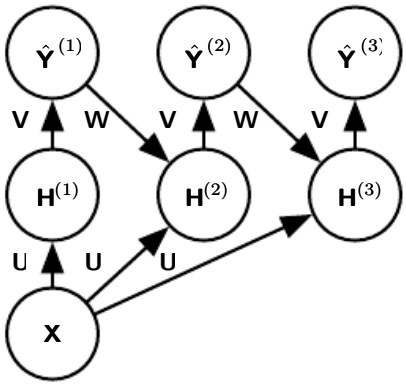

Figure 9.17: An example of a recurrent convolutional network for pixel labeling. The input is an image tensor $\mathbf{X}$, with axes corresponding to image rows, image columns, and channels (red, green, blue). The goal is to output a tensor of labels $\hat{Y}$, with a probability distribution over labels for each pixel. This tensor has axes corresponding to image rows, image columns, and the different classes. Rather than outputting $\hat{Y}$ in a single shot, the recurrent network iteratively refines its estimate $\hat{Y}$ by using a previous estimate of $\hat{Y}$ as input for creating a new estimate. The same parameters are used for each updated estimate, and the estimate can be refined as many times as we wish. The tensor of convolution kernels $\mathbf{U}$ is used on each step to compute the hidden representation given the input image. The kernel tensor $\mathbf{V}$ is used to produce an estimate of the labels given the hidden values. On all but the first step, the kernels $\mathbf{W}$ are convolved over $\hat{Y}$ to provide input to the hidden layer. On the first time step, this term is replaced by zero. Because the same parameters are used on each step, this is an example of a recurrent network, as described in chapter 10.

input plane, as shown in figure 9.13. In the kinds of architectures typically used for classification of a single object in an image, the greatest reduction in the spatial dimensions of the network comes from using pooling layers with large stride. In order to produce an output map of similar size as the input, one can avoid pooling altogether (Jain *et al.*, 2007). Another strategy is to simply emit a lower-resolution grid of labels (Pinheiro and Collobert, 2014, 2015). Finally, in principle, one could use a pooling operator with unit stride.

One strategy for pixel-wise labeling of images is to produce an initial guess of the image labels, then refine this initial guess using the interactions between neighboring pixels. Repeating this refinement step several times corresponds to using the same convolutions at each stage, sharing weights between the last layers of the deep net (Jain *et al.*, 2007). This makes the sequence of computations performed by the successive convolutional layers with weights shared across layers a particular kind of recurrent network (Pinheiro and Collobert, 2014, 2015). Figure 9.17 shows the architecture of such a recurrent convolutional network.