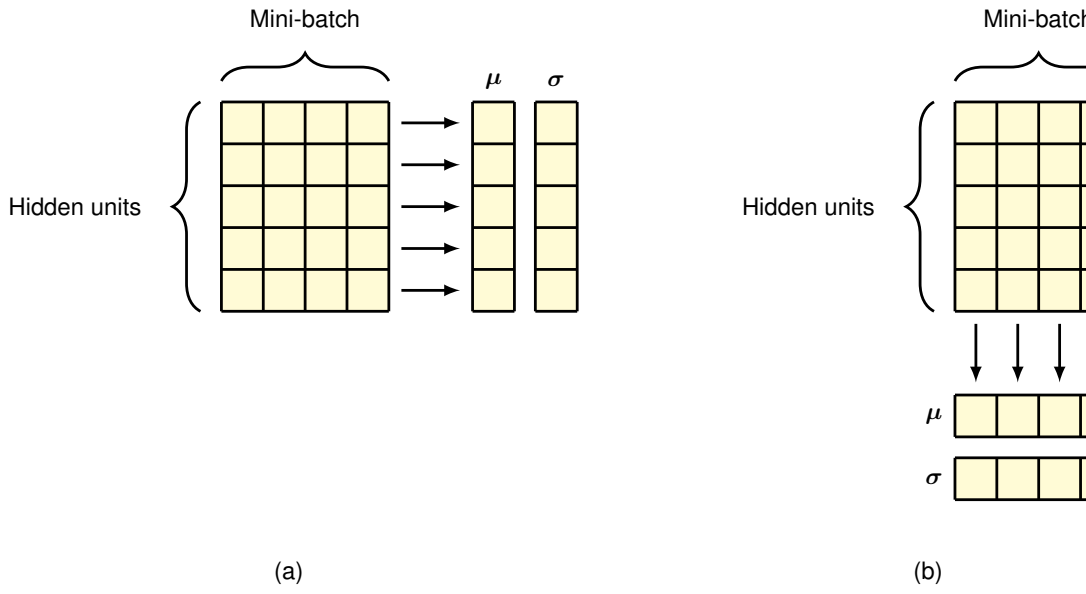

Figure 7.8 Illustration of batch normalization and layer normalization in a neural network. In batch normalization, shown in (a), the mean and variance are computed across the mini-batch separately for each hidden unit. In layer normalization, shown in (b), the mean and variance are computed across the hidden units separately for each data point.

reduce its representational capability. We can compensate for this by re-scaling the pre-activations of the batch to have mean β_i and standard deviation γ_i using

$$\widetilde{a}_{ni} = \gamma_i \widehat{a}_{ni} + \beta_i \tag{7.55}$$

where β_i and γ_i are adaptive parameters that are learned by gradient descent jointly with the weights and biases of the network. These learnable parameters represent a key difference compared to input data normalization.

Section 7.4.1

It might appear that the transformation (7.55) has simply undone the effect of the batch normalization since the mean and variance can now adapt to arbitrary values again. However, the crucial difference is in the way the parameters evolve during training. For the original network, the mean and variance across a mini-batch are determined by a complex function of all the weights and biases in the layer, whereas in the representation given by (7.55), they are determined directly by independent parameters β_i and γ_i, which turn out to be much easier to learn during gradient descent.

Equations (7.52) – (7.55) describe a transformation of the variables that is differentiable with respect to the learnable parameters β_i and γ_i. This can be viewed as an additional layer in the neural network, and so each standard hidden layer can be followed by a batch normalization layer. The structure of the batch-normalization process is illustrated in Figure 7.8.

Once the network is trained and we want to make predictions on new data, we