注意，大多数错误都是人们认为很容易出现的情况。

其他改进的方法

- LeNet 利用了以下的原则来保证知识的不变性：
 - 本地连接
 - 权重共享
 - 池化.
- 通过对输入图片进行变化以及其他的一些技巧错误能够减少到40个。
(Ranzato 2008)

- Ciresan et. al. (2010) 通过创建大量精心设计的额外训练数据来注入不变性知识：
 - 对于每张训练图像，他们通过应用许多不同的变换产生许多新的训练图片.
 - 然后，他们可以在 GPU 上训练一个大而深的网络。
 - 他们取得了35个错误.

CNN 设计原则

LeNet-5 带来的启发

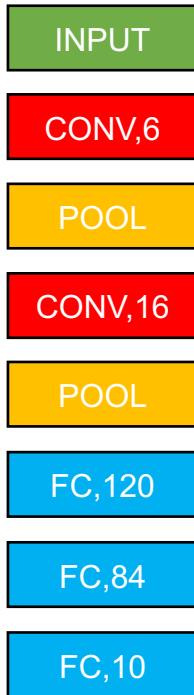
INPUT	32x32
CONV,6	5x5 卷积核, 无填充, 双曲正切激活
POOL	2x2 步长 & 池化大小
CONV,1 6	5x5 卷积核, 无填充, 双曲正切激活
POOL	2x2 步长 & 池化大小
FC,120	双曲正切激活
FC,84	双曲正切激活
FC,10	Softmax 激活

CNN 设计原则

LeNet-5的启发

三种基本的层: **卷积, 池化层 和全连接层.**

- 卷积层: 用于提取平面特征, 计算输入的局部域与卷积核对应的输入。
- 池化: 执行平面空间上的下采样的操作.
- 全连接层: 负责计算分类结果.



如何设计神经网络

若干卷积-非线性激活层堆叠起来, 接上池化层, 直到输入特征图大小很小。然后, 全连接层用于产生分类结果。

典型设计: **((CONV)*N-POOL)*M-(FC)*K-SOFTMAX**

M,N,K 常数

CNN 设计原则

LeNet-5的启发

INPUT

CONV,6

POOL

CONV,16

POOL

FC,120

FC,84

FC,10

- 不要用超过3层的FC层。FC不擅长发现层次化特征，堆FC层对提高网络性能没有帮助。
- 相反，堆卷积层对提升性能帮助很大。随卷积层数目增多感受野区域会增大。堆3-4层卷积之后跟池化就有可能取得非常好的结果。（但是堆得太多，参数的学习变得非常困难。）

这节课，我们学习了，

- 卷积神经网络

- 在3D图像上计算卷积
 - 感受野&共享权重连接.
 - 下采样模块

- 图像识别应用

- 手写字符识别: LeNet-5
 - CNN特征图的可视化

- 基本的CNN设计理念

- 堆卷积层而不是全连接层.
 - 典型的一个设计原则是:

((CONV)*N-POOL)*M-(FC)*K-SOFTMAX