To define tiled convolution algebraically, let k be a 6-D tensor, where two of the dimensions correspond to different locations in the output map. Rather than having a separate index for each location in the output map, output locations cycle through a set of t different choices of kernel stack in each direction. If t is equal to the output width, this is the same as a locally connected layer.

$$Z_{i,j,k} = \sum_{l,m,n} V_{l,j+m-1,k+n-1} K_{i,l,m,n,j\%t+1,k\%t+1}\,, \tag{9.10}$$

where $\%$ is the modulo operation, with $t\%t = 0$, $(t+1)\%t = 1$, etc. It is straightforward to generalize this equation to use a different tiling range for each dimension.

Both locally connected layers and tiled convolutional layers have an interesting interaction with max-pooling: the detector units of these layers are driven by different filters. If these filters learn to detect different transformed versions of the same underlying features, then the max-pooled units become invariant to the learned transformation (see figure 9.9). Convolutional layers are hard-coded to be invariant specifically to translation.

Other operations besides convolution are usually necessary to implement a convolutional network. To perform learning, one must be able to compute the gradient with respect to the kernel, given the gradient with respect to the outputs. In some simple cases, this operation can be performed using the convolution operation, but many cases of interest, including the case of stride greater than 1, do not have this property.

Recall that convolution is a linear operation and can thus be described as a matrix multiplication (if we first reshape the input tensor into a flat vector). The matrix involved is a function of the convolution kernel. The matrix is sparse and each element of the kernel is copied to several elements of the matrix. This view helps us to derive some of the other operations needed to implement a convolutional network.

Multiplication by the transpose of the matrix defined by convolution is one such operation. This is the operation needed to back-propagate error derivatives through a convolutional layer, so it is needed to train convolutional networks that have more than one hidden layer. This same operation is also needed if we wish to reconstruct the visible units from the hidden units (Simard *et al.*, 1992). Reconstructing the visible units is an operation commonly used in the models described in part III of this book, such as autoencoders, RBMs, and sparse coding. Transpose convolution is necessary to construct convolutional versions of those models. Like the kernel gradient operation, this input gradient operation can be