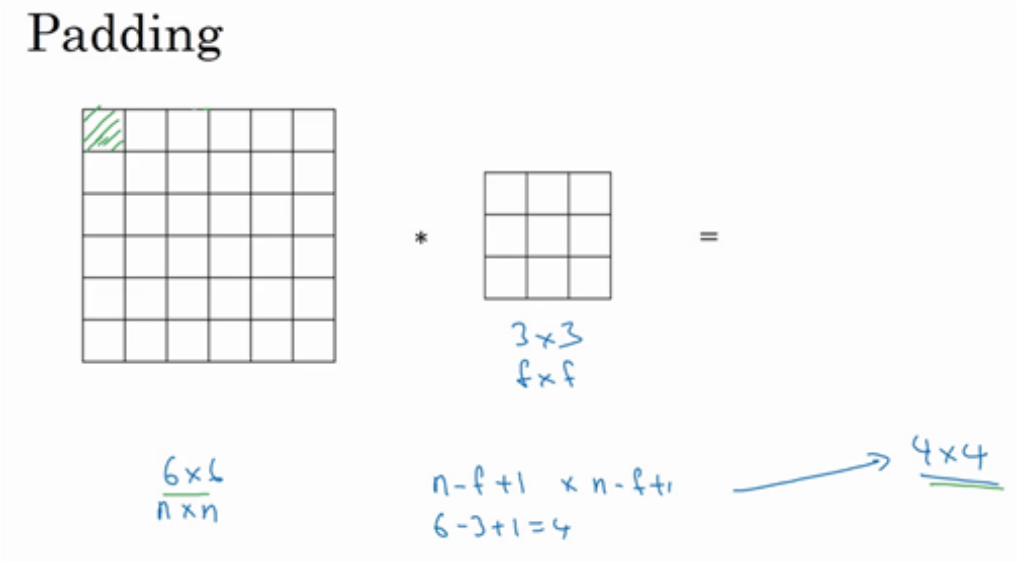
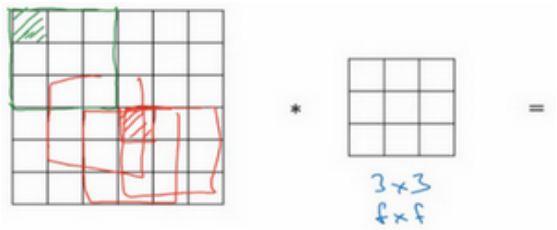
1.4 Padding

为了构建深度神经网络，你需要学会使用的一个基本的卷积操作就是 padding，让我们来看看它是如何工作的。



我们在之前视频中看到，如果你用一个 3×3 的过滤器卷积一个 6×6 的图像，你最后会得到一个 4×4 的输出，也就是一个 4×4 矩阵。那是因为你的 3×3 过滤器在 6×6 矩阵中，只可能有 4×4 种可能的位置。这背后的数学解释是，**如果我们有一个$n \times n$的图像，用$f \times f$的过滤器做卷积，那么输出的维度就是$(n-f+1) \times (n-f+1)$**。在这个例子里是$6-3+1=4$，因此得到了一个 4×4 的输出。

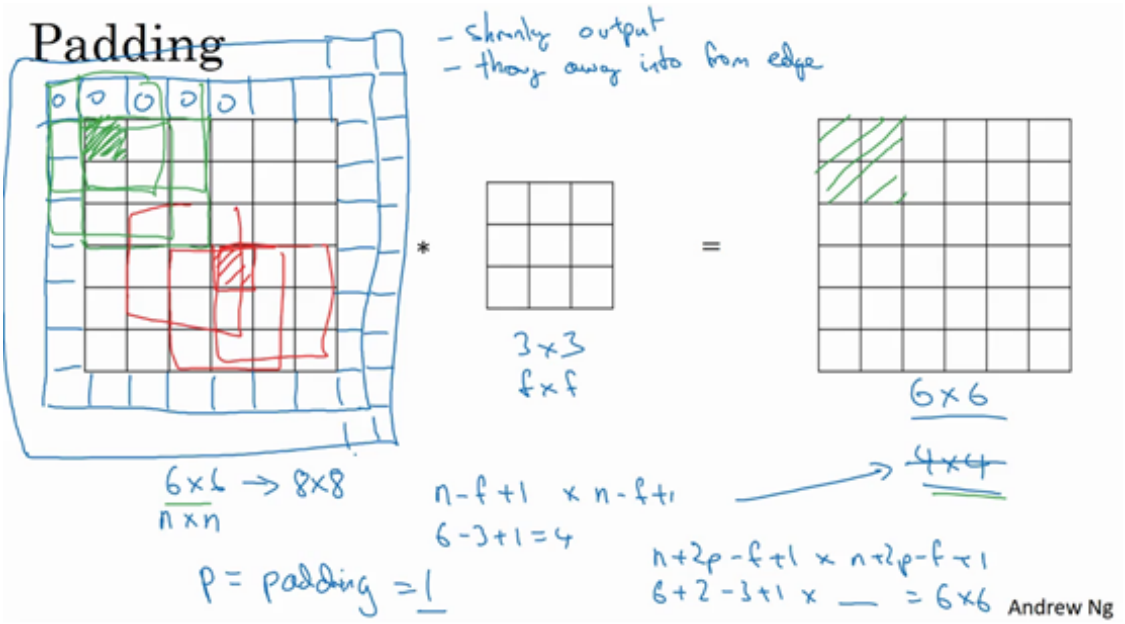
这样的话会有两个缺点，**第一个缺点是每次做卷积操作，你的图像就会缩小**，从 6×6 缩小到 4×4，你可能做了几次之后，你的图像就会变得很小了，可能会缩小到只有 1×1 的大小。你可不想让你的图像在每次识别边缘或其他特征时都缩小，这就是第一个缺点。



第二个缺点时，**如果你注意角落边缘的像素，这个像素点（绿色阴影标记）只被一个输出所触碰或者使用**，因为它位于这个 3×3 的区域的一角。但如果是在中间的像素点，比如这个（红色方框标记），就会有许多 3×3 的区域与之重叠。所以那些在角落或者边缘区域的像素点在输出中采用较少，意味着你丢掉了图像边缘位置的许多信息。

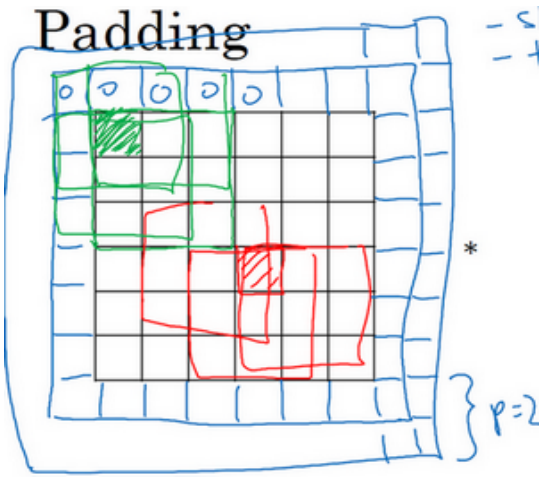
- Shrinks output
- throw away into from edge

为了解决这两个问题，一是输出缩小。当我们建立深度神经网络时，你就会知道你为什么不希望每进行一步操作图像都会缩小。比如当你有 100 层深层的网络，如果图像每经过一层都缩小的话，经过 100 层网络后，你就会得到一个很小的图像，所以这是个问题。另一个问题是图像边缘的大部分信息都丢失了。



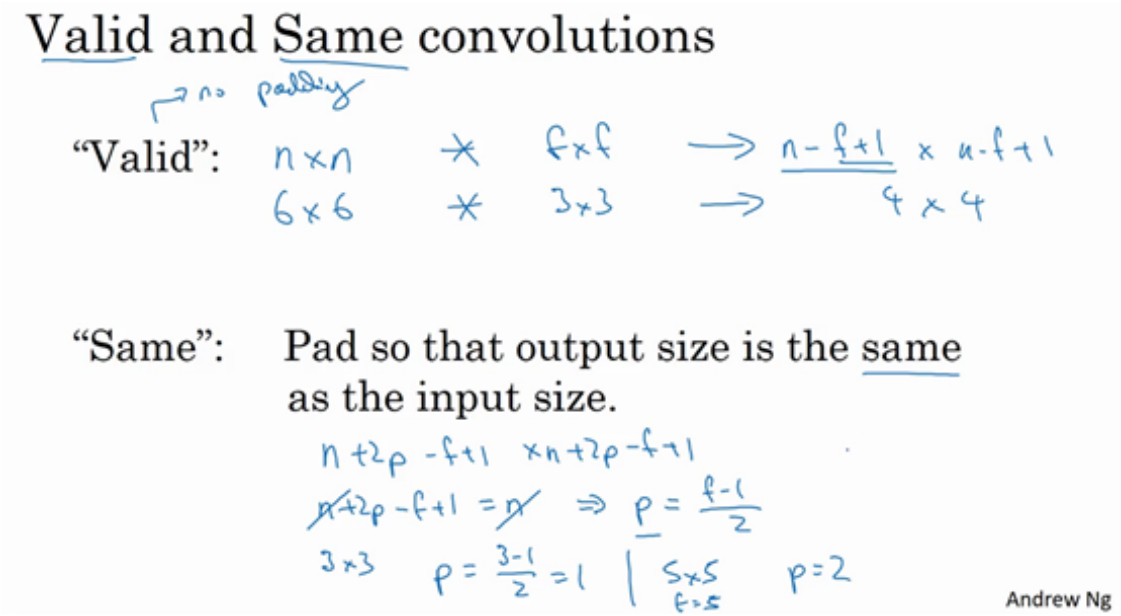
为了解决这些问题，你可以在卷积操作之前填充这幅图像。在这个案例中，你可以沿着图像边缘再填充一层像素。如果你这样操作了，那么 6×6 的图像就被你填充成了一个 8×8 的图像。如果你用 3×3 的图像对这个 8×8 的图像卷积，你得到的输出就不是 4×4 的，而是 6×6 的图像，你就得到了一个尺寸和原始图像 6×6 的图像。习惯上，你可以用 0 去填充，如果p是填充的数量，在这个案例中，$p = 1$，因为我们在周围都填充了一个像素点，输出也就变成了$(n + 2p - f + 1) \times (n + 2p - f + 1)$，所以就变成了$(6 + 2 \times 1 - 3 + 1) \times (6 + 2 \times 1 - 3 + 1) = 6 \times 6$，和输入的图像一样大。这个涂绿的像素点（左边矩阵）影响了输出中的这些格子（右边矩阵）。这样一来，丢失信息或者更准确来说角落或图像边缘的信息发挥的作用较小的这一缺点就被削弱了。

刚才我已经展示过用一个像素点来填充边缘，如果你想的话，也可以填充两个像素点，也就是说在这里填充一层。实际上你还可以填充更多像素。我这里画的这种情况，填充后$p = 2$。



至于选择填充多少像素，通常有两个选择，分别叫做 **Valid 卷积和 Same 卷积。**

Valid 卷积意味着不填充，这样的话，如果你有一个$n \times n$的图像，用一个$f \times f$的过滤器卷积，它将会给你一个$(n-f+1) \times (n-f+1)$维的输出。这类似于我们在前面的视频中展示的例子，有一个 6×6 的图像，通过一个 3×3 的过滤器，得到一个 4×4 的输出。



另一个经常被用到的填充方法叫做 **Same** 卷积，那意味你填充后，你的输出大小和输入大小是一样的。根据这个公式$n-f+1$，当你填充p个像素点，n就变成了$n+2p$，最后公式变为$n+2p-f+1$。因此如果你有一个$n \times n$的图像，用p个像素填充边缘，输出的大小就是这样的$(n+2p-f+1) \times (n+2p-f+1)$。如果你想让$n+2p-f+1=n$的话，使得输出和输入大小相等，如果你用这个等式求解p，那么$\boldsymbol{p=(f-1)/2}$。所以当f是一个奇数的时候，只要选择相应的填充尺寸，你就能确保得到和输入相同尺寸的输出。这也是为什么前面的例子，当过滤器是 3×3 时，和上一张幻灯片的例子一样，使得输出尺寸等于输入尺寸，所需要的填充是(3-1)/2，也就是 1 个像素。另一个例子，当你的过滤器是 5×5，如果$f=5$，

然后代入那个式子，你就会发现需要 2 层填充使得输出和输入一样大，这是过滤器 5×5 的情况。

习惯上，计算机视觉中，f通常是奇数，甚至可能都是这样。你很少看到一个偶数的过滤器在计算机视觉里使用，我认为有两个原因。

其中一个可能是，如果f是一个偶数，那么你只能使用一些不对称填充。只有f是奇数的情况下，**Same** 卷积才会有自然的填充，我们可以以同样的数量填充四周，而不是左边填充多一点，右边填充少一点，这样不对称的填充。

第二个原因是当你有一个奇数维过滤器，比如 3×3 或者 5×5 的，它就有一个中心点。有时在计算机视觉里，如果有一个中心像素点会更方便，便于指出过滤器的位置。

也许这些都不是为什么f通常是奇数的充分原因，但如果你看了卷积的文献，你经常会看到 3×3 的过滤器，你也可能会看到一些 5×5，7×7 的过滤器。后面我们也会谈到 1×1 的过滤器，以及什么时候它是有意义的。但是习惯上，我推荐你只使用奇数的过滤器。我想如果你使用偶数 f 也可能会得到不错的表现，如果遵循计算机视觉的惯例，我通常使用奇数值的f。

你已经看到如何使用 **padding** 卷积，为了指定卷积操作中的 **padding**，你可以指定p的值。也可以使用 **Valid** 卷积，也就是$p = 0$。也可使用 **Same** 卷积填充像素，使你的输出和输入大小相同。以上就是 **padding**，在接下来的视频中我们讨论如何在卷积中设置步长。