

In some cases, we do not actually want to use convolution, but rather locally connected layers (LeCun, 1986, 1989). In this case, the adjacency matrix in the graph of our MLP is the same, but every connection has its own weight, specified by a 6-D tensor  $\mathbf{W}$ . The indices into  $\mathbf{W}$  are respectively:  $i$ , the output channel,  $j$ , the output row,  $k$ , the output column,  $l$ , the input channel,  $m$ , the row offset within the input, and  $n$ , the column offset within the input. The linear part of a locally connected layer is then given by

$$Z_{i,j,k} = \sum_{l,m,n} [V_{l,j+m-1,k+n-1} w_{i,j,k,l,m,n}]. \quad (9.9)$$

This is sometimes also called **unshared convolution**, because it is a similar operation to discrete convolution with a small kernel, but without sharing parameters across locations. Figure 9.14 compares local connections, convolution, and full connections.

Locally connected layers are useful when we know that each feature should be a function of a small part of space, but there is no reason to think that the same feature should occur across all of space. For example, if we want to tell if an image is a picture of a face, we only need to look for the mouth in the bottom half of the image.

It can also be useful to make versions of convolution or locally connected layers in which the connectivity is further restricted, for example to constrain each output channel  $i$  to be a function of only a subset of the input channels  $l$ . A common way to do this is to make the first  $m$  output channels connect to only the first  $n$  input channels, the second  $m$  output channels connect to only the second  $n$  input channels, and so on. See figure 9.15 for an example. Modeling interactions between few channels allows the network to have fewer parameters in order to reduce memory consumption and increase statistical efficiency, and also reduces the amount of computation needed to perform forward and back-propagation. It accomplishes these goals without reducing the number of hidden units.

**Tiled convolution** (Gregor and LeCun, 2010a; Le *et al.*, 2010) offers a compromise between a convolutional layer and a locally connected layer. Rather than learning a separate set of weights at *every* spatial location, we learn a set of kernels that we rotate through as we move through space. This means that immediately neighboring locations will have different filters, like in a locally connected layer, but the memory requirements for storing the parameters will increase only by a factor of the size of this set of kernels, rather than the size of the entire output feature map. See figure 9.16 for a comparison of locally connected layers, tiled convolution, and standard convolution.