

to define a separate stride for each direction of motion. See figure 9.12 for an illustration.

One essential feature of any convolutional network implementation is the ability to implicitly zero-pad the input \mathbf{V} in order to make it wider. Without this feature, the width of the representation shrinks by one pixel less than the kernel width at each layer. Zero padding the input allows us to control the kernel width and the size of the output independently. Without zero padding, we are forced to choose between shrinking the spatial extent of the network rapidly and using small kernels—both scenarios that significantly limit the expressive power of the network. See figure 9.13 for an example.

Three special cases of the zero-padding setting are worth mentioning. One is the extreme case in which no zero-padding is used whatsoever, and the convolution kernel is only allowed to visit positions where the entire kernel is contained entirely within the image. In MATLAB terminology, this is called **valid** convolution. In this case, all pixels in the output are a function of the same number of pixels in the input, so the behavior of an output pixel is somewhat more regular. However, the size of the output shrinks at each layer. If the input image has width m and the kernel has width k , the output will be of width $m - k + 1$. The rate of this shrinkage can be dramatic if the kernels used are large. Since the shrinkage is greater than 0, it limits the number of convolutional layers that can be included in the network. As layers are added, the spatial dimension of the network will eventually drop to 1×1 , at which point additional layers cannot meaningfully be considered convolutional. Another special case of the zero-padding setting is when just enough zero-padding is added to keep the size of the output equal to the size of the input. MATLAB calls this **same** convolution. In this case, the network can contain as many convolutional layers as the available hardware can support, since the operation of convolution does not modify the architectural possibilities available to the next layer. However, the input pixels near the border influence fewer output pixels than the input pixels near the center. This can make the border pixels somewhat underrepresented in the model. This motivates the other extreme case, which MATLAB refers to as **full** convolution, in which enough zeroes are added for every pixel to be visited k times in each direction, resulting in an output image of width $m + k - 1$. In this case, the output pixels near the border are a function of fewer pixels than the output pixels near the center. This can make it difficult to learn a single kernel that performs well at all positions in the convolutional feature map. Usually the optimal amount of zero padding (in terms of test set classification accuracy) lies somewhere between “valid” and “same” convolution.