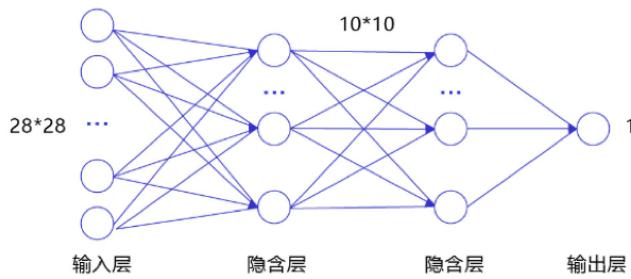


CNN

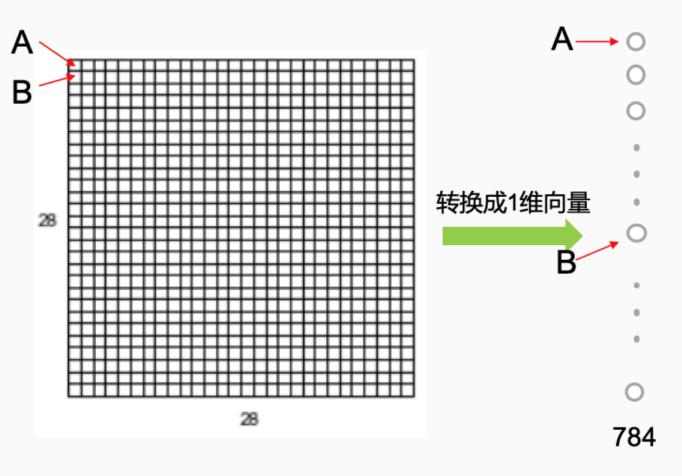
1 卷积

1.1 Back

全连接网络图：一张图片全部展开为一维向量输入网络， $28 \times 28 \rightarrow 784 \times 1$



1. 输入数据的空间信息被丢失。AB两点之间的空间相关性



2. 模型参数过多，过拟合（模型在训练数据上表现很好，在新的数据表现差）

每个像素点都要跟所有输出的神经元相连接。当图片尺寸变大时，输入神经元的个数会按图片尺寸的平方增大，导致模型参数过多，容易发生过拟合。（刷题）

例如：对于一幅 1000×1000 的输入图像而言，如果下一个隐含层的神经元数目为 10^6 个，那么将会有 $1000 \times 1000 \times 10^6 = 10^{12}$ 个权重参数。

解决上述问题：引入卷积对输入图像进行特征提取

1.2 卷积核/特征图/计算

- 卷积核（kernel）：也叫滤波器（filter）。卷积核中的数值为对输入图像中与卷积核大小相同的子块像素点进行卷积计算采用的权重。
- 特征图（feature map）：卷积滤波结果在卷积神经网络中被称为特征图（feature map）。

1	2	3
4	5	6
7	8	9

(a) $0 \times 1 + 1 \times 2 + 2 \times 4 + 3 \times 5 = 25$

1	2	3
4	5	6
7	8	9

(b) $0 \times 2 + 1 \times 3 + 2 \times 5 + 3 \times 6 = 31$

1	2	3
4	5	6
7	8	9

(c) $0 \times 4 + 1 \times 5 + 2 \times 7 + 3 \times 8 = 43$

1	2	3
4	5	6
7	8	9

(d) $0 \times 5 + 1 \times 6 + 2 \times 8 + 3 \times 9 = 49$

- 输入形状 $n_h \times n_w$, 卷积核形状 $k_h \times k_w$, 输出形状: $(n_h - k_h + 1) \times (n_w - k_w + 1)$

1.2.1.1 图像中目标的边缘检测

PYTHON

```
# 构造 6x8 的黑白图像, 中间 4 列为黑色 (0), 其余为白色 (1)
image = torch.ones((1, 1, 6, 8)) # (batch_size=1, channels=1, height=6, width=8)
image[:, :, :, 2:6] = 0 # 中间 4 列设为黑色
print("原始图像:")
print(image)

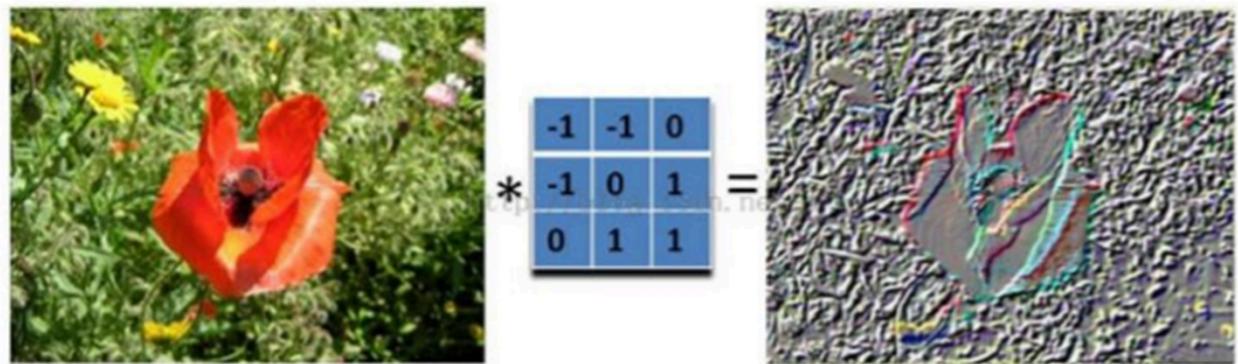
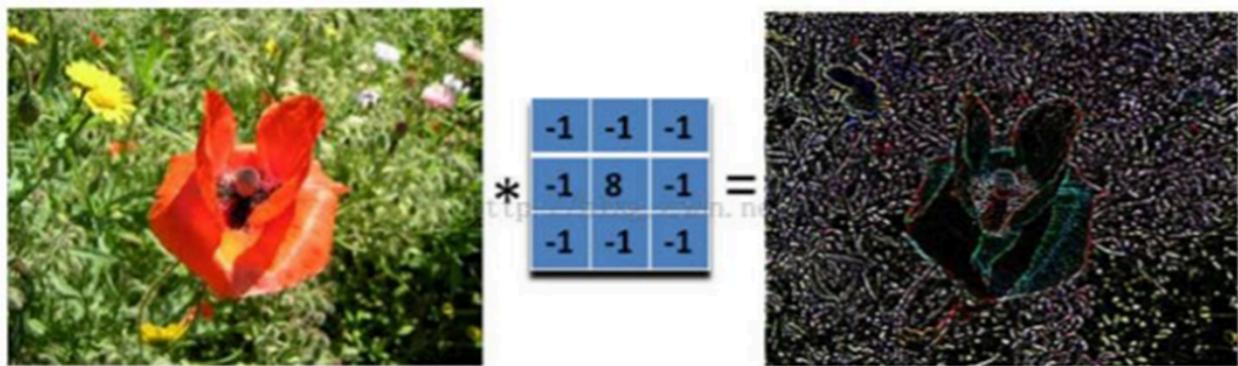
# 构造垂直边缘检测核 (高度=1, 宽度=2)
kernel = torch.tensor([[[[1.0, -1.0]]]]) # (out_channels=1, in_channels=1, height=1, width=2)

# 进行互相关运算
# 2D卷积, 不使用padding
result = F.conv2d(image, kernel, padding=0) print("\n互相关运算结果:")
print(result)
```

不同卷积核作用不同:

p1 : 检测图像的边缘

p2: 水平和对角线方向的边缘检测



1.3 填充和步幅

1.3.1 填充

多层卷积会损失边缘像素。解决办法之一 → 填充（在边缘像素点周围填充“0”（即0填充），使得输入图像的边缘像素也可以参与卷积计算）

输入图片尺寸： 4×4

0	0	0	0	0	0
0	13	14	15	16	0
0	9	10	11	12	0
0	5	6	7	8	0
0	1	2	3	4	0
0	0	0	0	0	0

(a)padding=1

0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	13	14	15	16	0	0	0
0	0	9	10	11	12	0	0	0
0	0	5	6	7	8	0	0	0
0	0	1	2	3	4	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0

(b)padding=2

filter:

1	0	1
0	1	0
1	0	1

- picture(a) : 输出图片尺寸为 4×4
- picture(b) : 输出图片尺寸为 6×6

输出形状变为： $(n_h - k_h + p_h * 2 + 1) \times (n_w - k_w + p_w * 2 + 1)$

p_h 为填充的行， p_w 为填充的列

卷积核大小通常使用1, 3, 5, 7这样的奇数，这样如果使用的填充大小为

$p_h = \frac{k_h - 1}{2}$, $p_w = \frac{k_w - 1}{2}$, 则可以使得卷积之后图像尺寸不变。例如当卷积核大小为3时 padding大小为1, 卷积之后图像尺寸不变; 同理, 如果卷积核大小为5, padding大小为2, 也能保持图像尺寸不变。

PYTHON

```

import torch
from torch import nn

def comp_conv2d(conv2d, X):
    X = X.reshape((1, 1) + X.shape) # PyTorch输入4D
    Y = conv2d(X) # 计算二维卷积, 输出4张量
    return Y.reshape(Y.shape[2:]) # 只返回四维张量的H、W

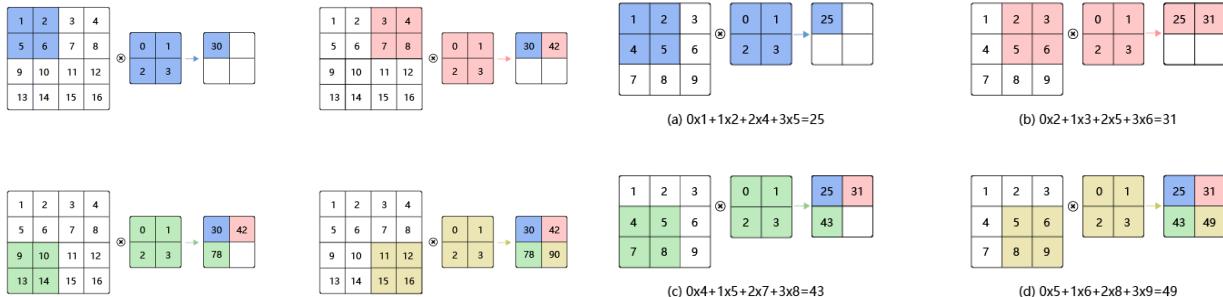
# 输入形状为 (batch_size, num_channels, height, width)
conv2d_1 = nn.Conv2d(1, 1, kernel_size=3, padding=1)
X_1 = torch.rand(size=(4, 4)) # 4x4的随机张量, 数值在 [0, 1] 之间
result_1 = comp_conv2d(conv2d_1, X_1).shape # 张量形状
print('padding=1时的输出尺寸', result_1)

conv2d_2 = nn.Conv2d(1, 1, kernel_size=3, padding=2)
X_2 = torch.rand(size=(4, 4))
result_2 = comp_conv2d(conv2d_2, X_2).shape
print('padding=2时的输出尺寸', result_2)

```

1.3.2 步幅

卷积操作 → 缩减采样次数或高效计算, 每次滑动多个元素。



- 左图步长为2的卷积过程, 每次移动两个像素点
- 右图步长为1的卷积过程, 每次移动一个像素点
输出特征图形状: (s_h 垂直步幅, s_w 水平步幅)

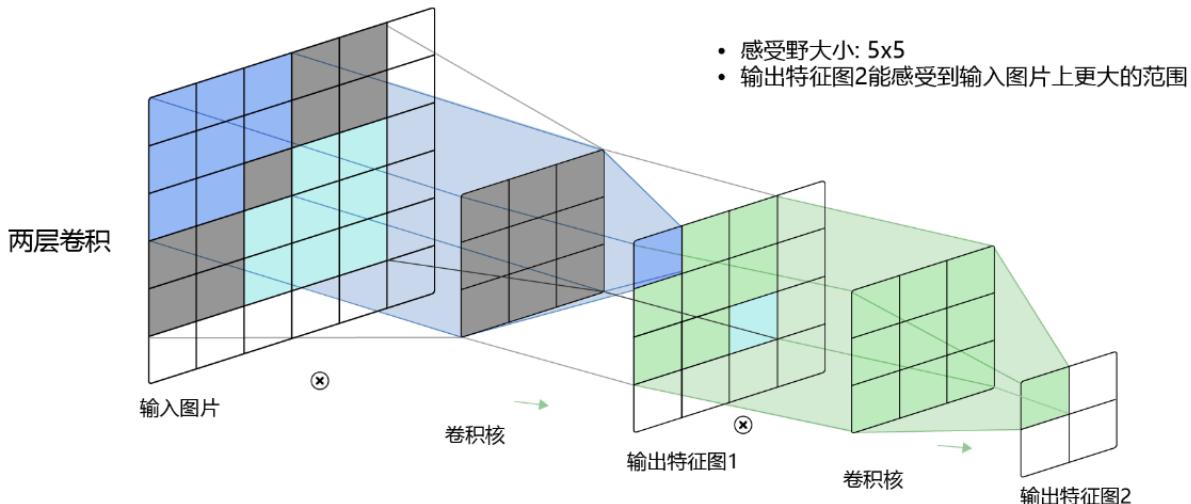
$$\left(\frac{n_h - k_h + p_h * 2 + s_h}{s_h} \right) \times \left(\frac{n_w - k_w + p_w * 2 + s_w}{s_w} \right)$$

```
conv2d_3 = nn.Conv2d(1, 1, kernel_size=2, stride=2)
X_3 = torch.rand(size=(4, 4))
result_3 = comp_conv2d(conv2d_3, X_3).shape
print('stride=2时的输出尺寸', result_3)
```

1.4 感受野

卷积所得结果当中，每个特征图像素点的取值依赖于输入图像的某个区域，该区域就称为感受野。

感受野 → 对应输入图片的像素尺寸，不是上一层输入尺寸

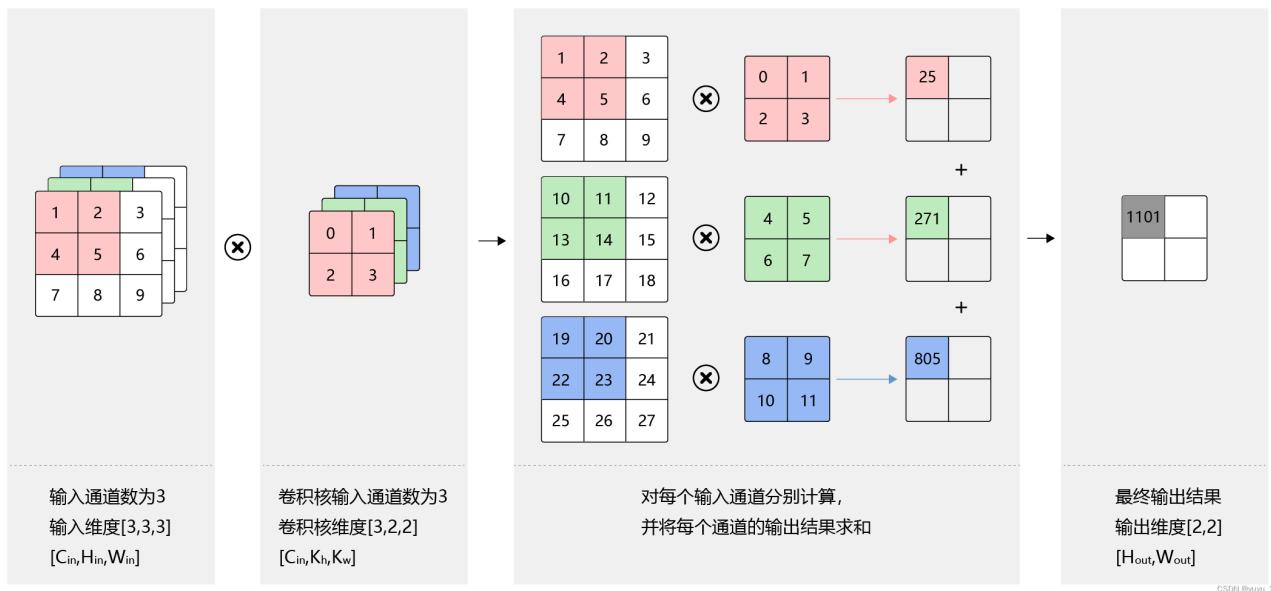


卷积网络越深，感受野越大，特征图中的一个像素点包含更多信息

1.5 多输入输出通道场景

灰度图单通道，彩色图片RGB三个通道，处理多输入通道场景，相应输出特征图也具有多个通道。

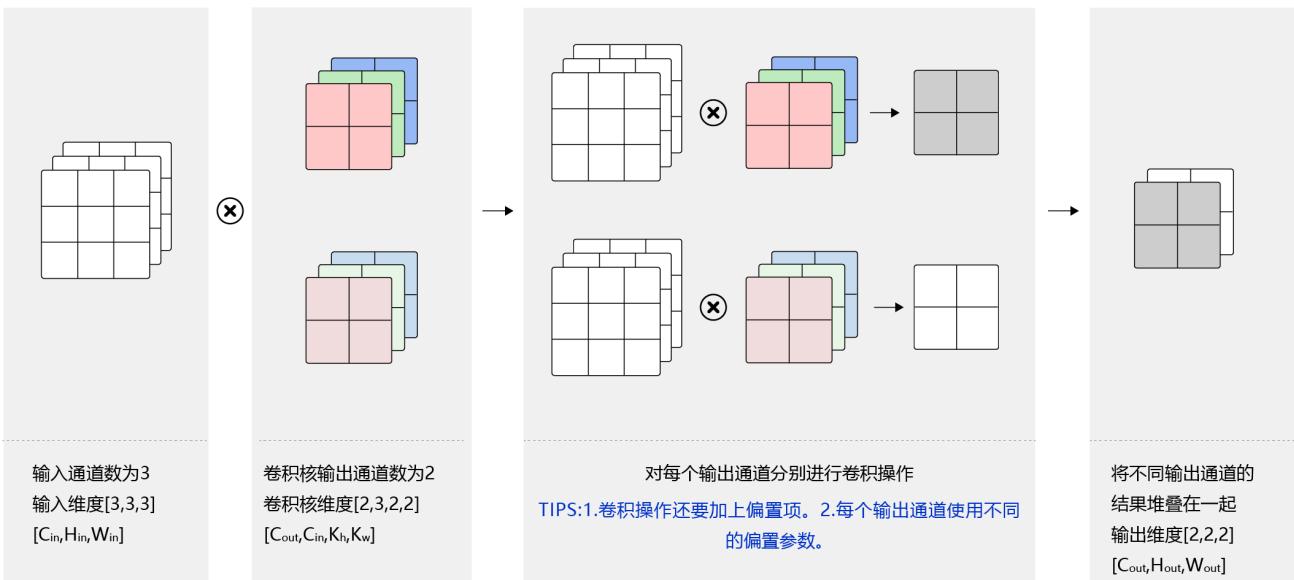
1.5.1.1 多输入通道场景



1.5.1.2 多输出通道场景

检测多种类型的特征，使用多种卷积核计算

- 输出通道的数目通常也被称作卷积核的个数，这里有两个卷积核。
- 红绿蓝代表第一个卷积核的三个输入通道；浅红浅绿浅蓝代表第二个卷积核的三个输入通道。

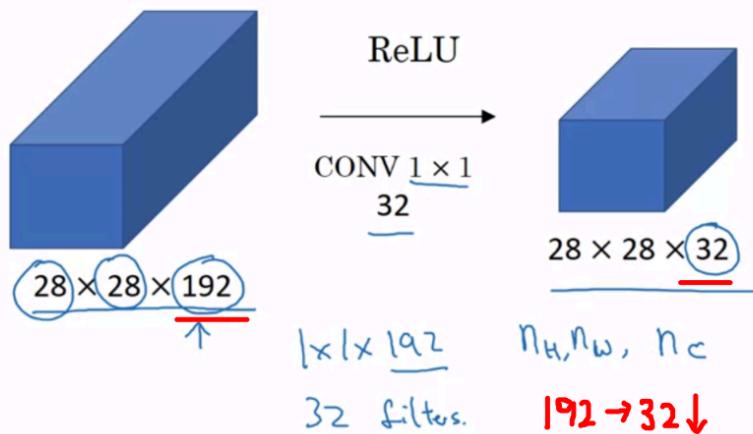


1.5.1.3 1×1卷积层

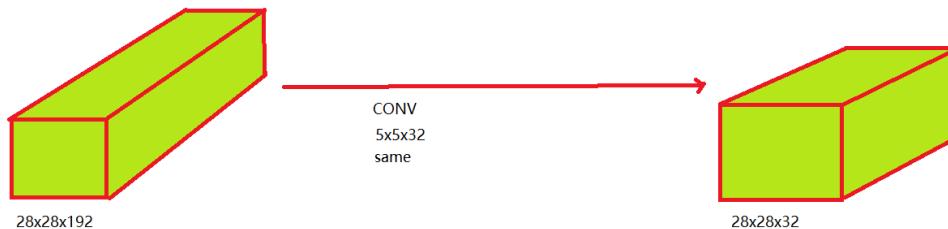
对卷积核通道数进行降维（减少参数量）和升维。

降维：

Using 1×1 convolutions

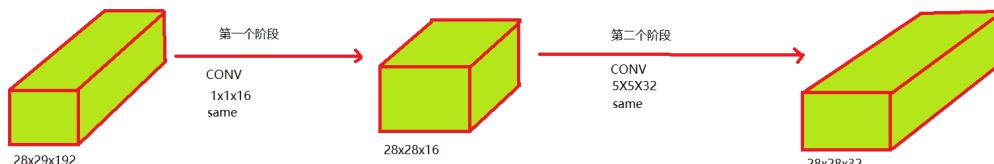


- 减少参数量:



总的参数量计算: $28 \times 28 \times 192 \times 5 \times 5 \times 32 = 120\,422\,400$ 约等于1.2亿

CSDN @Keep_Trying_Go



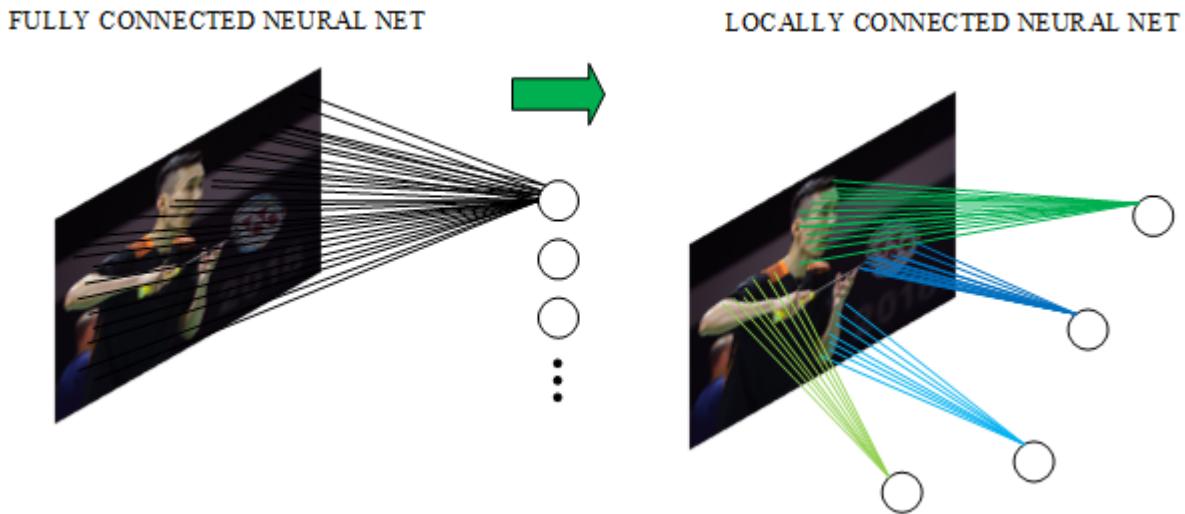
参数计算: 第一个阶段的参数量计算: $28 \times 28 \times 192 \times 1 \times 1 \times 16 = 2408\,448$ 约等于240万
第二个阶段的参数量计算: $28 \times 28 \times 16 \times 5 \times 5 \times 32 = 10\,035\,200$ 约等于1000万
总参数量: 12 443 648 约等于1200万

CSDN @Keep_Trying_Go

1.6 卷积优势

- 局部连接

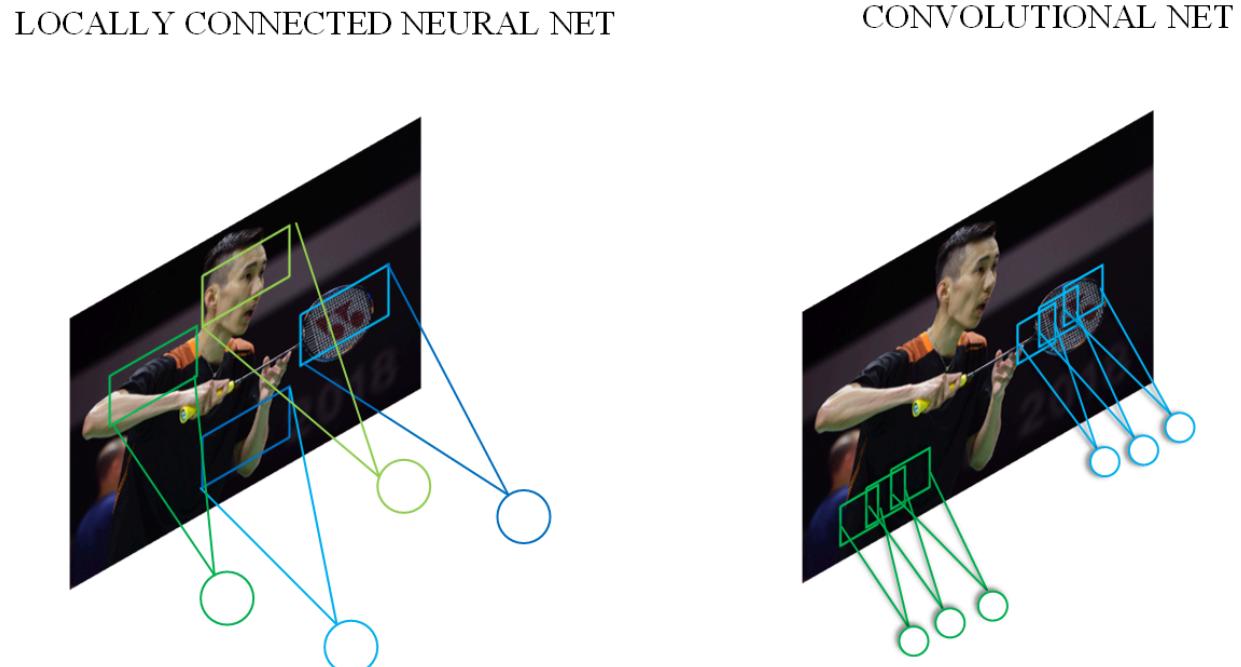
每个神经元只与局部的一块区域进行连接，训练后的滤波器能够对局部特征有强的响应，使得神经网络可以提取数据的局部特征



1000×1000的输入图像，下一个隐含层的神经元数目同样为 10^6 个，每个神经元与大小 10×10 的局部区域连接，权重参数为 $10\times 10\times 10^6 = 10^8$

- 权重共享

卷积计算 → 卷积核在图片上滑动，计算乘积求和。同一个卷积核计算过程，与图像计算过程中，权重是共享的

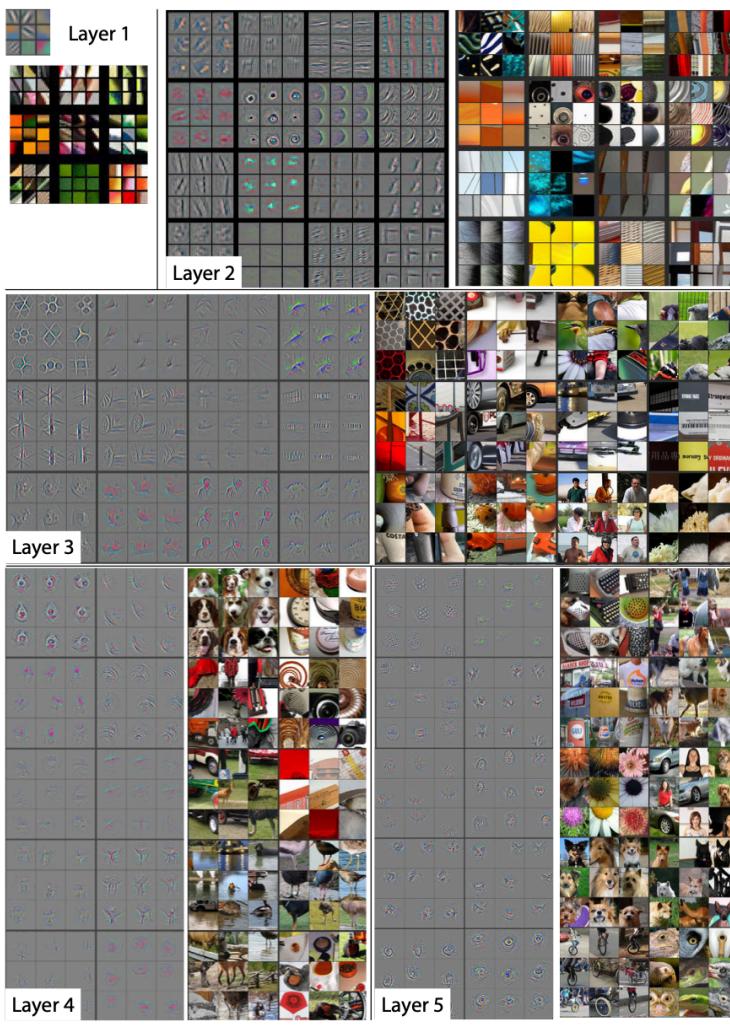


- 不同层级卷积提取不同特征

浅层卷积提取的是图像中的边缘等信息；

中层卷积提取的是图像中的局部信息；

深层卷积提取的则是图像中的全局信息。



Layer1和Layer2种，网络学到的基本上是边缘、颜色等底层特征；Layer3开始变的稍微复杂，学习到的是纹理特征；Layer4中，学习到了更高维的特征，比如：狗头、鸡脚等；Layer5则学习到了更加具有辨识性的全局特征。

卷积 (Convolution)

(https://paddlepedia.readthedocs.io/en/latest/tutorials/CNN/convolution_operator/Convolution.html).

卷积神经网络 (CNN) 详细介绍及其原理详解

(<https://blog.csdn.net/IronmanJay/article/details/128689946>).

深度学习基础学习-1x1卷积核的作用 (CNN中)

(https://blog.csdn.net/m0_47146037/article/details/127769028?ops_request_misc=%257B%2522request%255Fid%2522%253A%2522f2b9d857c35a2dd6e0e7d0f9a5a5d45c%2522%2522C%2522scm%2522%253A%252220140713.130102334..%2522%257D&request_id=f2b9d857c35a2dd6e0e7d0f9a5a5d45c&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~top_click~default-1-127769028-null-

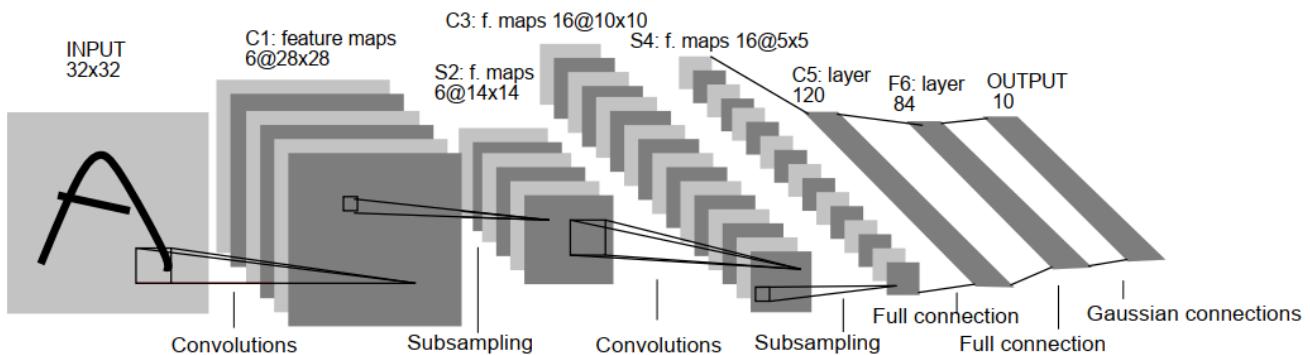
[null.142%5Ev102%5Epc_search_result_base1&utm_term=1*1%E5%8D%B7%E7%A7%AF%E6%A0%B8&spm=1018.2226.3001.4187](https://blog.csdn.net/m0_47146037/article/details/127769028?ops_request_misc=%257B%2522request%255Fid%2522%253A%2522f2b9d857c35a2dd6e0e7d0f9a5a5d45c%2522%2522C%2522scm%2522%253A%252220140713.130102334..%2522%257D&request_id=f2b9d857c35a2dd6e0e7d0f9a5a5d45c&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~top_click~default-1-127769028-null-)).

【深度学习】一文搞懂卷积神经网络 (CNN) 的原理 (超详细)

(https://blog.csdn.net/AI_dataloads/article/details/133250229?ops_request_misc=%257B%2522request%255Fid%2522%253A%2522f81ddc78c4590d0ee2faa3d936bd34e%2522%2522C%2522scm%2522%253A%252220140713.13010

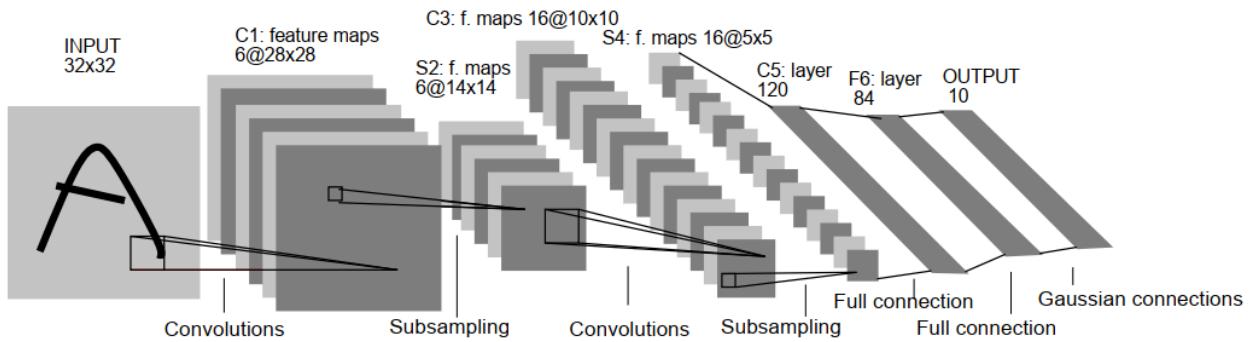
[2334..%2522%257D&request_id=8f81ddc78c4590d0ee2faa3d936bd34e&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~top_positive~default-5-133250229-null-null.142%5Ev102%5Epc_search_result_base1&utm_term=cnn&spm=1018.2226.3001.4187\).](#)

2 LeNet-5



- LeNet共分为7层
 - C1 - 卷积层
 - 输入: 32×32 图像
 - 卷积核种类 (决定输出特征图的数量) : 6
 - 卷积核大小: 5×5 (限制于当时计算机资源水平, 论文里使用的 5×5)
 - 特征图大小: 28×28 - 没有进行填充 (padding=0)
 - 可训练参数: $(5 \times 5 + 1) \times 6$
卷积核是一个 5×5 的矩阵, 里面的每一个数都是要通过训练得到的。在实际运算中还要加上一个偏置bias, 所以每一个卷积核需要训练 $5 \times 5 + 1$ 个参数, 6个卷积核就需要训练 $(5 \times 5 + 1) \times 6 = 156$ 个参数。

- S2 - 池化层



- 输入：6张 28×28 的特征图

- 采样区域： 2×2

LeNet-5 的池化层 S2 采用的是平均池化 (average pooling)

取该窗口内的 4 个像素值，计算均值后再乘以一个可训练的权重系数，再加上一个可训练的偏置，最后通过 Sigmoid 激活函数

- 输出特征图的大小： 14×14

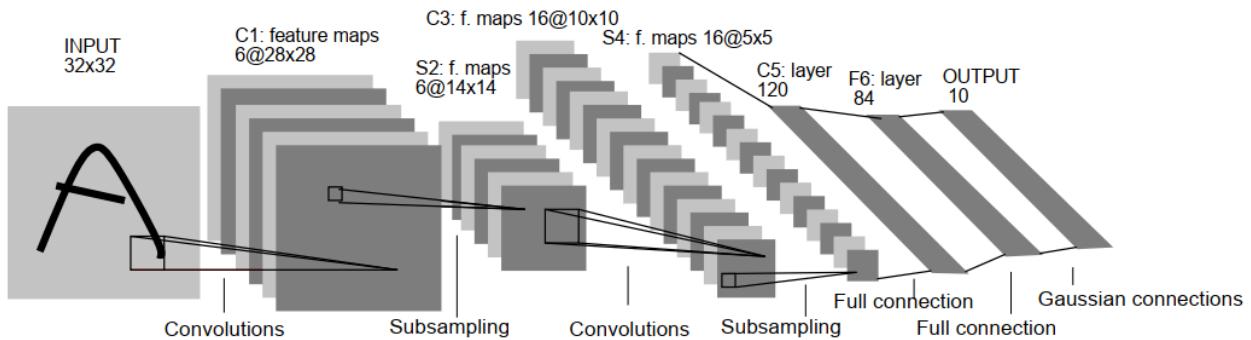
- 输出特征图数量：6

- **可训练参数： 2×6**

对于每张特征图来说，只有两个参数需要确定：用于相乘的“可训练参数 w ”与“可训练偏置 b ”，一共6张特征图，因此需要训练 2×6 个参数

- 池化基本思想其实就是使特征图尺度减小，降维，同时保留主要特征

• C3 - 卷积层



- 输入：14×14的特征图（6张）
- 卷积核种类：16
- 卷积核大小：5×5
- 输入特征图数量：6
- 输出特征图数量：16
- C3层卷积方式：

在C1卷积层中，卷积核有6种，输入图像只有一张，因此只需要将6个卷积核分别对一张图像进行卷积操作，最后得到6张特征图。

对于输入6张图像的处理方法是：“每个卷积核对多张特征图”进行处理” - 部分连接模式。

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	X		X	X	X			X	X	X	X	X	X	X	X	
1	X	X			X	X	X		X	X	X	X	X	X		
2	X	X	X			X	X	X		X		X	X	X	X	
3		X	X	X		X	X	X	X		X		X	X	X	
4		X	X	X		X	X	X	X		X	X		X	X	
5		X	X	X		X	X	X	X		X	X		X	X	

横轴为编号0~15的16个卷积核，纵轴为0~5张输入的特征图，X表示为连接。

- 输出特征图大小：10×10(padding=0)
- 输出特征图的边长为： $14 - 5 + 1 = 10$
- 可训练参数：1516

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	X		X	X	X			X	X	X	X	X	X	X	X	
1	X	X			X	X	X		X	X	X	X	X	X		
2	X	X	X			X	X	X		X		X	X	X	X	
3		X	X	X		X	X	X	X		X		X	X	X	
4		X	X	X		X	X	X	X		X	X		X	X	
5		X	X	X		X	X	X	X		X	X		X	X	

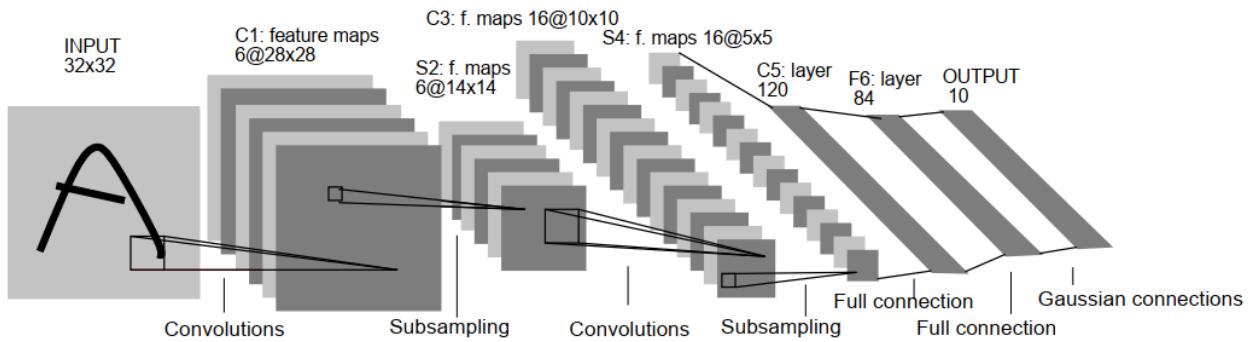
第一个红框，每个卷积核包含 5×5 个可训练参数；在标号为0~5的卷积核需要分别与3张特征图相连，最后还要加上一个偏置，因此需要训练 $3 \times 5 \times 5 + 1$ 个参数。第一个红框内有6个这样的卷积核，因此共需要训练 $6 \times (3 \times 5 \times 5 + 1)$ 个参数。

同理，第二个红框，共需要训练 $6 \times (4 \times 5 \times 5 + 1)$ 个参数；对于第三个红框，其共需要训练 $3 \times (4 \times 5 \times 5 + 1)$ 个参数；对于第四个红框，其共需要训练 $1 \times (6 \times 5 \times 5 + 1)$ 个参数。

总计可训练 $6 \times (3 \times 5 \times 5 + 1) + 6 \times (4 \times 5 \times 5 + 1) + 3 \times (4 \times 5 \times 5 + 1) + 1 \times (6 \times 5 \times 5 + 1) = 1516$ 个参数。

- 若采用全连接模式总参数数量：
 - 每个卷积核需要连接6个输入通道（6个输入特征图）
每个卷积核的参数数目是： $6 \times 5 \times 5 = 150$
 - 总参数量： $16 \times 150 = 2400$
- Summary - C3采用部分连接而不采用全连接
 - 计算复杂度降低（总参数量）
 - 模仿生物视觉的局部感受野（目的：不同的卷积核学习到不同的特征，增强多样性。）
LeNet-5当时设计的时候是受到生物视觉系统的启发。在视觉皮层中，不是所有神经元会连接到所有的输入信号，而是局部感受野的形式

- S4 - 池化层



- 输入: 10×10的特征图 (16张)

- 采样区域: 2×2

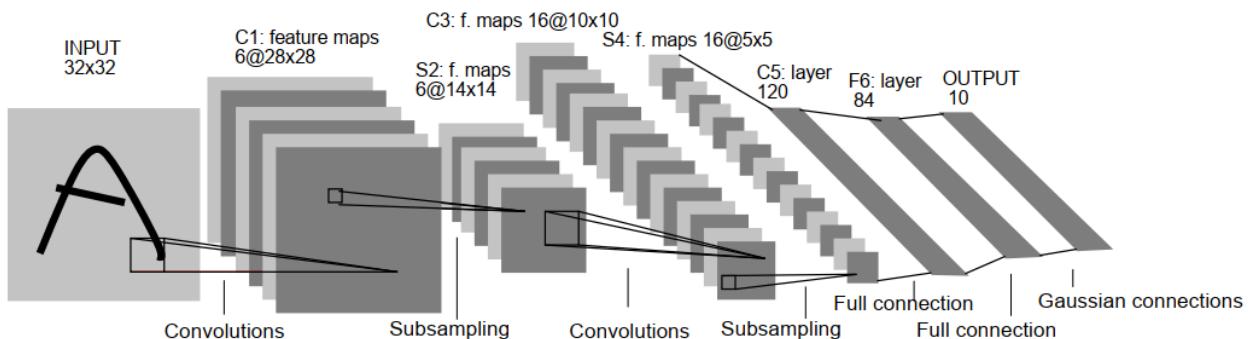
采样方式为4个输入相加, 乘以一个可训练参数, 再加上一个可训练偏置, 并将结果通过sigmoid函数。 (和S2相同)

- 输出: 5×5的特征图 (16张)

- 可训练参数: 2×16

对于每张特征图来说, 只有两个参数需要确定: 用于相乘的“可训练参数 w ”与“可训练偏置 b ”, 一共16张特征图, 因此要训练 $16 \times 2 = 32$ 个参数。

- C5 - 卷积层 (类似全连接层 - 卷积核大小和输入该层的特征图大小相等)



- 输入5×5的特征图 (16张)

- 卷积核大小: 5×5

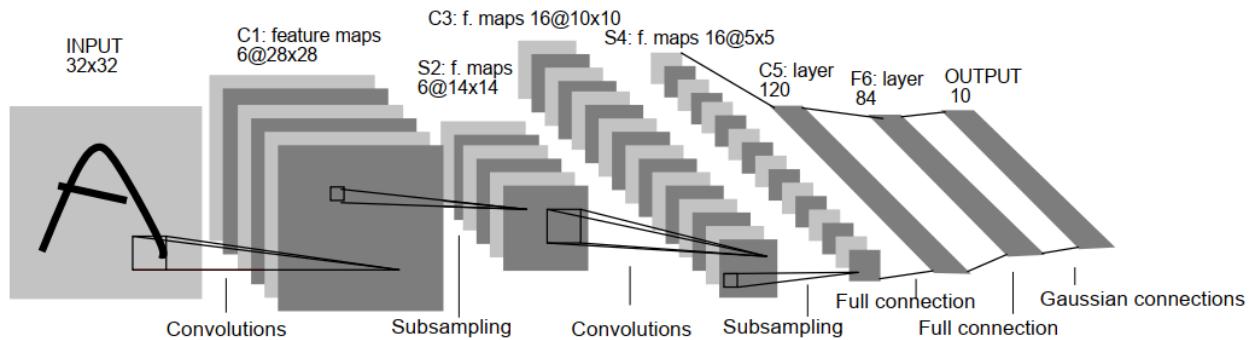
- 卷积核种类120

- 输出120维向量 (120 个 1x1 的特征图)

每个卷积核与16张特征图做卷积, 得到的结果求和, 再加上一个偏置, 结果通过sigmoid函数输出。

- 可训练参数: $(5 \times 5 \times 16 + 1) \times 120$

- F6 - 全连接层



- 输入: 120维向量
- 算法: 计算输入向量和权重向量之间的点积, 再加上一个偏置, 结果通过sigmoid函数输出。
 - 输入向量: 120×1
 - 权重向量: 84×120 (在全连接层中, 每个输出神经元都连接到所有输入神经元, 因此权重矩阵的大小取决于输入神经元数量和输出神经元数量)
- 输出: 84维向量 (矩阵乘法规则 - 两个矩阵的行数和列数相等)
- 可训练参数: $84 + 84 \times 120$
 - 权重参数: F6 层的输入是 120 维的向量, 输出是 84 维的向量。这意味着 F6 层需要学习从 120 维空间到 84 维空间的映射。因此, 对于每一维输出, 都需要与输入的 120 维特征进行连接。换句话说, 每输出一维就需要 120 个权重值。
- 总权重参数量 = $84 \times 120 = 10080$
- 偏置参数: 每个F6层的神经元都有一个独立的偏置参数, 因此
总偏置参数 = 84
- F6层有84个节点, 对应于一个 7×12 的比特图, -1表示白色, 1表示黑色, 这样每个符号的比特图的黑白色就对应于一个编码。

ASCII编码图如下:



- OUTPUT - 全连接层

- 输入: 84维向量
- 输出: 10维向量

OUTPUT层一共有10个节点, 分别对应着数字0到9。

采用径向基函数(RBF)的连接方式, 计算方式为:

$$y_i = \sum_{j=0}^{83} (x_j - w_{ij})^2$$

- y_i 计算的是输入特征向量 x 与数字 i 的比特图编码（模板）之间的距离：
- x_j : F6 层的第 j 个神经元输出值（输入特征向量）。
- w_{ij} : 值由数字 i 的比特图编码确定
- RBF 输出的值 (y_i) 越接近于 0，则越接近于 i ，即越接近于 i 的 ASCII 编码图，表示当前网络输入的识别结果是字符 i