深度学习卷积运算指南

Vincent Dumoulin[1]★ and Francesco Visin[2]★†

★MILA, Université de Montréal

†AIRLab, Politecnico di Milano

2018.1.12

[1]dumouliv@iro.umontreal.ca

[2]francesco.visin@polimi.it

All models are wrong, but some are useful.

George E. P. Box

Acknowledgements

The authors of this guide would like to thank David Warde-Farley, Guillaume Alain and Caglar Gulcehre for their valuable feedback. We are likewise grateful to all those who helped improve this tutorial with helpful comments, constructive criticisms and code contributions. Keep them coming!

Special thanks to Ethan Schoonover, creator of the Solarized color scheme,[1] whose colors were used for the figures.

Feedback

Your feedback is welcomed! We did our best to be as precise, informative and up to the point as possible, but should there be anything you feel might be an error or could be rephrased to be more precise or comprehensible, please don't refrain from contacting us. Likewise, drop us a line if you think there is something that might fit this technical report and you would like us to discuss – we will make our best effort to update this document.

Source code and animations

The code used to generate this guide along with its figures is available on GitHub.[2] There the reader can also find an animated version of the figures.

[1] http://ethanschoonover.com/solarized

[2] https://github.com/vdumoulin/conv_arithmetic

目录

第一章　引言

深度卷积神经网络 (CNNs) 一直是深度学习取得巨大进步的核心. 尽管卷积神经网络在九十年代就被用来解决字符识别任务 (Le Cun *et al.*, 1997)，它们当前的广泛应用是由于最近的工作，当时一个深层 CNN 被用来打败最先进的 ImageNet 图像分类挑战 (Krizhevsky *et al.*, 2012).

因此卷积神经网络为机器学习的专业人士构建了一个非常有用的工具. 然而，第一次学习使用卷积神经网络通常是一种 "可怕" 的经历. 一个卷积层的输出形状受它的输入形状及卷积核的形状、零填充和步长的选择所影响，并且这些特性之间的关系不可忽视. 这不同于输出尺寸依赖于输入尺寸的全连接层. 另外，卷积神经网络通常还具有一个池化 (pooling) 阶段，对全连接网络来说增加了另一层的复杂性. 最后，所谓的转置卷积层（也被称为分步卷积层）近期已经在越来越多的工作 (Zeiler *et al.*, 2011; Zeiler and Fergus, 2014; Long *et al.*, 2015; Radford *et al.*, 2015; Visin *et al.*, 2015; Im *et al.*, 2016), 中被采用，并且它们与卷积层的关系已经从多个角度解释清楚了.

本指南的目标有两个:

1. 解释卷积层和转置卷积层.
2. 直观地理解卷积、池化和转置卷积层中输入形状、卷积核形状、零填充、步长和输出形状之间的关系.

为了保持普适性，本指南中所示结果不依赖实现细节，并适用于所有常用的机器学习框架，例如 Theano (Bergstra *et al.*, 2010; Bastien *et al.*, 2012), Torch (Collobert *et al.*, 2011), Tensorflow (Abadi *et al.*, 2015) 和 Caffe (Jia *et al.*, 2014).

本章简要回顾卷积神经网络的主要组成部分，即离散卷积和池化. 为了深入研究这一问题，请看深度学习教材的第九章 (Goodfellow *et al.*, 2016).

1.1　离散卷积

神经网络的中心思想是 *仿射变换* (affine transformations): 一个向量被接收为输入（input），并与矩阵相乘以产生输出（在进行非线性传递之前，通常会添加一个偏置向量）. 这适用于任何类型的输入，无论是图像、音频片段还是无序的特征集合：无论它们的维数如何，在变换之前，它们的表示都可以被展成向量.

图像、音频片段和许多其他相似类型的数据都有一个本质的结构. 从形式上讲，它们共享这些重要的性质：

- 它们被存储为多维数组.
- 它们以一个或多个轴为特征，它们之间的顺序很重要（例如，图像的宽度和高度轴，音频片段的时间轴）。
- 一个轴称为通道轴 (channel axis)，用于访问数据的不同视图（例如，彩色图像的红色，绿色和蓝色通道，或立体声音频轨道的左右通道）。

在应用仿射变换之前，这些性质还未被利用；事实上，所有的轴都以相同的方式处理，而且没有考虑拓扑的信息. 尽管如此，利用数据的隐式结构可能在完成某些任务，如计算机视觉和语义识别时非常方便，在这些情况下最好保存这些数据. 这就是离散卷积发挥作用的地方.

离散卷积是一种线性变换，它保留了顺序这一概念. 离散卷积是稀疏的（只有几个输入单元对给定的输出单元有贡献），并且重用参数（相同的权重应用于输入的多个位置）.

Figure 1.1 给出了一个离散卷积的例子. 浅蓝色的网格称为*输入特征映射* (input feature map). 为了保持绘图的简洁，该图代表了一个输入特征映射，但多个特征映射一个置于另一个上也很常见. [1] *卷积核* （阴影区域）在输入特征映射中滑动. 在每个位置，计算卷积核的每个元素与它重叠的输入元素之间的乘积，

0	1	2
2	2	0
0	1	2

[1] 在这方面的一个例子就是之前提到的图像和音频片段*通道* (channels) .

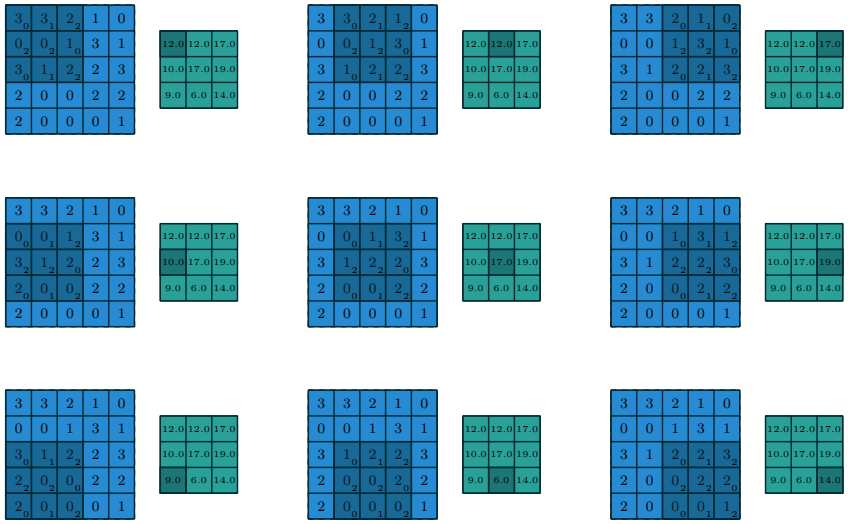


图 1.1: 计算离散卷积的输出值.

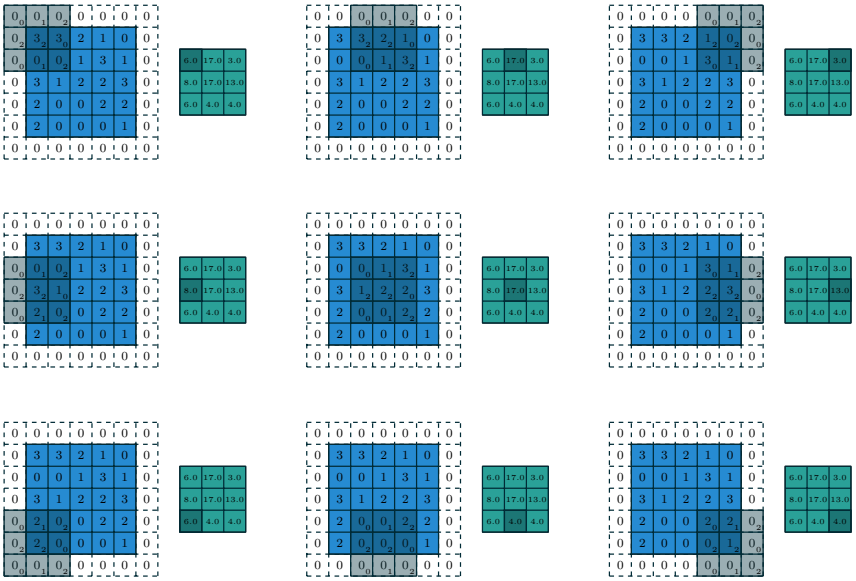


图 1.2: 对 $N = 2$, $i_1 = i_2 = 5$, $k_1 = k_2 = 3$, $s_1 = s_2 = 2$, 和 $p_1 = p_2 = 1$ 计算离散卷积的输出值.

并对结果进行求和. 可以用不同的卷积核重复该过程，以生成所需的输出特征映射 (Figure 1.3). 这个过程的最终输出被称为*输出特征映射* (output feature maps). [2]

如果有多个输入特征映射，则核必须是三维的——或者等价于每个特征映射都用一个不同的核进行卷积，并且生成的特征映射主元素相加，以生成输出特征映射。

卷积（见图Figure 1.1）是二维卷积的一种形式，但是它可以生成 N 维卷积. 例如，在三维卷积中，卷积核可以是一个*立方体* (cuboid)，而且会在输入特征映射的的高度、宽度和深度上滑动.

定义了一批离散卷积的核有一个形状对应 $(n, m, k_1, \ldots, k_N)$ 的某个排列，其中

$$
\begin{aligned}
n &\equiv \text{输出特征映射的数量}, \\
m &\equiv \text{输入特征映射的数量}, \\
k_j &\equiv \text{沿 } j \text{ 轴核的尺寸}.
\end{aligned}
$$

以下属性会影响沿 j 轴方向卷积层的输出尺寸 o_j :

- i_j: 沿 j 轴的输入尺寸,
- k_j: 沿 j 轴的核尺寸,
- s_j: 沿 j 轴的步长 (核的两个连续位置之间的距离),
- p_j: 沿 j 轴方向的零填充 (在轴开始和结尾处连接的 0 的数目).

例如, Figure 1.2 展示了一个应用于 5×5 输入的 3×3 卷积核，使用了 2×2 的步长填充了 1×1 的零边界.

注意，步长构成了一种 *子采样* (subsampling) 形式. 步长作为衡量核被翻译多少的一种替代方法，它也可以看作保留了多少输出. 例如，以两个跃点移动卷积核相当于以一个跃点移动卷积核，但只保留奇数个输出元素 (Figure 1.4).

[2] 虽然从信号处理的角度来看，卷积和交叉相关是有区别的，但当学习了卷积核时，两者就可以互换了. 为了简单起见，以及与大多数机器学习文献保持一致，我们将在这个指南中使用*卷积* (convolution) 这一术语.

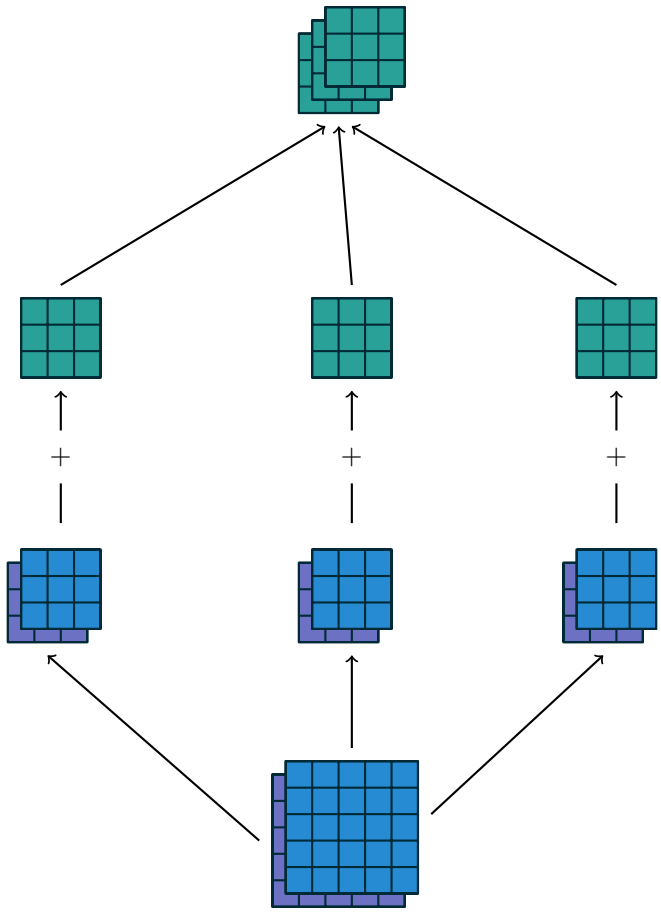


图 1.3:　一种从两个输入特征映射到三个输出特征映射的卷积映射，使用 $3 \times 2 \times 3 \times 3$ 的一批卷积核 $\mathbf{w}$. 在左边的路径中，输入特征映射 1 与核 $\mathbf{w}_{1,1}$ 做卷积且输入特征映射 2 与核 $\mathbf{w}_{1,2}$ 做卷积，并将结果逐元素相加，形成第一个输出特征映射. 对中间和右侧的路径重复相同的步骤，以形成第二和第三个特征映射，并将所有的三个输出特征映射组合在一起形成输出.

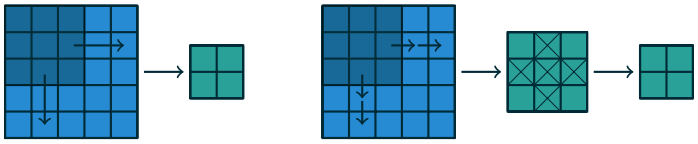


图 1.4:　另一种步长可视方法. 不是以 $s = 2$ （左边）的增量去转换 3×3 的卷积核，而是以 1 为增量进行转换，且只保留一个 $s = 2$ 的输出元素（右边）.

1.2　池化

除了离散卷积本身，*池化* (pooling) 操作是构成卷积神经网络的另一个重要部分. 池化操作通过使用某种函数来汇总子区域，例如取平均值或最大值，以此减少特征映射的大小.

池化的工作原理是在输入中滑动一个窗口，并将窗口的内容提供给*池化函数* (pooling function). 从某种意义上讲，池化的工作原理与离散卷积十分类似，但它用其他函数代替了卷积核形成的线性组合. Figure 1.5 给出了一个平均池化的例子，而 Figure 1.6 给出了最大值池化的例子.

以下属性会影响沿 j 轴池化层的输出大小 o_j:

- i_j: 沿 j 轴的输入尺寸,
- k_j: 沿 j 轴的池化窗口大小,
- s_j: 沿 j 轴的步长 (池化窗口的两连续位置之间的距离).

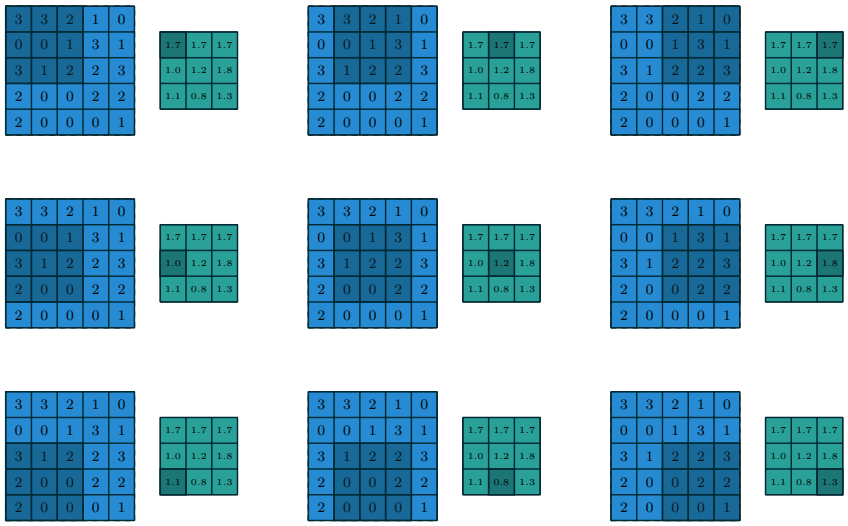


图 1.5: 使用 1×1 步长计算 5×5 输入、3×3 平均池化操作的输出值.

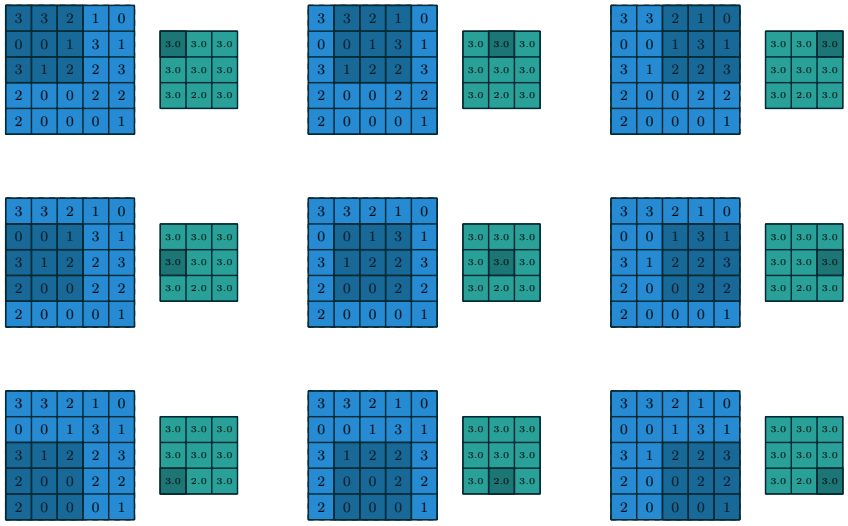


图 1.6: 使用 1×1 步长计算 5×5 输入、3×3 最大值池化操作的输出值.

第二章　卷积运算

卷积层属性之间的关系分析起来很容易，因为它们不跨轴交互，即沿 j 轴选定的卷积核大小、步长和零填充只影响 j 轴的输出大小. 因此，本章将重点介绍以下的简化设置:

- 二维离散卷积 $(N = 2)$,
- 方形输入 $(i_1 = i_2 = i)$,
- 方形核尺寸 $(k_1 = k_2 = k)$,
- 沿着两轴的相同步长 $(s_1 = s_2 = s)$,
- 沿着两轴的相同零填充 $(p_1 = p_2 = p)$.

这有助于分析及可视化，但请记住，这里概述的结果也可推广到 N 维和非方形情形.

2.1　无零填充，单位步长

要分析的最简单情形是卷积核在输入 (即 $s = 1$ 且 $p = 0$ 时) 的每个位置滑动. Figure 2.1 提供了当 $i = 4$ 且 $k = 3$ 时的例子.

在这种情况下，定义输出大小的一种方法是确定在输入层上可放置卷积核的个数. 让我们考虑宽度轴：内核从输入特征映射的最左边开始，一步步滑动，直到它接触输入的右侧. 输出的大小等于所执行步数加一，即内核的初始位置 (Figure 2.8a). 相同的逻辑应用于高度轴.

更正式地说，可以推断出以下关系:

关系 1. *对任何 i 和 k, 以及对 $s = 1$ 和 $p = 0$,*

$$o = (i - k) + 1.$$

2.2　有零填充，单位步长

为了考虑零填充 (即仅限于 $s = 1$), 让我们考虑其对有效输入大小的影响: 使用 p 零填充将有效输入大小从 i 改为 $i + 2p$. 在一般情况下, 关系 1 可推出以下关系:

关系 2. *对任何 i, k 和 p, 且对 $s = 1$, 有*

$$o = (i - k) + 2p + 1.$$

Figure 2.2 给出了当 $i = 5$, $k = 4$ 且 $p = 2$ 时的一个例子.

实际上，零填充的两个特定实例由于各自的性质而被广泛使用. 让我们更详细地讨论一下.

2.2.1　半填充 (相似填充)

输出大小和输入大小相同 (即 $o = i$) 是一个理想的性质:

关系 3. *对任何 i 和奇数 k ($k = 2n + 1, \quad n \in \mathbb{N}$), $s = 1$, $p = \lfloor k/2 \rfloor = n$, 有*

$$\begin{aligned} o &= i + 2\lfloor k/2 \rfloor - (k - 1) \\ &= i + 2n - 2n \\ &= i. \end{aligned}$$

这有时被称为*半填充* (或*相似填充*). Figure 2.3 给出了当 $i = 5$, $k = 3$ 且 (因此) $p = 1$ 时的一个例子.

2.2.2　全填充

虽然对一个核做卷积时，通常会减小 相对于输入的输出大小，但有时需要相反的结果. 这可以通过适当的零填充来实现:

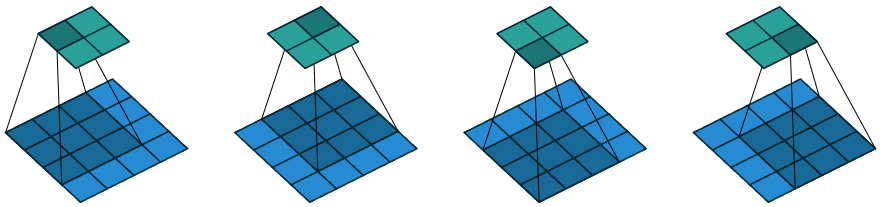


图 2.1: (无零填充，单位步长) 用单位步长 (即 $i = 4, k = 3, s = 1$ 和 $p = 0$)，将 3×3 的卷积核与 4×4 的输入做卷积.

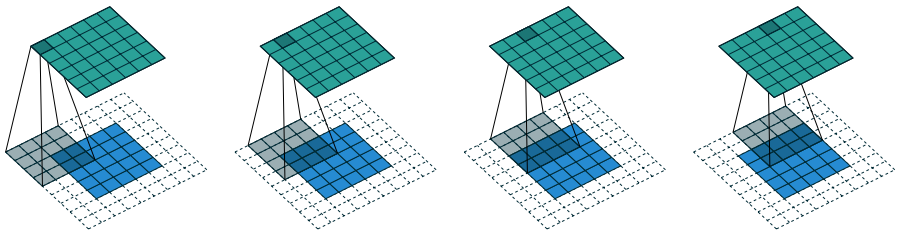


图 2.2: (任意填充, 单位步长) 用单位步长 (即 $i = 5, k = 4, s = 1$ 和 $p = 2$)，将 4×4 的卷积核与 5×5 的输入做卷积，并用 2×2 的零边界填充.

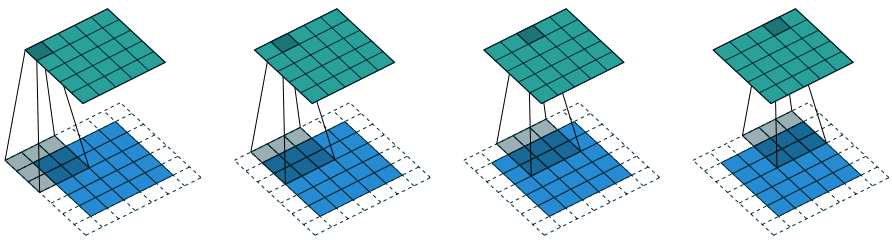


图 2.3: (半填充, 单位步长) 用半填充和单位步长 (即 $i = 5, k = 3, s = 1$ 和 $p = 2$)，将 3×3 的卷积核与 5×5 的输入做卷积.

关系 4. 对任何 i 和 k, 且对 $p = k - 1$ 和 $s = 1$, 有

$$\begin{aligned} o &= i + 2(k-1) - (k-1) \\ &= i + (k-1). \end{aligned}$$

这有时被称为全 填充, 因为在这个设定中，输入特征映射上卷积核的每一个可能的部分或完全重叠都被考虑在内. Figure 2.4 给出了当 $i = 5$, $k = 3$ 且 (因此) $p = 2$ 的一个例子.

2.3　无零填充，非单位步长

迄今为止导出的所有关系仅适用于单位步长卷积. 合并非单位步长需要另一种进一步的推理. 为了便于分析，让我们暂时忽略零填充 (即 $s > 1$ 且 $p = 0$). Figure 2.5 给出了当 $i = 5$, $k = 3$ 且 $s = 2$ 时的一个例子.

同样，输出大小可以根据卷积核在输入上的可能位置的数量来定义. 让我们考虑宽度轴：卷积核像往常一样从输入的最左边开始，但是这次它以大小为 s 滑动，直到它接触到输入的右侧. 这个输出的大小再次等于所执行的步骤数加一，说明了卷积核的初始位置 (Figure 2.8b). 相同的逻辑应用于高度轴.

由此可以推断出以下关系:

关系 5. 对任何 i, k 和 s, 并对 $p = 0$, 有

$$o = \left\lfloor \frac{i-k}{s} \right\rfloor + 1.$$

下取整函数说明了这样一个事实：有时最后可能的步骤与到达输入层末尾的卷积核不 一致, 也就是说, 一些输入单元被忽略了 (关于这种情况，参见 Figure 2.7).

2.4　有零填充，非单位步长

最一般的情况（使用非单位步长在零填充的输入上做卷积）可以通过在大小为 $i + 2p$ 的有效输入上应用 关系 5 来推导，类似于 关系 2 所做的工作：

关系 6. 对任何 i, k, p 和 s,

$$o = \left\lfloor \frac{i + 2p - k}{s} \right\rfloor + 1.$$

如前所述，下取整函数意味着在某些情况下，卷积将为多个输入大小生成相同的输出大小. 更具体地讲，如果 $i + 2p - k$ 是 s 的倍数, 所有输出大小 $j = i + a, \quad a \in \{0, \ldots, s - 1\}$ 都将产生相同的输出大小. 注意，这种二义性仅应用于 $s > 1$.

Figure 2.6 给出了当 $i = 5$, $k = 3$, $s = 2$ 和 $p = 1$ 时的一个例子, 而 Figure 2.7 给出了当 $i = 6$, $k = 3$, $s = 2$ 且 $p = 1$ 时的例子. 有趣的是，尽管输入大小不同，这些卷积依然共享同一输出大小. 虽然这不影响卷积 (convolutions) 的分析，但这将使转置卷积 (transposed convolutions) 的分析复杂化.

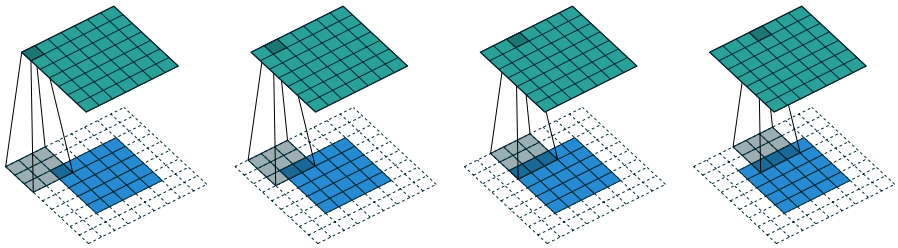


图 2.4: (全填充，单位步长) 使用全填充和单位步长 (即 $i = 5$, $k = 3$, $s = 1$ 和 $p = 2$)，将 3×3 的卷积核与 5×5 的输入做卷积.

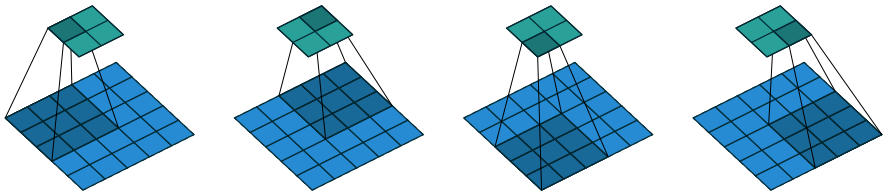


图 2.5: (无零填充，任意步长) 使用 2×2 步长 (即 $i = 5$, $k = 3$, $s = 2$ 和 $p = 0$) 将 3×3 的卷积核与 5×5 的输入做卷积.

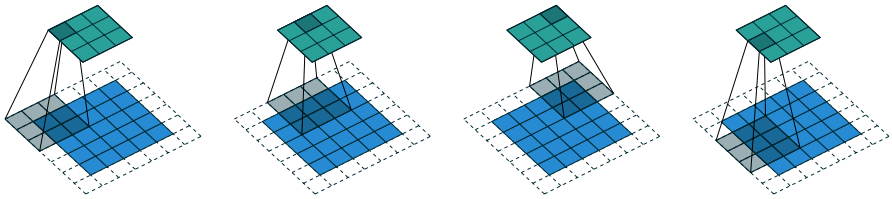


图 2.6: (任意填充和步长) 使用 2×2 步长 (即 $i = 5$, $k = 3$, $s = 2$ 和 $p = 1$) 将 3×3 的卷积核与填充了 1×1 零边界的 5×5 输入做卷积. 在这种情况下，卷积核不覆盖零填充输入的底部行和右边列

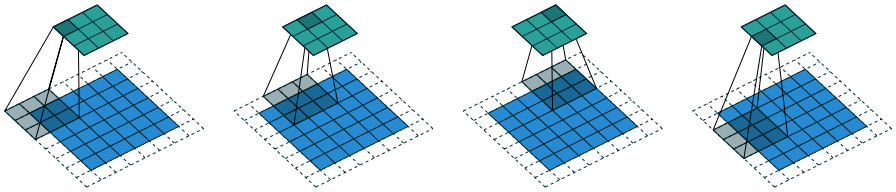
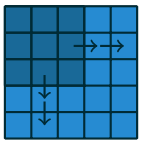
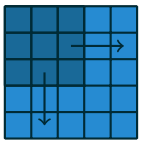


图 2.7:　(任意填充和步长) Convolving a 3×3 kernel over a 6×6 input padded with a 1×1 border of zeros using 2×2 strides (即, $i = 6$, $k = 3$, $s = 2$ and $p = 1$). In this case, the bottom row and right column of the zero padded input are not covered by the kernel.



(a)　卷积核必须向右滑动两步才能接触到输入的右侧（等价于向下）. 加一以说明初始卷积核位置，输出大小为 3×3.



(b)　内核必须向右滑动一步（大小为 2）才能接触到输入的右侧（相当于向下）. 加一以说明初始卷积核的位置，输出大小为 2×2.

图 2.8: 计算卷积核位置.

第三章　池化运算

在神经网络中，池化层对输入的微小转换满足不变性. 最常见的一种池化是最大值池化 (max pooling)，它包括将输入拆分为（通常不重叠）块并输出每个块的最大值. 其他类型的池化也存在，例如平均值池化，它们都具有相同的思想，即通过对某些块应用非线性来汇总局部输入 (Boureau *et al.*, 2010a,b, 2011; Saxe *et al.*, 2011).

一些读者可能已经注意到，卷积运算的处理仅仅依赖于一个假设，即某些函数被重复应用到输入的子集上. 这意味着上一章中导出的关系可以在池化算法下重用. 由于池化不涉及零填充，描述一般情况的关系如下：

关系 7. *对任何 i, k 和 s, 有*

$$o = \left\lfloor \frac{i-k}{s} \right\rfloor + 1.$$

这种关系适用于任何类型的池化.

第四章　转置卷积运算

转置卷积用于与正常卷积做相反方向变换的情况，即从具有某种卷积输出形状到具有其输入形状的情况，同时保持与所述卷积兼容的连接模式. 例如，可以使用这样的转换作为卷积自动编码器的解码层，或者将特征映射投影到更高维空间.

同样，卷积的情况比全连接的情况复杂得多，后者只需要使用形状被转置的权重矩阵. 然而，由于每一个卷积都可以归结为一个矩阵运算的有效实现，因此从全连接情形中获得的见解对于解决卷积情形是有用的.

与卷积算法一样，转置卷积算法的论文由于转置卷积不跨轴交互的性质而得以简化.

本章将着重于以下设定:

- 2 维转置卷积 ($N = 2$),
- 方形输入 ($i_1 = i_2 = i$),
- 方形核大小 ($k_1 = k_2 = k$),
- 沿着两轴的相同步长 ($s_1 = s_2 = s$),
- 沿着两轴的相同零填充 ($p_1 = p_2 = p$).

结果将再次推广到 N 维的情形.

4.1　作为矩阵运算的卷积

以 Figure 2.1 表示的卷积为例. 如果输入和输出从左至右、从上到下展开为向量，卷积可以表示为稀疏矩阵 $\mathbf{C}$，其中非零元素是卷积核的元素 $w_{i,j}$

(i 和 j 分别是卷积核的行和列):

$$\begin{pmatrix} w_{0,0} & w_{0,1} & w_{0,2} & 0 & w_{1,0} & w_{1,1} & w_{1,2} & 0 & w_{2,0} & w_{2,1} & w_{2,2} & 0 & 0 & 0 & 0 & 0 \\ 0 & w_{0,0} & w_{0,1} & w_{0,2} & 0 & w_{1,0} & w_{1,1} & w_{1,2} & 0 & w_{2,0} & w_{2,1} & w_{2,2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & w_{0,0} & w_{0,1} & w_{0,2} & 0 & w_{1,0} & w_{1,1} & w_{1,2} & 0 & w_{2,0} & w_{2,1} & w_{2,2} & 0 \\ 0 & 0 & 0 & 0 & 0 & w_{0,0} & w_{0,1} & w_{0,2} & 0 & w_{1,0} & w_{1,1} & w_{1,2} & 0 & w_{2,0} & w_{2,1} & w_{2,2} \end{pmatrix}$$

此线性运算将展平后的输入矩阵作为 16 维向量，并生成一个 4 维向量，该向量随后变为形状是 2×2 的输出矩阵.

使用这种表示方法，通过转置 $\mathbf{C}$ 很容易得到反向传递; 换句话说，通过将损失乘以 $\mathbf{C}^T$，可以反向传播误差. 这一操作以一个 4 维向量作为输入，生成一个 16 维向量作为输出，并且其连接模式通过一定的构造与 $\mathbf{C}$ 兼容.

值得注意的是，卷积核 $\mathbf{w}$ 定义了用于向前和向后传递的矩阵 $\mathbf{C}$ 和 $\mathbf{C}^T$.

4.2　转置卷积

现在让我们考虑一下从 4 维空间映射到 16 维空间,并同时保持Figure 2.1中描述的卷积连接模式反过来又需要什么？ 这一操作称为*转置卷积* (transposed convolution).

*转置卷积*也叫*分步卷积* (fractionally strided convolutions) 或*逆卷积* (deconvolutions)[1]，通过交换卷积的前向和反向传递来工作. 一种说法是卷积核定义了一个卷积，但它是直接卷积还是转置卷积取决于前向和反向传递的计算方式.

例如，虽然卷积核 $\mathbf{w}$ 定义了一个卷积，其正向和反向传播分别通过与 $\mathbf{C}$ 和 $\mathbf{C}^T$ 相乘来计算，但它也 定义了一个转置的卷积，其正向和反向传递分别通过与 $\mathbf{C}^T$ 和 $(\mathbf{C}^T)^T = \mathbf{C}$ 相乘来计算.[2]

最后要注意的是，我们总是可以用直接卷积来模拟转置卷积. 缺点是，它通常需要向输入中添加许多列和零行，导致实现效率大大降低.

在目前所介绍的基础上，本章将从卷积算法一章开始，通过引用与其共享卷积核的直接卷积来推导每个转置卷积的性质，并定义等效的直接卷积.

[1]“逆卷积”(deconvolution) 一词有时在文献中使用，但我们不推荐这样用，理由是逆卷积在数学上被定义为卷积的逆，它不同于转置卷积.

[2]转置卷积操作可以被认为是某些卷积相对于其输入的梯度，这通常是转置卷积在实际应用中的实现方式.

4.3　无零填充，单位步长，转置

考虑给定输入上转置卷积的最简单方法，是将这种输入想象为应用于某些初始特征映射的直接卷积的结果. 因此，卷积可以看作是恢复初始特征映射 *形状* (shape) [3] 的操作.

考虑在单位步长且无填充的 4×4 输入 (即 $i=4$, $k=3$, $s=1$ 且 $p=0$) 上应用 3×3 的卷积核. 就像 Figure 2.1中描述的那样，这产生了一个 2×2 的输出. 当应用于 2×2 的输入时，这个卷积的转置将产生形状为 4×4 的输出.

另一种获得转置卷积结果的方法是应用一个等价（但效率要低得多）的直接卷积. 到目前为止描述的例子可以通过使用单位步长 (即 $i'=2$, $k'=k$, $s'=1$ 且 $p'=2$) 将 3×3 的卷积核与一个 2×2 的输入做卷积，这个输入使用单位步长且填充了 2×2 的零边界，如Figure 4.1 展示的那样. 值得注意的是，卷积核和步长的大小保持不变，但是转置卷积的输入现在是零填充.[4]

理解零填充背后逻辑的一个方法是考虑转置卷积的连接模式，并用它来指导等效卷积的设计. 例如，直接卷积输入的左上角像素只贡献给输出的左上角像素，右上角像素只连接到右上角的输出像素等.

为了在等价的卷积中保持相同的连接模式，有必要对输入进行零填充，使卷积核的第一个（左上）应用只接触左上角像素，即填充必须等于核的大小减 1.

以同样的方式进行，可以确定图像其他元素类似的观察结果，从而得到以下关系:

> **关系 8.** 由 $s=1$, $p=0$ 和 k 描述的卷积具有由 $k'=k$, $s'=s$ 和 $p'=k-1$ 描述的相关转置卷积，并且其输入大小为
>
> $$o'=i'+(k-1).$$

有趣的是，这对应于单位步长的全填充卷积.

[3] 注意到转置卷积并不保证恢复输入本身，因为它不是定义为卷积的逆，而是返回具有相同宽度和高度的特征映射.

[4] 注意，尽管相当于应用转置矩阵，但这种以补零的形式可视化添加了大量的零乘法. 这里这样做是为了说明问题，但是效率低下，软件实现通常不会执行无用的零乘法.

4.4　有零填充，单位步长，转置

已知无填充卷积的转置等价于对一个零填充输入做卷积，就可以合理地假设零填充卷积的转置等价于对较少的零填充输入做卷积.

确实如此，对 $i = 5$, $k = 4$ 和 $p = 2$，如 Figure 4.2 所示.

在形式上，以下关系适用于零填充卷积:

> **关系 9.** 一个由 $s = 1$, k 和 p 描述的卷积与一个由 $k' = k$, $s' = s$ 和 $p' = k - p - 1$ 描述的转置卷积相关联. 且其输出大小为
>
> $$o' = i' + (k - 1) - 2p.$$

4.4.1　半 (相似) 填充，转置

通过应用与前面相同的归纳推理，可以合理推测半填充卷积转置的等效卷积就是半填充卷积自身，因为半填充卷积的输出大小与其输入大小相同. 因此，以下关系适用:

> **关系 10.** *A convolution described by* $k = 2n + 1, \quad n \in \mathbb{N}$, $s = 1$ *and* $p = \lfloor k/2 \rfloor = n$ *has an associated transposed convolution described by* $k' = k$, $s' = s$ *and* $p' = p$ *and its output size is*
>
> $$\begin{aligned} o' &= i' + (k - 1) - 2p \\ &= i' + 2n - 2n \\ &= i'. \end{aligned}$$

Figure 4.3 给出了当 $i = 5$, $k = 3$ 且（因此）$p = 1$ 的一个例子.

4.4.2　全填充，转置

已知非填充转置卷积的等价卷积涉及全填充，所以全填充转置卷积的等价卷积是非填充卷积就不足为奇了:

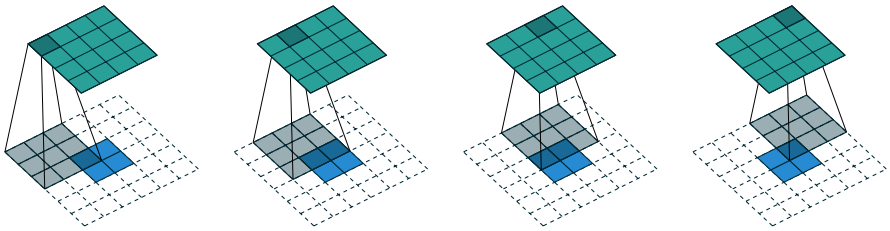


图 4.1: The transpose of convolving a 3×3 kernel over a 4×4 input using unit strides (即 $i = 4$, $k = 3$, $s = 1$ and $p = 0$). It is equivalent to convolving a 3×3 kernel over a 2×2 input padded with a 2×2 border of zeros using unit strides (即, $i' = 2$, $k' = k$, $s' = 1$ and $p' = 2$).

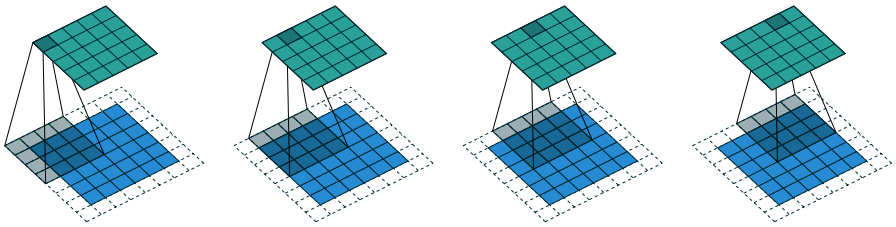


图 4.2: The transpose of convolving a 4×4 kernel over a 5×5 input padded with a 2×2 border of zeros using unit strides (即, $i = 5$, $k = 4$, $s = 1$ and $p = 2$). It is equivalent to convolving a 4×4 kernel over a 6×6 input padded with a 1×1 border of zeros using unit strides (即 $i' = 6$, $k' = k$, $s' = 1$ 且 $p' = 1$).

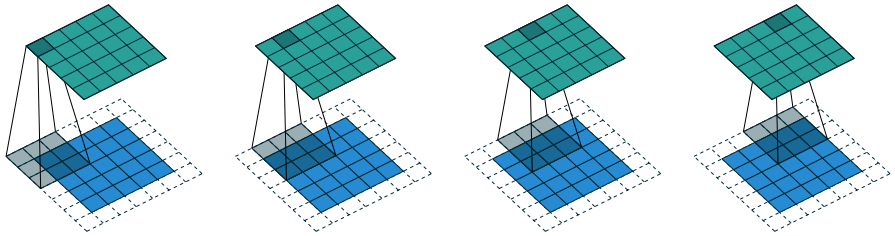


图 4.3: The transpose of convolving a 3×3 kernel over a 5×5 input using half padding and unit strides (即 $i = 5$, $k = 3$, $s = 1$ and $p = 1$). It is equivalent to convolving a 3×3 kernel over a 5×5 input using half padding and unit strides (即 $i' = 5$, $k' = k$, $s' = 1$ and $p' = 1$).

关系 11. *一种卷积被描述为 $s = 1$, k 和 $p = k - 1$ has an associated transposed convolution described by $k' = k$, $s' = s$ and $p' = 0$ and its output size is*

$$\begin{aligned} o' &= i' + (k-1) - 2p \\ &= i' - (k-1) \end{aligned}$$

Figure 4.4 给出了当 $i = 5$, $k = 3$ 且（因此）$p = 2$ 时的一个例子.

4.5　无零填充，非单位步长，转置

使用与零填充卷积相同的归纳逻辑，可以推测 $s > 1$ 时卷积的转置涉及到 $s < 1$ 时的等效卷积. 正如将要解释的，这是一个有效的直观感受，这就是为什么转置卷积有时被称为分步卷积.

Figure 4.5 给出了当 $i = 5$, $k = 3$ 且 $s = 2$ 时的一个例子，这有助于理解分步包含的内容: 在输入单元之间插入零，这使得内核的移动速度比单位步长慢.[5]

现在假设卷积是无填充的 ($p = 0$)，并且它的输入大小 i 使得 $i - k$ 是 s 的倍数. 在这种情况下，以下关系成立:

关系 12. *由 $p = 0$, k 和 s 描述的一种卷积，其输入大小使 $i - k$ 为 s 的倍数，它有一个由 $\tilde{i}'$, $k' = k$, $s' = 1$ 和 $p' = k - 1$ 所描述的相关转置卷积，其中 $\tilde{i}'$ 是通过在每个输入单元之间添加 $s - 1$ 个零得到的扩展输入大小，其输出大小是*

$$o' = s(i' - 1) + k.$$

4.6　零填充，非单位步长，转置

当卷积的输入大小为 i 时，如果 $i + 2p - k$ 是 s 的倍数，则可以通过结合 关系 9 和 关系 12将分析扩展到零填充的情况:

[5] 这样做是低效的，并且实际的实现避免了无用的零乘法，但从概念上讲，这是怎样考虑步长卷积的转置.

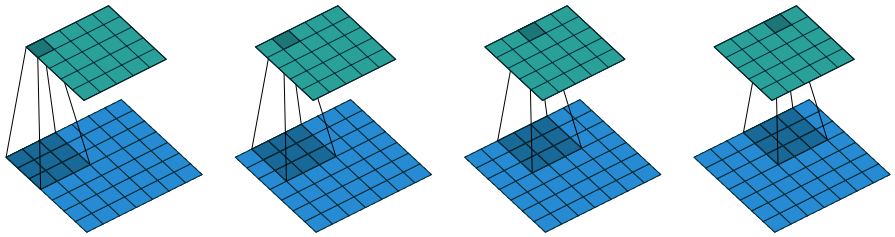


图 4.4: The transpose of convolving a 3×3 kernel over a 5×5 input using full padding and unit strides (即 $i = 5$, $k = 3$, $s = 1$ 且 $p = 2$). It is equivalent to convolving a 3×3 kernel over a 7×7 input using unit strides (即 $i' = 7$, $k' = k$, $s' = 1$ 且 $p' = 0$).

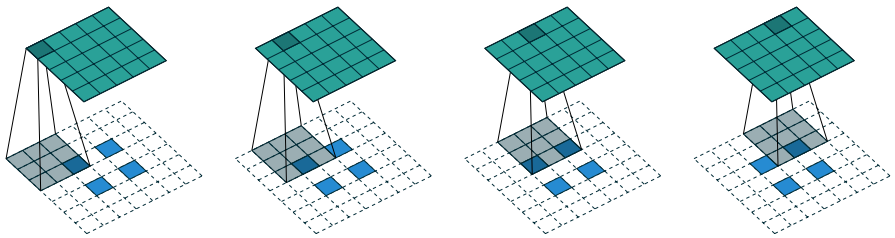


图 4.5: The transpose of convolving a 3×3 kernel over a 5×5 input using 2×2 strides (即 $i = 5$, $k = 3$, $s = 2$ and $p = 0$). It is equivalent to convolving a 3×3 kernel over a 2×2 input (with 1 zero inserted between inputs) padded with a 2×2 border of zeros using unit strides (即 $i' = 2$, $\tilde{i}' = 3$, $k' = k$, $s' = 1$ and $p' = 2$).

关系 13. 一种由 k, s 和 p 描述的卷积，其输入大小 i 使得 $i + 2p - k$ 是 s 的倍数，它有一个由 $\tilde{i}'$, $k' = k$, $s' = 1$ 和 $p' = k - p - 1$ 描述的相关转置卷积，其中 $\tilde{i}'$ 是通过在每个输入单元之间添加 $s - 1$ 个零得到的扩展输入的大小，其输出大小为

$$o' = s(i' - 1) + k - 2p.$$

Figure 4.6 给出了当 $i = 5$, $k = 3$, $s = 2$ 且 $p = 1$ 时的一个例子.

输入 i 大小的约束条件可以通过引入另一个参数 $a \in \{0, \ldots, s - 1\}$ 来放宽，该参数允许区分在 s 不同的情况下，所有结果都导致相同的 i':

关系 14. 一种由 k, s 和 p 描述的卷积，*has an associated transposed convolution described by* a, $\tilde{i}'$, $k' = k$, $s' = 1$ 且 $p' = k - p - 1$, *where* $\tilde{i}'$ *is the size of the stretched input obtained by adding* $s - 1$ *zeros between each input unit, and* $a = (i + 2p - k) \mod s$ *represents the number of zeros added to the bottom and right edges of the input*，且其输出大小为

$$o' = s(i' - 1) + a + k - 2p.$$

Figure 4.7 提供了当 $i = 6$, $k = 3$, $s = 2$ 且 $p = 1$ 时的一个例子.

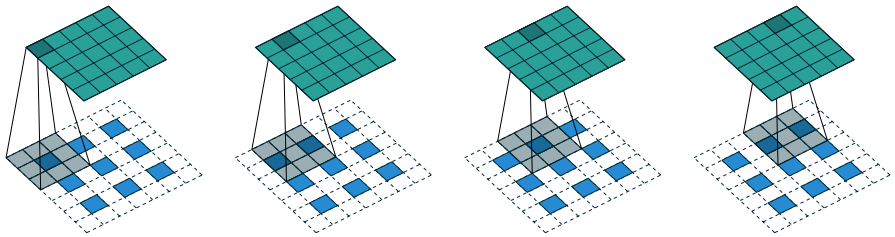


图 4.6: The transpose of convolving a 3×3 kernel over a 5×5 input padded with a 1×1 border of zeros using 2×2 strides (即, $i = 5$, $k = 3$, $s = 2$ and $p = 1$). It is equivalent to convolving a 3×3 kernel over a 3×3 input (with 1 zero inserted between inputs) padded with a 1×1 border of zeros using unit strides (即, $i' = 3$, $\tilde{i}' = 5$, $k' = k$, $s' = 1$ and $p' = 1$).

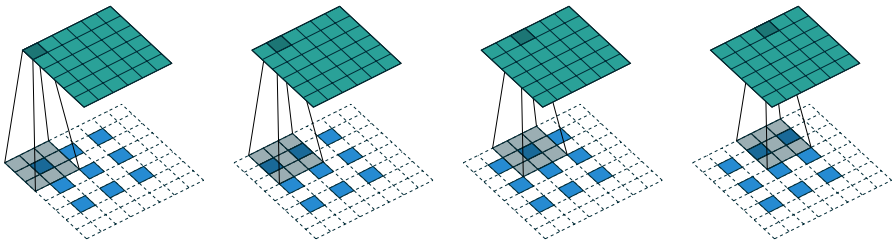


图 4.7: The transpose of convolving a 3×3 kernel over a 6×6 input padded with a 1×1 border of zeros using 2×2 strides (即, $i = 6$, $k = 3$, $s = 2$ and $p = 1$). It is equivalent to convolving a 3×3 kernel over a 2×2 input (with 1 zero inserted between inputs) padded with a 1×1 border of zeros (with an additional border of size 1 added to the bottom and right edges) using unit strides (即, $i' = 3$, $\tilde{i}' = 5$, $a = 1$, $k' = k$, $s' = 1$ and $p' = 1$).

第五章　其他卷积

5.1　扩张卷积

Readers familiar with the deep learning literature may have noticed the term "dilated convolutions" (or "atrous convolutions", from the French expression *convolutions à trous*) appear in recent papers. Here we attempt to provide an intuitive understanding of dilated convolutions. For a more in-depth description and to understand in what contexts they are applied, see Chen *et al.* (2014); Yu and Koltun (2015).

Dilated convolutions "inflate" the kernel by inserting spaces between the kernel elements. The dilation "rate" is controlled by an additional hyperparameter d. Implementations may vary, but there are usually $d-1$ spaces inserted between kernel elements such that $d=1$ corresponds to a regular convolution.

Dilated convolutions are used to cheaply increase the receptive field of output units without increasing the kernel size, which is especially effective when multiple dilated convolutions are stacked one after another. For a concrete example, see Oord *et al.* (2016), in which the proposed WaveNet model implements an autoregressive generative model for raw audio which uses dilated convolutions to condition new audio frames on a large context of past audio frames.

To understand the relationship tying the dilation rate d and the output size o, it is useful to think of the impact of d on the *effective kernel size*. A kernel of size k dilated by a factor d has an effective size

$$\hat{k} = k + (k-1)(d-1).$$

This can be combined with 关系 6 to form the following relationship for dilated convolutions:

关系 15. *For any i, k, p and s, and for a dilation rate d,*

$$o = \left\lfloor \frac{i + 2p - k - (k-1)(d-1)}{s} \right\rfloor + 1.$$

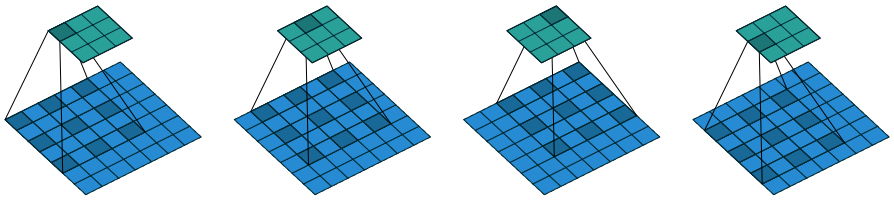


图 5.1: (Dilated convolution) Convolving a 3×3 kernel over a 7×7 input with a dilation factor of 2 (即, $i = 7$, $k = 3$, $d = 2$, $s = 1$ and $p = 0$).

Figure 5.1 provides an example for $i = 7$, $k = 3$ and $d = 2$.

参考文献

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., *et al.* (2015). Tensorflow: Large-scale machine learning on heterogeneous systems. *Software available from tensorflow.org*.

Bastien, F., Lamblin, P., Pascanu, R., Bergstra, J., Goodfellow, I., Bergeron, A., Bouchard, N., Warde-Farley, D., and Bengio, Y. (2012). Theano: new features and speed improvements. *arXiv preprint arXiv:1211.5590*.

Bergstra, J., Breuleux, O., Bastien, F., Lamblin, P., Pascanu, R., Desjardins, G., Turian, J., Warde-Farley, D., and Bengio, Y. (2010). Theano: A cpu and gpu math compiler in python. In *Proc. 9th Python in Science Conf*, pages 1–7.

Boureau, Y., Bach, F., LeCun, Y., and Ponce, J. (2010a). Learning mid-level features for recognition. In *Proc. International Conference on Computer Vision and Pattern Recognition (CVPR'10)*. IEEE.

Boureau, Y., Ponce, J., and LeCun, Y. (2010b). A theoretical analysis of feature pooling in vision algorithms. In *Proc. International Conference on Machine learning (ICML'10)*.

Boureau, Y., Le Roux, N., Bach, F., Ponce, J., and LeCun, Y. (2011). Ask the locals: multi-way local pooling for image recognition. In *Proc. International Conference on Computer Vision (ICCV'11)*. IEEE.

Chen, L.-C., Papandreou, G., Kokkinos, I., Murphy, K., and Yuille, A. L. (2014). Semantic image segmentation with deep convolutional nets and fully connected crfs. *arXiv preprint arXiv:1412.7062*.

Collobert, R., Kavukcuoglu, K., and Farabet, C. (2011). Torch7: A matlab-like environment for machine learning. In *BigLearn, NIPS Workshop*, number EPFL-CONF-192376.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). Deep learning. Book in preparation for MIT Press.

Im, D. J., Kim, C. D., Jiang, H., and Memisevic, R. (2016). Generating images with recurrent adversarial networks. *arXiv preprint arXiv:1602.05110*.

Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S., and Darrell, T. (2014). Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the ACM International Conference on Multimedia*, pages 675–678. ACM.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

Le Cun, Y., Bottou, L., and Bengio, Y. (1997). Reading checks with multilayer graph transformer networks. In *Acoustics, Speech, and Signal Processing, 1997. ICASSP-97., 1997 IEEE International Conference on*, volume 1, pages 151–154. IEEE.

Long, J., Shelhamer, E., and Darrell, T. (2015). Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3431–3440.

Oord, A. v. d., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A., and Kavukcuoglu, K. (2016). Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*.

Radford, A., Metz, L., and Chintala, S. (2015). Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*.

Saxe, A., Koh, P. W., Chen, Z., Bhand, M., Suresh, B., and Ng, A. (2011). On random weights and unsupervised feature learning. In L. Getoor and

T. Scheffer, editors, *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, ICML '11, pages 1089–1096, New York, NY, USA. ACM.

Visin, F., Kastner, K., Courville, A. C., Bengio, Y., Matteucci, M., and Cho, K. (2015). Reseg: A recurrent neural network for object segmentation.

Yu, F. and Koltun, V. (2015). Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv:1511.07122*.

Zeiler, M. D. and Fergus, R. (2014). Visualizing and understanding convolutional networks. In *Computer vision–ECCV 2014*, pages 818–833. Springer.

Zeiler, M. D., Taylor, G. W., and Fergus, R. (2011). Adaptive deconvolutional networks for mid and high level feature learning. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2018–2025. IEEE.