

that copies its input \mathbf{x} to an approximate reconstruction \mathbf{r} using the function $\mathbf{W}^\top \mathbf{W} \mathbf{x}$. It is common for more general autoencoders to use multiplication by the transpose of the weight matrix just as PCA does. To make such models convolutional, we can use the function h to perform the transpose of the convolution operation. Suppose we have hidden units \mathbf{H} in the same format as \mathbf{Z} and we define a reconstruction

$$\mathbf{R} = h(\mathbf{K}, \mathbf{H}, s). \quad (9.14)$$

In order to train the autoencoder, we will receive the gradient with respect to \mathbf{R} as a tensor \mathbf{E} . To train the decoder, we need to obtain the gradient with respect to \mathbf{K} . This is given by $g(\mathbf{H}, \mathbf{E}, s)$. To train the encoder, we need to obtain the gradient with respect to \mathbf{H} . This is given by $c(\mathbf{K}, \mathbf{E}, s)$. It is also possible to differentiate through g using c and h , but these operations are not needed for the back-propagation algorithm on any standard network architectures.

Generally, we do not use only a linear operation in order to transform from the inputs to the outputs in a convolutional layer. We generally also add some bias term to each output before applying the nonlinearity. This raises the question of how to share parameters among the biases. For locally connected layers it is natural to give each unit its own bias, and for tiled convolution, it is natural to share the biases with the same tiling pattern as the kernels. For convolutional layers, it is typical to have one bias per channel of the output and share it across all locations within each convolution map. However, if the input is of known, fixed size, it is also possible to learn a separate bias at each location of the output map. Separating the biases may slightly reduce the statistical efficiency of the model, but also allows the model to correct for differences in the image statistics at different locations. For example, when using implicit zero padding, detector units at the edge of the image receive less total input and may need larger biases.

9.6 Structured Outputs

Convolutional networks can be used to output a high-dimensional, structured object, rather than just predicting a class label for a classification task or a real value for a regression task. Typically this object is just a tensor, emitted by a standard convolutional layer. For example, the model might emit a tensor \mathbf{S} , where $S_{i,j,k}$ is the probability that pixel (j, k) of the input to the network belongs to class i . This allows the model to label every pixel in an image and draw precise masks that follow the outlines of individual objects.

One issue that often comes up is that the output plane can be smaller than the