iterate until you find the right number of layers to reuse. If you have plenty of training data, you may try replacing the top hidden layers instead of dropping them, and even add more hidden layers.

Model Zoos

Where can you find a neural network trained for a task similar to the one you want to tackle? The first place to look is obviously in your own catalog of models. This is one good reason to save all your models and organize them so you can retrieve them later easily. Another option is to search in a *model zoo*. Many people train Machine Learning models for various tasks and kindly release their pretrained models to the public.

TensorFlow has its own model zoo available at *https://github.com/tensorflow/models*. In particular, it contains most of the state-of-the-art image classification nets such as VGG, Inception, and ResNet (see Chapter 13, and check out the *models/slim* directory), including the code, the pretrained models, and tools to download popular image datasets.

Another popular model zoo is Caffe's Model Zoo. It also contains many computer vision models (e.g., LeNet, AlexNet, ZFNet, GoogLeNet, VGGNet, inception) trained on various datasets (e.g., ImageNet, Places Database, CIFAR10, etc.). Saumitro Dasgupta wrote a converter, which is available at *https://github.com/ethereon/caffe-tensorflow*.

Unsupervised Pretraining

Suppose you want to tackle a complex task for which you don't have much labeled training data, but unfortunately you cannot find a model trained on a similar task. Don't lose all hope! First, you should of course try to gather more labeled training data, but if this is too hard or too expensive, you may still be able to perform *unsupervised pretraining* (see Figure 11-5). That is, if you have plenty of unlabeled training data, you can try to train the layers one by one, starting with the lowest layer and then going up, using an unsupervised feature detector algorithm such as *Restricted Boltzmann Machines* (RBMs; see Appendix E) or autoencoders (see Chapter 15). Each layer is trained on the output of the previously trained layers (all layers except the one being trained are frozen). Once all layers have been trained this way, you can fine-tune the network using supervised learning (i.e., with backpropagation).

This is a rather long and tedious process, but it often works well; in fact, it is this technique that Geoffrey Hinton and his team used in 2006 and which led to the revival of neural networks and the success of Deep Learning. Until 2010, unsupervised pretraining (typically using RBMs) was the norm for deep nets, and it was only after the vanishing gradients problem was alleviated that it became much more common to train DNNs purely using backpropagation. However, unsupervised pretraining (today typically using autoencoders rather than RBMs) is still a good option when

you have a complex task to solve, no similar model you can reuse, and little labeled training data but plenty of unlabeled training data.[9]

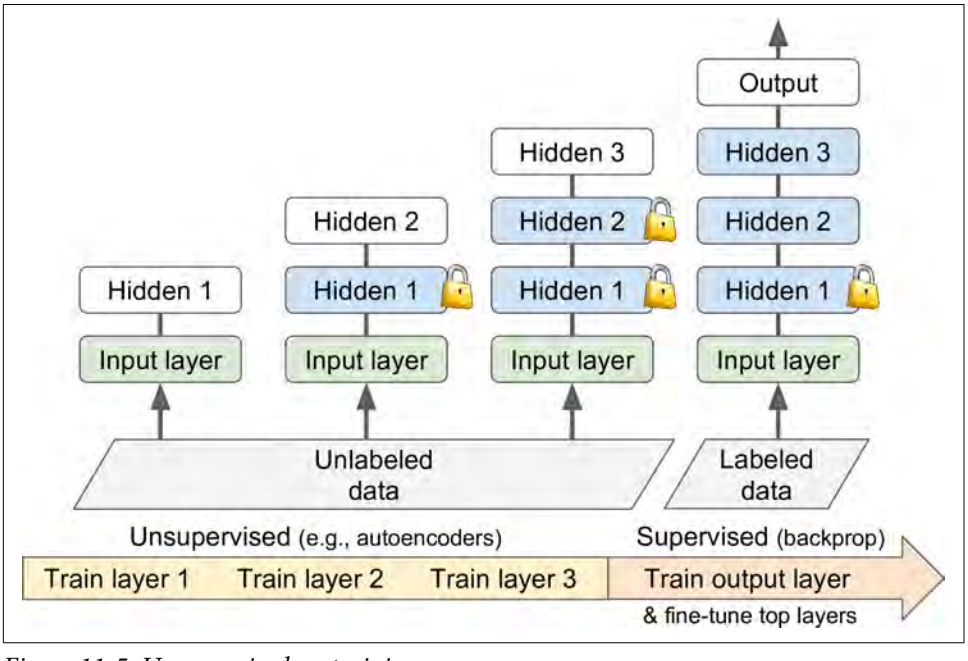

Figure 11-5. Unsupervised pretraining

Pretraining on an Auxiliary Task

One last option is to train a first neural network on an auxiliary task for which you can easily obtain or generate labeled training data, then reuse the lower layers of that network for your actual task. The first neural network's lower layers will learn feature detectors that will likely be reusable by the second neural network.

For example, if you want to build a system to recognize faces, you may only have a few pictures of each individual—clearly not enough to train a good classifier. Gathering hundreds of pictures of each person would not be practical. However, you could gather a lot of pictures of random people on the internet and train a first neural network to detect whether or not two different pictures feature the same person. Such a

9 Another option is to come up with a supervised task for which you can easily gather a lot of labeled training data, then use transfer learning, as explained earlier. For example, if you want to train a model to identify your friends in pictures, you could download millions of faces on the internet and train a classifier to detect whether two faces are identical or not, then use this classifier to compare a new picture with each picture of your friends.