智能之门

神经网络和深度学习入门

（基于Python的实现）

STEP 8 卷积神经网络

第 17 章

卷积神经网络原理

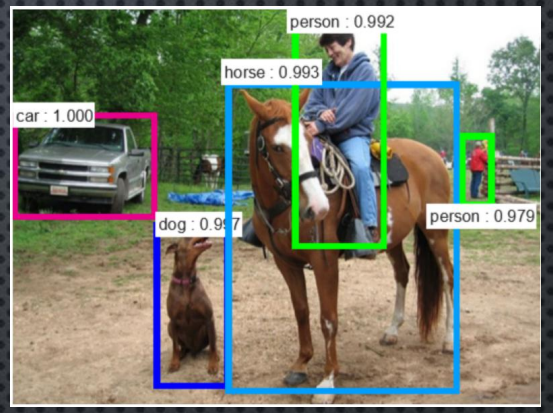
卷积神经网络是深度学习中的一个里程碑式的技术，有了这个技术，才会让计算机有能力理解图片和视频信息，才会有计算机视觉的众多应用。

在本部分的学习中，我们将会逐步介绍卷积的前向计算、卷积的反向传播、池化的前向计算与反向传播，然后用代码实现一个卷积网络并训练一些实际数据。

17.1 卷积神经网络概述

卷积神经网络是神经网络的类型之一，在图像识别和分类领域中取得了非常好的效果，比如识别人脸、物体、交通标识等，这就为机器人、自动驾驶等应用提供了坚实的技术基础。

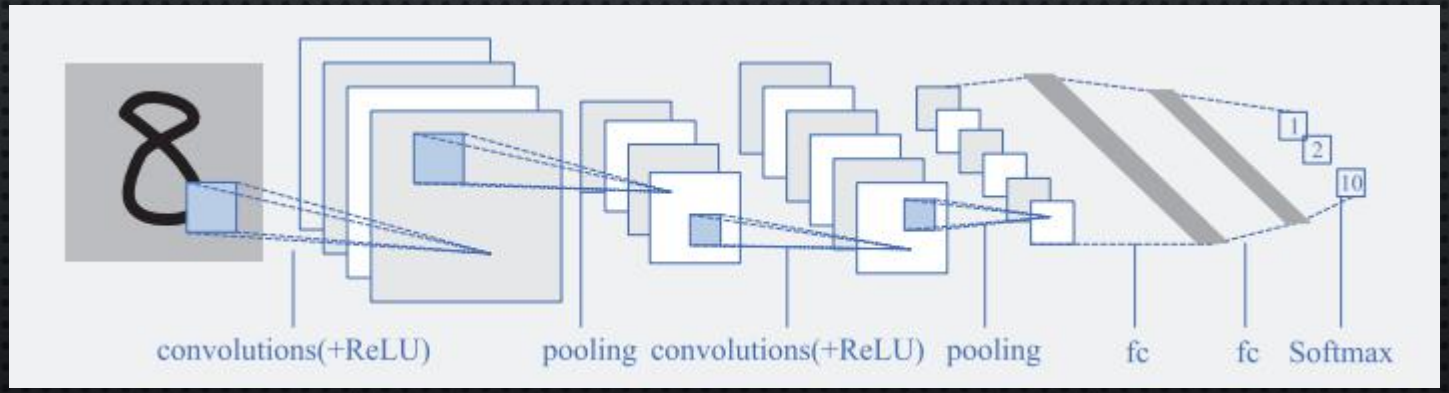
卷积神经网络可以识别出上面两张图中的物体和场景。当然，识别物体和给出简要的场景描述是两套系统配合才能完成的任务，第一个系统只负责识别，第二个系统可以根据第一个系统的输出形成摘要文字。



17.1 卷积神经网络概述

➢ 卷积神经网络的典型结构

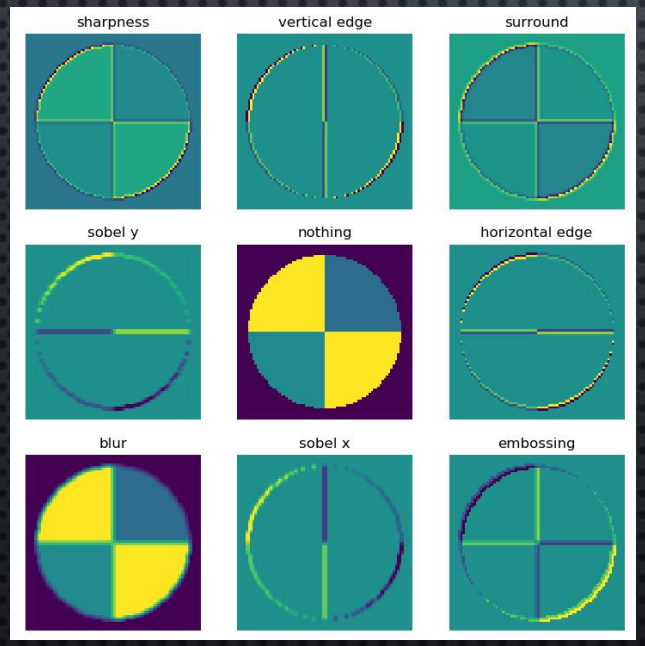
- 在一个典型的卷积神经网络中，会至少包含以下几个层：
  - ✓ 卷积层
  - ✓ 激活函数层
  - ✓ 池化层
  - ✓ 全连接分类层



17.1 卷积神经网络概述

➢ 卷积核

- 卷积网络之所以能工作，完全是卷积核的功劳。



	1	2	3
1	0,-1, 0 -1, 5,-1 0,-1, 0	0, 0, 0 -1, 2,-1 0, 0, 0	1, 1, 1 1,-9, 1 1, 1, 1
	sharpness	vertical edge	surround
2	-1,-2, -1 0, 0, 0 1, 2, 1	0, 0, 0 0, 1, 0 0, 0, 0	0,-1, 0 0, 2, 0 0,-1, 0
	sobel y	nothing	horizontal edge
3	0.11,0.11,0.11 0.11,0.11,0.11 0.11,0.11,0.11	-1, 0, 1 -2, 0, 2 -1, 0, 1	2, 0, 0 0,-1, 0 0, 0,-1
	blur	sobel x	embossing

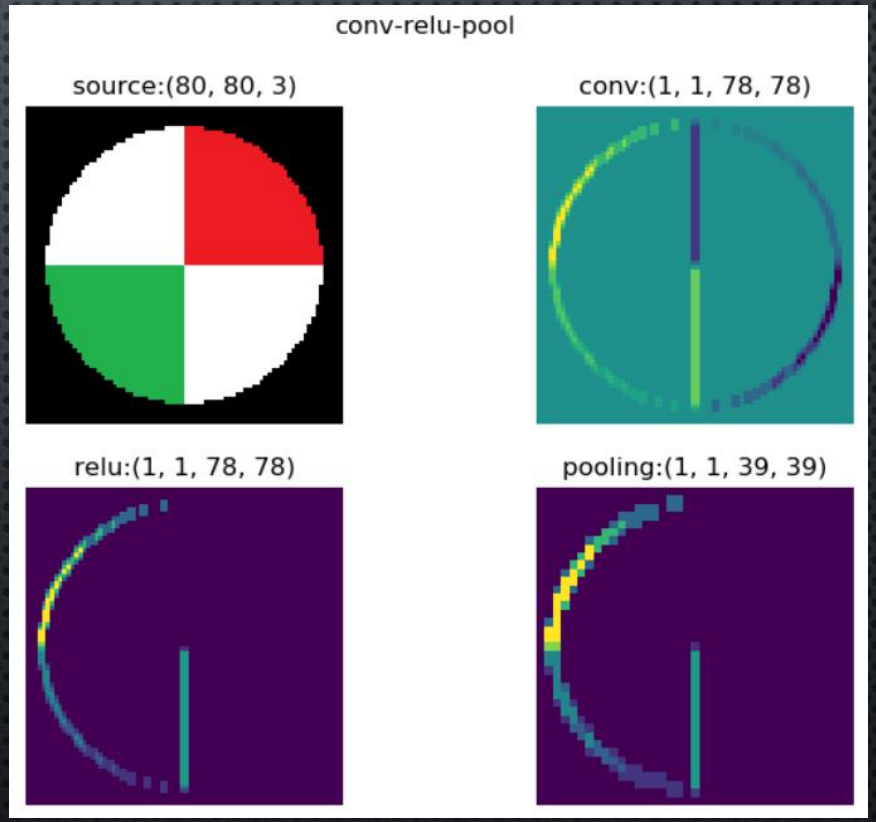
17.1 卷积神经网络概述

- 各个卷积核的作用

序号	名称	说明
1	锐化	如果一个像素点比周围像素点亮，则此算子会令其更亮
2	检测竖边	检测出了十字线中的竖线，由于是左侧和右侧分别检查一次，所以得到两条颜色不一样的竖线
3	周边	把周边增强，把同色的区域变弱，形成大色块
4	Sobel-Y	纵向亮度差分可以检测出横边，与横边检测不同的是，它可以使得两条横线具有相同的颜色，具有分割线的效果
5	Identity	中心为1四周为0的过滤器，卷积后与原图相同
6	横边检测	检测出了十字线中的横线，由于是上侧和下侧分别检查一次，所以得到两条颜色不一样的横线
7	模糊	通过把周围的点做平均值计算而"杀富济贫"造成模糊效果
8	Sobel-X	横向亮度差分可以检测出竖边，与竖边检测不同的是，它可以使得两条竖线具有相同的颜色，具有分割线的效果
9	浮雕	形成大理石浮雕般的效果

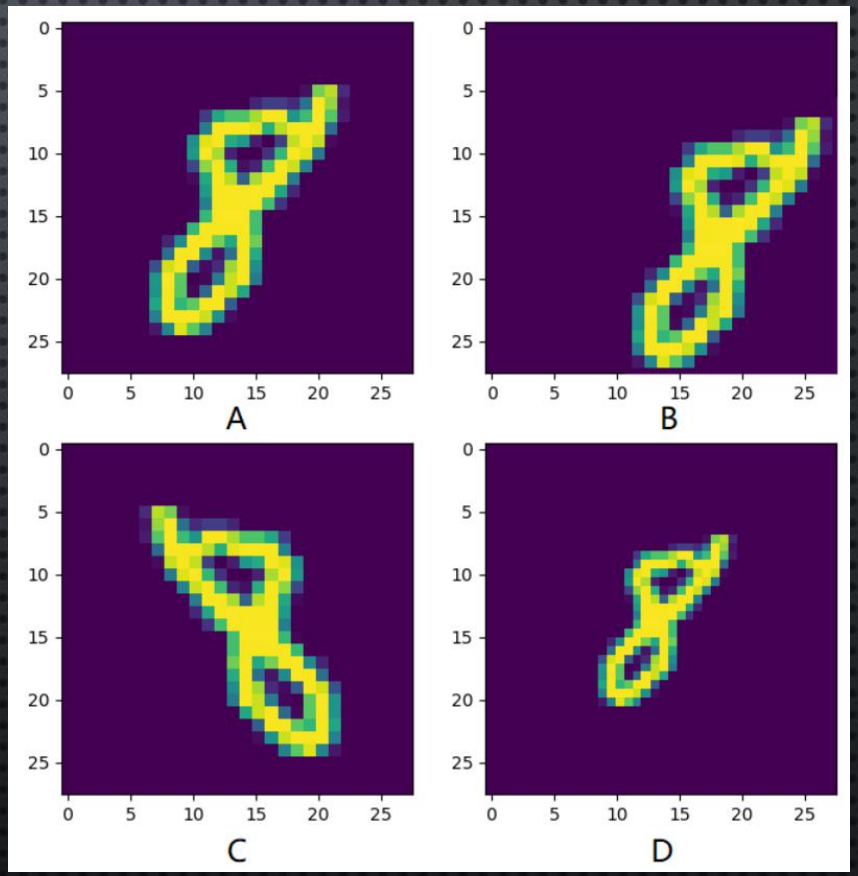
17.1 卷积神经网络概述

- **卷积的后续运算**
  - 四个子图展示如下结果：
    - ✓ 原图
    - ✓ 卷积结果
    - ✓ 激活结果
    - ✓ 池化结果



17.1 卷积神经网络概述

- **卷积神经网络的学习**
  - 平移不变性
  - 旋转不变性
  - 尺度不变性



17.2 卷积的前向计算

➢ 卷积的数学定义

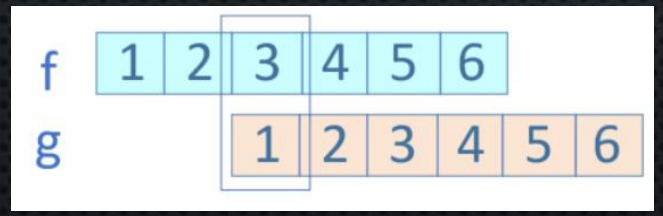
- 连续型定义

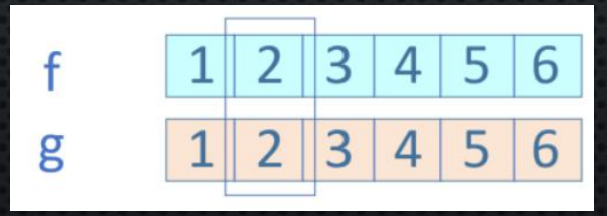
$$h(x) = (f * g)(x) = \int_{-\infty}^{\infty} f(t)g(x-t)dt$$

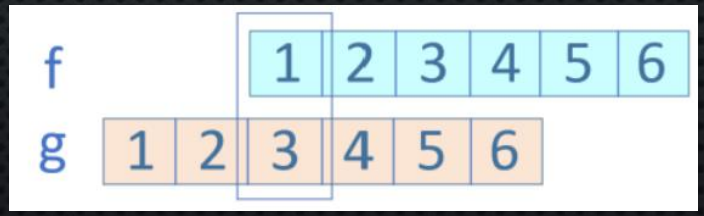
- 离散型定义

$$h(x) = (f * g)(x) = \sum_{t=-\infty}^{\infty} f(t)g(x-t)$$

➢ 一维卷积实例







17.2 卷积的前向计算

➢ 单入单出的二维卷积

- 记图像为I，卷积核为K，则目标图像的第(i,j)个像素的卷积为

$$h(i,j)=(I*K)(x)=\sum_m\sum_n I(m,n)K(i-m,j-n)=\sum_m\sum_n I(i-m,j-n)K(m,n)$$

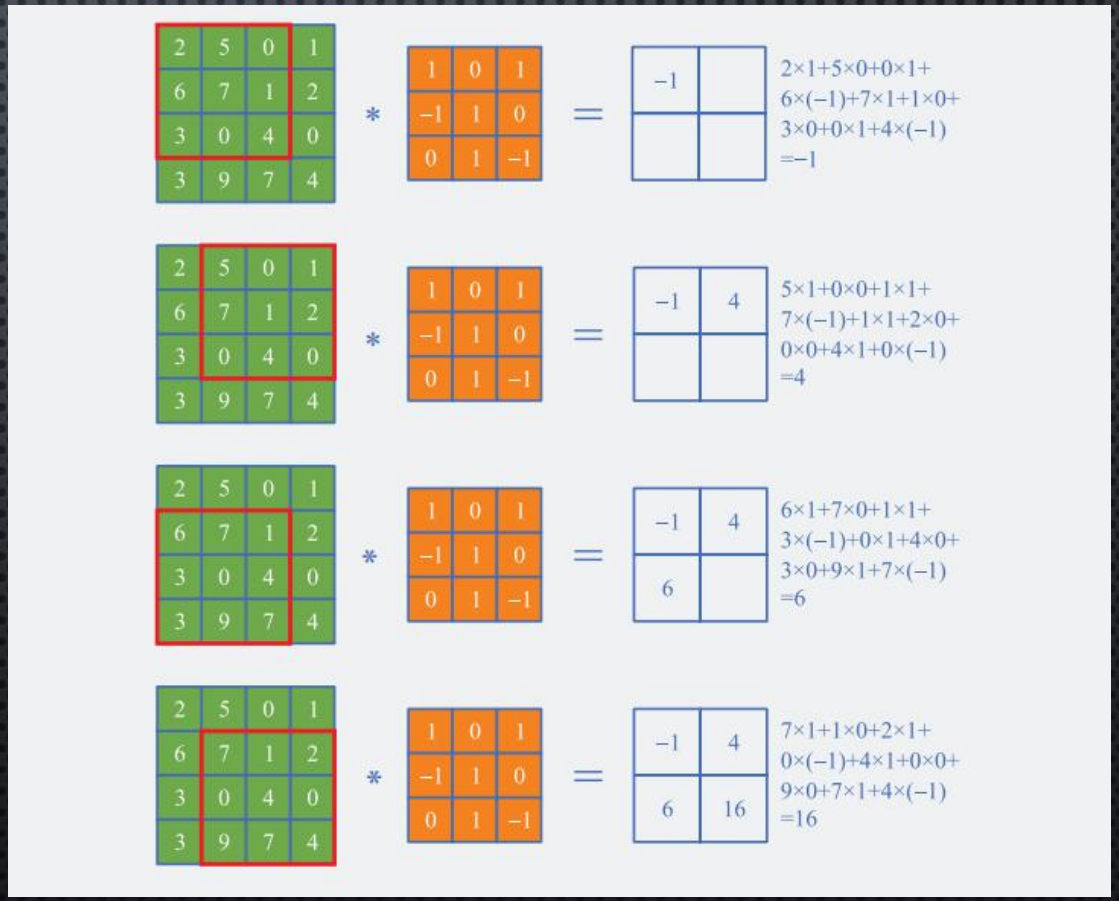
- 不翻转卷积核的互相关函数

$$h(i,j)=(I*K)(x)=\sum_m\sum_n I(i+m,j+n)K(m,n)$$

- 互相关函数的运算，是两个序列滑动相乘，两个序列都不翻转。卷积运算也是滑动相乘，但是其中一个序列需要先翻转，再相乘。所以，从数学意义上说，机器学习实现的是互相关函数，而不是原始含义上的卷积。我们实现的卷积操作不是原始数学含义的卷积，而是工程上的卷积，可以简称为卷积。在实现卷积操作时，并不会反转卷积核。

17.2 卷积的前向计算

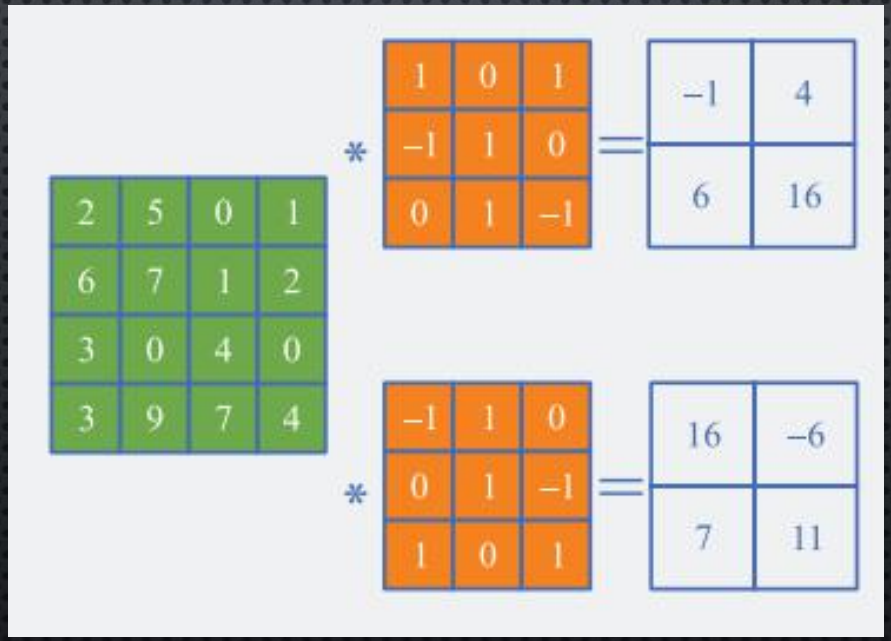
- 卷积运算的过程



17.2 卷积的前向计算

➢ 单入多出的升维卷积

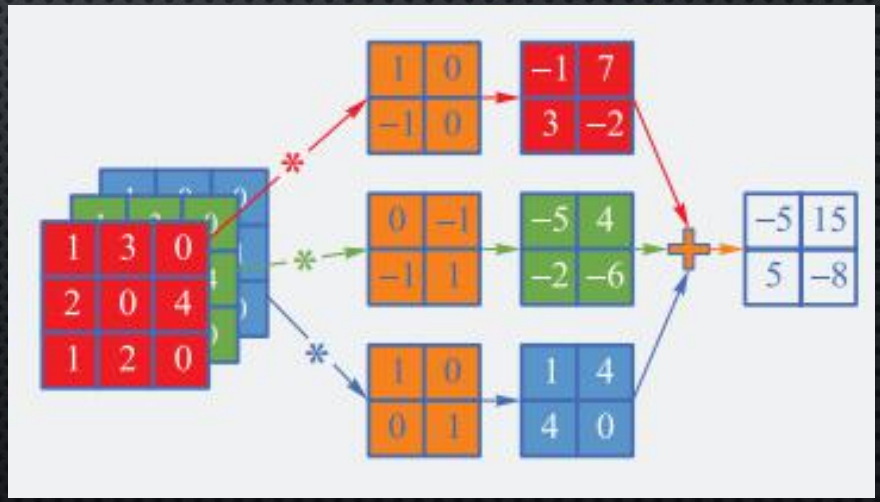
- 原始输入是一维的图片，但是我们可以用多个卷积核分别对其计算，得到多个特征输出。



17.2 卷积的前向计算

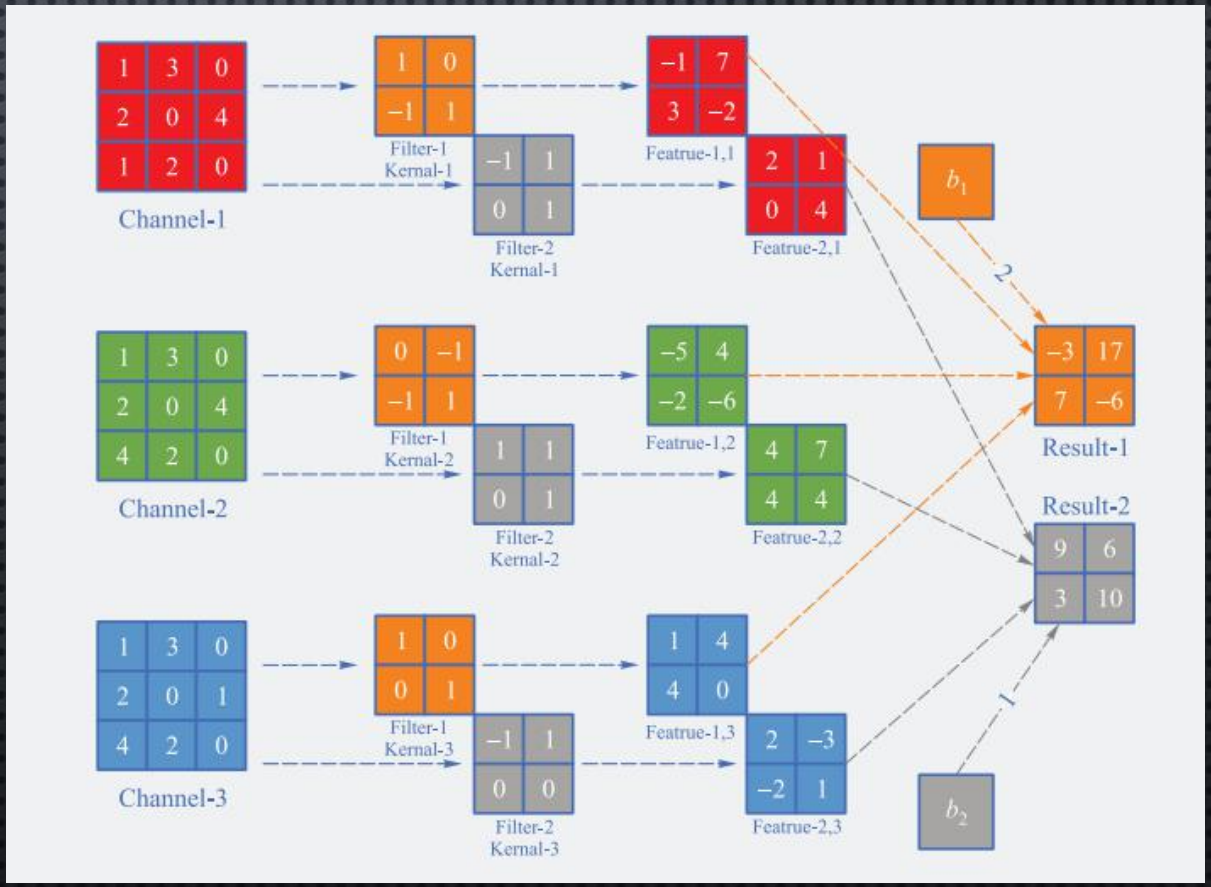
➢ 多入单出的降维卷积

- 一张图片，通常是彩色的，具有红绿蓝三个通道。我们可以有两个选择来处理：
  - ✓ 变成灰度的，每个像素只剩下一个值，就可以用二维卷积
  - ✓ 对于三个通道，每个通道都使用一个卷积核，分别处理红绿蓝三种颜色的信息
- 显然第2种方法可以从图中学习到更多的特征，于是出现了三维卷积，即有三个卷积核分别对应三个通道，三个子核的尺寸是一样的。
- 对三个通道各自做卷积后，得到右侧的三张特征图，然后再按照原始值不加权地相加在一起，得到最右侧的白色特征图。



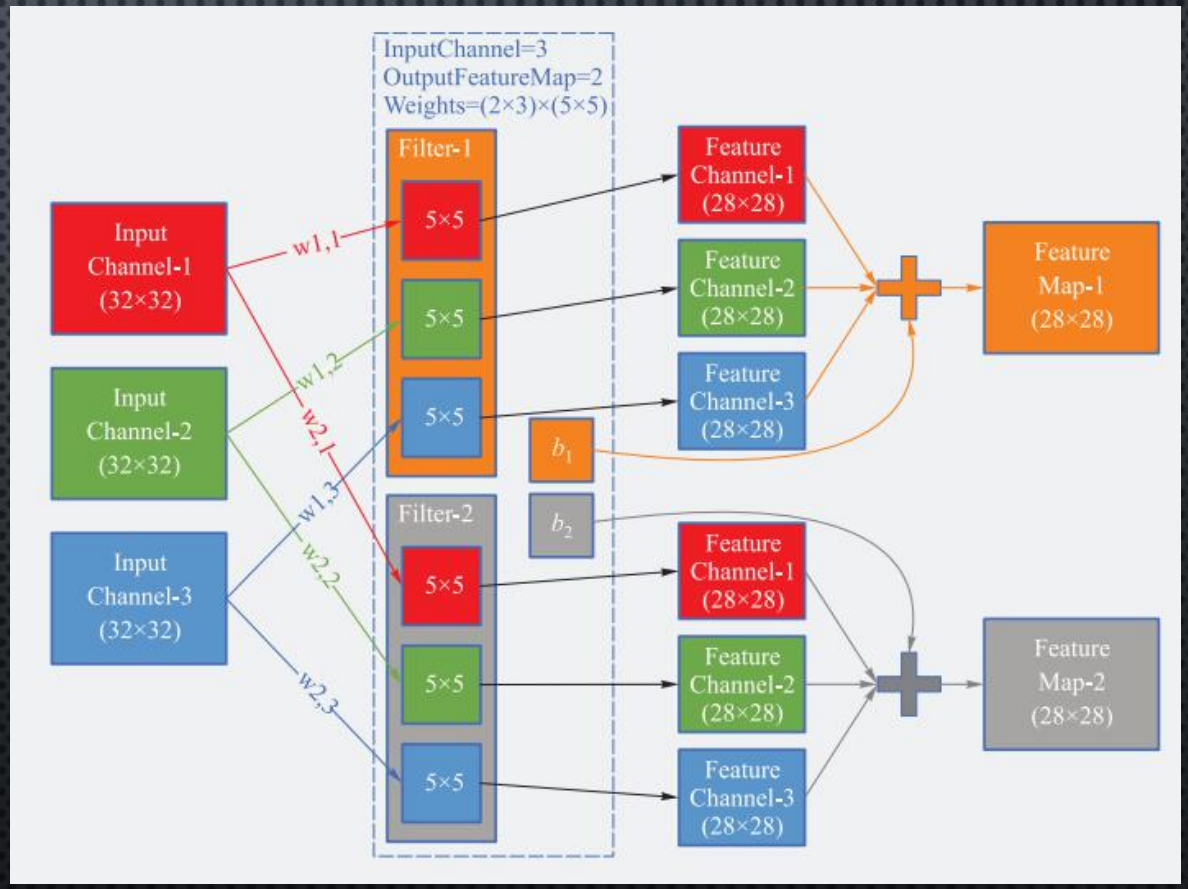
17.2 卷积的前向计算

- **多入多出的同维卷积**
  - 假设有一张 3 × 3 的彩色图片，如果有两组 3 × 2 × 2 的卷积核，则卷积计算过程如右图所示。



17.2 卷积的前向计算

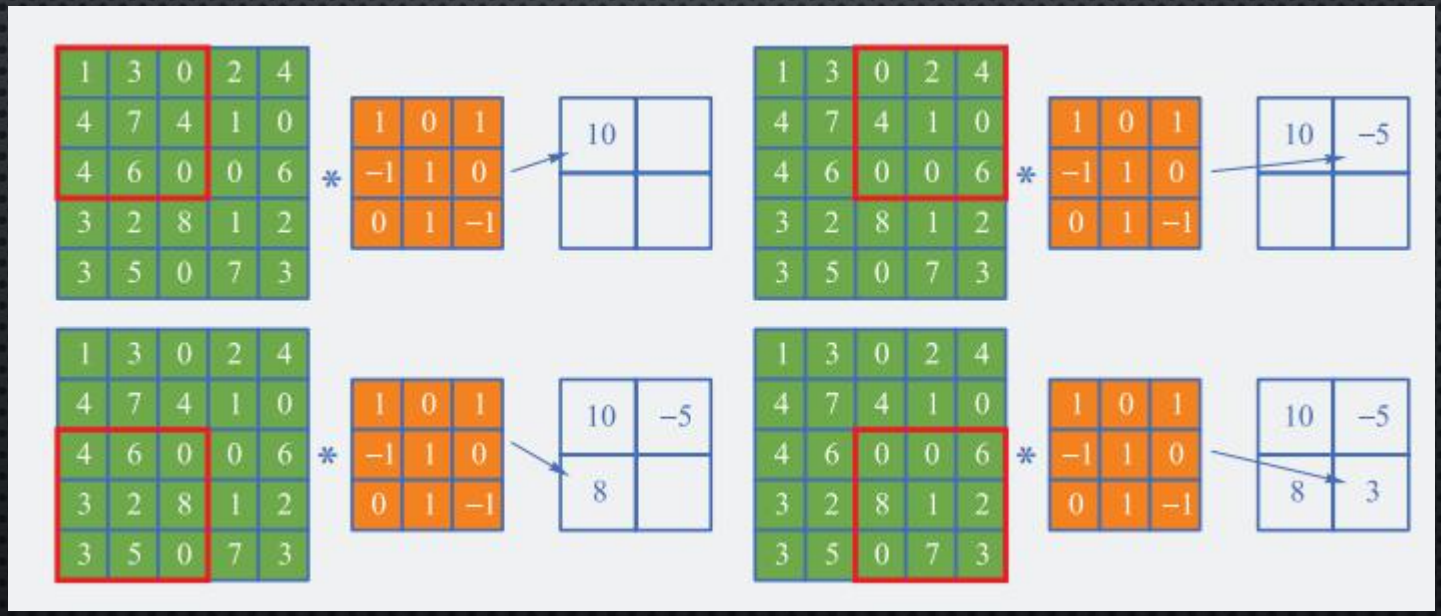
- **卷积编程模型**
  - 输入 Input Channel
  - 卷积核组 Weights,Bias
  - 过滤器 Filter
  - 卷积核 Kernel
  - 输出 Feature Map



17.2 卷积的前向计算

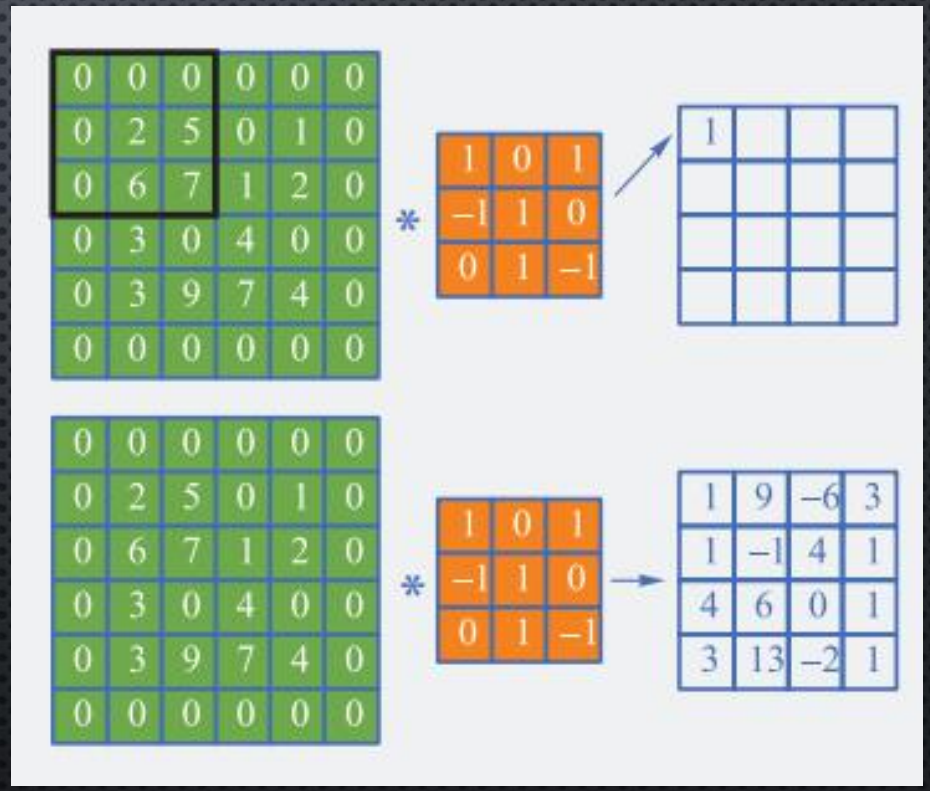
➢ 步长 Stride

- 下图的卷积操作中，卷积核每次向右或向下移动两个单元，即 stride = 2。在后续的步骤中，由于每次移动两格，所以最终得到一个 2×2 的图片。



17.2 卷积的前向计算

- **填充 Padding**
  - 如果原始图为4×4，用3×3的卷积核进行卷积后，目标图片变成了2×2。如果我们想保持目标图片和原始图片为同样大小，一般我们会向原始图片周围填充一圈0，然后再做卷积。



17.2 卷积的前向计算

➢ 输出结果

- 综合以上所有情况，可以得到卷积后的输出图片的大小的公式：

$$H_{Output} = \frac{H_{Input} - H_{Kernal} + 2 \cdot Padding}{Stride} + 1$$

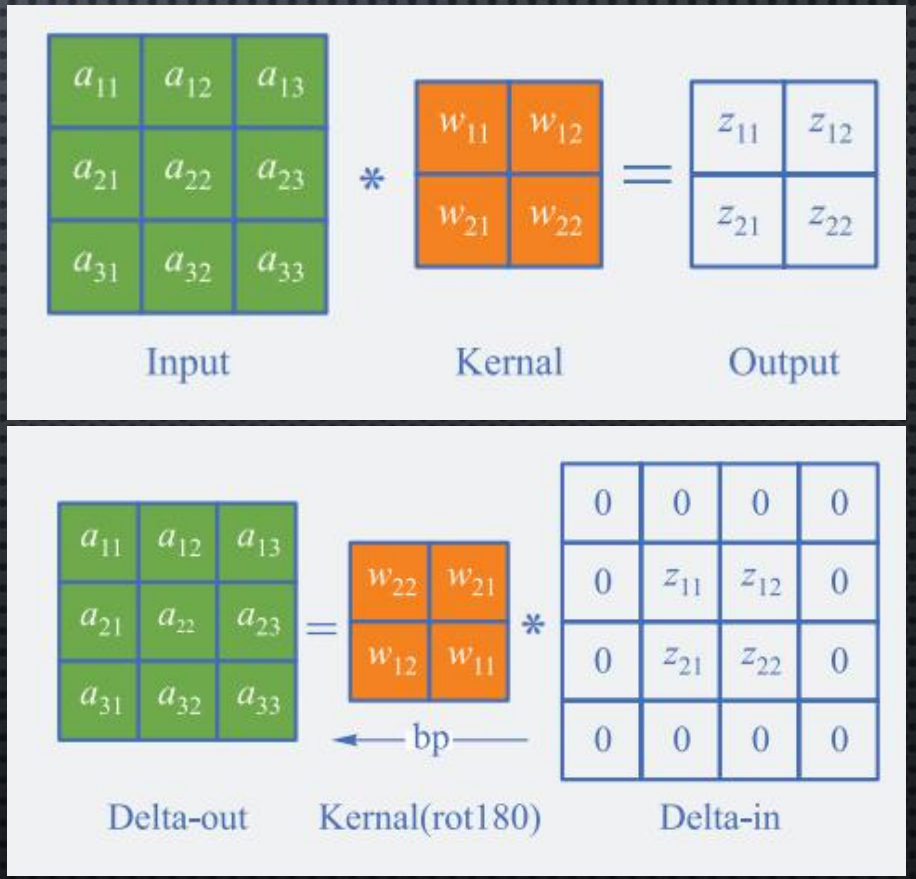
$$W_{Output} = \frac{W_{Input} - W_{Kernal} + 2 \cdot Padding}{Stride} + 1$$

- 一般情况下，我们用正方形的卷积核，且为奇数。
- 如果计算出的输出图片尺寸为小数，则取整，不做四舍五入。

17.3 卷积层的训练

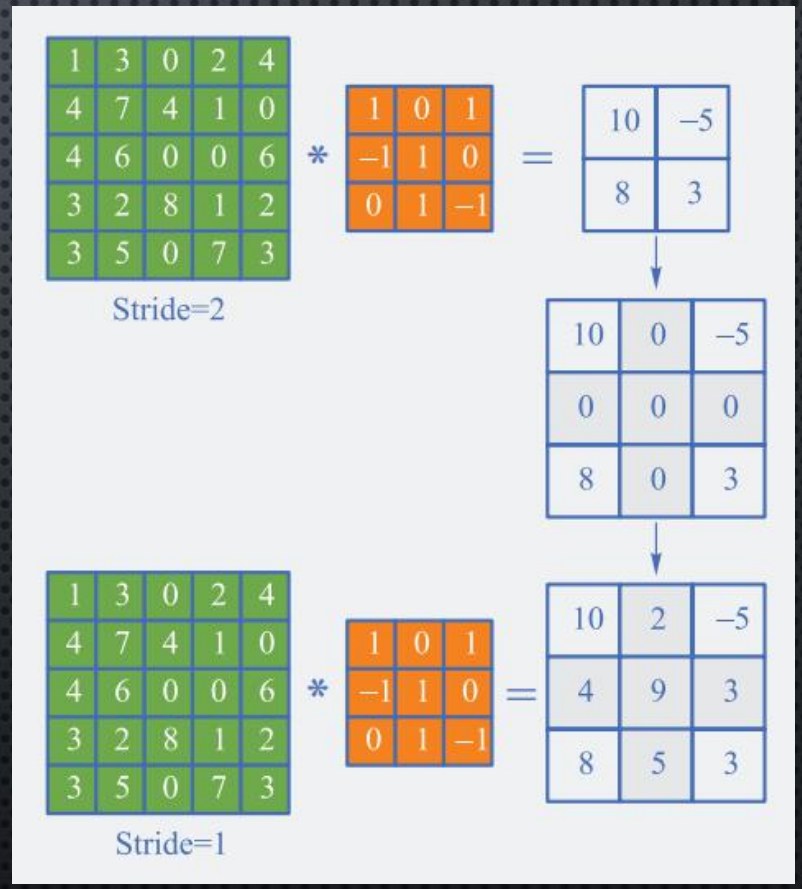
➢ 计算反向传播梯度矩阵

- 正向公式：$Z = W * A + b$
- 反向传播：$\delta_{out} = \delta_{in} * W^{rot180}$
  - ✓ 把传入的误差矩阵 δ_{in} 做一个zero padding，再乘以旋转180度的卷积核，就是要传出的误差矩阵δ_{out}。
  - ✓ 当Weights是$N \times N$时， δ_{in}需要padding=$N-1$，即加$N-1$圈0。



17.3 卷积层的训练

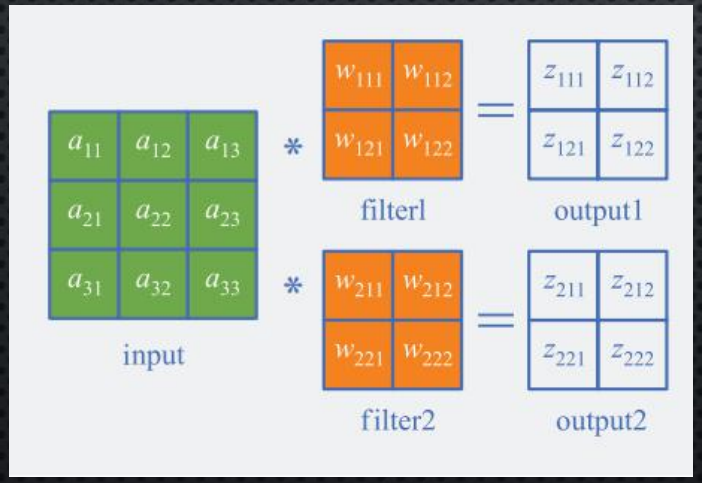
- **步长不为1时的梯度矩阵还原**
  - 右图为步长为2和1时，卷积结果的差异。
  - 当前的卷积层步长为 S （$S>1$）时：
    - ✓ 得到从上层回传的误差矩阵形状，假设为$M \times N$；
    - ✓ 初始化一个$(M \cdot S) \times (N \cdot S)$ 的零矩阵；
    - ✓ 把传入的误差矩阵的第一行值放到零矩阵第0行的 $0, S, 2S, 3S, \ldots$ 位置；
    - ✓ 然后把误差矩阵的第二行的值放到零矩阵第S行的 $0, S, 2S, 3S, \ldots$位置……



17.3 卷积层的训练

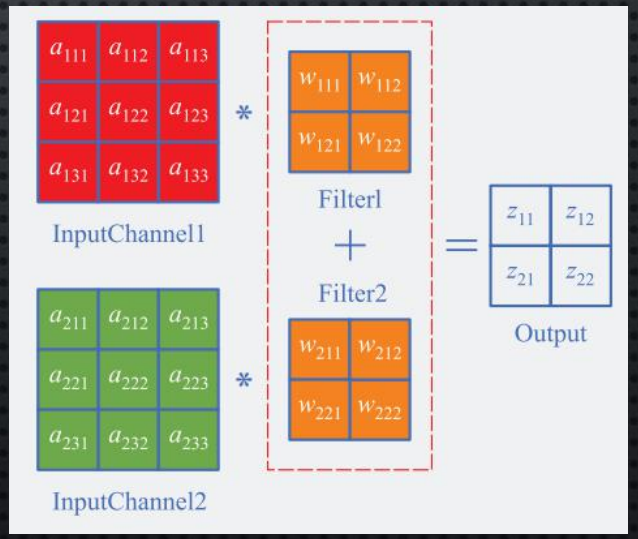
➢ 有多个卷积核时的梯度计算

- 有多个卷积核也就意味着有多个输出通道。
- 梯度公式： $\delta_{out} = \sum_m \delta_{in_m} * W_m^{rot180}$



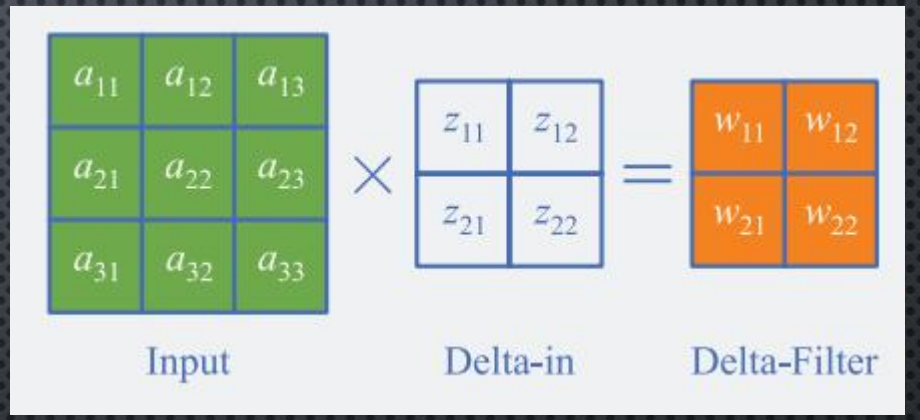
➢ 有多个输入时的梯度计算

- 当输入层是多个图层时，每个图层必须对应一个卷积核。
- 梯度公式： $\delta_{out_k} = \delta_{in} * W_k^{rot180}$



17.3 卷积层的训练

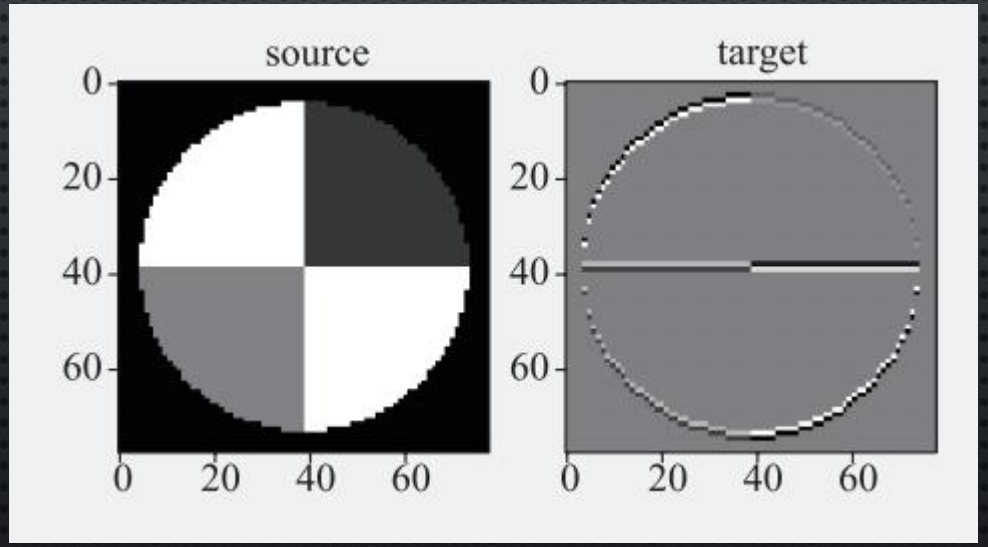
- **权重（卷积核）梯度计算**
  - 梯度公式： $\delta_w = A * \delta_{in}$
- **偏移的梯度计算**
  - 梯度公式： $\delta_b = \delta_{in}$



17.3 卷积层的训练

➢ 实例说明

- 先制作一张样本图片，然后使用“横边检测”算子卷积核对该样本进行卷积，得到对比图。
- 由于算子是横边检测，所以只保留了原始图片中的横边。



17.3 卷积层的训练

➢ 直线拟合与图像拟合的对比

- 直线拟合中的均方差，是计算预测值与样本点之间的距离；图片拟合中的均方差，可以直接计算两张图片对应的像素点之间的差值。
- 若令$b = 0$，求卷积核的梯度，则得到：

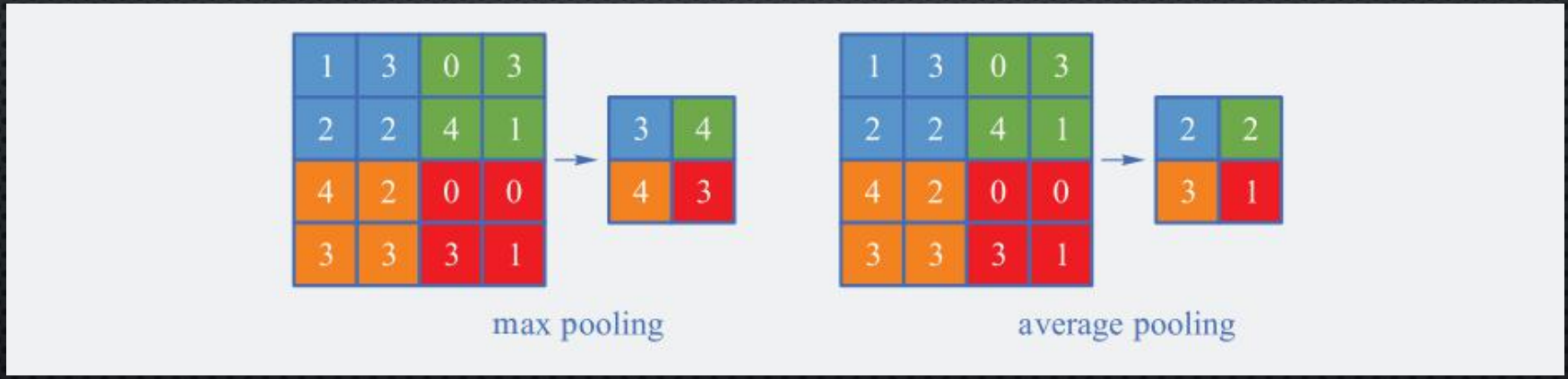
$$\frac{\partial loss}{\partial w} = x * (z - y)$$

项目 / 拟合方式	样本数据	标签数据	预测数据	公式	损失函数
直线拟合	样本点 *x*	标签值 *y*	预测直线 *z*	*Z=x×w+b*	均方差
图片拟合	原始图片 *x*	目标图片 *y*	预测图片 *z*	*Z=x×w+b*	均方差

17.4 池化层

➢ 池化 pooling

- 池化又称为下采样，downstream sampling or sub-sampling。池化方法分为两种，一种是最大值池化 Max Pooling，一种是平均值池化 Mean/Average Pooling。
- 最大值池化，是取当前池化视野中所有元素的最大值，输出到下一层特征图中。
- 平均值池化，是取当前池化视野中所有元素的平均值，输出到下一层特征图中。



17.4 池化层

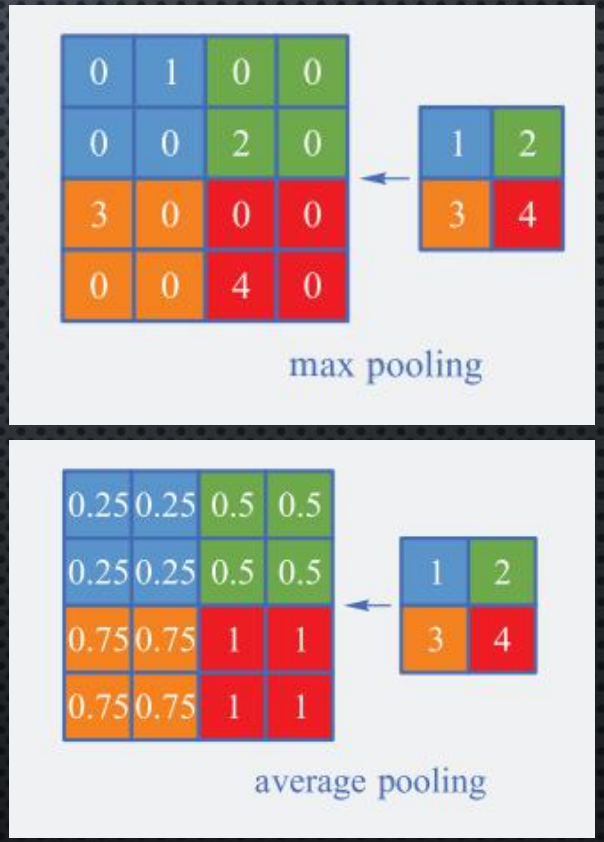
- 目的：
  - ✓ 扩大视野：就如同先从近处看一张图片，然后离远一些再看同一张图片，有些细节就会被忽略。
  - ✓ 降维：在保留图片局部特征的前提下，使得图片更小，更易于计算。
  - ✓ 平移不变性，轻微扰动不会影响输出。
  - ✓ 维持同尺寸图片，便于后端处理：假设输入的图片不是一样大小的，就需要用池化来转换成同尺寸图片。
- 池化的其它方式
  - ✓ 假设输入图片的形状是 $W \times H \times D$，其中 W 是图片宽度，H 是图片高度，D 是图片深度（多个图层），F 是池化的视野（正方形），S是池化的步长，则输出图片的形状是：

$$W_2 = \frac{W_1 - F}{S} + 1, \qquad H_2 = \frac{H_1 - F}{S} + 1, \qquad D_2 = D_1$$

17.4 池化层

➢ 池化层的训练

- 对于最大值池化，残差值会回传到当初最大值的位置上，而其它三个位置的残差都是0。
- 对于平均值池化，残差值会平均到原始的各个位置上。
- 无论是最大值池化还是平均值池化，都没有要学习的参数，所以在卷积网络的训练中，池化层需要做的只是把误差项向后传递，不需要计算任何梯度。



THE END

谢谢！