深度学习入门

基于Python的理论与实现

Stephen CUI

March 14, 2023

Chapter 1

卷积神经网络

本章的主题是卷积神经网络（Convolutional Neural Network，CNN）。 CNN被用于图像识别、语音识别等各种场合。

1.1 整体结构

CNN和之前介绍的神经网络一样，可以像乐高积木一样通过组装层来构建。不过， CNN中新出现了卷积层（Convolution层）和池化层（Pooling层）。

之前介绍的神经网络中，相邻层的所有神经元之间都有连接，这称为**全连接**（fully-connected）。另外，我们用Affine层实现了全连接层。如果使用这个Affine层，一个5层的全连接的神经网络就可以通过 Figure 1.1所示的网络结构来实现。

如 Figure 1.2 所示CNN 中新增了 Convolution 层和 Pooling 层。CNN 的层的连接顺序是“Convolution - ReLU -（Pooling）”（Pooling 层有时会被省略）。这可以理解为之前的“Affi ne - ReLU”连接被替换成了“Convolution - ReLU -（Pooling）”连接。

还需要注意的是，靠近输出的层中使用了之前的“Affi ne - ReLU”组合。此外，最后的输出层中使用了之前的“Affi ne - Softmax”组合。这些都是一般的CNN中比较常见的结构。

1.2 卷积层

CNN中出现了一些特有的术语，比如填充、步幅等。此外，各层中传递的数据是有形状的数据（比如，3维数据）。

1.2.1 全连接层存在的问题

之前介绍的全连接的神经网络中使用了全连接层（Affine层）。在全连接层中，相邻层的神经元全部连接在一起，输出的数量可以任意决定。

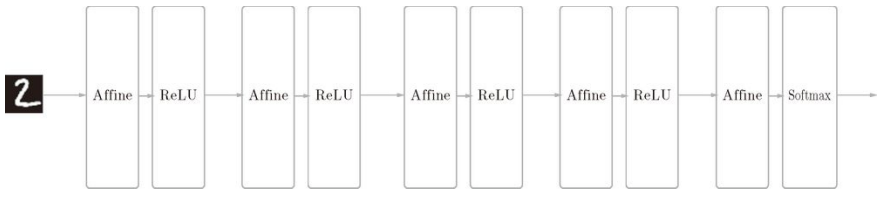

Figure 1.1: An example of a network based on a fully connected layer (Affine layer)

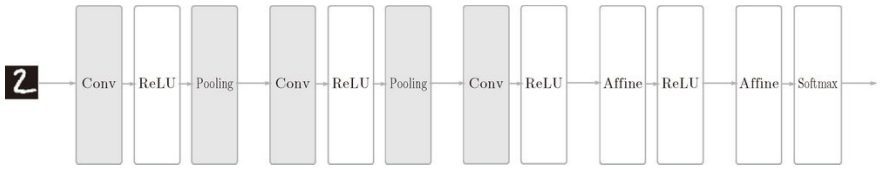

Figure 1.2: Examples of CNN-based networks

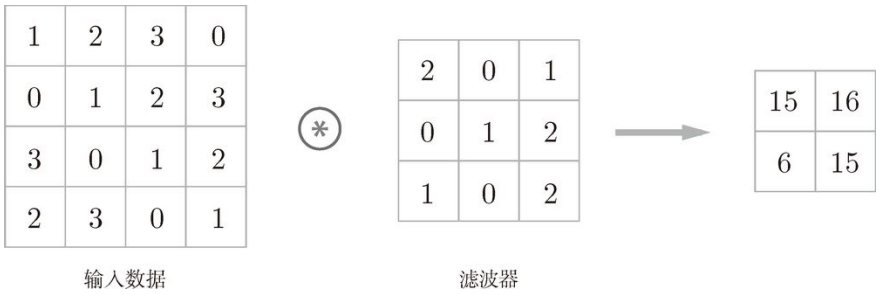

Figure 1.3: Example of convolution operation

全连接层存在什么问题呢？那就是数据的形状被“忽视”了。比如，输入数据是图像时，图像通常是高、长、通道方向上的3维形状。但是，向全连接层输入时，需要将3维数据拉平为1维数据。实际上，前面提到的使用了 MNIST 数据集的例子中，输入图像就是 1 通道、高 28 像素、长 28 像素的（1, 28, 28）形状，但却被排成1列，以784个数据的形式输入到最开始的 Affine层。

> 图像是3维形状，这个形状中应该含有重要的空间信息。比如，空间上邻近的像素为相似的值、RGB的各个通道之间分别有密切的关联性、相距较远的像素之间没有什么关联等，3维形状中可能隐藏有值得提取的本质模式。但是，因为全连接层会忽视形状，将全部的输入数据作为相同的神经元（同一维度的神经元）处理，所以无法利用与形状相关的信息。

而卷积层可以保持形状不变。因此，在 CNN 中，可以（有可能）正确理解图像等具有形状的数据。

CNN 中，有时将卷积层的输入输出数据称为**特征图**（feature map）。其中，卷积层的输入数据称为**输入特征图**（input feature map），输出数据称为**输出特征图**（output feature map）。

1.2.2 卷积运算

卷积层进行的处理就是卷积运算。卷积运算相当于图像处理中的“滤波器运算”(Figure 1.3)。

对于输入数据，卷积运算以一定间隔滑动滤波器的窗口并应用。这里所说的窗口是指 Figure 1.4 中灰色的3×3的部分。如 Figure 1.4 所示，将各个位置上滤波器的元素和输入的对应元素相乘，然后再求和（有时将这个计算称为**乘积累加运算**）。然后，将这个结果保存到输出的对应位置。将这个过程在所有位置都进行一遍，就可以得到卷积运算的输出。

在全连接的神经网络中，除了权重参数，还存在偏置。CNN中，滤波器的参数就对应之前的权重。并且，CNN中也存在偏置。Figure 1.3 的卷积运算的例子一直展示到了应用滤波器的阶段。包含偏置的卷积运算的处理流如图 Figure 1.5 所示。

如 Figure 1.5 所示，向应用了滤波器的数据加上了偏置。偏置通常只有1个（1×1）（本例中，相对于应用了滤波器的4个数据，偏置只有1个），这个值会被加到应用了滤波器的所有元素上。

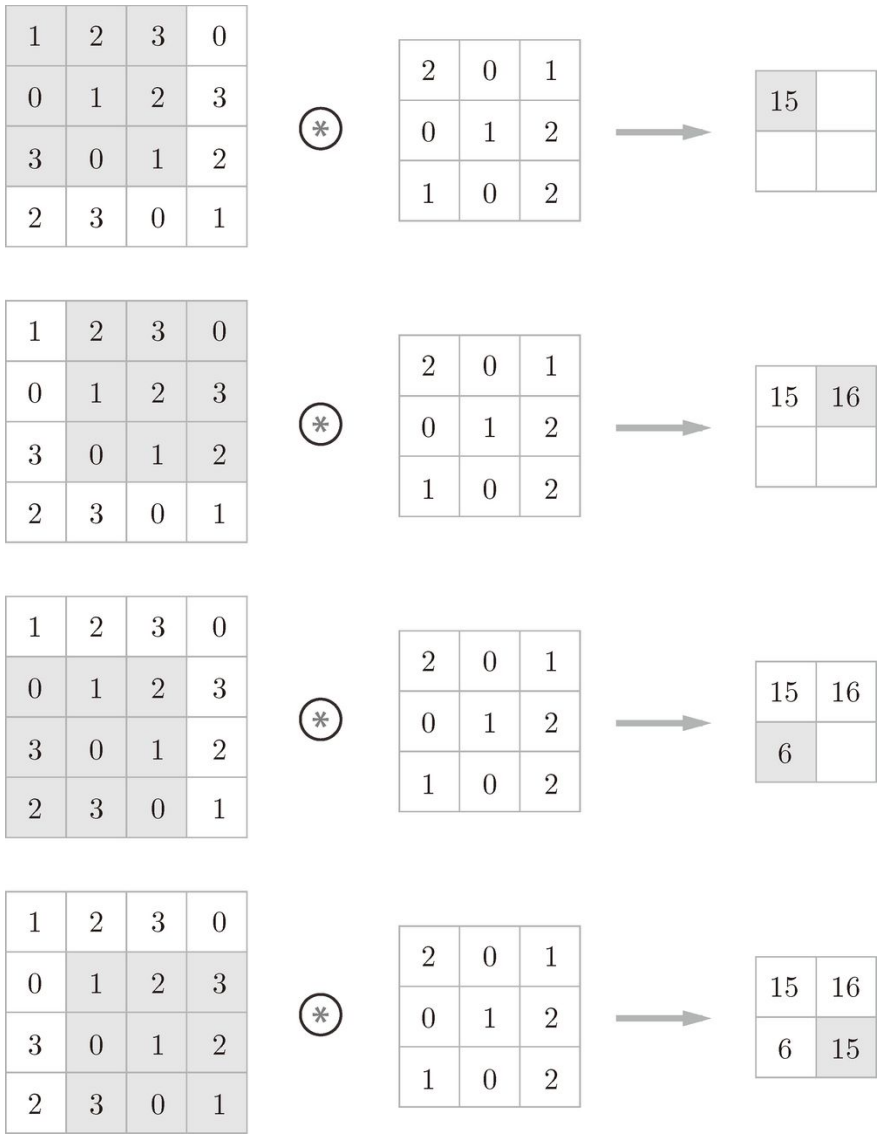

Figure 1.4: Calculation order of convolution operation

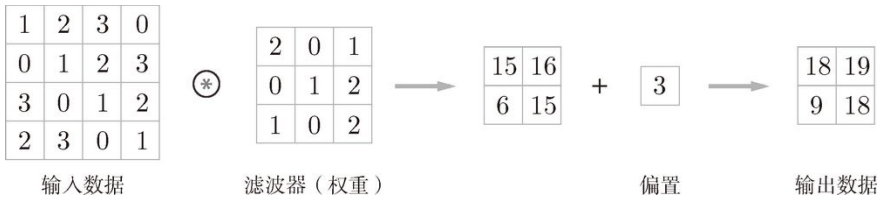

Figure 1.5: The bias of the convolution operation

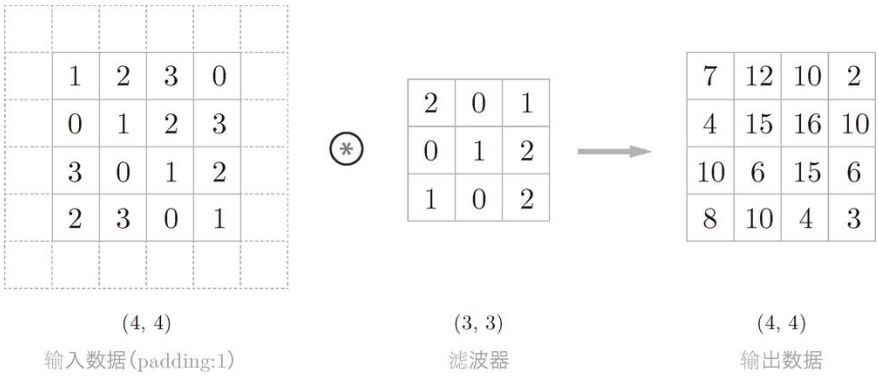

Figure 1.6: Filling processing of convolution operation

1.2.3 填充

在进行卷积层的处理之前，有时要向输入数据的周围填入固定的数据（比如0等），这称为**填充**（padding），是卷积运算中经常会用到的处理。

如 Figure 1.6 所示，通过填充，大小为(4,4)的输入数据变成了(6,6)的形状。然后，应用大小为(3,3)的滤波器，生成了大小为(4,4)的输出数据。这个例子中将填充设成了1，不过填充的值也可以设置成2、3等任意的整数。

使用填充主要是为了调整输出的大小。比如，对大小为(4,4)的输入数据应用(3,3)的滤波器时，输出大小变为(2,2)，相当于输出大小比输入大小缩小了2个元素。这在反复进行多次卷积运算的深度网络中会成为问题。为什么呢？因为如果每次进行卷积运算都会缩小空间，那么在某个时刻输出大小就有可能变为1，导致无法再应用卷积运算。为了避免出现这样的情况，就要使用填充。在刚才的例子中，将填充的幅度设为1，那么相对于输入大小(4,4)，输出大小也保持为原来的(4,4)。因此，卷积运算就可以在保持空间大小不变的情况下将数据传给下一层。

1.2.4 步幅

应用滤波器的位置间隔称为步幅（stride）。在 Figure 1.7 的例子中，对输入大小为(7,7)的数据，以步幅2应用了滤波器。通过将步幅设为2，输出大小变为(3,3)。像这样，步幅可以指定应用滤波器的间隔。

综上，增大步幅后，输出大小会变小。而增大填充后，输出大小会变大。

假设输入大小为(H,W)，滤波器大小为(FH,FW)，输出大小为 (OH,OW)，填充为P，步幅为S。此时，输出大小可通过 Equation 1.1 进行计算。

$$\begin{aligned} OH &= \frac{H+2P-FH}{S}+1 \\ OW &= \frac{W+2P-FW}{S}+1 \end{aligned} \tag{1.1}$$

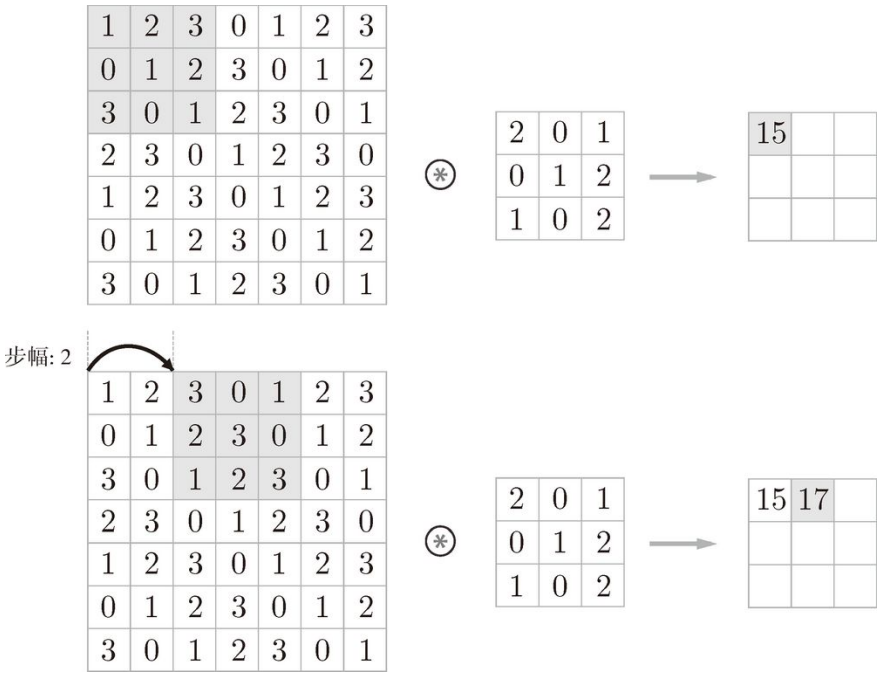

Figure 1.7: Example of convolution operation with stride 2