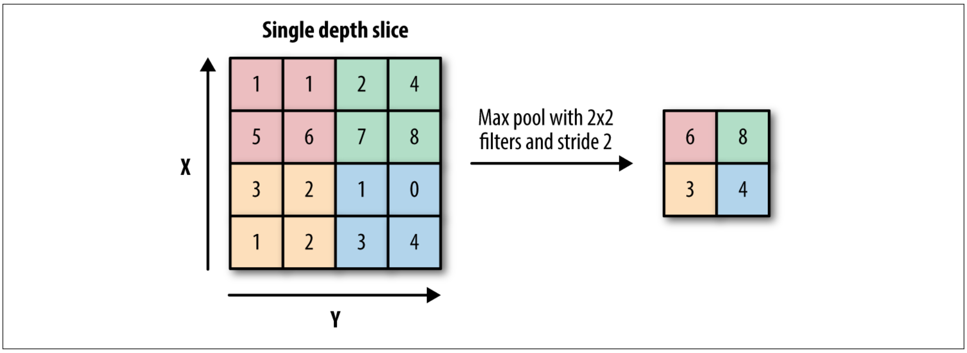
Pooling Layers

In the previous section, we introduced the notion of convolutional kernels. These kernels apply learnable nonlinear transformations to local patches of inputs. These transformations are learnable, and by the universal approximation theorem, capable of learning arbitrarily complex input transformations on local patches. This flexibility gives convolutional kernels much of their power. But at the same time, having many learnable weights in a deep convolutional network can slow training.

Instead of using a learnable transformation, it's possible to instead use a fixed nonlinear transformation in order to reduce the computational cost of training a convolutional network. A popular fixed nonlinearity is "max pooling." Such layers select and output the maximally activating input within each local receptive patch. Figure 6-6 demonstrates this process. Pooling layers are useful for reducing the dimensionality of input data in a structured fashion. More mathematically, they take a local receptive field and replace the nonlinear activation function at each portion of the field with the max (or min or average) function.



1	1	2	4
5	6	7	8
3	2	1	0
1	2	3	4

6	8
3	4

Figure 6-6. An illustration of a max pooling layer. Notice how the maximal value in each colored region (each local receptive field) is reported in the output.

Pooling layers have become less useful as hardware has improved. While pooling is still useful as a dimensionality reduction technique, recent research tends to avoid using pooling layers due to their inherent lossiness (it's not possible to back out of pooled data which pixel in the input originated the reported activation). Nonetheless, pooling appears in many standard convolutional architectures so it's worth understanding.

Constructing Convolutional Networks

A simple convolutional architecture applies a series of convolutional layers and pooling layers to its input to learn a complex function on the input image data. There are a lot of details in forming these networks, but at its heart, architecture design is simply an elaborate form of Lego stacking. Figure 6-7 demonstrates how a convolutional architecture might be built up out of constituent blocks.

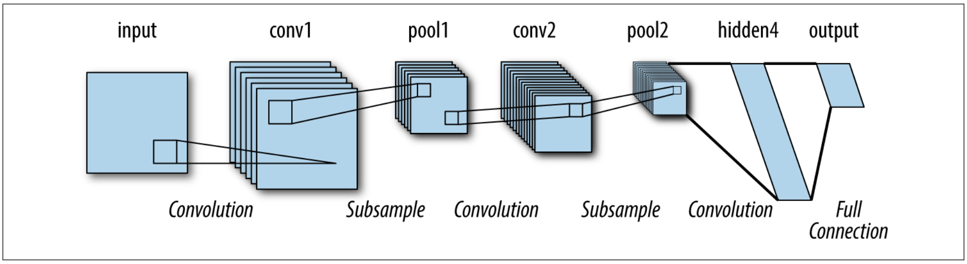


Figure 6-7. An illustration of a simple convolutional architecture constructed out of stacked convolutional and pooling layers.

Dilated Convolutions

Dilated or atrous convolutions are a newly popular form of convolutional layer. The insight here is to leave gaps in the local receptive field for each neuron (atrous means *a trous*, or "with holes" in French). The basic concept is an old one in signal processing that has recently found some good traction in the convolutional literature.

The core advantage to the atrous convolution is the increase in visible area for each neuron. Let's consider a convolution architecture whose first layer is a vanilla convolutional with 3 × 3 local receptive fields. Then a neuron one layer deeper in the architecture in a second vanilla convolutional layer has receptive depth 5 × 5 (each neuron in a local receptive field of the second layer itself has a local receptive field in the first layer). Then, a neuron two layers deeper has receptive view 7 × 7. In general, a neuron N layers within the convolutional architecture has receptive view of size $(2N + 1) \times (2N + 1)$. This linear growth in receptive view is fine for smaller images, but quickly becomes a liability for large images.

The atrous convolution enables exponential growth in the visible receptive field by leaving gaps in its local receptive fields. A "1-dilated" convolution leaves no gaps, while a "2-dilated" convolution leaves one gap between each local receptive field element. Stacking dilated layers leads to exponentially increasing local receptive field sizes. Figure 6-8 illustrates this exponential increase.

Dilated convolutions can be very useful for large images. For example, medical images can stretch thousands of pixels in every dimension. Creating vanilla convolutional networks that have global understanding could require unreasonably deep networks. Using dilated convolutions could enable networks to better understand the global structure of such images.