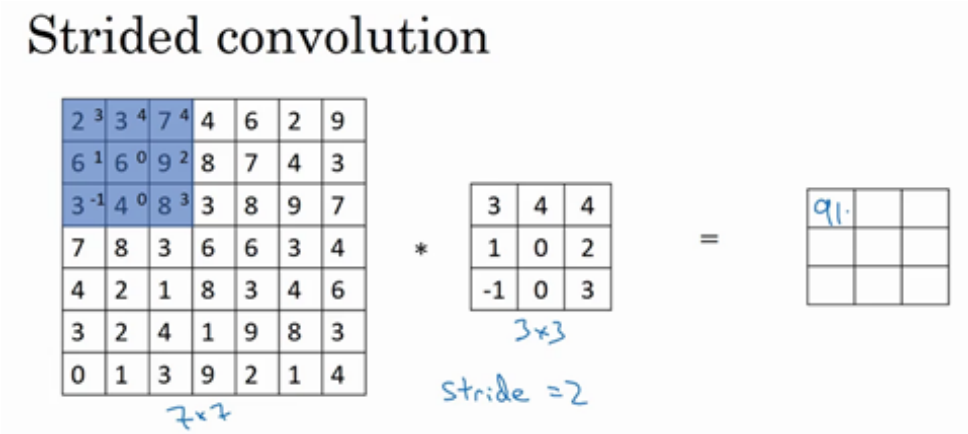
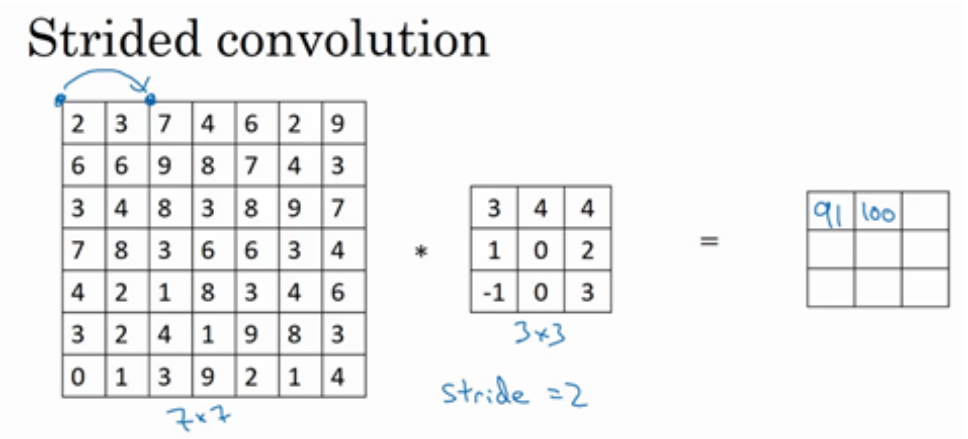
1.5 卷积步长（Strided convolutions）

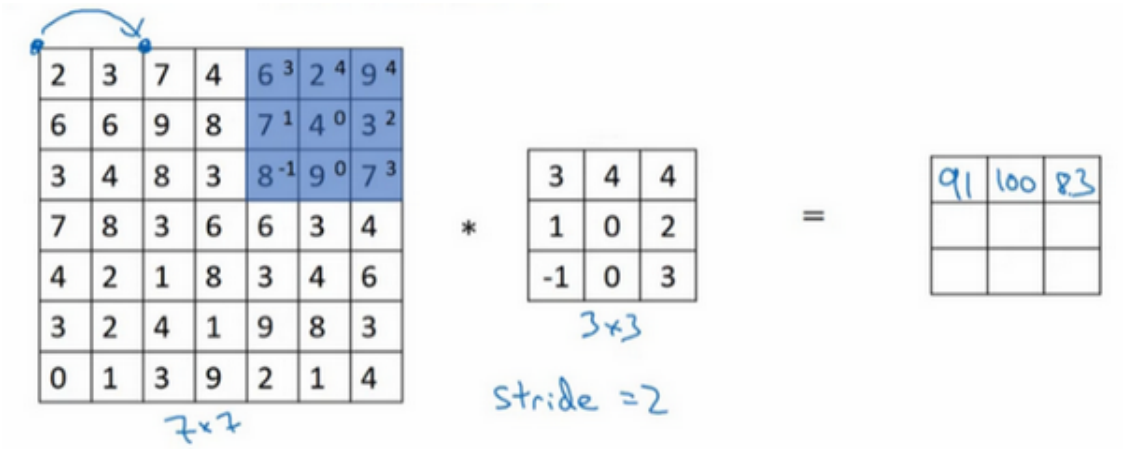
卷积中的步幅是另一个构建卷积神经网络的基本操作，让我向你展示一个例子。



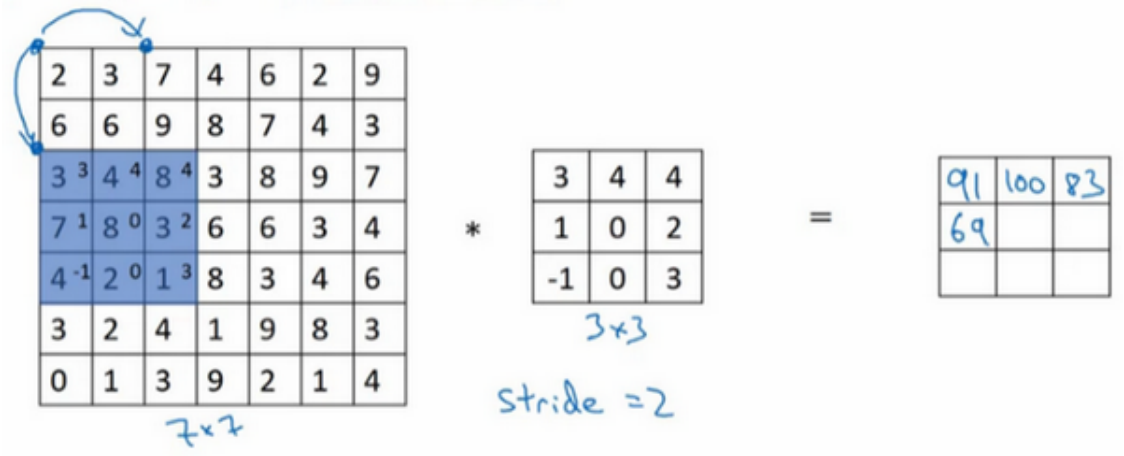
如果你想用 3×3 的过滤器卷积这个 7×7 的图像，和之前不同的是，我们把步幅设置成了 2。你还和之前一样取左上方的 3×3 区域的元素的乘积，再加起来，最后结果为 91。



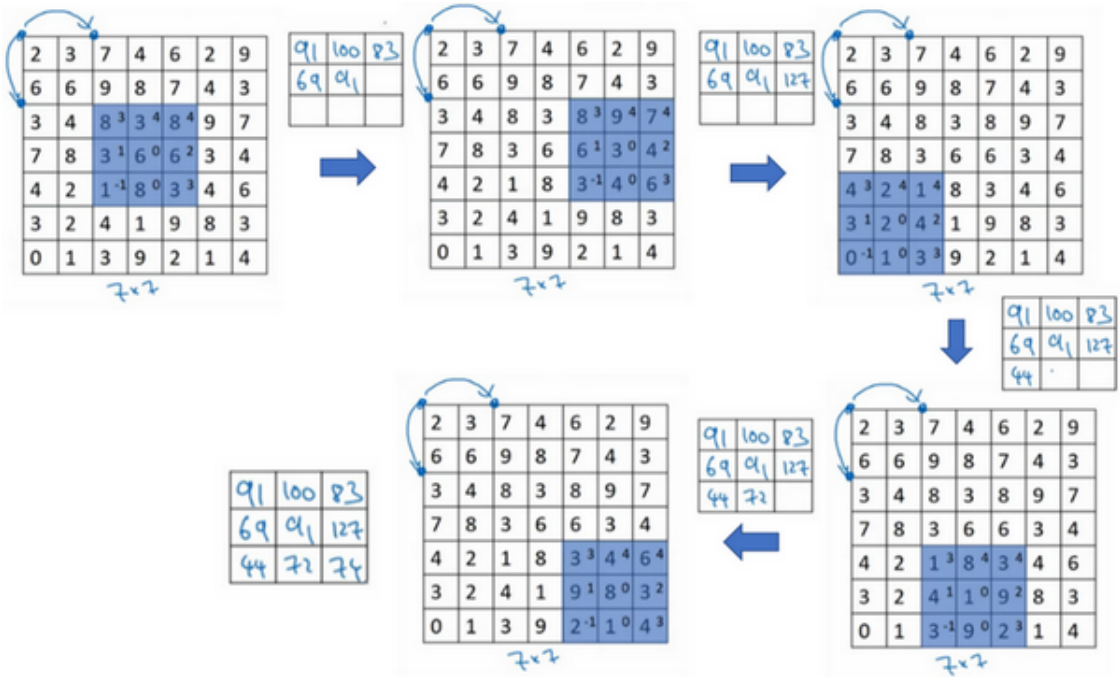
只是之前我们移动蓝框的步长是 1，现在移动的步长是 2，我们让过滤器跳过 2 个步长，注意一下左上角，这个点移动到其后两格的点，跳过了一个位置。然后你还是将每个元素相乘并求和，你将会得到的结果是 100。



现在我们继续，将蓝色框移动两个步长，你将会得到 83 的结果。当你移动到下一行的时候，你也是使用步长 2 而不是步长 1，所以我们将蓝色框移动到这里：



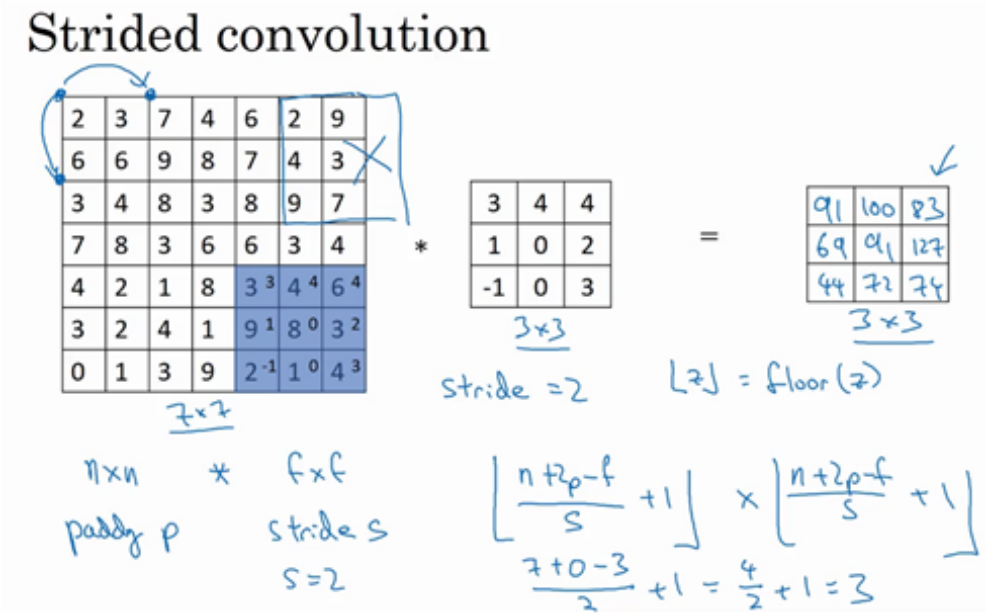
注意到我们跳过了一个位置，得到 69 的结果，现在你继续移动两个步长，会得到 91，127，最后一行分别是 44，72，74。



所以在这个例子中，我们用 3×3 的矩阵卷积一个 7×7 的矩阵，得到一个 3×3 的输出。输入和输出的维度是由下面的公式决定的。

如果你用一个$f \times f$的过滤器卷积一个$n \times n$的图像，你的 padding 为p，步幅为s，在这个例子中$s = 2$，你会得到一个输出，因为现在你不是一次移动一个步子，而是一次移动s个步子，输出于是变为$\frac{n+2p-f}{s} + 1 \times \frac{n+2p-f}{s} + 1$

在我们的这个例子里，$n = 7$，$p = 0$，$f = 3$，$s = 2$，$\frac{7+0-3}{2} + 1 = 3$，即 3×3 的输出。



现在只剩下最后的一个细节了，如果商不是一个整数怎么办？在这种情况下，我们向下取整。⌊ ⌋这是向下取整的符号，这也叫做对z进行地板除(floor)，这意味着z向下取整到最近

的整数。这个原则实现的方式是，你只在蓝框完全包括在图像或填充完的图像内部时，才对它进行运算。如果有任意一个蓝框移动到了外面，那你就不要进行相乘操作，这是一个惯例。你的 3×3 的过滤器必须完全处于图像中或者填充之后的图像区域内才输出相应结果，这就是惯例。因此正确计算输出维度的方法是向下取整，以免$\frac{n+2p-f}{s}$不是整数。

Summary of convolutions

$n \times n$ image $\quad f \times f$ filter

padding p $\quad$ stride s

Output Size:

$$\left\lfloor \frac{n+2p-f}{s} + 1 \right\rfloor \times \left\lfloor \frac{n+2p-f}{s} + 1 \right\rfloor$$

总结一下维度情况，如果你有一个$n \times n$的矩阵或者$n \times n$的图像，与一个$f \times f$的矩阵卷积，或者说$f \times f$的过滤器。**Padding** 是p，步幅为s没输出尺寸就是这样：

Summary of convolutions

$n \times n$ image $\quad f \times f$ filter

padding p $\quad$ stride s

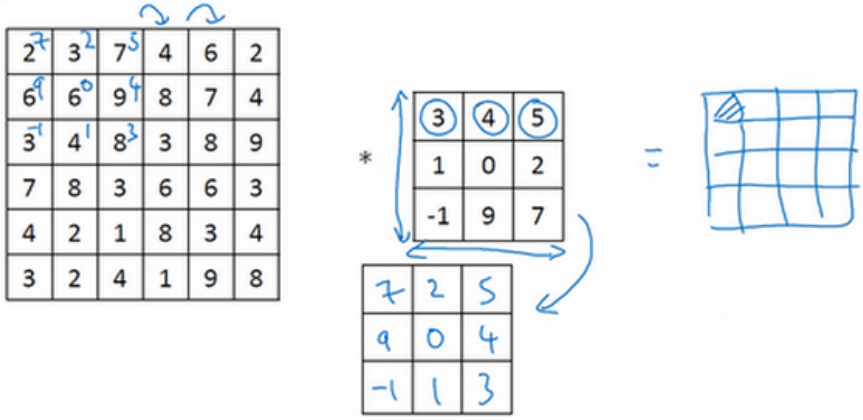
Output Size:

$$\left\lfloor \frac{n+2p-f}{s} + 1 \right\rfloor \times \left\lfloor \frac{n+2p-f}{s} + 1 \right\rfloor$$

可以选择所有的数使结果是整数是挺不错的，尽管一些时候，你不必这样做，只要向下取整也就可以了。你也可以自己选择一些n，f，p和s的值来验证这个输出尺寸的公式是对的。

Technical note on cross-correlation vs. convolution

Convolution in math textbook:



在讲下一部分之前，这里有一个关于互相关和卷积的技术性建议，这不会影响到你构建卷积神经网络的方式，但取决于你读的是数学教材还是信号处理教材，在不同的教材里符号可能不一致。**如果你看的是一本典型的数学教科书，那么卷积的定义是做元素乘积求和，实际上还有一个步骤是你首先要做的，也就是在把这个 6×6 的矩阵和 3×3 的过滤器卷积之前，首先你将 3×3 的过滤器沿水平和垂直轴翻转，所以** $\begin{bmatrix} 3 & 4 & 5 \\ 1 & 0 & 2 \\ -1 & 9 & 7 \end{bmatrix}$ **变为** $\begin{bmatrix} 7 & 2 & 5 \\ 9 & 0 & 4 \\ -1 & 1 & 3 \end{bmatrix}$**，这相当于将 3×3 的过滤器做了个镜像**。然后你再把这个翻转后的矩阵复制到这里（左边的图像矩阵），你要把这个翻转矩阵的元素相乘来计算输出的 4×4 矩阵左上角的元素，如图所示。然后取这 9 个数字，把它们平移一个位置，再平移一格，以此类推。

所以我们在这些视频中定义卷积运算时，我们跳过了这个镜像操作。从技术上讲，我们实际上做的，我们在前面视频中使用的操作，有时被称为互相关（**cross-correlation**）而不是卷积（**convolution**）。**但在深度学习文献中，按照惯例，我们将这（不进行翻转操作）叫做卷积操作。**

总结来说，按照机器学习的惯例，我们通常不进行翻转操作。从技术上说，这个操作可能叫做互相关更好。但在大部分的深度学习文献中都把它叫做卷积运算，因此我们将在这些视频中使用这个约定。如果你读了很多机器学习文献的话，你会发现许多人都把它叫做卷积运算，不需要用到这些翻转。

事实证明在信号处理中或某些数学分支中，在卷积的定义包含翻转，使得卷积运算符拥有这个性质，即$(A * B) * C = A * (B * C)$，这在数学中被称为结合律。这对于一些信号处理应用来说很好，但对于深度神经网络来说它真的不重要，因此省略了这个双重镜像操作，就

简化了代码，并使神经网络也能正常工作。

根据惯例，我们大多数人都叫它卷积，尽管数学家们更喜欢称之为互相关，但这不会影响到你在编程练习中要实现的任何东西，也不会影响你阅读和理解深度学习文献。

现在你已经看到了如何进行卷积，以及如何使用填充，如何在卷积中选择步幅。但到目前为止，我们所使用的是关于矩阵的卷积，例如 6×6 的矩阵。在下一集视频中，你将看到如何对立体进行卷积，这将会使你的卷积变得更加强大，让我们继续下一个视频。