```
# Loop through training steps.
for step in xrange(int(num_epochs * train_size) // BATCH_SIZE):
  # Compute the offset of the current minibatch in the data.
  # Note that we could use better randomization across epochs.
  offset = (step * BATCH_SIZE) % (train_size - BATCH_SIZE)
  batch_data = train_data[offset:(offset + BATCH_SIZE), ...]
  batch_labels = train_labels[offset:(offset + BATCH_SIZE)]
  # This dictionary maps the batch data (as a NumPy array) to the
  # node in the graph it should be fed to.
  feed_dict = {train_data_node: batch_data,
               train_labels_node: batch_labels}
  # Run the optimizer to update weights.
  sess.run(optimizer, feed_dict=feed_dict)
  # print some extra information once reach the evaluation frequency
  if step % EVAL_FREQUENCY == 0:
    # fetch some extra nodes' data
    l, lr, predictions = sess.run([loss, learning_rate,
                                   train_prediction],
                                  feed_dict=feed_dict)
    elapsed_time = time.time() - start_time
    start_time = time.time()
    print('Step %d (epoch %.2f), %.1f ms' %
          (step, float(step) * BATCH_SIZE / train_size,
           1000 * elapsed_time / EVAL_FREQUENCY))
    print('Minibatch loss: %.3f, learning rate: %.6f' % (l, lr))
    print('Minibatch error: %.1f%%'
          % error_rate(predictions, batch_labels))
    print('Validation error: %.1f%%' % error_rate(
        eval_in_batches(validation_data, sess), validation_labels))
    sys.stdout.flush()
# Finally print the result!
test_error = error_rate(eval_in_batches(test_data, sess),
                        test_labels)
print('Test error: %.1f%%' % test_error)
```

## Challenge for the Reader

Try training the network yourself. You should be able to achieve test error of < 1%!

# Review

In this chapter, we have shown you the basic concepts of convolutional network design. These concepts include convolutional and pooling layers that constitute core building blocks of convolutional networks. We then discussed applications of convolutional architectures such as object detection, image segmentation, and image generation. We ended the chapter with an in-depth case study that showed you how to train a convolutional architecture on the MNIST handwritten digit dataset.