Hardware engineers have noted this fact for years, and there exist a variety of alternative hardware for working with deep networks. Such hardware can be broadly divided into *inference only* or *training and inference*. Inference-only hardware cannot be used to train new deep networks, but can be used to deploy trained models in production, allowing for potentially orders-of-magnitude increases in performance. Training and inference hardware allows for models to be trained natively. Currently, Nvidia's GPU hardware holds a dominant position in the training and inference market due to significant investment in software and outreach by Nvidia's teams, but a number of other competitors are snapping at the GPU's heels. In this section, we will briefly cover some of these newer hardware alternatives. With the exception of GPUs and CPUs, most of these alternative forms of hardware are not yet widely available, so much of this section is forward looking.

## CPU Training

Although CPU training is by no means state of the art for training deep networks, it often does quite well for smaller models (as you've seen firsthand in this book). For reinforcement learning problems, a multicore CPU machine can even outperform GPU training.

CPUs also see wide usage for inference-only applications of deep networks. Most companies have invested heavily in developing cloud servers built primarily on Intel server boxes. It's very likely that the first generation of deep networks deployed widely (outside tech companies) will be primarily deployed into production on such Intel servers. While such CPU-based deployment isn't sufficient for heavy-duty deployment of learning models, it is often plenty for first customer needs. Figure 9-1 illustrates a standard Intel CPU.

*Figure 9-1. A CPU from Intel. CPUs are still the dominant form of computer hardware and are present in all modern laptops, desktops, servers, and phones. Most software is written to execute on CPUs. Numerical computations (such as neural network training) can be executed on CPUs, but might be slower than on customized hardware optimized for numerical methods.*

## GPU Training

GPUs were first developed to perform computations needed by the graphics community. In a fortuitous coincidence, it turned out that the primitives used to define graphics shaders could be repurposed to perform deep learning. At their mathematical hearts, both graphics and machine learning rely critically on matrix multiplications. Empirically, GPU matrix multiplications offer speedups of an order of magnitude or two over CPU implementations. How do GPUs succeed at this feat? The trick is that GPUs make use of thousands of identical threads. Clever hackers have succeeded in decomposing matrix multiplications into massively parallel operations that can offer dramatic speedups. Figure 9-2 illustrates a GPU architecture.

Although there are a number of GPU vendors, Nvidia currently dominates the GPU market. Much of the power of Nvidia's GPUs stems from its custom library CUDA (compute unified device architecture), which offers primitives that make it easier to write GPU programs. Nvidia offers a CUDA extension, CUDNN, for speeding up deep networks (Figure 9-2). TensorFlow has built-in CUDNN support, so you can make use of CUDNN to speed up your networks as well through TensorFlow.