

Two quantities are displayed during training: the loss of the network over the training data, and the accuracy of the network over the training data.

We quickly reach an accuracy of 0.989 (98.9%) on the training data. Now let's check that the model performs well on the test set, too:

```
>>> test_loss, test_acc = network.evaluate(test_images, test_labels)
>>> print('test_acc:', test_acc)
test_acc: 0.9785
```

The test-set accuracy turns out to be 97.8%—that's quite a bit lower than the training set accuracy. This gap between training accuracy and test accuracy is an example of *overfitting*: the fact that machine-learning models tend to perform worse on new data than on their training data. Overfitting is a central topic in chapter 3.

This concludes our first example—you just saw how you can build and train a neural network to classify handwritten digits in less than 20 lines of Python code. In the next chapter, I'll go into detail about every moving piece we just previewed and clarify what's going on behind the scenes. You'll learn about tensors, the data-storing objects going into the network; tensor operations, which layers are made of; and gradient descent, which allows your network to learn from its training examples.