

# 卷积神经网络

1. 卷积神经网络 convolutional neural network:CNN：是指那些至少在网络的某一层中使用了卷积运算来代替一般的矩阵乘法运算的神经网络。
2. 卷积神经网络专门处理具有类似网格结构的数据的神经网络。如：时间序列是一维网格，图像数据是二维网格。

## 一、卷积运算

### 1.1 数学卷积

#### 1.1.1 卷积定义

1. 示例：一个激光传感器输出  $x(t)$ ，表示宇宙飞船在时刻  $t$  的位置的观测结果。假设传感器包含噪声，则  $x(t)$  与飞船在时刻  $t$  的真实位置有偏离。

可以利用观测结果的均值来估计飞船的位置。假设越近的观测结果越相关，于是对最近的观测结果赋予更高的权重。

令  $w(a)$  为权重函数，其中  $a$  表示观测结果距离当前时刻的间隔，则得到时刻  $t$  飞船真实位置的估计：

$$s(t) = \int x(a)w(t-a)da$$

这种运算就称作卷积 convolution，用符号星号  $s(t) = (x * w)(t)$  表示。

理论上  $w(\cdot)$  可以为任意的实值函数，但是在这个示例中要求：

- $w(\cdot)$  是个有效的概率密度函数，否则  $s(t)$  就不是一个加权平均。
  - $w(\cdot)$  在自变量为负数时，取值为零。否则涉及到未来函数，因为激光传感器只能输出  $t$  时刻之前的观测结果。
2. 通常当计算机处理数据时，连续的数据会被离散化，因此时间  $t$  只能取离散值。

假设  $x(\cdot), w(\cdot)$  都是定义在整数时刻  $t$  上，则得到离散形式的卷积：

$$s(t) = (x * w)(t) = \sum_{a=-\infty}^{a=\infty} x(a)w(t-a)$$

实际操作中，因为只能存储有限的数据，所以这些函数的值在有限的点之外均为零。因此无限级数的求和最终是有限级数的求和。

3. 在卷积神经网络中，函数  $x(\cdot)$  称作输入，函数  $w(\cdot)$  称作核函数，输出有时被称作特征图 feature map。
4. 可以对多个维度进行卷积运算。

如果二维图像  $\mathbf{I}$  作为输入，则需要使用二维核函数  $\mathbf{K}$ ，卷积运算的输出为：

$$\mathbf{S}(i, j) = (\mathbf{I} * \mathbf{K})(i, j) = \sum_m \sum_n \mathbf{I}(m, n) \mathbf{K}(i-m, j-n)$$

其中  $(m, n)$  表示二维图像  $\mathbf{I}$  的像素点的坐标， $\mathbf{I}(m, n)$  表示该坐标处的像素值。

- 通常  $\mathbf{I}$  的尺寸较大，如  $\mathbf{I} \in \mathbb{R}^{100 \times 200}$ ；而  $\mathbf{K}$  的尺寸较小，如  $\mathbf{K} \in \mathbb{R}^{5 \times 5}$
- 因为卷积是可交换的，所以可以等价写作：

$$\mathbf{S}(i, j) = (\mathbf{I} * \mathbf{K})(i, j) = \sum_m \sum_n \mathbf{I}(i - m, j - n) \mathbf{K}(m, n)$$

这称作翻转 `flip` 了核。

卷积的可交换性在数学证明中有用，但是在神经网络中很少使用。

### 1.1.2 数学卷积与矩阵乘法

1. 离散卷积可以视作输入矩阵与一个特殊的核矩阵的乘法。

- 对于一维的离散卷积，核矩阵的每一行必须和上一行移动一个元素后相等。

这种类型的矩阵叫做 `Toeplitz` 矩阵。

- 对于二维的离散卷积，核矩阵对应着一个双重块循环矩阵。

该矩阵大部分元素相等，且非常稀疏（几乎所有元素都为零）。

2. 卷积运算可以转换成矩阵乘法，所以不需要对神经网络库的实现作出大的修改。

#### 1.1.2.1 一维卷积和矩阵乘法

1. 循环矩阵的定义：

$$\mathbf{C} = \begin{bmatrix} c_0 & c_{n-1} & \cdots & c_2 & c_1 \\ c_1 & c_0 & \cdots & c_3 & c_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ c_{n-1} & c_{n-2} & \cdots & c_1 & c_0 \end{bmatrix}$$

可以利用循环矩阵求一维卷积。

2. 假设有两个长度分别为  $M$  和  $N$  的序列  $x(i)$  和  $w(i)$ ，则一维卷积为：

$$s(i) = x(i) * w(i) = \sum_j x(j)w(i-j)$$

卷积的长度为  $L = M + N - 1$ 。

- 首先用 0 扩充序列  $x, w$ :

$$x_p(i) = \begin{cases} x(i) & , 0 \leq i \leq M-1 \\ 0 & , M-1 < i \leq L-1 \end{cases}$$

$$w_p(i) = \begin{cases} w(i) & , 0 \leq i \leq N-1 \\ 0 & , N-1 < i \leq L-1 \end{cases}$$

- 由于用  $w$  取卷积  $x$ ，因此构造  $w$  的循环矩阵：

$$\mathbf{W} = \begin{bmatrix} w_p(0) & w_p(L-1) & w_p(L-2) & \cdots & w_p(1) \\ w_p(1) & w_p(0) & w_p(L-1) & \cdots & w_p(2) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ w_p(L-1) & w_p(L-2) & w_p(L-3) & \cdots & w_p(0) \end{bmatrix}$$

这里列优先，因此第一列是完全顺序的。

- 一维卷积为：

$$\vec{s} = \mathbf{W} \cdot x_p = \begin{bmatrix} w_p(0) & w_p(L-1) & w_p(L-2) & \cdots & w_p(1) \\ w_p(1) & w_p(0) & w_p(L-1) & \cdots & w_p(2) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ w_p(L-1) & w_p(L-2) & w_p(L-3) & \cdots & w_p(0) \end{bmatrix} \cdot \begin{bmatrix} x_p(0) \\ x_p(1) \\ \vdots \\ x_p(L-1) \end{bmatrix}$$

其中  $\vec{s} = (s(0), s(1), \dots, s(L-1))^T$ 。

### 1.1.2.2 二维卷积和矩阵乘法

1. 二维卷积：

$$\mathbf{S}(i, j) = (\mathbf{I} * \mathbf{K})(i, j) = \sum_m \sum_n \mathbf{I}(m, n) \mathbf{K}(i - m, j - n)$$

假设  $\mathbf{I} \in \mathbb{R}^{M_I, N_I}$ ,  $\mathbf{K} \in \mathbb{R}^{M_K, N_K}$  :

$$\mathbf{I} = \begin{bmatrix} I_{1,1} & I_{1,2} & \cdots & I_{1,N_I} \\ I_{2,1} & I_{2,2} & \cdots & I_{2,N_I} \\ \vdots & \vdots & \ddots & \vdots \\ I_{M_I,1} & I_{M_I,2} & \cdots & I_{M_I,N_I} \end{bmatrix}, \quad \mathbf{K} = \begin{bmatrix} K_{1,1} & K_{1,2} & \cdots & K_{1,N_K} \\ K_{2,1} & K_{2,2} & \cdots & K_{2,N_K} \\ \vdots & \vdots & \ddots & \vdots \\ K_{M_K,1} & K_{M_K,2} & \cdots & K_{M_K,N_K} \end{bmatrix}$$

- 先将  $\mathbf{I}, \mathbf{K}$  扩充到  $M \times N$  维 :  $M = M_I + M_K - 1$ ,  $N = N_I + N_K - 1$ 。扩充之后的新矩阵为

$\mathbf{I}_p, \mathbf{K}_p$ 。其中：

$$\mathbf{I}_p = \begin{bmatrix} I_{1,1} & I_{1,2} & \cdots & I_{1,N_I} & 0 & \cdots & 0 \\ I_{2,1} & I_{2,2} & \cdots & I_{2,N_I} & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & 0 & \cdots & 0 \\ I_{M_I,1} & I_{M_I,2} & \cdots & I_{M_I,N_I} & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & 0 & \cdots & 0 \end{bmatrix}$$

$$\mathbf{K}_p = \begin{bmatrix} K_{1,1} & K_{1,2} & \cdots & K_{1,N_K} & 0 & \cdots & 0 \\ K_{2,1} & K_{2,2} & \cdots & K_{2,N_K} & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & 0 & \cdots & 0 \\ K_{M_K,1} & K_{M_K,2} & \cdots & K_{M_K,N_K} & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & 0 & \cdots & 0 \end{bmatrix}$$

- 用  $\mathbf{I}_p$  构造一个列向量  $f_p$ ：将  $\mathbf{I}_p$  的第一行转置之后将其成为  $f_p$  的前  $N$  个元素；接下来是第二行的转置....第  $M$  行的转置。

$$f_p = (I_{1,1}, I_{1,2}, \dots, I_{1,N_I}, 0, \dots, I_{M_I,1}, I_{M_I,2}, \dots, I_{M_I,N_I}, 0, \dots)^T$$

- 将  $\mathbf{K}_p$  中的每一行，都按照一维卷积中介绍的循环矩阵生成的方法构成一个  $N \times N$  的循环矩阵。这些矩阵记做： $\mathbf{G}_1, \mathbf{G}_2, \dots, \mathbf{G}_M$ 。

$$\mathbf{G}_m = \begin{bmatrix} K_{m,1} & 0 & \cdots & K_{m,2} \\ K_{m,2} & K_{m,1} & \cdots & K_{m,3} \\ \vdots & \vdots & \ddots & \vdots \\ K_{m,N_K} & K_{m,N_K-1} & \cdots & 0 \\ 0 & K_{m,N_K} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & K_{m,1} \end{bmatrix}, \quad m = 1, 2, \dots, M$$

- 用这些循环矩阵构造一个大的块循环矩阵：

$$\mathbf{G}_b = \begin{bmatrix} [\mathbf{G}_1] & [\mathbf{G}_M] & \cdots & [\mathbf{G}_2] \\ [\mathbf{G}_2] & [\mathbf{G}_1] & \cdots & [\mathbf{G}_3] \\ \vdots & \vdots & \ddots & \vdots \\ [\mathbf{G}_M] & [\mathbf{G}_{M-1}] & \cdots & [\mathbf{G}_1] \end{bmatrix}$$

- 计算： $h_b = \mathbf{G}_b \cdot f_p$ 。将  $h_b$  的结果分配到  $\mathbf{S}$  的各行（与构造  $f_p$  相反的过程），即得到二维卷积。

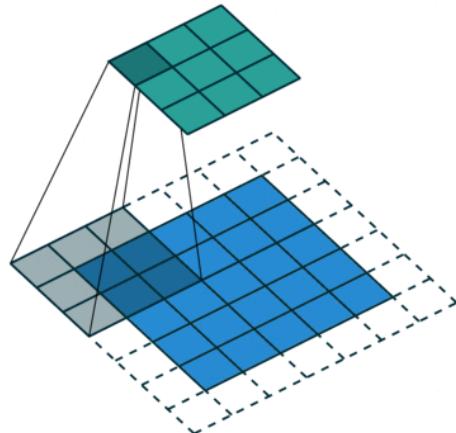
## 1.2 神经网络卷积

### 1.2.1 卷积定义

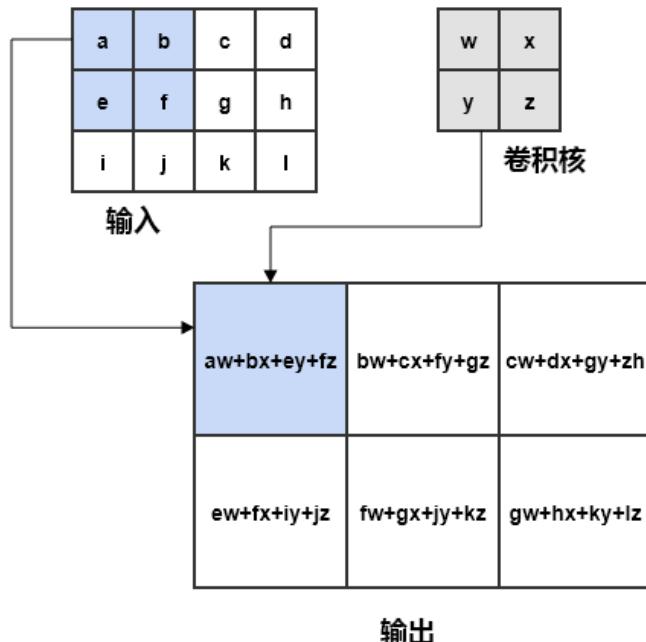
- 许多神经网络库会实现一个与卷积有关的函数，称作互相关函数 `cross-correlation`。它类似于卷积：

$$\mathbf{S}(i, j) = (\mathbf{I} * \mathbf{K})(i, j) = \sum_m \sum_n \mathbf{I}(i + m, j + n) \mathbf{K}(m, n)$$

有些机器学习库将它称作卷积。事实上在神经网络中，卷积指的就是这个函数（而不是数学意义上的卷积函数）。



- 神经网络的2维卷积的示例：



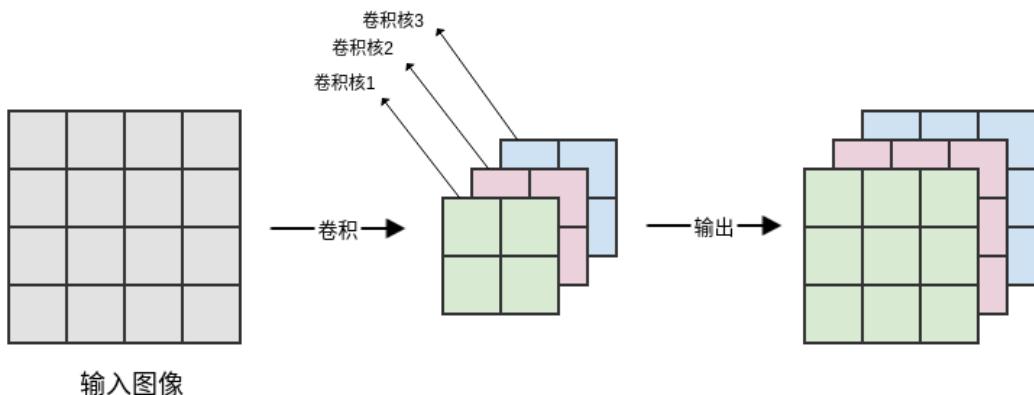
这里采用的是神经网络中卷积的定义： $\mathbf{S}(i, j) = (\mathbf{I} * \mathbf{K})(i, j) = \sum_{m=0}^1 \sum_{n=0}^1 \mathbf{I}(i+m, j+n) \mathbf{K}(m, n)$ 。其中， $m$  和  $n$  由核函数决定。因为  $\mathbf{K} \in \mathbb{R}^{2 \times 2}$ ，所以他们的取值范围是  $[0, 2)$ ， $[0, 2)$ 。

### 3. 单个卷积核只能提取一种类型的特征。

如果希望卷积层能够提取多个特征，则可以并行使用多个卷积核，每个卷积核提取一种特征。我们称输出的 `feature map` 具有多个通道 `channel`。

`feature map` 特征图是卷积层的输出的别名，它由多个通道组成，每个通道代表通过卷积提取的某种特征。

事实上，当输入为图片或者 `feature map` 时，池化层、非线性激活层、`Batch Normalization` 等层的输出也可以称作 `feature map`。卷积神经网络中，非全连接层、输出层以外的几乎所有层的输出都可以称作 `feature map`。



### 4. 神经网络中，卷积运算的作用就类似于滤波，因此也称卷积核为 `filter` 滤波器。

- 滤波器可以从原始的像素特征中抽取某些特征，如：边缘、角度、形状等。

如：`sobel` 算子：

$$\mathbf{K}_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} \quad \mathbf{K}_y = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

其中  $\mathbf{K}_x$  表示检测垂直边缘的滤波器，它沿着水平方向做卷积； $\mathbf{K}_y$  表示检测水平边缘的滤波器，它沿着垂直的方向做卷积。

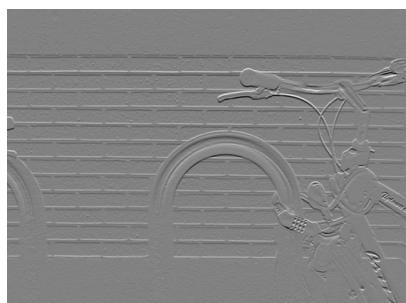
下图所示为一张原始的灰度图：



经过  $\mathbf{K}_x$  卷积之后：



经过  $\mathbf{K}_y$  卷积之后：



- 实际上，在卷积神经网络中我们并不会手工设计卷积核，而是通过学习算法自动学得卷积核中每个位置的值。

## 1.2.2 输入填充

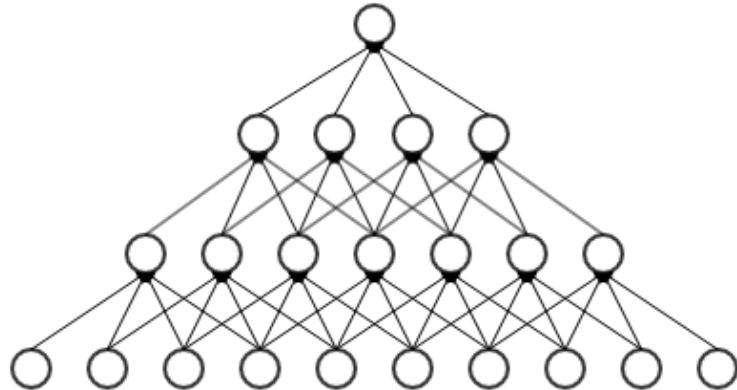
- 在卷积神经网络中，可以隐式地对输入填充零，使其得到加宽。

如果未填充零，则网络每一层的宽度会逐层递减。根据卷积的性质，网络每一层宽度减少的数量等于卷积核的宽度减1。

- 如果卷积核尺寸较大，则网络的宽度迅速缩减，这限制了卷积神经网络的网络深度。
  - 如果卷积核尺寸较小，则可用的卷积核的数量大幅度降低，这限制了卷积神经网络的表达能力。
- 对输入  $\mathbf{V}$  有三种填充零的方式：`valid` 填充、`same` 填充、`full` 填充。
  - `valid` 填充：不使用零来填充输入，卷积核只允许访问那些图像中能完全包含整个核的位置。

在 `valid` 填充模式中，输出的大小在每一层都缩减。假设核的宽度是  $k$ ，则每经过一层，输出的宽度减少了  $k - 1$ 。

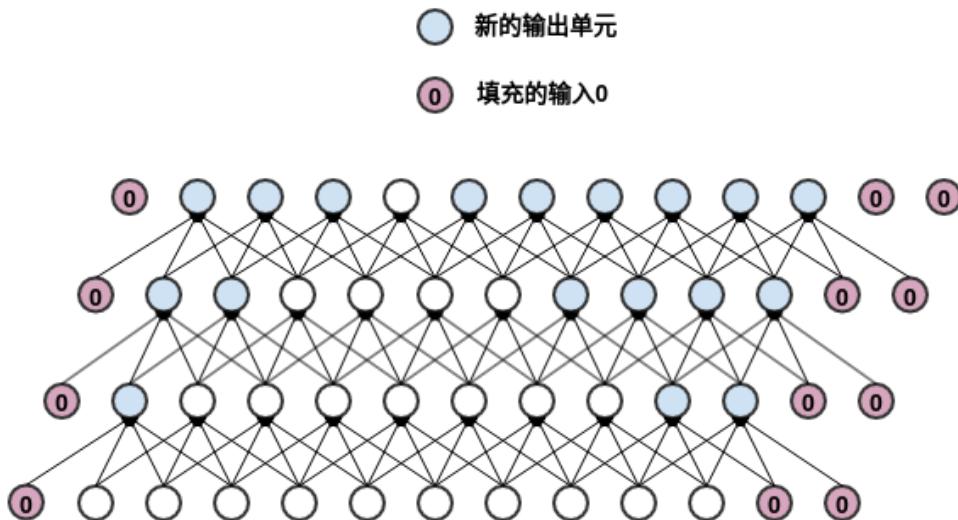
如果输入图像的宽度是  $m$ ，则网络经过了  $d$  层之后，输出的宽度变成  $m - (k - 1) \times d$ 。如果核的宽度  $k$  非常大时，缩减非常明显。最终网络会缩减到 1。



4. `same` 填充：使用足够的零来填充，使得输出和输入保持相同的大小。这是最常见的填充方式。

- 在 `same` 填充模式中，网络可以包含任意多的卷积层，因为它不存在网络输出宽度缩减的问题。
- `same` 填充模式的一个问题是：输入的边缘单元可能存在一定程度上的欠表达。

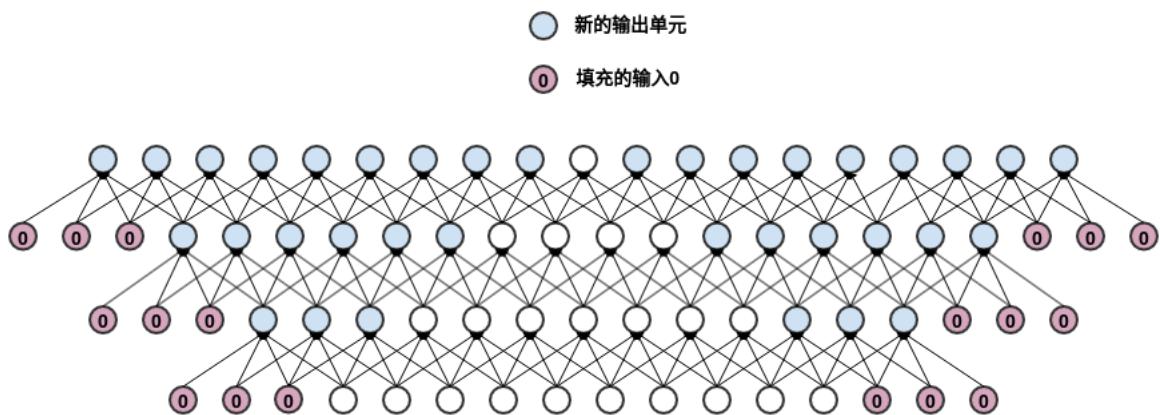
因为输入的中间区域的单元的影响域为全部的输出单元，这意味着这些输入单元的信息会被很多输出单元所编码。而输入的边缘区域的单元的影响域只是输出单元的一部分，这意味着这些输入单元的信息仅被少量输出单元所编码。



5. `full` 填充：在输入的两端各填充  $k - 1$  个零，使得每个输入单元都恰好被卷积核访问  $k$  次。其中  $k$  为卷积核的宽度。

- 它将从卷积核和输入开始相交的时候开始做卷积。
  - 假设核的宽度是  $k$ ，则每经过一层，输出的宽度增加了  $k - 1$ 。
- 如果输入图像的宽度是  $m$ ，则网络经过了  $d$  层之后，输出的宽度变成  $m + (k - 1) \times d$ 。
- 它使得输入的边缘单元也能够得到充分表达。
  - `full` 填充的一个问题是：输出的边界单元依赖于更少的输入单元。

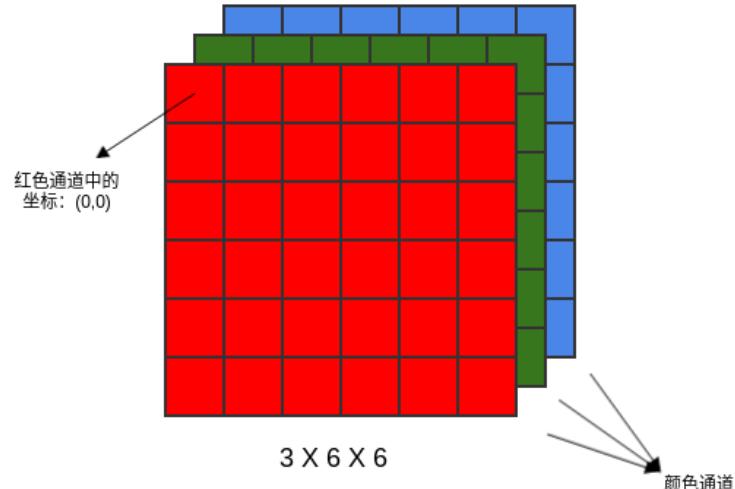
这使得学习到的结果在输出的中间部分表现较好，边缘部分的表现较差。



### 1.2.3 三维卷积

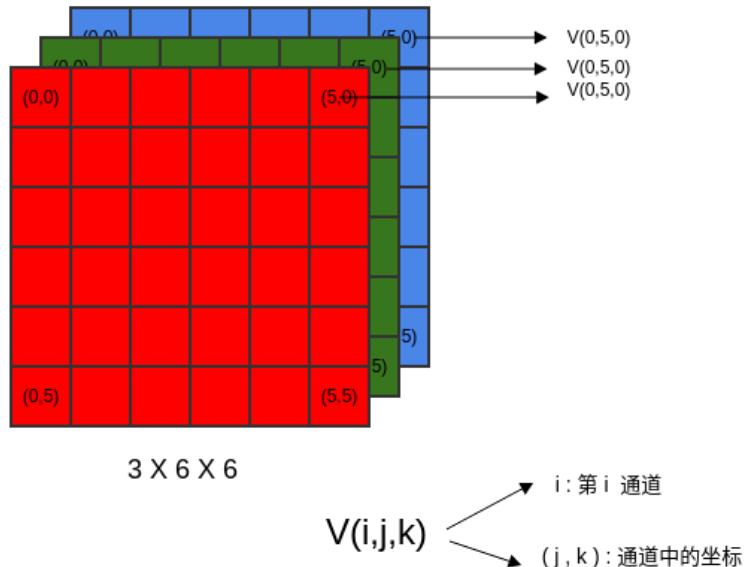
1. 卷积神经网络的输入图片可以是二维（黑白图片），也可以是三维的（彩色图片）。

对于三维彩色图片，一个维度来表示不同的颜色通道（如红绿蓝），另外两个维度表示在每个通道上的空间坐标。



2. 对于三维卷积：

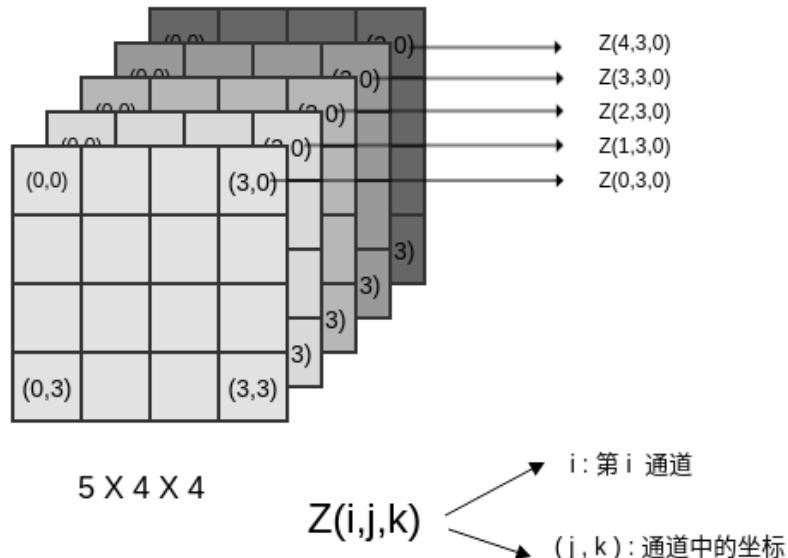
- 假设输入为张量  $V$ ，每个元素是  $V_{i,j,k}$ 。其中： $i$  表示输入单元位于  $i$  通道， $j, k$  表示通道中的坐标。



- 假设输出为张量  $Z$ ，每个元素为  $Z_{i,j,k}$ 。其中： $i$  表示输出单元位于  $i$  通道， $j, k$  表示通道中的坐标。

注意：输出的通道数量通常与输入的通道数量不等。

输出有多个通道的原因是：使用了多个卷积核，每个卷积核会输出一个通道。



- 假设核张量为4维的张量  $K$ ，每个元素是  $K_{i,l,j,k}$ 。其中： $i$  表示输出单元位于  $i$  通道， $l$  表示输入单元位于  $l$  通道， $j, k$  表示通道中的坐标。

则三维卷积可以表示为：

$$Z_{i,j,k} = \sum_l \sum_m \sum_n V_{l,j+m,k+n} K_{i,l,m,n}$$

其中：

- $\sum_m \sum_n$  遍历了图像平面上的所有坐标。
- $\sum_l$  遍历了输入的所有通道。

■  $K_{i,:,:}$  是单个三维卷积的核，多个并行的核的卷积结果组成了输出的多个通道。

3. 上述表述中，张量  $\mathbf{V}$ 、 $\mathbf{Z}$  的通道索引位于坐标索引之前，这称作通道优先 channel-first。

还有另一种模式：通道索引位于坐标索引之后，这称作 channel-last。

如：`tensorflow` 框架采用 channel-last 的模式，`theano` 框架采用 channel-first 的模式。这里默认采用 channel-last 的方式来描述。

## 1.2.4 降采样

1. 如果对卷积层的输出进行降采样，则表示跳过图片中的一些位置。

- 优点：可以降低计算开销。因为它降低了卷积层的输出单元数量，也就降低了高层网络的输入单元数量。
- 缺点：提取的特征可能没有那么好，因为跳过的位置可能包含一些关键信息。

2. 假设希望对输出的每个方向上，每隔  $s$  个像素进行采样，则：

$$Z_{i,j,k} = \sum_l \sum_m \sum_n V_{l,j \times s + m, k \times s + n} K_{i,l,m,n}$$

这里  $s$  称作降采样卷积的步幅。

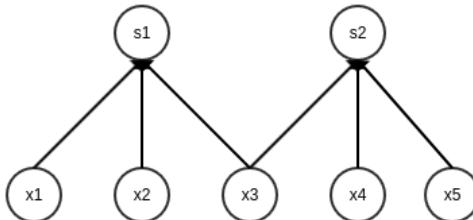
3. 可以对不同的方向定义不同的步幅。

假设  $j$  方向的步幅为  $s_1$ ， $k$  方向的步幅为  $s_2$ ，则有：

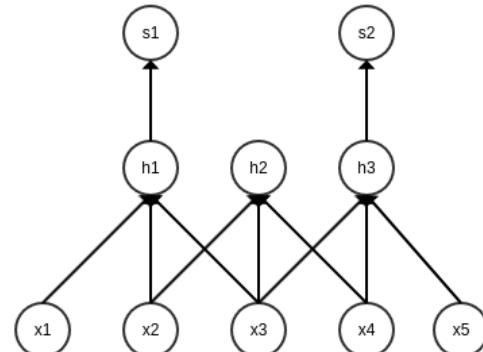
$$Z_{i,j,k} = \sum_l \sum_m \sum_n V_{l,j \times s_1 + m, k \times s_2 + n} K_{i,l,m,n}$$

4. 降采样卷积有两种实现形式：

- 通过直接实现步幅为  $s$  的卷积。
- 先进行完整的卷积，再降采样。这种做法会造成计算上的大量浪费，不建议采用。



降采样：直接实现



降采样：先卷积，再降采样

## 1.2.5 梯度计算

1. 实现卷积神经网络时，为了能够学习模型，必须能够计算核的梯度。

在某些简单情况下，核的梯度可以通过卷积来实现；大多数情况下（如：步幅大于1时），核的梯度无法通过卷积来实现。

2. 卷积是一种线性运算，所以可以表示成矩阵乘法形式，涉及的矩阵是卷积核的函数。

该矩阵有两个特性：该矩阵是稀疏的；卷积核的每个元素都复制到该矩阵的很多个位置。

3. 假设要训练一个卷积神经网络，它包含步幅为  $s$  的步幅卷积，卷积核为  $\mathbf{K}$ ，作用于多通道的图像  $\mathbf{V}$ 。则卷积输出为：

$$Z_{i,j,k} = \sum_l \sum_m \sum_n V_{l,j \times s + m, k \times s + n} K_{i,l,m,n}$$

假设需要最小化某个损失函数  $J(\mathbf{V}, \mathbf{K})$ ，则：

- 前向传播过程：计算  $\mathbf{Z}$ ，然后将  $\mathbf{Z}$  传递到网络的其余部分来计算  $J$ 。
- 反向传播过程：假设得到一个张量  $\mathbf{G}$ ： $G_{i,j,k} = \frac{\partial J}{\partial Z_{i,j,k}}$ 。为了训练网络，需要对过滤器的权重求导。

令  $g(\mathbf{G}, \mathbf{V}, s)_{i,l,m,n} = \frac{\partial J}{\partial K_{i,l,m,n}}$ ，则有：

$$g(\mathbf{G}, \mathbf{V}, s)_{i,l,m,n} = \frac{\partial J}{\partial K_{i,l,m,n}} = \sum_{i'} \sum_j \sum_k \frac{\partial J}{\partial Z_{i',j,k}} \frac{\partial Z_{i',j,k}}{\partial K_{i,l,m,n}}$$

根据  $Z_{i',j,k}$  的定义，有：

$$\frac{\partial Z_{i',j,k}}{\partial K_{i,l,m,n}} = \begin{cases} V_{l,j \times s + m, k \times s + n}, & i' = i \\ 0, & i' \neq i \end{cases}$$

则有：

$$g(\mathbf{G}, \mathbf{V}, s)_{i,l,m,n} = \frac{\partial J}{\partial K_{i,l,m,n}} = \sum_j \sum_k G_{i,j,k} V_{l,j \times s + m, k \times s + n}$$

- 如果该层不是网络的输入层，则需要对  $\mathbf{V}$  求梯度来使得误差进一步反向传播。

令  $h(\mathbf{K}, \mathbf{G}, s)_{l,j,k} = \frac{\partial J}{\partial V_{l,j,k}}$ ，则有：

$$h(\mathbf{K}, \mathbf{G}, s)_{l,j,k} = \frac{\partial J}{\partial V_{l,j,k}} = \sum_i \sum_{j'} \sum_{k'} \frac{\partial J}{\partial Z_{i,j',k'}} \frac{\partial Z_{i,j',k'}}{\partial V_{l,j,k}}$$

根据  $Z_{i,j',k'}$  的定义，有：

$$\frac{\partial Z_{i,j',k'}}{\partial V_{l,j,k}} = \sum_{\substack{m' \\ s.t. j' \times s + m' = j}} \sum_{\substack{n' \\ s.t. k' \times s + n' = k}} K_{i,l,m',n'}$$

则有：

$$h(\mathbf{K}, \mathbf{G}, s)_{l,j,k} = \frac{\partial J}{\partial V_{l,j,k}} = \sum_i \sum_{j'} \sum_{k'} G_{i,j',k'} \sum_{\substack{m' \\ s.t. j' \times s + m' = j}} \sum_{\substack{n' \\ s.t. k' \times s + n' = k}} K_{i,l,m',n'}$$

## 二、卷积层、池化层

1. 卷积运算在神经网络中通过卷基层来实现。
2. 对于卷积层的分层有两种观点：
  - 卷积层是由复杂的、三个阶段的子层组成的（如左图所示）。
  - 第一阶段子层：执行卷积运算。

这个阶段是线性变换，其作用是从输入中提取特征。

- 第二阶段子层：执行非线性的激活函数（如 `ReLU` 单元）。

这个阶段是非线性变换，其作用是引入非线性。

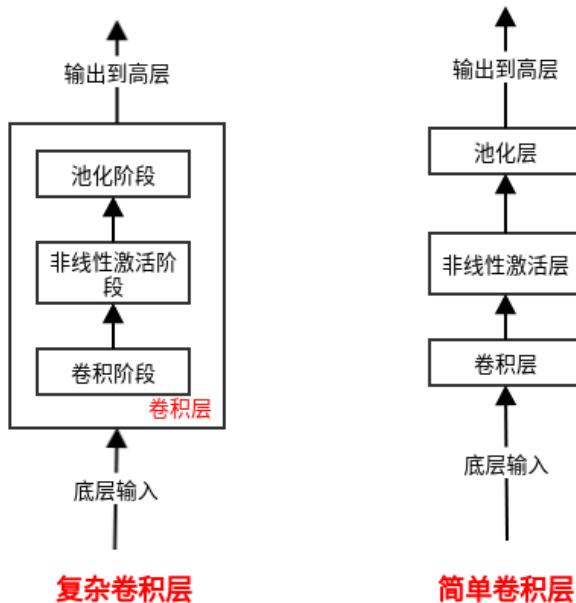
实际应用中，这一阶段可以被丢弃，因为第三阶段也可以引入非线性。

- 第三阶段子层：通过池化函数来调整输出。

这个阶段也是非线性变换，其作用是降低输出的维度，但保留了大部分重要的信息。

- 卷积层是简单的，仅仅包含卷积运算（如右图所示）。这是目前最流行的观点。

在这个观点中，卷积层、非线性激活层、池化层都是简单的网络层，它们相互配合使用。它们的作用参考前面的三个阶段子层。



## 2.1 卷积层

1. 传统的网络层要求输入的每个样本各维度是固定长度的，卷积层可以处理各维度非固定长度的样本数据，如：输入的图片可以为不同的分辨率。
2. 卷积层的卷积运算主要包含了三个重要的思想：稀疏交互 `sparse interactions`、参数共享 `parameter sharing`、等变表示 `equivariant representation`。

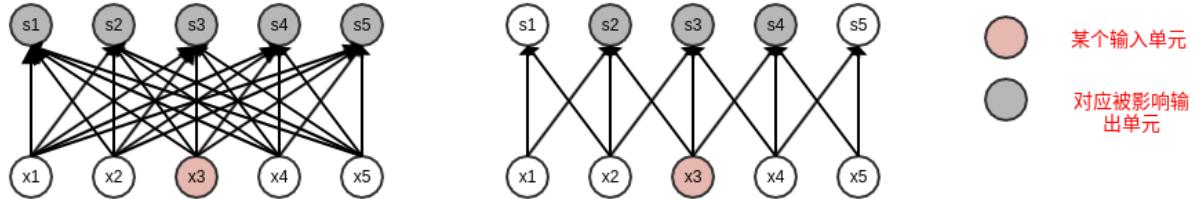
### 2.1.1 稀疏交互

1. 传统的网络层是全连接的，使用矩阵乘法来建立输入与输出的连接关系。矩阵的每个参数都是独立的，它描述了每个输入单元与输出单元的交互。这意味着每个输出单元与所有的输入单元都产生关联。  
卷积层通过使用核矩阵来实现稀疏交互（也称作稀疏连接，或者稀疏权重），每个输出单元仅仅与少量的输入单元产生关联。

这降低了网络的参数和计算量，不仅减少了模型的存储需求，也降低了计算复杂度。

2. 每个输入单元影响的输出单元：

- 对于传统全连接层，每个输入单元影响了所有的输出单元。
- 对于卷积层，每个输入单元只影响了3个输出单元（核尺寸为3时）。

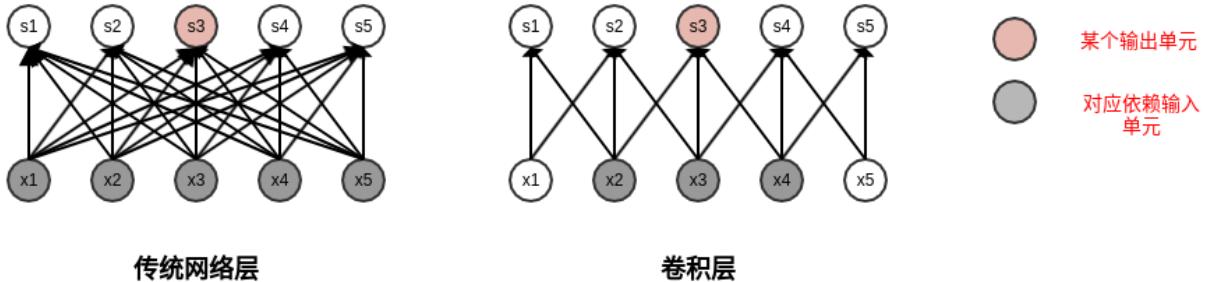


传统网络层

卷积层

3. 每个输出单元依赖的输入单元：

- 对于传统全连接层，每个输出单元依赖所有的输入单元。
- 对于卷积层，每个输出单元只依赖3个输入单元（核尺寸为3时）。

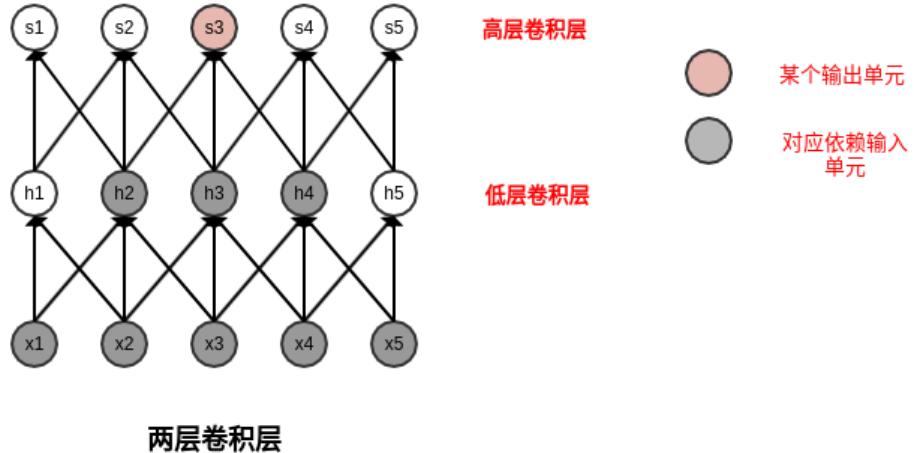


传统网络层

卷积层

4. 在卷积神经网络中，虽然卷积层每个输出单元只依赖于少量的直接输入单元，但是它可能间接依赖于大部分的间接输入单元。

处在卷积网络更深层的单元，其接受域（即影响它的神经元）要比处在浅层的单元的接受域更大。



两层卷积层

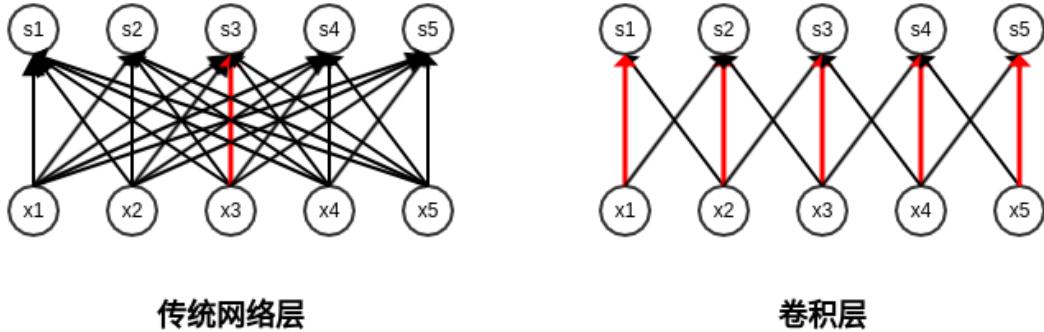
## 2.1.2 参数共享

1. 参数共享：在模型的多个位置使用相同的参数。

- 传统的全连接层中，权重矩阵不同位置处的参数相互独立。
- 卷积层中，同一个核会在输入的不同区域做卷积运算。
  - 核函数会在输入的不同区域之间共享，这些区域的大小就是核函数的大小。
  - 物理意义：将小的、局部区域上的相同的线性变换（由核函数描述），应用到输入的很多区域上。
- 卷积运算在存储需求和计算效率方面要大大优于传统网络层的稠密矩阵的乘法运算。

### 3. 网络参数比较：

- 对于传统全连接层： $x_3 \rightarrow s_3$  的权重  $w_{3,3}$  只使用了一次。
- 对于卷积层： $x_3 \rightarrow s_3$  的权重  $w_{3,3}$  被共享到  $x_i \rightarrow s_i, i = 1, 2, 4, 5$ 。



### 2.1.3 等变表示

- 如果一个函数满足：输入发生改变，输出也以同样的方式改变，则称它是等变的 **equivariant**。  
如果函数  $f(x), g(x)$  满足  $f(g(x)) = g(f(x))$ ，则称  $f(x)$  对于变换  $g$  具有等变性。
- 对于卷积层，它具有平移等变的性质：如果  $g$  是输入的任何一个平移函数（如：将图片向右移动一个像素），则下面的两个操作的结果是相同的：
  - 先对图片进行卷积之后，再对结果应用平移函数  $g$
  - 先对图片应用平移函数  $g$ ，再对图片进行卷积

因为根据定义有：

$$\begin{aligned} \mathbf{S}(i, j) &= (\mathbf{I} * \mathbf{K})(i, j) = \sum_m \sum_n \mathbf{I}(i + m, j + n) \mathbf{K}(m, n) \\ &\rightarrow \sum_m \sum_n \mathbf{I}(i + m + \Delta_i, j + n + \Delta_j) \mathbf{K}(m, n) = \mathbf{S}(i + \Delta_i, j + \Delta_j) \end{aligned}$$

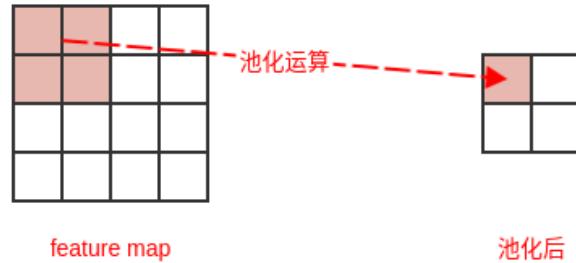
例如：

- 当处理时间序列时，如果把输入中的一个事件向后延时，则卷积的输出中也发生相应的延时。
  - 当处理图像时，如果平移了输入的图片，则卷积的输出结果也平移了相同的量。
- 卷积对于某些变换并不是等变的：如缩放变换（改变图像的尺寸）、角度变换（旋转）。

这些非等变的变换需要其它机制来处理（而不是卷积）。

## 2.2 池化层

- 池化运算也叫亚采样或者下采样。池化运算用一个矩阵区域内部的某个总体统计量来作为神经网络在该矩阵区域的输出，它综合了该矩阵区域的信息。
  - 最大池化：定义一个窗口，并从窗口内取出最大值作为总体统计量。
  - 均值池化：定义一个窗口，并从窗口内取出平均值作为总体统计量。
  - 其他常见的还有： $L^2$  范数以及基于中心像素距离的加权平均函数作为总体统计量。



## 2. 池化层就是对输入专门执行池化运算的网络层。

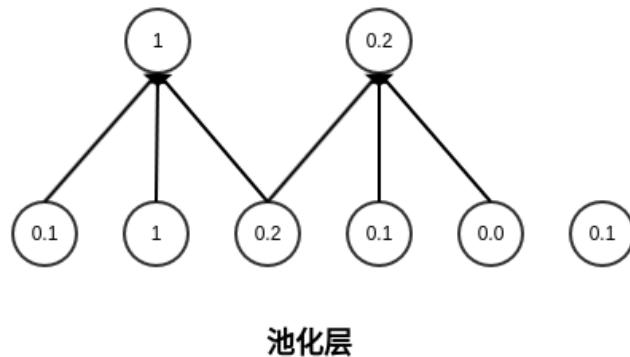
池化层可以减少该层的输出数量。这意味着减少了网络下一层的输入数量，可以减少网络整体参数数量，降低计算量，减少参数存储需求，提高网络计算效率。

### 3. 池化层输出的数量由池化的宽度 $W$ 和步幅 $S$ 决定。设输入单元的数量为 $N$ ，设池化的位置为 $0, 1, \dots, n$ ，则有：

$$n \times S + (W - 1) \leq (N - 1) \rightarrow n \leq \frac{N - W}{S}$$

每个位置对应一个输出单元，因此输出单元的数量为  $\lfloor \frac{N-W}{S} \rfloor + 1$ ，其中  $\lfloor \cdot \rfloor$  为向下取整数。

如下所示，池化层的输入单元数量为 6，池的宽度为 3、步幅为 2，池化层的输出单元数量为 2。



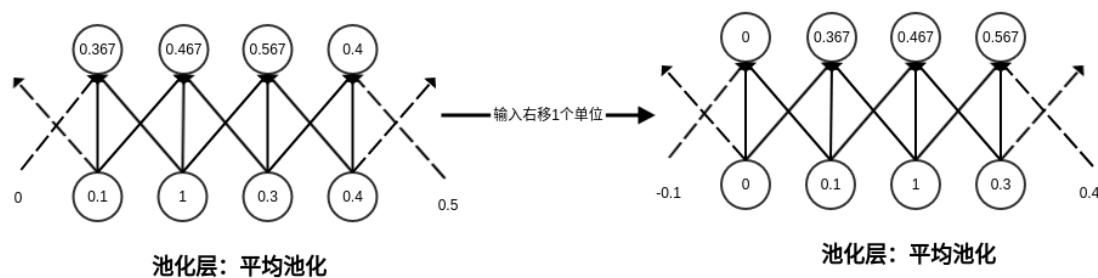
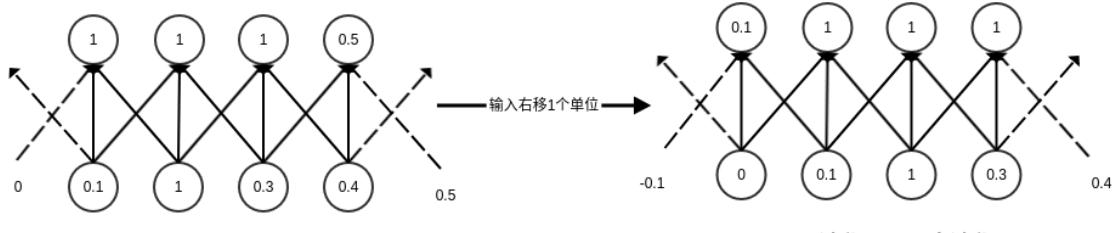
## 4. 池化层的宽度 $W$ 可以随着输入图片尺寸的不同而动态调整，这可以解决图像任务中输入图像尺寸不同的问题。

### 2.2.1 平移近似不变性

#### 1. 当输入做出了少量的平移时，最大池化能够获得输出的近似不变性。即：当输入平移一个微小的量时，经过最大池化的输出近似不变。

下图给出了输入右移了一个单位时，最大池化的输出。可以看到：当输入的每个位置的值都发生改变时，最大池化的输出只有一半的值发生了改变。

- 最大池化平移近似不变性的原因：最大池化单元只对周围存在的最大值比较敏感，而不关心最大值的精确位置。
- 如下图所示，平均池化并没有平移近似不变性。事实上，最大池化、平均池化、以及其它池化对于平移变换均具有等变性。



2. 局部平移不变性是个很重要的性质。该性质表明：网络只关心某个特征是否出现，而不关心它出现的具体位置。如：判断是否人脸时，并不关心眼睛的具体位置，只需要知道一只眼睛在脸的左边、一只眼睛在脸的右边。

但是有些领域，特征的具体位置很重要。如：判断两条线段是否相交。

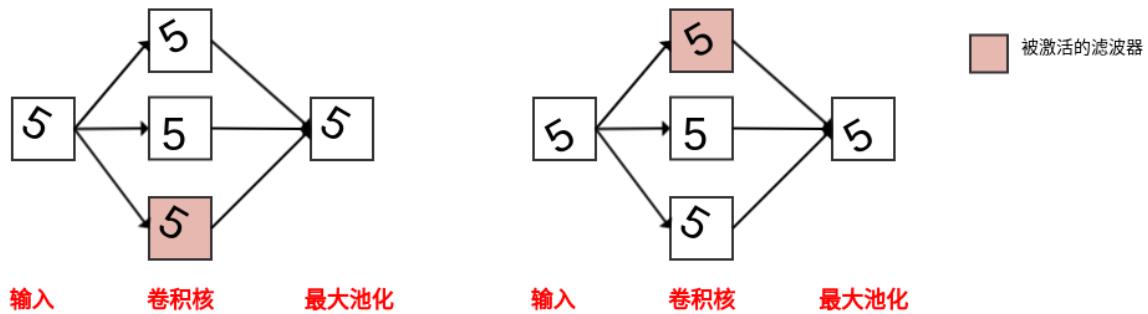
## 2.2.2 模拟其它不变性

1. 最大池化只能对空间的平移具有不变性，而无法对空间的旋转具有不变性。

可以将空间的旋转角度进行离散化，每个角度对应一个卷积，然后采用最大池化。这种做法也可以实现空间的旋转不变性。

下图中，使用多个滤波器和一个最大池化单元可以学得旋转不变性。

- 当输入旋转某个角度时，对应的滤波器被激活。
- 只要任意一个过滤器被激活，最大池化单元也相应被激活。



2. 最大池化的空间平移不变性是原生的，其他变换的不变性则必须采用这种多通道的方法来实现。

## 2.3 卷积层、池化层的先验

1. 先验有强弱之分：

- 弱的先验具有较高的熵，即较大的不确定性。如：方差很大的高斯分布（方差无穷大就成了均匀分布）。

- 强的先验具有较低的熵，即很小的不确定性。如：方差很小的高斯分布（方差很小意味着随机变量取值几乎确定）。

一个无限强的先验对于参数在参数空间的某些取值的概率是0。即：永远不支持参数取这些值。

2. 卷积层也可以看做是一个全连接层，但是它的权值有两个无限强的先验：

- 层内的权重都是重复的，同一个权重将复用多次。
- 层输出单元的接受区域内的权重非零，其它权重都是零。

这些先验表明：卷积层学得的函数只包含了局部连接关系，并且具有稀疏性、平移等变性。

3. 最大池化层也是一个无限强的先验：每个最大池化单元都具有对少量平移的不变性。

4. 卷积与池化只有当先验的假设合理，且正确时才有用。

- 如果任务依赖于保存精确的空间信息，那么使用最大池化将增大训练误差。
- 如果任务需要对输入中相隔较远的信息进行合并，那么卷积所需要的先验可能就不准确。

因为卷积拥有平移等变性，相隔的远近没有意义。

## 2.4 计算复杂度

1. 对于卷积层，假设输入的 `feature map` 为：宽度  $W_I$ 、高度  $H_I$ 、输入通道数  $C_I$ 。假设一共  $C_O$  个卷积核，每个卷积核的宽度为  $W_K$ 、高度为  $H_K$ 。假设沿着宽度方向卷积的步幅为  $S_W$ ，沿着高度方向卷积的步幅为  $S_H$ 。

则输出的 `feature map` 的几个参数为（其中  $\lfloor \cdot \rfloor$  为向下取整数）：

- 宽度： $W_O = \lfloor \frac{W_I - W_K}{S_W} \rfloor + 1$

其推导过程参考池化层的推导。

- 高度： $H_O = \lfloor \frac{H_I - H_K}{S_H} \rfloor + 1$

- 输出通道数  $C_O$

卷积过程中的几个指标：

- 参数数量： $C_I \times W_K \times H_K \times C_O$ ，它也就是核张量的尺寸。
- 计算量（以一次 `乘-加` 计算为单位）： $C_I \times W_K \times H_K \times C_O \times W_O \times H_O$

考虑输出 `feature map` 中的每一个值，每个值都是通过一次卷积计算得到。每个卷积计算的 `乘-加` 运算量为  $C_I \times W_K \times H_K$ 。一共有  $C_O \times W_O \times H_O$  这样的值，则最终计算复杂度为上式。

2. 对于池化层，假设输入的 `feature map` 为：宽度  $W_I$ 、高度  $H_I$ 、输入通道数  $C_I$ 。假设池化窗口的宽度为  $W_P$ 、高度为  $H_P$ 。假设沿着宽度方向池化的步幅为  $S_W$ ，沿着高度方向池化的步幅为  $S_H$ 。

则输出 `feature map` 的几个参数为：

- 宽度： $W_O = \lfloor \frac{W_I - W_P}{S_W} \rfloor + 1$
- 高度： $H_O = \lfloor \frac{H_I - H_P}{S_H} \rfloor + 1$
- 输出通道数  $C_O = C_I$

池化过程中的几个指标：

- 参数数量：0。因为池化过程是无参数的。
- 计算量（以一次求最大值或者均值的计算为单位）： $C_I \times W_O \times H_O$

考虑输出 `feature map` 中的每一个值，每个值都是通过一次池化计算得到。每个池化计算的运算量为 1 个单位。一共有  $C_O \times W_O \times H_O$  这样的值，则最终计算复杂度为上式，其中  $C_O = C_I$ 。

事实上，单次求最大值或者均值的计算量要小于单次`乘-加`的计算量，因此池化层的计算量要远远小于卷积层的计算量。

3. 对于普通的全连接层，假设输入单元数量为  $C_I \times W_I \times H_I$  个，输出单元数量为  $C_O \times W_O \times H_O$  个，其中： $W_O = \lfloor \frac{W_I - W_K}{S_W} \rfloor + 1$ ， $H_O = \lfloor \frac{H_I - H_K}{S_H} \rfloor + 1$ 。

则：

- 参数数量： $C_I \times W_I \times H_I \times C_O \times W_O \times H_O$ 。

因此，全连接层的参数数量是卷积层参数数量的  $\frac{W_I \times H_I \times W_O \times H_O}{W_K \times H_K}$  倍。

当输入 `feature map` 宽高为 `32x32`，卷积核宽高为 `2x2`，步长为 1 时，全连接层的参数数量是卷积层参数数量的 24.6 万倍。

- 计算量：(以一次`乘-加`计算为单位)： $C_I \times W_I \times H_I \times C_O \times W_O \times H_O$ 。

因此，全连接层的计算量是卷积层计算量的  $\frac{W_I \times H_I}{W_K \times H_K}$  倍。

当输入 `feature map` 宽高为 `32x32`，卷积核宽高为 `2x2`，步长为 1 时，全连接层的计算量是卷积层计算量的 256 倍。

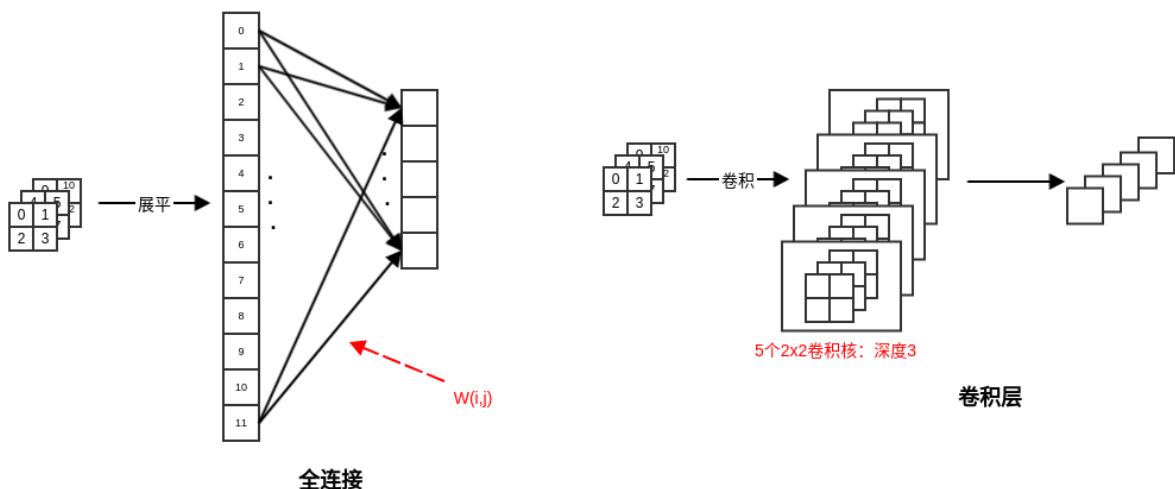
## 2.5 全连接层转卷积层

1. 通常情况下，卷积神经网络的最后几层都是全连接层加一个输出层（如：一个 `softmax` 层）。

事实上，全连接层可以和卷积层相互转换。

2. 假设在全连接之前，网络的 `feature map` 的形状为  $C_I \times W \times H$ ，即：通道数为  $C_I$ 、宽度为  $W$ 、高度为  $H$ 。假设输出一个长度为  $C_O$  的一维向量。

- 如果后接全连接层，则需要将其展平为长度为  $C_I \times W \times H$  的一维向量。权重  $\mathbf{W} = (W_{i,j})$ ， $i = 1, 2, \dots, C_I \times W \times H$ ； $j = 1, 2, \dots, C_O$ ，一共有  $C_I \times W \times H \times C_O$  个参数。
- 如果后接卷积层，则需要  $C_O$  个卷积核，每个卷积核的尺寸都是  $W \times H$ 。一共有  $C_I \times W \times H \times C_O$  个参数。
- 如果将  $W_{:,j}$  重新 `reshape` 成一个尺寸为  $W \times H$ 、输入通道为  $C_I$  的卷积核，则它恰好与后接卷积层的第  $j$  个卷积核相同。而且二者都是输入的线性组合，因此二者可以相互转换。



3. 全连接层转卷积层的优点：可以适应尺寸多变的输入图像。

- 如果是全连接层，则全连接层的输入大小是固定的。  
如果网络的输入尺寸发生变化，则该变化会传递到全连接层，而全连接层无法处理可变的输入。
- 如果是卷积层，则它对输入大小没有限制。

4. 全卷积网络 FCN 就是将 CNN 网络后面的几个全连接层替换为卷积层，它可以高效的对测试图像做滑动窗口式的预测。

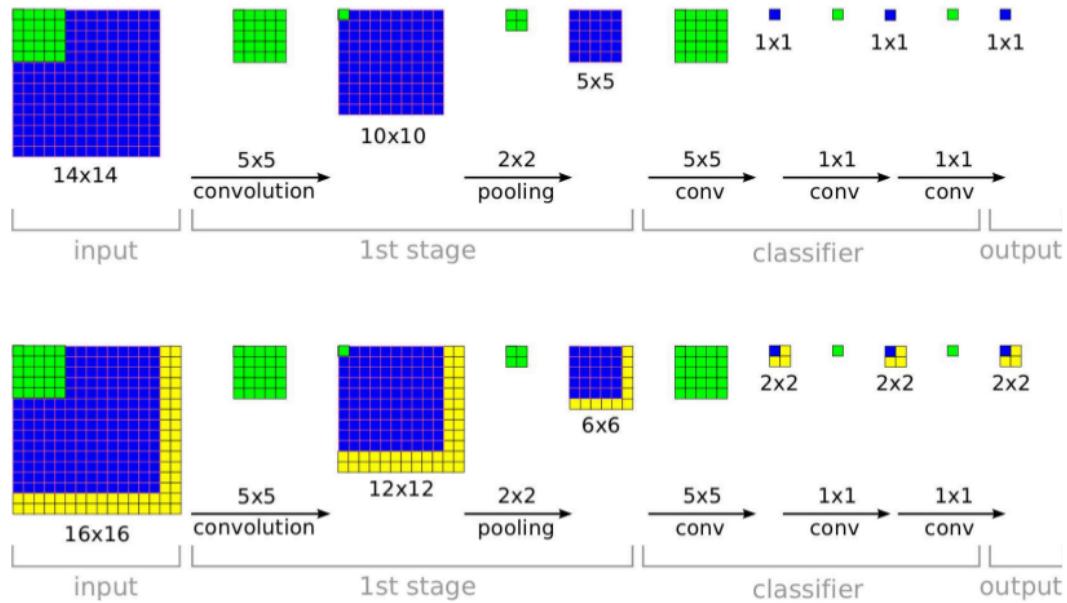
- 在测试阶段，如果图像尺寸大于网络的输入尺寸，则通常需要对图像进行多次裁剪。通常裁剪左上、左下、右上、右下、中间，以及翻转之后的这5个位置，一共10个位置进行预测。  
即：每张大图片需要裁剪10张标准输入大小的图片，然后取这10次预测的综合结果作为该张图片的预测结果。
- 如果采用全卷积网络，则不需要做任何裁剪就可以直接预测，这大大提高了预测效率。

下图中，绿色表示卷积核，蓝色表示 feature map 。

- 上半图：测试图像的尺寸与网络的输入尺寸相同，输出一个概率。
- 下半图：测试图像的尺寸较大时，输出一组概率。

它等效于：先将测试图像先执行所有可能的裁剪（这里是  $2 \times 2 = 4$  种）；然后预测所有裁剪图片的概率；然后将这一组概率取平均，即可得到测试图像的概率。

它的计算效率远远高于全连接的原始方法，因为它节省了大量的共享计算。



5. 例：在 VGG-net 中，训练图像尺寸是 224x224 的，展平之前的 feature map 尺寸为 512x7x7。

如果测试图像尺寸是 384x384 的，展平之前的 feature map 尺寸为 512x12x12。

- 如果使用全连接层，则无法处理这种尺寸的输入（展平后全连接层的输入尺寸不匹配）。
- 如果后接卷积层，则卷积核的尺寸为 7x7、通道数 1000，最终的输出的结果尺寸为 1000x6x6。

它表示分别对测试图像的36个位置使用了原始的 CNN，平均这36个位置的各类别概率，则得到了最终的 1000 个类别的概率。

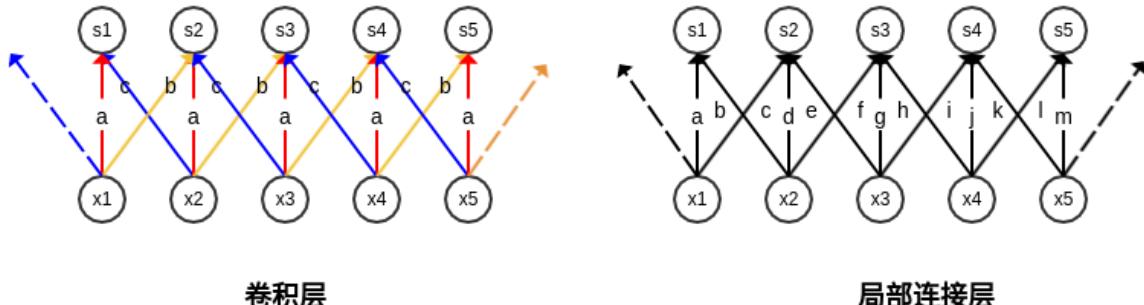
它比等效的执行36次原始的 CNN + 全连接的形式节省了大量的共享计算。

## 三、基本卷积的变体

### 3.1 局部连接

1. 局部连接与卷积很类似：局部连接也是连接受限的，但是每个连接都有自己的权重。

即：局部连接实现了稀疏交互，但是没有实现参数共享。



2. 假设局部连接的权重矩阵为一个 6 维的张量  $\mathbf{W}$ ，其元素为  $W_{i,j,k,l,m,n}$ ，其中： $i$  为输出的通道； $j, k$  为输出通道中的位置； $l$  为输入的通道； $m, n$  为输入通道中的位置。

则局部连接可以表示为：

$$Z_{i,j,k} = \sum_l \sum_m \sum_n V_{l,j+m,k+n} W_{i,j,k,l,m,n}$$

当权重共享时， $W_{i,j,k,l,m,n} = K_{i,l,m,n}$ ，此时局部连接操作退化到卷积操作。

### 3. 局部连接也称作非共享卷积，因为它并不横跨位置来共享参数。

- 与基本卷积相比，权重参数由  $K_{i,l,m,n} \rightarrow W_{i,j,k,l,m,n}$ 。

这说明  $\mathbf{W}$  中，不同位置的输出的计算过程中，采用了不同的权重。这意味着局部连接不满足输入的等变表示。

- 与全连接相比，局部连接实现了稀疏交互。

### 4. 如果知道特征是一小部分区域的函数，而不是整个区域的函数时，局部连接层很有用。此时只需要处理部分输入即可。如：如果需要辨别一张图片是否人脸图像，则只需要在图像的下部中央部分寻找即可。

卷积也可以处理局部特征，但是对于不满足平移不变性的特征，卷积层无能为力。此时需要使用局部连接层。

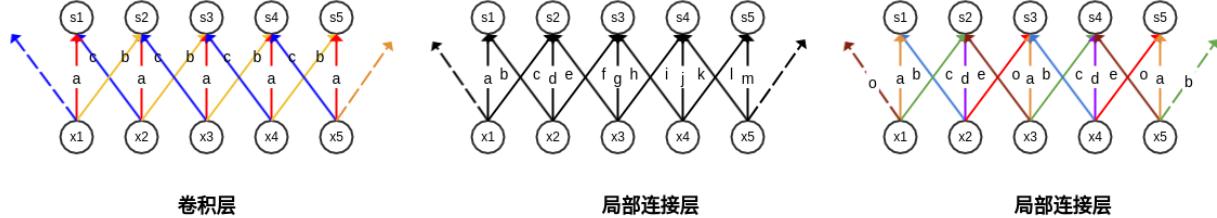
在图片中识别是否存在人脸，这满足平移不变性，因此也可以使用卷积来处理。

### 5. 有时候，可以进一步限制卷积层或者局部连接层。如：限制输出的通道 $i$ 仅仅利用了一部分输入通道（而不是全部输入通道）的数据。

这种方案减少了通道之间的连接，使得模型的参数更少，降低了存储消耗，减少了计算量。

## 3.2 拼接卷积

### 1. 拼接卷积 tiled convolution 对卷积和局部连接进行了折中：学习一组核，使得当核在空间移动时，它们可以循环利用。



- 拼接卷积在相邻的位置上拥有不同的过滤器，就像局部连接层一样。
- 拼接卷积每隔一定的位置，使用相同的过滤器，就像卷积层一样。
- 拼接卷积的参数仅仅会增长常数倍，常数就是过滤器集合的大小。

### 2. 假设拼接卷积的权重矩阵为一个 6 维的张量 $\mathbf{W}$ ，其元素为 $W_{i,j,k,l,m,n}$ ，其中： $i$ 为输出的通道； $j, k$ 为输出通道中的位置； $l$ 为输入的通道； $m, n$ 为输入通道中的位置。

则拼接卷积可以表示为：

$$Z_{i,j,k} = \sum_l \sum_m \sum_n V_{l,j+m,k+n} W_{i,j\%t,k\%t,l,m,n}$$

这里百分号是取模运算， $t$  为不同的核的数量（它也就是核的轮换周期）。

- 如果  $t$  等于输入的尺寸，则拼接卷积退化成局部连接。
- 如果  $t$  等于 1，则拼接卷积退化成卷积。

### 3. 通常在卷积层会引入非线性运算，而在非线性运算中，需要加入偏置项。

- 对于局部连接，每个输入单元都有各自的偏置。
- 对于拼接卷积，偏置项通过与核一样的拼接模式来共享。
- 对于常规卷积，通常在输入通道级别上共享偏置。即：同一个通道使用一个偏置项。

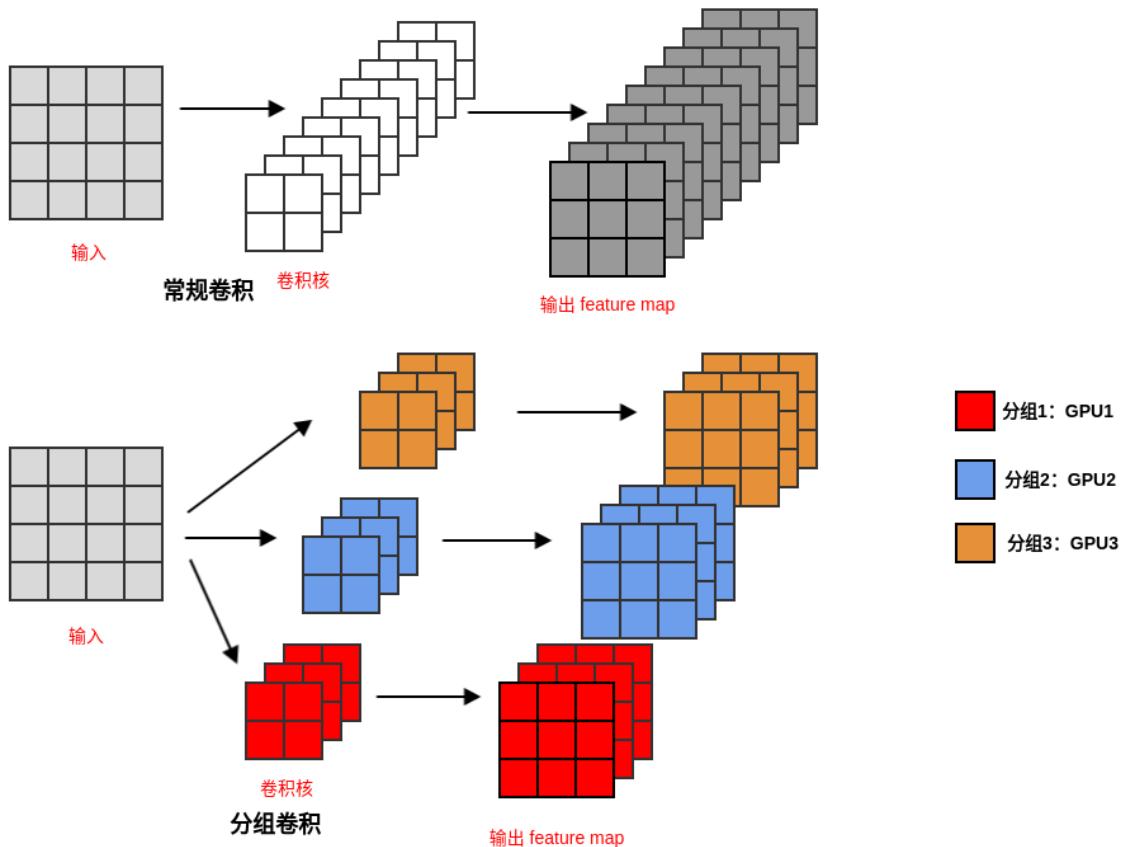
如果输入是固定大小的，也可以在每个输入位置上学习一个单独的偏置。其好处是：允许模型校正输入图像中不同位置的差异。

### 3.3 分组卷积

1. 分组卷积 **Group convolution**：将多个卷积核拆分为分组，每个分组单独执行一系列运算之后，最终在全连接层再拼接在一起。

- 通常每个分组会在单独的 **GPU** 中训练，从而利用多 **GPU** 来训练。
- 分组卷积的重点不在于 **卷积**，而在于 **分组**：在执行卷积之后，将输出的 **feature map** 执行分组。然后在每个组的数据会在各个 **GPU** 上单独训练。

对卷积的输出 **feature map** 分组，等价于在卷积过程中对卷积核进行分组。



2. 分组卷积在网络的全连接层才进行融合，这使得每个 **GPU** 中只能看到部分通道的数据，这降低了模型的泛化能力。

如果每次分组卷积之后，立即融合卷积的结果则可以解决这个问题。

3. 分组卷积降低了模型的参数数量以及计算量。

假设输入 **feature map** 具有  $C_I$  的输入通道、宽/高分别为  $W_I, H_I$ ，假设卷积核的宽/高分别为  $W_K, H_K$ ，有  $C_O$  个卷积核。则：

- 参数数量： $W_K \times H_K \times C_I \times C_O$
- 计算量（以一次 乘-加 计算为单位）： $W_K \times H_K \times C_I \times W_O \times H_O \times C_O$ 。其中  $W_O, H_O$  分别为输出 **feature map** 的宽/高

假设采用分组卷积，将输入通道分成了  $G$  组，则分组之后：

- 参数数量： $G \times W_K \times H_K \times \frac{C_I}{G} \times \frac{C_O}{G}$
- 计算量(以一次乘-加计算为单位)： $G \times W_K \times H_K \times \frac{C_I}{G} \times W_O \times H_O \times \frac{C_O}{G}$

因此分组卷积的参数数量、计算量均为标准卷积计算的  $\frac{1}{G}$ 。

考虑到全连接层的参数数量在网络中占据主导地位，因此即使采取分组卷积，网络最终的参数数量的减小幅度不会很大。

因此分组卷积主要降低的是计算量。

- 分组卷积最早在 AlexNet 中出现。由于当时的硬件资源的限制，训练 AlexNet 时卷积操作无法全部放在同一个 GPU 中处理。因此，通过分组来在多个 GPU 上分别处理，然后将多个 GPU 的处理结果融合。

### 3.4 小卷积核替代

- 在 AlexNet 中用到了一些非常大的卷积核，如  $11 \times 11$ 、 $5 \times 5$  等尺寸的卷积核。

- 卷积核的尺寸越大，则看到的图片信息越多，因此获得的特征会越好。

但是卷积核的尺寸越大，模型的参数数量会暴涨，不利于模型的深度的增加，计算量和存储量也大幅上升。

- 卷积核的尺寸越小，模型的参数数量越少，模型可以越深。

但是卷积核的尺寸太小，则只能看到图片的一个非常小的局部区域，获得的特征越差。

- 一种解决方案是：用多个小卷积层的堆叠来代替较大的卷积核。

假设大卷积核的宽度是  $k$ ，则每经过一层，输出的宽度减少了  $k - 1$ 。假设希望通过  $n$  个宽度为  $k'$  的小卷积核来代替，则为了保持输出的大小一致，需要满足：

$$k - 1 = n(k' - 1)$$

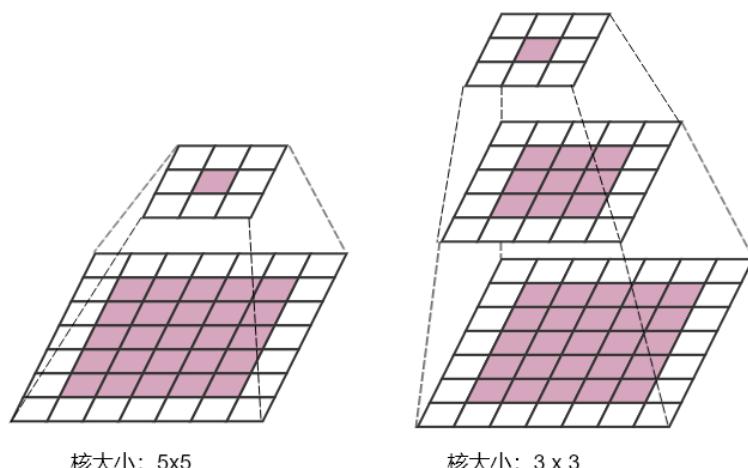
- 当  $k' = 3$  时，即用尺寸为 3 的卷积核代替尺寸为  $k$  的卷积核时，有： $n = \frac{k-1}{2}$

- 如：用 2 个  $3 \times 3$  的卷积核来代替 1 个  $5 \times 5$  的卷积核。

假设输入通道数为  $C_I$ ，输出通道数为  $C_O$ ，则  $5 \times 5$  卷积核的参数数量为  $C_O \times C_I \times 5 \times 5$ ；

而 2 个  $3 \times 3$  卷积核的参数数量为  $2 \times C_O \times C_I \times 3 \times 3$ ，是前者的 72%。

- 如果用 5 个  $3 \times 3$  的卷积核来代替 1 个  $11 \times 11$  的卷积核，则替代后的卷积核的参数数量是替代前的 37%。



### 3. 用多个小卷积层的堆叠代替一个大卷积层的优点：

- 可以实现与大卷积层相同的感受野。
- 具有更大的非线性，网络表达能力更强。

虽然卷积是线性的，但是卷积层之后往往跟随一个 `ReLU` 激活函数。这使得多个小卷积层的堆叠注入了更大的非线性。

- 具有更少的参数数量。

### 4. 小卷积层堆叠的缺点是：加深了网络的深度，容易引发梯度消失等问题，从而使得网络的训练难度加大。

### 5. 用多个小卷积层的堆叠代替一个大卷积层可以看作是一种正则化：要求大卷积核通过多个小卷积核进行分解（同时在小卷积层之间注入非线性）。

### 6. 感受野：一个特定的 `CNN` 输出单元在输入空间所受影响的区域。上图中，染色的区域为某个输出单元的感受野。

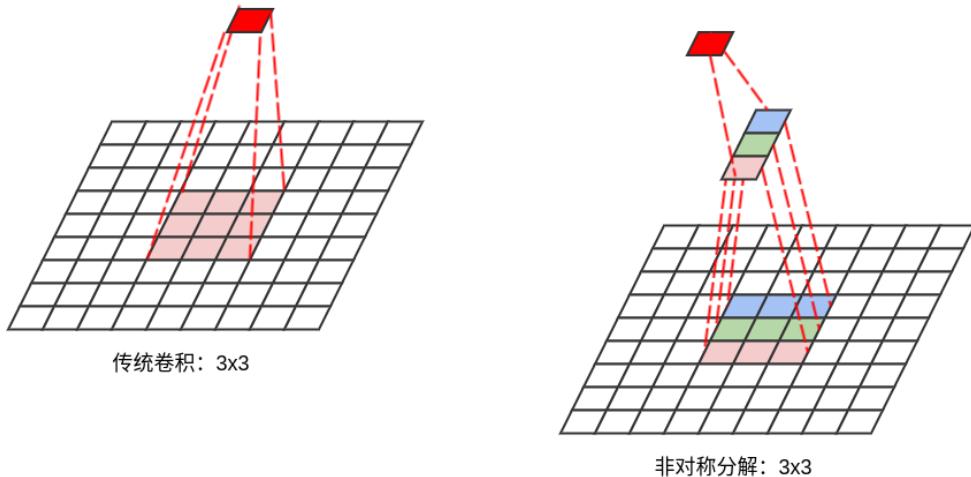
- 一个感受野可以用中心位置和大小来表示。
- 用多个小卷积核来代替大卷积核时，输出单元的感受野不会受到影响。

### 7. 通常选择使用 `3x3` 卷积核的堆叠：

- `1x1` 的卷积核：它无法提升感受野，因此多个 `1x1` 卷基层的堆叠无法实现大卷积层的感受野。
- `2x2` 的卷积核：如果希望输入的 `feature map` 尺寸和输出的 `feature map` 尺寸不变，则需要对输入执行非对称的 `padding`。此时有四种 `padding` 方式，填充方式的选择又成了一个问题。
- `3x3` 的卷积核：可以提升感受野，对称性填充（不需要考虑填充方式），且尺寸足够小。

## 3.5 非对称卷积核

### 1. 在卷积核分解过程中，还有一种分解方式：非对称卷积核分解，将 `nxn` 卷积替换为 `1xn` 卷积和 `nx1` 卷积。



### 2. 非对称卷积核的分解有以下优点：

- 感受野保持不变。
- 节省计算成本，尤其是当 `n` 较大时。

假设输入通道数和输出通道数都为  $C$ ，原始卷积 `nxn` 的参数数量为： $n \times n \times C \times C = n^2 C^2$ 。

假设非对称卷积的 `1xn` 的输出通道数也是  $C$ ，则非对称分解时参数数量为：

$1 \times n \times C \times C + n \times 1 \times C \times C = 2nC^2$ 。它是原始卷积的参数数量的  $\frac{2}{n}$ 。

3. 在 Inception v2 论文中作者指出：对于较大的特征图，这种分解不能很好的工作；但是对于中等大小的特征图（尺寸在 12~20 之间），这种分解效果非常好。

因此非对称卷积分解通常用在较高的网络层。

### 3.6 多尺寸卷积核

1. 图像中目标对象的大小可能差别很大。如下图所示，每张图像中狗占据区域都是不同的。

由于信息区域的巨大差异，为卷积操作选择合适的卷积核尺寸就非常困难。

- 信息分布更具有全局性的图像中，更倾向于使用较大的卷积核。如最左侧的图片所示。
- 信息分布更具有局部性的图像中，更倾向于使用较小的卷积核。如最右侧的图片所示。

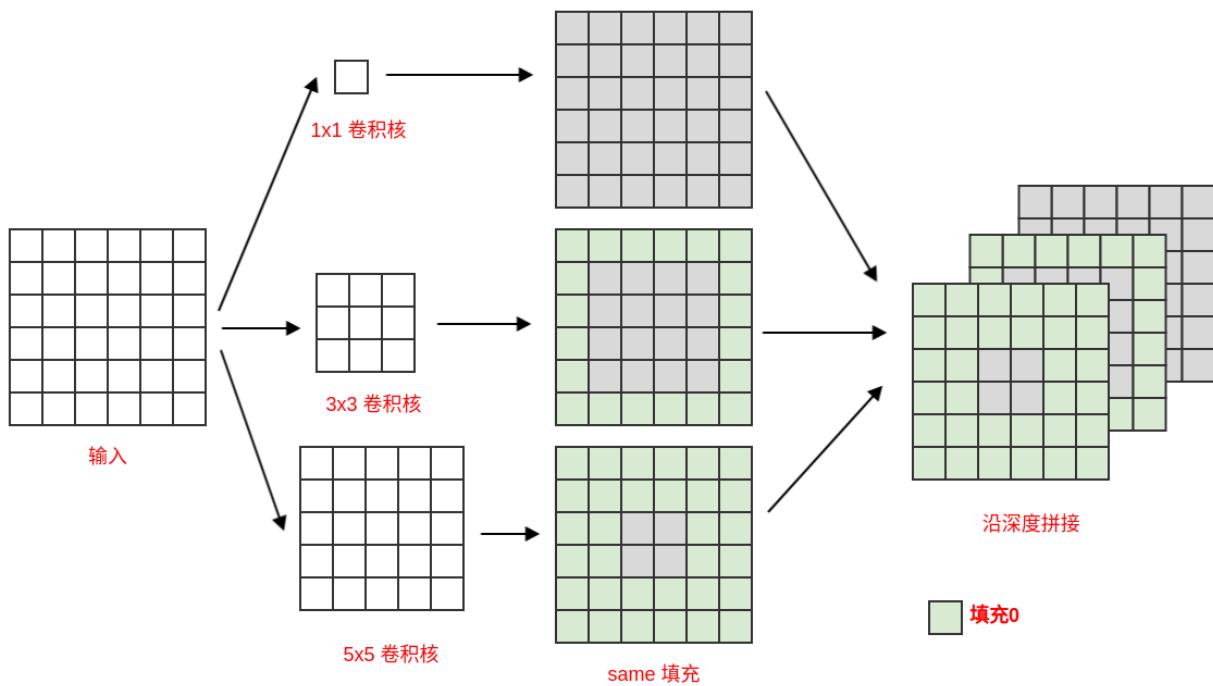


2. 一个解决方案是：分别使用多个不同尺寸的卷积核从而获得不同尺度的特征。然后将这些特征拼接起来。

- 在 Inception 系列的网络中，大量使用这种思想。

在最初版本的 Inception 结构中，一个输入图片会分别同时经过  $1 \times 1, 3 \times 3, 5 \times 5$  的卷积核的处理；得到的特征再组合起来。

- 通过多种尺度的卷积核，无论感兴趣的信息区域尺寸多大，总有一种尺度的卷积核与之匹配。这样总可以提取到合适的特征。



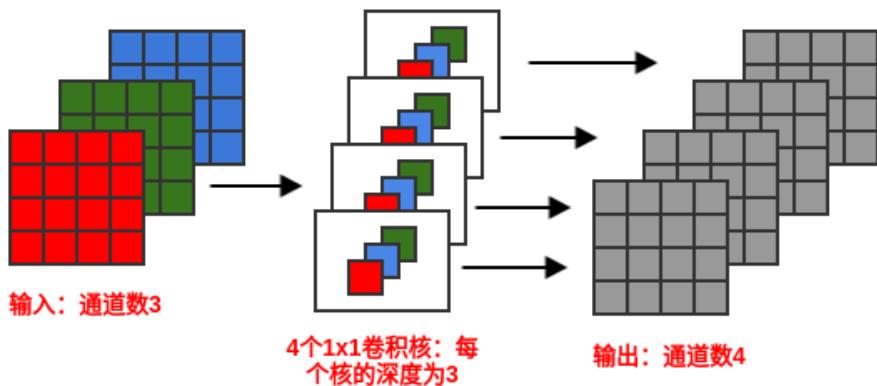
3. 多尺寸卷积核存在一个严重的问题：参数数量比单尺寸卷积核要多很多，这就使得计算量和存储量都大幅增长。

### 3.7 1x1 卷积核

1. **1x1** 卷积并不是复制输入，它会进行跨通道的卷积。它有三个作用：

- 实现跨通道的信息整合。
- 进行通道数的升维和降维。
- 在不损失分辨率的前提下（即：**feature map** 尺寸不变），大幅增加非线性。

事实上 **1x1** 卷积本身是通道的线性组合，但是通常会在 **1x1** 卷积之后跟随一个 **ReLU** 激活函数。



2. 假设输入张量为  $\mathbf{I} \in \mathbb{R}^{(C_I \times W_I \times H_I)}$ ，即： $C_I$  个通道、宽度为  $W_I$ 、高度为  $H_I$ 。

- 如果图片直接通过一个宽度为  $W_K$ 、高度为  $H_K$ 、输出通道为  $C_O$  的卷积层，则参数数量为：

$$C_I \times C_O \times W_K \times H_K$$

- 如果图片先通过一个 **1x1**、输出通道为  $\sqrt{C_O}$  的卷积层，再经过一个  $W_K \times H_K$ 、输出通道为  $\sqrt{C_O}$  的卷积层；最后经过一个 **1x1**、输出通道为  $C_O$  的卷积层。

这里中间卷积层的输出通道不一定为  $\sqrt{C_O}$ ，但是一定是一个比  $C_O$  小的数。其作用是起到了信息压缩的作用（类似于 **PCA** 降维）。

则参数数量为：

$$\begin{aligned} & C_I \times 1 \times 1 \times \sqrt{C_O} + \sqrt{C_O} \times W_K \times H_K \times \sqrt{C_O} + \sqrt{C_O} \times 1 \times 1 \times C_O \\ & = C_I \sqrt{C_O} + W_K H_K C_O + C_O^{3/2} \end{aligned}$$

- 当  $C_I \approx C_O$  时（输入通道数与输出通道数接近）， $C_I \times C_O \times W_K \times H_K \approx W_K H_K C_O^2$ ，以及  $C_I \sqrt{C_O} + W_K H_K C_O + C_O^{3/2} \approx 2C_O^{3/2} + W_K H_K C_O$ 。

则二者参数的数量比例为： $\frac{2C_O^{3/2} + W_K H_K C_O}{W_K H_K C_O^2} \approx \frac{1}{C_O}$ 。因此后者的参数数量远远小于前者。

3. **1x1** 卷积层通常会形成瓶颈层 **bottleneck layer**。瓶颈层指的是网络中信息被压缩的层。

- 输入 **feature map** 中每个元素值代表某个特征，将所有图片在该 **feature map** 上的取值扩成为矩阵：

$$\mathbf{P} = \begin{bmatrix} P_{1,1} & P_{1,2} & \cdots & P_{1,n} \\ P_{2,1} & P_{2,2} & \cdots & P_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ P_{N,1} & P_{N,2} & \cdots & P_{N,n} \end{bmatrix}$$

其中  $N$  为样本的数量， $n = W \times H \times C$ 。即：行索引代表样本，列索引代表特征。所有特征由 `feature map` 展平成一维得到。

通常  $N \gg n$ ，则输入矩阵  $\mathbf{P}$  的秩  $\text{rank}(\mathbf{P}) < n$ 。假设输入矩阵  $\mathbf{P}$  的秩为  $r$ 。

- 不考虑 `1x1` 卷积的非线性层，则 `1x1` 卷积是输入特征的线性组合。输出 `featuremap` 以矩阵描述为：

$$\mathbf{P}^* = \begin{bmatrix} P_{1,1}^* & P_{1,2}^* & \cdots & P_{1,n^*}^* \\ P_{2,1}^* & P_{2,2}^* & \cdots & P_{2,n^*}^* \\ \vdots & \vdots & \ddots & \vdots \\ P_{N,1}^* & P_{N,2}^* & \cdots & P_{N,n^*}^* \end{bmatrix}$$

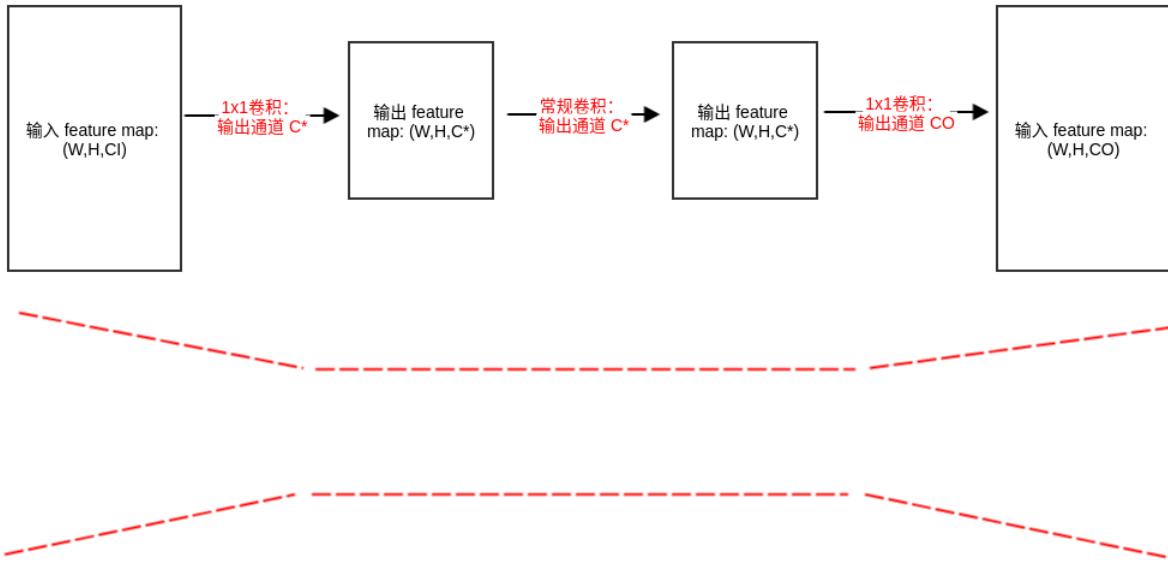
其中  $n^* = W \times H \times C^*$ 。

- 信息膨胀过程：当  $C^* > C$  时， $\mathbf{P}^*$  的容量要大于  $\mathbf{P}$  的信息，因此所有的有效信息都可以得到保留。
- 信息压缩过程：当  $C^* < C$  时， $\mathbf{P}^*$  的容量要小于  $\mathbf{P}$  的信息，这时需要参考  $\mathbf{P}$  的有效信息。

$\mathbf{P}$  的有效信息由矩阵  $\mathbf{P}$  的秩  $r$  决定。

- 当矩阵  $\mathbf{P}$  是满秩时，即： $r = n$ ，此时对  $\mathbf{P}$  的任何压缩都会丢失有效信息。
- 当矩阵  $\mathbf{P}$  不是满秩时：
  - 当  $C^*$  非常小时， $\mathbf{P}^*$  的容量要小于  $\mathbf{P}$  的有效信息，此时会导致有效信息丢失。
  - 当  $C^*$  较大时， $\mathbf{P}^*$  的容量要大于等于  $\mathbf{P}$  的有效信息，此时不会丢失有效信息。这是 `1x1` 卷积相当于线性降维。
- 在前文提到的例子（示意图如下所示）中，输入 `feature map` 先经过 `1x1` 卷积的压缩，这会导致该段信息容量的下降；然后经过常规卷积，此段信息容量不变；最后经过 `1x1` 卷积的膨胀，恢复了信息容量。

整体而言模型的信息容量很像一个 `bottleneck`，因此 `1x1` 卷积层也被称作瓶颈层。



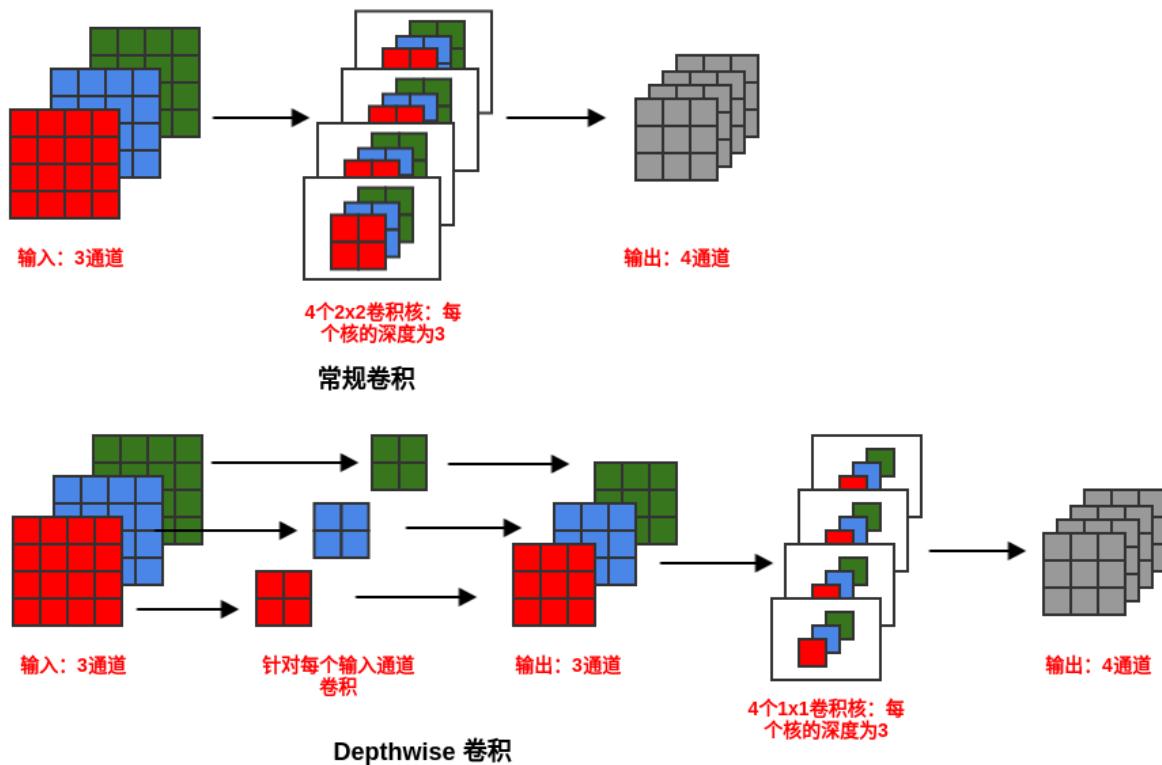
4. 事实上，不仅 `1x1` 卷积层会形成 `bottleneck`，任何会降低模型信息容量的层都会形成瓶颈。

因此在卷积神经网络中，通常每经过一个卷积层，输出尺寸减半、输出通道数翻倍。

5. 瓶颈层中的信息膨胀阶段不是必须存在，通常信息膨胀过程是为了保持整个瓶颈层的输入尺寸、输出尺寸满足某些约束。如：输出尺寸等于输入尺寸。

### 3.8 DepthWise 卷积

1. 标准的卷积会考虑所有的输入通道，而 `DepthWise` 卷积会针对每一个输入通道进行卷积操作，然后接一个 `1x1` 的跨通道卷积操作。



2. `DepthWise` 卷积与分组卷积的区别在于：

- 分组卷积是一种通道分组的方式，它改变的是对输入的 `feature map` 处理的方式。
- `Depthwise` 卷积是一种卷积的方式，它改变的是卷积的形式。
- `Depthwise` 分组卷积结合了两者：首先沿着通道进行分组，然后每个分组执行 `DepthWise` 卷积。

3. 假设输入张量为  $\mathbf{I} \in \mathbb{R}^{(C_I \times W_I \times H_I)}$ ，即： $C_I$  个通道、宽度为  $W_I$ 、高度为  $H_I$ 。

- 假设使用标准卷积，输入通道的数量为  $C_I$ ，输出通道的数量为  $C_O$ ，卷积核的尺寸为  $W_K \times H_K$ 。则需要的参数数量为  $C_I \times W_K \times H_K \times C_O$ 。
- 使用 `Depthwise` 卷积时，图像的每个通道先通过一个  $W_K \times H_K$  的 `depthwise` 卷积层，再经过一个 `1x1`、输出通道为  $C_O$  的卷积层。

参数数量为：

$$C_I \times W_K \times H_K + C_I \times 1 \times 1 \times C_O = W_K H_K C_I + C_I C_O$$

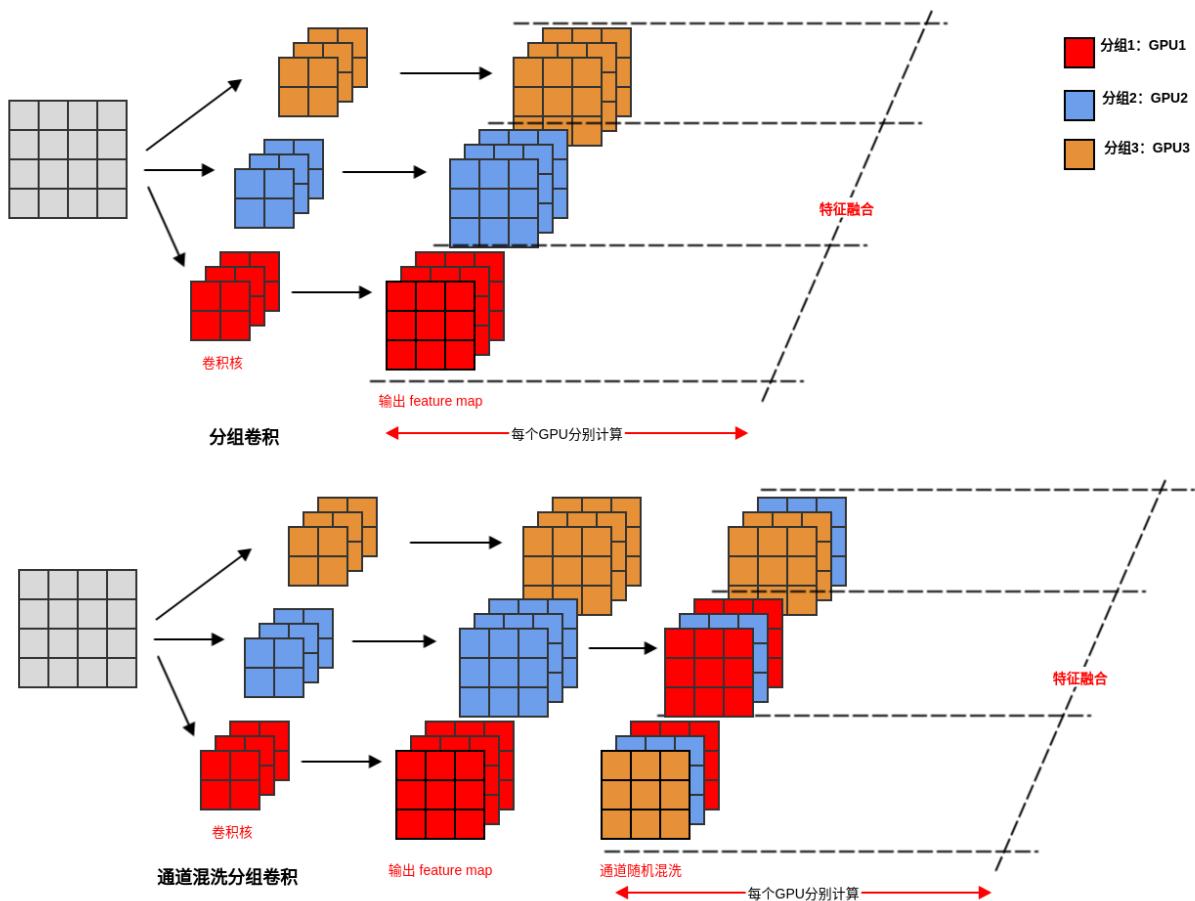
其参数数量是标准卷积的  $\frac{1}{C_O} + \frac{1}{W_K H_K}$ 。因此 `depthwise` 卷积的参数数量远远小于标准卷积。

4. `DepthWise` 卷积有几种变形的形式：

- 只有对每个输入通道执行单通道卷积，没有后续的 `1x1` 的跨通道卷积。
- 对输入通道执行单通道卷积的结果执行 `BN` 和 `ReLU`，再后接 `1x1` 的跨通道卷积。这会引入更多的非线性。

## 3.9 通道混洗分组卷积

1. 在分组卷积中，特征的通道被平均分配到不同的分组中。如果融合的时刻非常靠后，则对模型的泛化性相当不利，因为如果能在早期知道其它通道的一些信息，则很可能得到更有效的特征。
2. 通道混洗分组卷积在每一次分组卷积之后执行一次通道混洗，被混洗过的通道被分配到不同的分组中。  
经过通道混洗之后，每个分组输出的特征能够考虑到更多的通道，因此输出特征的表达能力更强。  
在 `ShuffleNet` 中，大量运用了这种通道混洗分组卷积。



3. 在 AlexNet 的分组卷积中，执行的是标准卷积操作。

在 ShuffleNet 中，分组卷积执行的是 `depthwise` 卷积，从而使得参数更少。

### 3.10 通道加权卷积

1. 在常规卷积中，各通道产生的特征都是不分权重直接结合的。通道加权卷积中，不同的通道具有不同的权重，各通道产生的特征经过加权之后再结合。

所用到的权重是输入的函数。

注意：因为卷积是线性过程，因此卷积计算的通道加权等价于对输入的 `feature map` 的通道加权。

2. SEnet ( Squeeze-and-Excitation Networks ) 网络大量使用通道加权卷积。在 SEnet 中存在三个关键的操作：

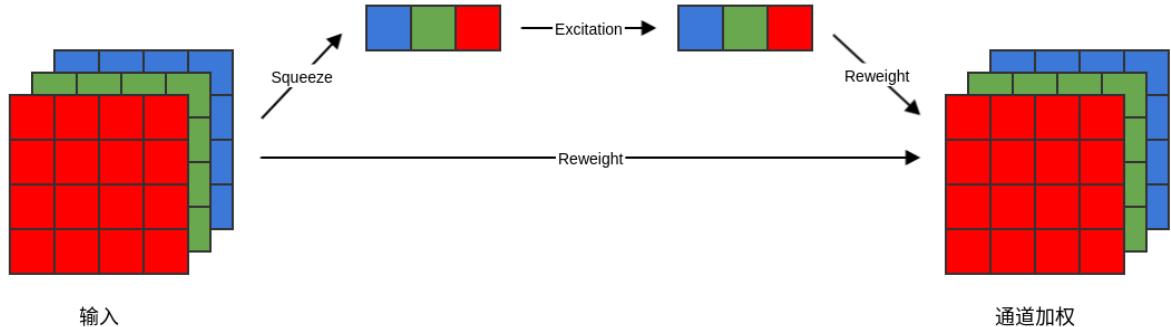
- `Squeeze` 操作：沿着空间维度压缩特征，将每个二维的 `feature map` 通道压缩成一个实数。

该实数具有全局的感受野，表征了在该 `feature map` 通道上响应的全局分布。

- `Excitation` 操作：通过一个类似循环神经网络中的门机制，用一个 `sigmoid` 激活函数的全连接层获取每个 `feature map` 通道的权重。

实际上，`Excitation` 操作使用了两个全连接层来获取通道权重。

- `Reweight` 操作：将特征通道的权重通过乘法逐通道的加权到先前的 `feature map` 上。



### 3.11 空洞卷积

1. 在图像分割任务，图像输入到传统卷积层，然后再输入到池化层。由于图像分割是逐像素的输出，因此需要将池化层的输出（一个较小的尺寸）升采样（一般使用反卷积操作）到原始的图像尺寸来进行预测。

但是这里有几个问题：

- 升采样（如：线性插值）是确定性的映射，无法学习（也没有参数要学习）。
  - 在这个图像的先减少尺寸、再增大尺寸过程中，有一些信息损失。
  - 小物体信息无法重建。假设有4个池化层，每个池化层的尺寸为2、步长为2，理论上任何小于  $2^4 = 16$  个像素的物体信息将无法被重建。

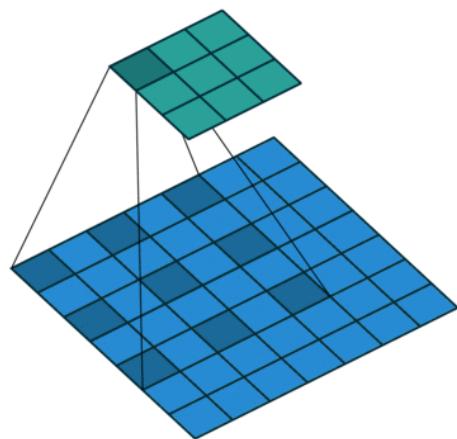
解决方案是空洞卷积。

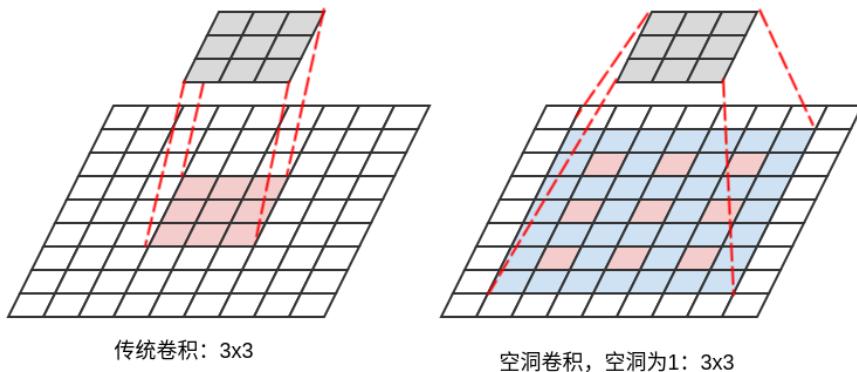
2. 空洞卷积：对于空洞数为  $d$  的空洞卷积，卷积结果为：

$$\mathbf{S}(i, j) = \sum_m \sum_n \mathbf{I}(i + m(d+1) + 1, j + n(d+1) + 1) \mathbf{K}(m, n)$$

它实际上等价于一个卷积核为  $(d + 1)K + 1$  的新的卷积核，其中  $K$  为当前卷积核的大小。新的卷积核的特点是：每隔  $d$  个位置，权重非零；否则权重为零。另外首行、首列、尾行、尾列权重均为零。

$d + 1$  称作膨胀比 dilation rate。



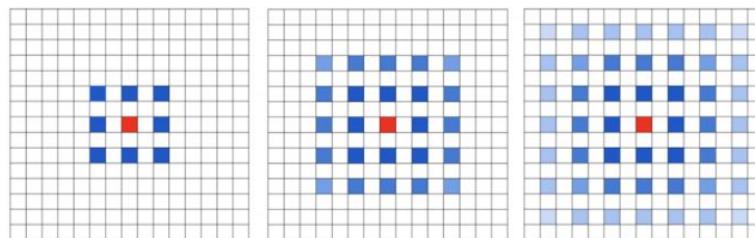


3. 空洞卷积的优点：在不做池化损失信息的情况下，加大感受野，让每个卷积的输出都包含较大范围的信息。

在图像需要全局信息，或者语音、文本需要较长序列信息的问题中，空洞卷积都能很好的应用。

4. 空洞卷积的缺点：

- 网格效应( Gridding Effect )。如果仅仅多次叠加多个 `dilation rate=2` 的 `3x3` 的卷积核时，会发现：并不是所有的输入像素都得到计算，也就是卷积核不连续。  
这对于逐像素的预测任务来说，是致命的问题。



- 长距离信息可能与任务无关。采用空洞卷积可能对大物体的分割有效果，但是对于小物体的分割可能没有好处。

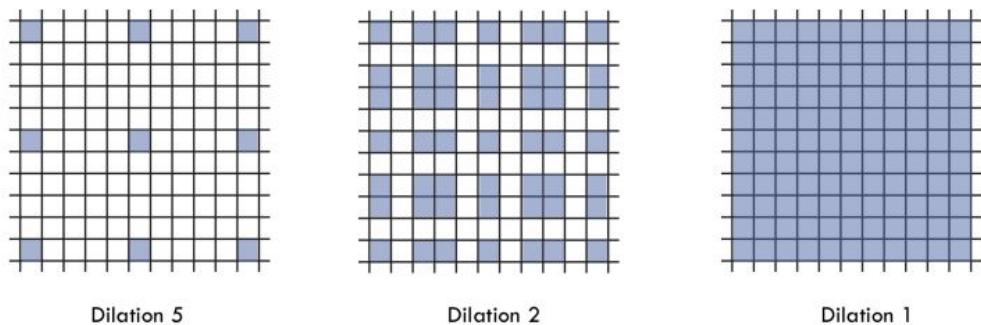
如何同时处理不同大小的物体，则是设计好空洞卷积网络的关键。

5. 为了解决空洞卷积的缺点，人们提出了一种混合空洞卷积的结构 (`Hybrid Dilated Convolution:HDC`)。

该结构有三个特性：

- 叠加的空洞卷积的 `dilation rate` 不能有大于1的公约数。这是为了对抗网格效应。  
如：`[2,4,6]` 不是一个好的三层空洞卷积，因为会出现网格效应。
- 将 `dilation rate` 设计成锯齿状结构。这是为了同时满足小物体、大物体的分割要求。  
如 `[1,2,5,1,2,5]` 的循环结构。
- 最后一层的空洞卷积的 `dilation rate` 最大，且 `dilation rate` 小于等于卷积核的大小。  
这也是为了对抗网格效应。

下面是一组 `dilation rate` 分别为 `[1,2,5]` 的卷积核，卷积核的尺寸为 `3x3`

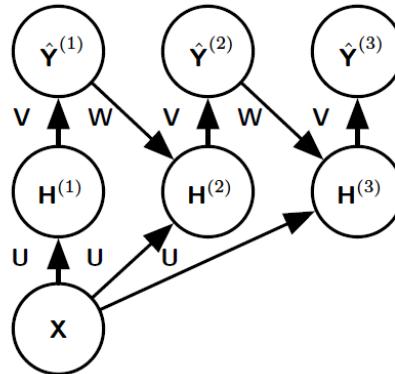


## 四、应用

### 4.1 逐像素输出

1. 卷积神经网络不仅可以输出分类任务的标签或者回归任务的实数值，还可以输出高维的结构化对象。如：图片上每个像素属于各个类别的概率。这允许模型标记图像中的每个像素，并绘制单个物体的精确轮廓。  
这种结构化对象用张量表示。如：张量  $\mathbf{S}$ ，其中  $S_{i,j,k}$  是网络的输入像素  $(j, k)$  属于类  $i$  的概率。
2. 对图像进行逐个像素标记的一种策略是：先产生图像标记的一个原始猜测；然后使用相邻像素之间的校验来修正该原始猜测；重复上述修正步骤直到收敛。

如下图所示：输入图像张量  $\mathbf{X}$ ，输出每个像素的类别张量  $\hat{\mathbf{Y}}$ 。



- 该网络并不是一次性输出结果  $\hat{\mathbf{Y}}$ ，而是使用前一轮的输出  $\hat{\mathbf{Y}}^{(t-1)}$  来改善结果  $\hat{\mathbf{Y}}^{(t)}$ 。
  - 每一步对  $\mathbf{X}$  执行卷积的卷积核都是张量  $\mathbf{U}$ 。
  - 每一步产生的  $\mathbf{H}^{(t)}$  都需要两个输入：
    - 一个输入是通过对图像  $\mathbf{X}$  采用核  $\mathbf{U}$  来卷积。
    - 一个输入是通过对前一个输出  $\hat{\mathbf{Y}}^{(t-1)}$  采用核  $\mathbf{W}$  进行卷积。
  - 第一次产生  $\mathbf{H}^{(1)}$  时，这一项为零，因为还没有前一次输入。
  - 张量  $\mathbf{V}$  用于产生从  $\mathbf{H}^{(t)}$  到  $\hat{\mathbf{Y}}^{(t)}$  的输出。
3. 每一次重复修正相当于再一次执行同样的卷积（卷积核为  $\mathbf{U}$ ）。
- 很多个这样的卷积组成了一个深层网络，该网络的最后几层之间存在跨层的权值连接（连接权重为  $\mathbf{W}$ ）。这种跨层的权值连接构成了反馈，因此这种深层网络形成了一个特殊的循环神经网络。
4. 一旦对每个像素都进行了一次预测，就可以用各种方法来进一步处理这些预测的结果。
- 常规思路是：假设大片相连的像素对应于相同的标签。

## 4.2 可变输入类型

1. 卷积神经网络使用的数据通常包含多个通道：每个通道都是时间/空间上一个点的某个角度的观测量。

同一个点，观测的角度不同，就产生了不同的通道。如：

- 三维的单通道：立体成像的数据。每个点代表了三维空间的一个点。
    - 三维：空间的三个维度。
    - 单通道：数据通道。
  - 三维的多通道：彩色视频数据。时间维度+二维空间（图像）+色彩通道（红绿蓝三通道）
    - 三维：时间维度（一维）+ 图像维度（二维）。
    - 多通道：色彩通道（红绿蓝三通道）。
2. 卷积神经网络还可以处理具有变化的空间尺度的输入。如：输入图片的尺寸可能各不相同。
- 这种不同尺寸大小的输入，无法使用传统的基于矩阵乘法的神经网络来表示。  
因为不同样本的输入的维度可能不同，所以权重矩阵的形状无法确定。
  - 卷积神经网络可以处理这种情况：根据输入图片尺寸的大小，核会被自动的使用不同次数。
    - 如果要求网络的输出尺寸和输入尺寸是一样的（如：为每个输入像素分配类别标签），则无需做额外的工作。
    - 如果要求网络的输出尺寸是固定的（如：为整个图像分配一个类别标签），此时需要插入一个池化层：池化区域的大小要和输入的大小成比例，从而保持固定数量的池化输出。
3. 卷积能处理可变大小的输入，但这种“可变”必须是因为同一个事物在同一个角度下的、不同数量的观察导致的。

如：时间角度下的、数量上的不同观察导致时间维度可变，空间角度下的、数量上的不同观察导致空间维度可变。

这种可变并不包括特征数量（即：多少个观察角度）的可变。

如：某个样本具有“年龄、学历、性别”特征，另一个样本只具有“年龄、学历”特征。则卷积对于这种类型的数据集无能为力。

## 4.3 高效的卷积算法

1. 设计更快的卷积、或者近似卷积而不降低模型准确率的方法是一个活跃的研究领域。

甚至仅提高前向传播效率的技术也是有用的。因为在商业环境中，通常对模型的推断有性能要求或者限制。

### 4.3.1 傅里叶变换

1. 卷积等效于：使用傅里叶变换将输入和核都转换到频域，然后在频域将输入和核进行逐点相乘，最后把相乘的结果使用傅里叶逆变换转换回时域。

对于某些规模的问题，这种算法可能比直接计算离散卷积效率更高。

### 4.3.2 可分离卷积

1. 对于一个  $d$  维的核矩阵，如果可以表示成  $d$  个一维向量的外积时，称该核是可分离的。

这里的外积不同于代数意义上的叉乘，而是：

$$\vec{a} \otimes \vec{b} = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_M \end{bmatrix} \otimes \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_N \end{bmatrix} = \begin{bmatrix} a_1 b_1 & a_1 b_2 & \cdots & a_1 b_N \\ a_2 b_1 & a_2 b_2 & \cdots & a_2 b_N \\ \vdots & \vdots & \ddots & \vdots \\ a_M b_1 & a_M b_2 & \cdots & a_M b_N \end{bmatrix}$$

2. 当核  $\mathbf{K}$  是可分离时，假设有： $\mathbf{K} = \vec{\mathbf{k}}_1 \otimes \vec{\mathbf{k}}_2 \cdots \otimes \vec{\mathbf{k}}_d$ ,  $\vec{\mathbf{k}}_i \in \mathbb{R}^w$ 。

- 直接使用  $\mathbf{K}$  进行卷积运算是非常低效的。它等价于连续的对  $d$  个一维向量  $\vec{\mathbf{k}}_i$  执行卷积。  
由于  $\mathbf{K} \in \mathbb{R}^{w \times w \times \cdots \times w}$ ，因此直接计算卷积需要  $O(w^d)$  个参数，需要  $O(w^d)$  的运行时间和存储空间。
- 如果使用可分离卷积，则只需要连续的对  $d$  个一维向量  $\vec{\mathbf{k}}_i$  执行卷积。  
此时需要  $O(w \times d)$  个参数，只需要  $O(w \times d)$  的运行时间和存储空间。
- 但遗憾的是：并不是每个核都是可分离的。

## 4.4 非监督的特征

1. 通常卷积神经网络训练中代价最高的是学习卷积核，而输出层的学习代价相对较低。
  - 因为卷积核的输入单元相对较多，而且卷积神经网络中可能需要学习多个卷积核。
  - 经过了若干个池化层之后，输出层的输入单元的数量要小的多。
2. 每个卷积核都可以提取某个特征，因此学习卷积核就是要学习特征。
3. 降低卷积神经网络训练成本的方法是：使用那些非监督方式训练得到特征。
4. 有三种基本策略来避免监督训练而得到特征：
  - 简单地随机初始化特征。
  - 人工设计特征。如：人工设计一个检测图像边缘的卷积核。
  - 用无监督训练来学习特征。

### 4.5.1 随机初始化特征

1. 随机初始化特征经常在卷积网络中表现的出乎意料的好。
2. 随机初始化特征训练卷积神经网络的步骤是：
  - 给出多个随机权重，生成一组候选的卷积核（这些卷积核可以是不同尺寸的）。
  - 仅仅训练输出层来评估这一组候选卷积核的性能，挑选表现最好的那个卷积核。  
训练过程中，卷积核的权重固定，不会被调整。
  - 使用表现最好的那个卷积核的结构和权重，并重新训练整个网络。  
训练过程中，卷积核的权重会被调整。

### 4.5.2 无监督学习特征

1. 使用无监督学习特征来训练卷积神经网络时，允许其结构与输出层相分离。

其步骤是：

- 使用无监督学习特征。
- 提取训练集的全部特征，构建一个新的训练集。
- 将这个新的训练集作为输出层的输入，训练一个简单的神经网络（可能只有一个输出层，也可能添加一层隐层）。

2. 无监督学习特征可以使用一些特殊的方法来学习，它不需要在每个梯度步骤中都完整的前向和反向传播。

如：逐层贪心预训练。逐层贪心预训练的经典模型是卷积深度信念网络。

3. 使用无监督学习特征来训练卷积神经网络时，训练过程中可以完全不使用卷积。

通过该方法可以训练非常大的模型，并且只在前向传播期间产生高计算成本（反向传播阶段计算成本较低，因为大量的参数并不参与训练）。

4. 当前大多数卷积神经网络以纯粹有监督的方式训练，因为计算能力比以前大幅度提升。

有监督的方式训练的预测能力更强。

5. 无监督学习特征的优点难以说清：

- 使用无监督学习特征可以提供一些相对于监督训练的正则化。
- 使用无监督学习特征可以训练更大的网络结构，因为它的学习方式减少了计算成本。

## 五、历史和现状

### 5.1 历史

1. 卷积神经网络是第一个解决重要商业应用的神经网络。

2. 卷积神经网络是用反向传播训练的第一个有效的深度神经网络之一。

3. 卷积神经网络提供了一种方法来特化神经网络，从而处理具有网格结构拓扑的数据。

这种方法在二维图像上是最成功的。

为了处理一维序列数据，往往采用另一种强大的特化网络：循环神经网络。

### 5.2 神经科学基础

1. 图像传输到大脑的流程可以简化为：

- 图像从光到达眼睛并刺激视网膜。
- 视网膜中的神经元对图像进行一些简单的预处理，但是基本不改变图像的表达方式。
- 图像通过视神经，以及称作外侧膝状体的脑部区域。
- 然后大脑的 V1 部分（也称作主要视觉皮层）开始处理图像。

2. 神经生理学家 David Hubel 和 Torsten Wiesel 观察了猫的脑内神经元的视觉响应发现：处于视觉系统较为前面的神经元对于特定的光模式反应最强烈，但是对于其它光模式几乎完全没有反应。

3. 卷积层被设计为描述 V1 的三个性质：

- V1 分布在空间中。

它实际上具有二维结构来映射视网膜中的图像，视网膜下半部的光仅仅影响 V1 相应的一半。

卷积网络通过用二维映射定义特征的方式来描述该特性。

- V1 包含许多简单细胞。

这些简单细胞的行为简单概括为：是在一个空间上小的、局部的接受域内的图像的线性函数。

卷积网络的卷积单元被设计为模拟简单细胞。

- V1 还包括许多复杂细胞。

- 复杂细胞对于特征的位置的微小偏移具有不变性。

这通过卷积网络的最大池化单元来刻画。

- 复杂细胞对于照明中的一些变化也是不变的。

它不能简单地通过在空间位置上池化来刻画。它也激发了卷积网络中的一些跨通道池化策略，如 maxout 单元。

4. 一般认为：类似于 V1 的原理也适用于视觉系统的其他区域。

5. 在大脑中人们找到了响应一些特定概念的细胞，并且这种细胞对于输入的许多种变换都具有不变性。

这些细胞被称作祖母细胞，存在于内侧颞叶的区域。

一个人可能有这样的一个神经元，当他看到祖母的照片时，该神经元被激活。无论祖母出现在照片的哪个位置、无论是祖母的脸部还是全身、无论是光亮还是黑暗。

6. 与卷积网络最后一层最接近的类比是：颞下皮质的脑区。

- 当查看一个对象时，信息从视网膜经过 LGN 流到 V1，然后到 V2，V4，然后是颞下皮质。  
这发生在瞥见对象的前 100ms 内。
- 如果允许一个人继续观察对象更多的时间，那么信息将开始向后流动（即前面过程的反馈路径）。  
因为大脑使用自上而下的反馈来更新较低级脑区中的激活。
- 如果打断人的注视，并且只观察前 100ms 内的大多数前向传播路径，则颞下皮质与卷积网络的最后一层非常相似。

7. 动物的视觉系统与卷积网络的主要区别：

- 人眼大部分是非常低的分辨率，除了一个被称作中央凹的小块（手臂远的拇指大小的区域）。而大多数卷积网络实际上接收到的是一张高分辨率的照片。
  - 虽然人们觉得可以看到高分辨率的整个场景，但是这是大脑的潜意识创建的错觉。因为大脑缝合了人们瞥见的若干个小区域。
  - 人类大脑控制几次眼动（称作扫视），从而瞥见场景中最显眼的或者任务相关的部分。这称作注意力机制。  
目前注意力机制在自然语言处理中大获成功。
- 人类视觉系统集成了许多其他感觉（如听觉，以及心情想法之类的因素），而卷积网络目前为止纯粹是视觉的。
- 人类视觉系统不仅用于识别对象，它还能够理解整个场景：包括多个对象、对象之间的关系、人们的身体与世界交互所需要的丰富的三维几何信息。而卷积神经网络在这些问题上还是起步阶段。
- 即使像 V1 这样简单的大脑区域也受到来自较高级别的反馈的严重影响。虽然神经网络模型也探索反馈机制，但是目前没有提供引人瞩目的改进。
- 大脑可能使用非常不同的激活函数、卷积函数、池化函数。单个神经元的激活可能并不能通过单个线性过滤器的响应来很好的表征。