Note that the use of convolution for processing variable sized inputs only makes sense for inputs that have variable size because they contain varying amounts of observation of the same kind of thing—different lengths of recordings over time, different widths of observations over space, etc. Convolution does not make sense if the input has variable size because it can optionally include different kinds of observations. For example, if we are processing college applications, and our features consist of both grades and standardized test scores, but not every applicant took the standardized test, then it does not make sense to convolve the same weights over both the features corresponding to the grades and the features corresponding to the test scores.

9.8 Efficient Convolution Algorithms

Modern convolutional network applications often involve networks containing more than one million units. Powerful implementations exploiting parallel computation resources, as discussed in section 12.1, are essential. However, in many cases it is also possible to speed up convolution by selecting an appropriate convolution algorithm.

Convolution is equivalent to converting both the input and the kernel to the frequency domain using a Fourier transform, performing point-wise multiplication of the two signals, and converting back to the time domain using an inverse Fourier transform. For some problem sizes, this can be faster than the naive implementation of discrete convolution.

When a d-dimensional kernel can be expressed as the outer product of d vectors, one vector per dimension, the kernel is called **separable**. When the kernel is separable, naive convolution is inefficient. It is equivalent to compose d one-dimensional convolutions with each of these vectors. The composed approach is significantly faster than performing one d-dimensional convolution with their outer product. The kernel also takes fewer parameters to represent as vectors. If the kernel is w elements wide in each dimension, then naive multidimensional convolution requires $O(w^d)$ runtime and parameter storage space, while separable convolution requires $O(w \times d)$ runtime and parameter storage space. Of course, not every convolution can be represented in this way.

Devising faster ways of performing convolution or approximate convolution without harming the accuracy of the model is an active area of research. Even techniques that improve the efficiency of only forward propagation are useful because in the commercial setting, it is typical to devote more resources to deployment of a network than to its training.