传统是一次卷积一次池化，重复多次，可以设计（怎么搭配，怎么pool）

一、基础概念

CNN：包含卷积层的神经网络

> 至少在一层中使用卷积运算，替代一般的矩阵乘法运算

这种运算就叫做**卷积**（convolution）。卷积运算通常用星号表示：

$$s(t) = (x * w)(t).$$

- x：input
- w：核函数

> 由学习算法优化得到的张量的参数
> 二维图像需要二维的核K

- 输出：特征映射（frature map）
- 张量：多维数组

离散卷积看作矩阵乘法

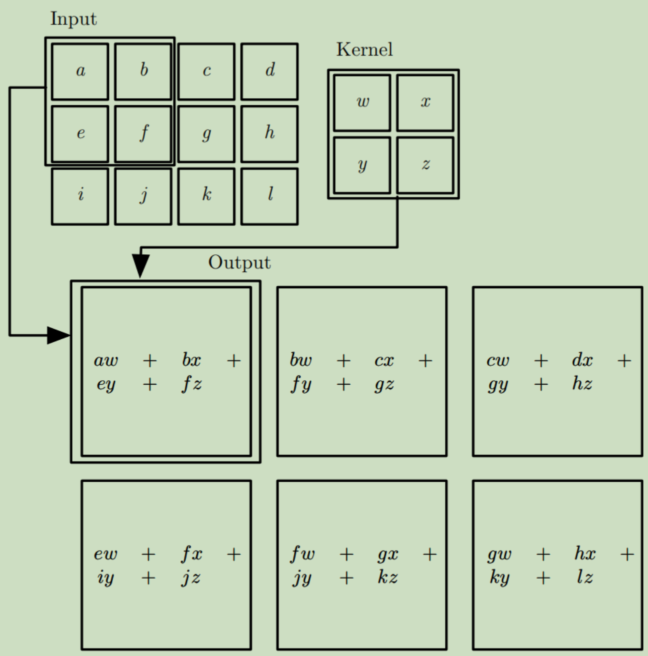


图 9.1: 一个 2 维卷积的例子（没有对核进行翻转）。我们限制只对核完全处在图像中的位置进行输出，在一些上下文中称为 “有效” 卷积。我们用画有箭头的盒子来说明输出张量的左上角元素是如何通过对输入张量相应的左上角区域应用核进行卷积得到的。

filter

> channel*卷积核=过滤器
> 一般的图像都是三通道

- kernel 是filter 的基本元素， 多张kernel 组成一个filter。卷积层数据都是三维，分解成多个二维的feature map
- filter输入通道是包含128个特征时， 即包含kernel 数是128张，
- N个filter 提取 N 个 特征
- 上一层每个feature map跟N个卷积核做卷积，下层产生N个feather map

二、技巧

1、稀疏链接

往是很显著的。如果有 m 个输入和 n 个输出，那么矩阵乘法需要 $m \times n$ 个参数并且相应算法的时间复杂度为 $O(m \times n)$（对于每一个例子）。如果我们限制每一个输出拥有的连接数为 k，那么稀疏的连接方法只需要 $k \times n$ 个参数以及 $O(k \times n)$ 的运行时间。在很多实际应用中，只需保持 k 比 m 小几个数量级，就能在机器学习的

- 核宽度为3的卷积：

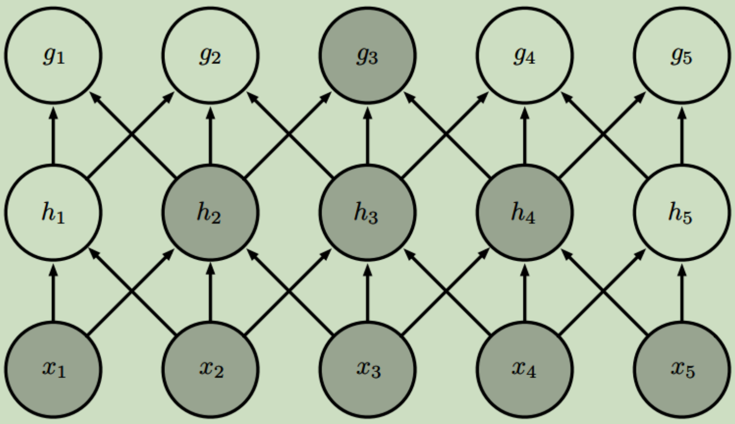


图 9.4: 处于卷积网络更深的层中的单元，它们的接受域要比处在浅层的单元的接受域更大。如果网络还包含类似步幅卷积（图 9.12）或者池化（第 9.3 节）之类的结构特征，这种效应会加强。这意味着在卷积网络中尽管直接连接都是很稀疏的，但处在更深的层中的单元可以间接地连接到全部或者大部分输入图像。

2、参数共享

显著地把模型的存储需求降低至 k 个参数

- 使神经网络对平移等变：equivariance

> 其他的一些变换并不是天然等变的，例如对于图像的放缩或者旋转变换，
> 需要其他的一些机制来处理这些变换

的性质。如果一个函数满足输入改变，输出也以同样的方式改变这一性质，我们就说它是等变 (equivariant) 的。特别地，如果函数 $f(x)$ 与 $g(x)$ 满足 $f(g(x)) = g(f(x))$，我们就说 $f(x)$ 对于变换 g 具有等变性。对于卷积来说，如果令 g 是输入的任意平移函数，那么卷积函数对于 g 具有等变性。举个例子，令 I 表示图像在整数坐标上

> 时间序列中：如果把输入中的一个事件向后延时，在输出中仍然
> 会有完全相同的表示。只是时间延后
> 处理图像时：在卷积网络的第一层进行图像的边缘检测很有用
> 相同的边缘或多或少地散落在图像各处，所以应对整个图像进行参数共享

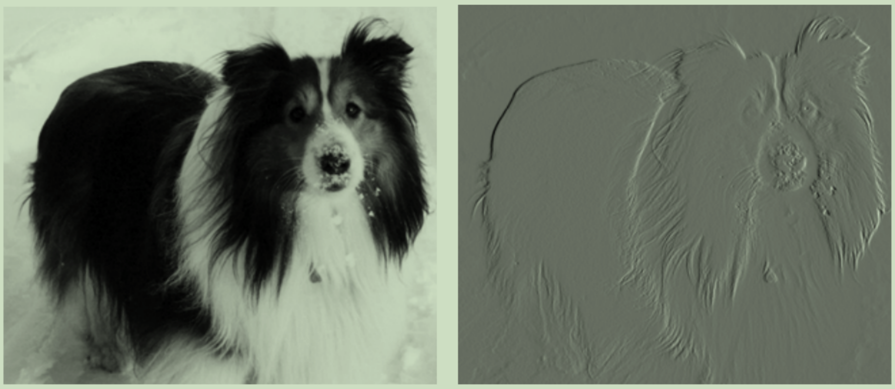


图 9.6: 边缘检测的效率。右边的图像是通过先获得原始图像中的每个像素，然后减去左边相邻像素的值而形成的。这个操作给出了输入图像中所有垂直方向上的边缘的强度，对目标检测来说是有用的。两个图像的高度均为 280 个像素。输入图像的宽度为 320 个像素，而输出图像的宽度为 319 个像素。这个变换可以通过包含两个元素的卷积核来描述，使用卷积需要 $319 \times 280 \times 3 = 267,960$ 次浮点运算（每个输出像素需要两次乘法和一次加法）。为了用矩阵乘法描述相同的变换，需要一

3、池化

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

- 最大池化给出相邻矩形区域内的最大值。
- 相邻矩形区域内的平均值
- L2范数
- 基于据中心像素距离的加权平均函数

作用:

- 当输入作出少量平移时，池化能够帮助输入的表示近似 **不变invariant**

- 使用池化可以看作是增加了一个无限强的先验：这一层学得的函数必须具有对少量平移的不变性
- 减小规模：池化综合了全部邻居的反馈，使得池化单元少于探测单元成为可能
- 统一规模：如想对不同大小的图像进行分类时，分类层的输入必须是固定的大小，通过调整池化区域的偏置大小来实现，这样分类层总能接收到相同数量的统计特征而不管最初的输入大小

> 局部平移不变性很有用，尤其是当我们关心某个特征是否出现，
> 而不关心出现位置时。例如，当判定一张图像中是否包含人脸

4、步幅stride（stride倍下采样）

我们有时会希望跳过核中的一些位置来降低计算的开销（相应的代价是提取特征没有先前那么好了）。我们可以把这一过程看作是对全卷积函数输出的下采样 (downsampling)。如果我们只想在输出的每个方向上每间隔 s 个像素进行采样，那么我们可以定义一个下采样卷积函数 c 使得

$$Z_{i,j,k} = c(\mathbf{K}, \mathbf{V}, s)_{i,j,k} = \sum_{l,m,n} [V_{l,(j-1)\times s+m,(k-1)\times s+n} K_{i,l,m,n}]. \tag{9.8}$$

我们把 s 称为下采样卷积的**步幅**（stride）。当然也可以对每个移动方向定义不同的步幅。图 9.12 演示了一个实例。

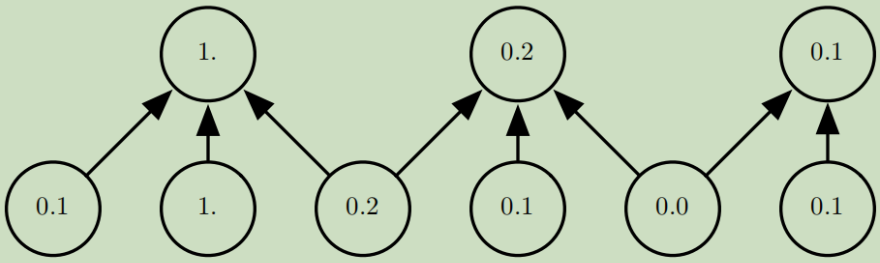


图 9.10: 带有降采样的池化。这里我们使用最大池化，池的宽度为三并且池之间的步幅为二。这使得表示的大小减少了一半，减轻了下一层的计算和统计负担。注意到最右边的池化区域尺寸较小，

5、padding

> 像素填充

图像：

- Valid卷积：padding=0

> n×n的图像，用f×f的过滤器卷积，(n-f+1)×(n-f+1)维的输出

- Same卷积：不变（前提步幅为1）

> 扩展原K*K的feature map，使得滑动后大小仍为K*K，不会减小

零填充pad

> 通常零填充的最优数量处于 有效卷积 和 相同卷积 之间

1.有效卷积：不填充

- 输出的大小在每一层都会缩减
- 如果输入的图像宽度是 m，核的宽度是k，那么输出的宽度就会变成m-k+1

2.相同卷积：保持相同

- 只进行足够的零填充来保持输出和输入具有相同的大小
- 输入像素中靠近边界的部分，相比于中间部分对于输出像素的影响更小，可能导致边界像素欠表示

3.全卷积

- 进行足够多的零填充，使每个像素在每个方向上恰好被访问了 k 次
- 最终输出图像的宽度为 m + k − 1
- 输出像素中靠近边界的部分相比于中间部分是更少像素的函数。

> 用隐含零填充时，图像边缘的探测单元接收到较少输入，因此需要较大的偏置

- 导致学得一个在卷积特征映射的所有位置都表现不错的单核更为困难

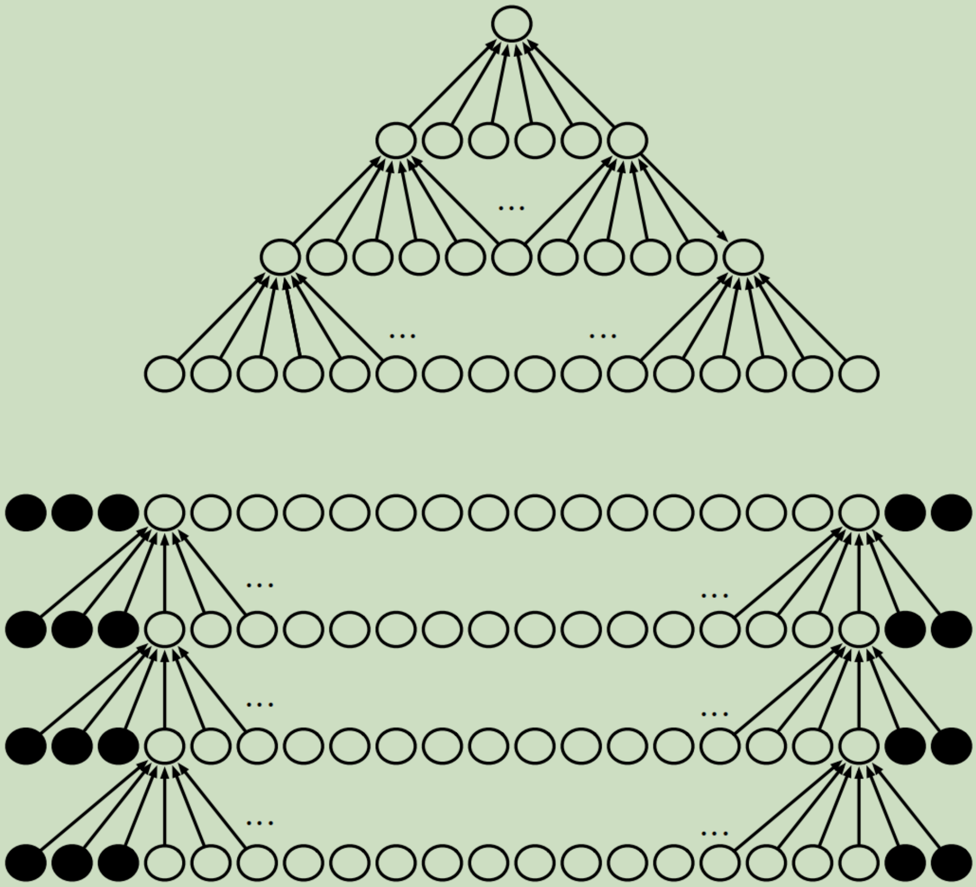


图 9.13: 零填充对网络大小的影响。考虑一个卷积网络，每层有一个宽度为六的核。在这个例子中，我们不使用任何池化，所以只有卷积操作本身缩小网络的大小。*(上)* 在这个卷积网络中，我们不使用任何隐含的零填充。这使得表示在每层缩小五个像素。从十六个像素的输入开始，我们只能有三个卷积层，并且最后一层不能移动核，所以可以说只有两层是真正的卷积层。可以通过使用较小的核来减缓收缩速率，但是较小的核表示能力不足，并且在这种结构中一些收缩是不可避免的。*(下)* 通过向每层添加五个隐含的零，我们防止了表示随深度收缩。这允许我们设计一个任意深的卷积网络。

三、变体卷积

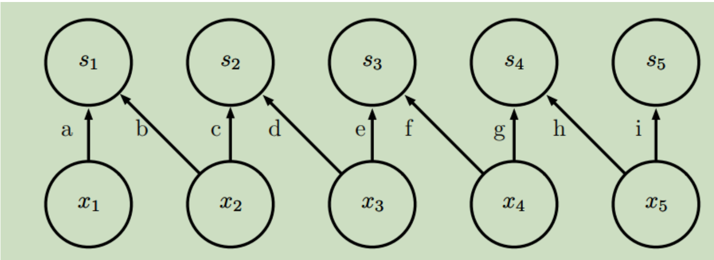
单个核的卷积只能提取一种特征

1、局部链接层LeCun

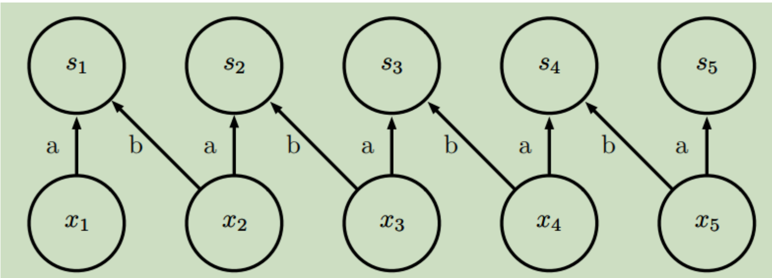
当每一个特征都是一小块空间的函数，并且相同的特征不会出现在所有空间上时

> 偏置的设置方式与权重类似

- 局部链接：每条边不同权重

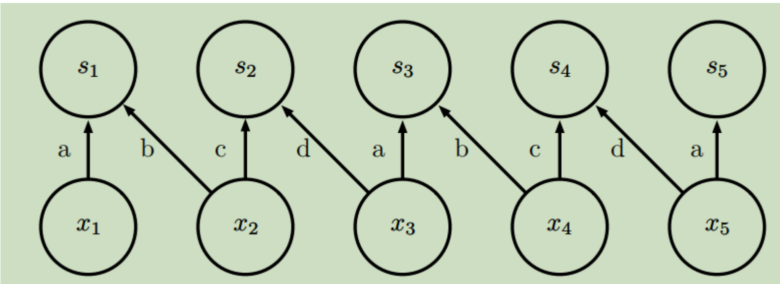


- 卷积：核宽度为2，参数共享



- 平铺卷积：上两个的折中，循环利用一组核

> 如果两个输出单元间隔 t 个步长的倍数，则它们共享参数



四、各层参数和链接数计算

计算特征图尺寸

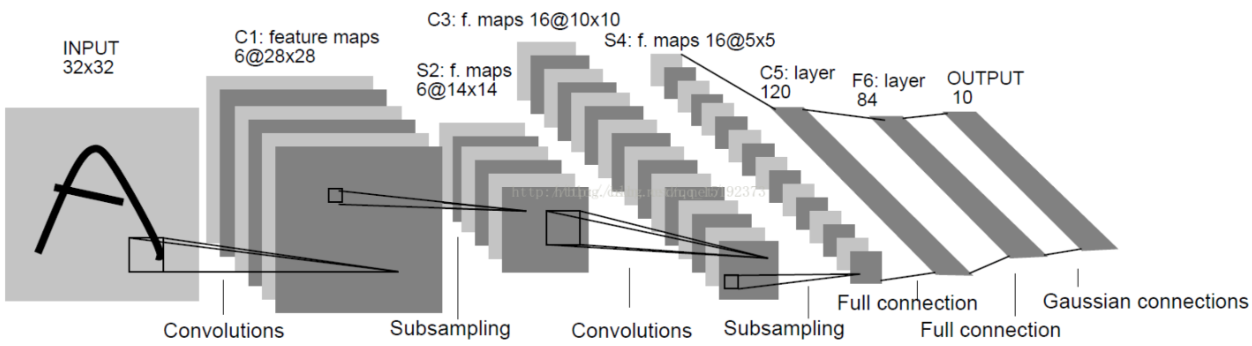
`一个尺寸 a*a 的特征图，经过 b*b 的卷积层，步幅（stride）=c，填充（padding）=d，池化层尺寸bs*bs`

$$\text{输出特征图尺寸} = \frac{a-b+2d}{c} + 1$$

池化层输出=（a-bs)/c + 1（不整除就向下取整）

例一：LeNet-5

下面以最经典的LeNet-5例子来逐层分析各层的参数及连接个数。



C1卷积层

- 有6个卷积核，特征图大小（32-5+1=28）
- 滤波器5*5*6个参数，bias1*6

S2下采样层

- 有6个卷积核，特征图大小（28/2=14）
- 是否有参数，要根据不同的采样方式

C3卷积层

- 使用S2中不同map的组合，生成了16个卷积核

> 破坏了网络的对称性。由于不同的特征图有不同的输入，所以迫使他们抽取不同的特征（希望是互补的）。

2. 参数个数：例如，存在的一个方式是：

C3的前6个特征图：相当于需要6组滤波器，每组以S2中 3个相邻 特征图子集 为输入，共享一个偏置。(C3每个特征图 由 S2中3个特征图分别用不同滤波器 再加和得到)

C3的接下来6个特征图：相当于需要6组滤波器，每组以S2中 4个相邻 特征图子集 为输入，共享一个偏置。(1对4)

C3的接下来3个特征图：相当于需要3组滤波器，每组以S2中 4个不相邻 特征图子集 为输入，共享一个偏置。(1对4)

C3的最后1个特征图：相当于需要1组滤波器，每组将S2中所有 特征图 为输入，共享一个偏置。(1对6)

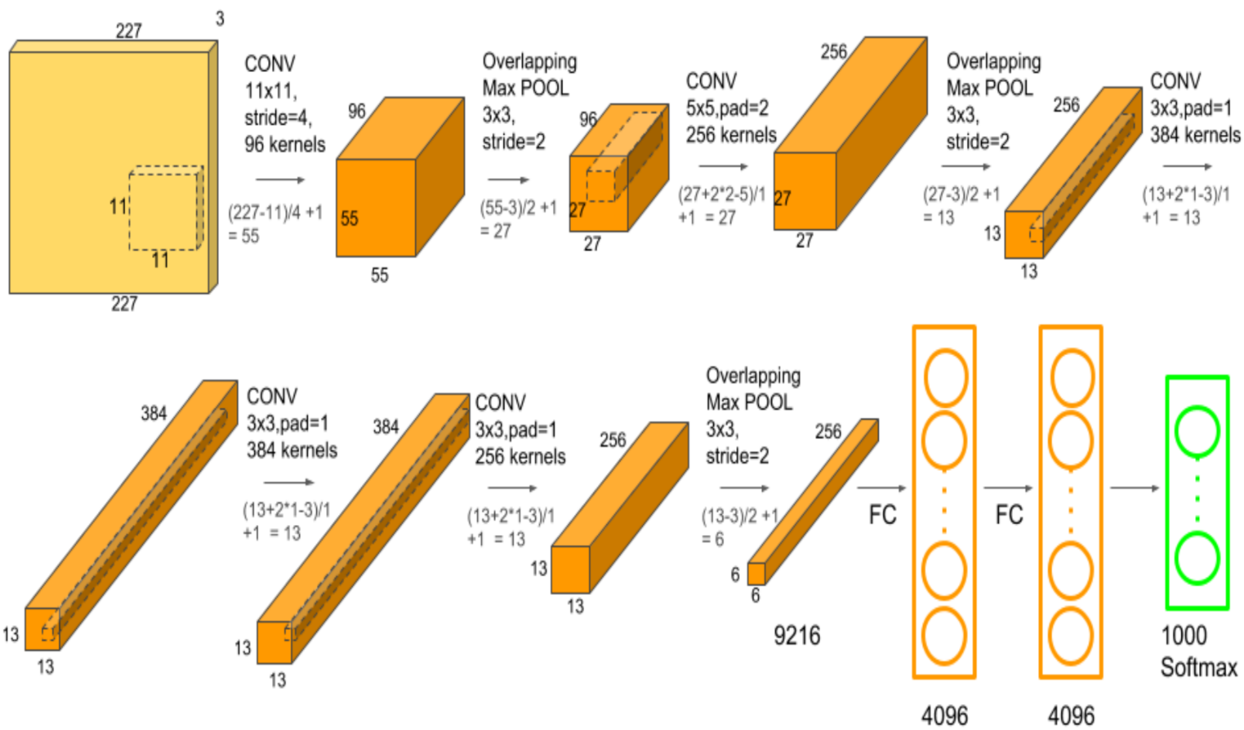
这样C3层有1516个可训练参数。计算：6*（3*25+1）+6*（4*25+1）+3*（4*25+1）+（25*6+1）=1516。此处，C3个特征图由 S2中n个卷积结果合并，然后共享1个b，组合计算得到。

C5卷积&恰好全链接层

> 全连接层分两种，这种是不是很纯血的

- 为什么120个?
- 特征图大小1*1
- 滤波器有16*120个，所以w有120*16*5*5个，同组16个滤波器共用一个b，所以有120个b

例二：AlexNet



计算参数：

卷积层：w=K^2乘上通道数C（前层核数）和当前层的核数N，b只要核数N

假全连接层：w=K^2*前层核数*后层神经元数，b=后层神经元数

真全连接层：w=前层神经元数*后层神经元数，b=后层神经元数

池化层没有参数，都是超参数

AlexNet网络中总共有5个卷积层和3个全连接层.总共有62,378,344个参数.以下是汇总表.

Layer Name	Tensor Size	Weights	Biases	Parameters
Input Image	227x227x3	0	0	0
Conv-1	55x55x96	34,848	96	34,944
MaxPool-1	27x27x96	0	0	0
Conv-2	27x27x256	614,400	256	614,656
MaxPool-2	13x13x256	0	0	0
Conv-3	13x13x384	884,736	384	885,120
Conv-4	13x13x384	1,327,104	384	1,327,488
Conv-5	13x13x256	884,736	256	884,992
MaxPool-3	6x6x256	0	0	0
FC-1	4096×1	37,748,736	4,096	37,752,832
FC-2	4096×1	16,777,216	4,096	16,781,312
FC-3	1000×1	4,096,000	1,000	4,097,000
Output	1000×1	0	0	0
Total				**62,378,344**

- 第一个卷积层，通道数就没了