no longer have the training mini-batches available, and we cannot determine a mean and variance from just one data example. To solve this, we could in principle evaluate μ_i and σ_i^2 for each layer across the whole training set after we have made the final update to the weights and biases. However, this would involve processing the whole data set just to evaluate these quantities and is therefore usually too expensive. Instead, we compute moving averages throughout the training phase:

$$\overline{\mu}_i^{(\tau)} = \alpha\overline{\mu}_i^{(\tau-1)} + (1-\alpha)\mu_i \tag{7.56}$$

$$\overline{\sigma}_i^{(\tau)} = \alpha\overline{\sigma}_i^{(\tau-1)} + (1-\alpha)\sigma_i \tag{7.57}$$

where $0 \leqslant \alpha \leqslant 1$. These moving averages play no role during training but are used to process new data points during the inference phase.

Although batch normalization is very effective in practice, there is uncertainty as to why it works so well. Batch normalization was originally motivated by noting that updates to weights in earlier layers of the network change the distribution of values seen by later layers, a phenomenon called *internal covariate shift*. However, later studies (Santurkar *et al.*, 2018) suggest that covariate shift is not a significant factor and that the improved training results from an improvement in the smoothness of the error function landscape.

7.4.3 Layer normalization

With batch normalization, if the batch size is too small then the estimates of the mean and variance become too noisy. Also, for very large training sets, the mini-batches may be split across different GPUs, making global normalization across the mini-batch inefficient. An alternative to normalizing across examples within a mini-batch for each hidden unit separately is to normalize across the hidden-unit values for each data point separately. This is known as *layer normalization* (Ba, Kiros, and Hinton, 2016). It was introduced in the context of recurrent neural networks where the distributions change after each time step making batch normalization infeasible. However, it is useful in other architectures such as transformer networks.

Section 12.2.5

Chapter 12

By analogy with batch normalization, we therefore make the following transformation:

$$\mu_n = \frac{1}{M}\sum_{i=1}^{M} a_{ni} \tag{7.58}$$

$$\sigma_n^2 = \frac{1}{M}\sum_{i=1}^{M} (a_{ni} - \mu_i)^2 \tag{7.59}$$

$$\widehat{a}_{ni} = \frac{a_{ni} - \mu_n}{\sqrt{\sigma_n^2 + \delta}} \tag{7.60}$$

where the sums $i = 1, \ldots, M$ are taken over all hidden units in the layer. As with batch normalization, additional learnable mean and standard deviation parameters are introduced for each hidden unit separately in the form (7.55). Note that the same normalization function can be employed during training and during inference, and