CHAPTER 1

Introduction to Deep Learning

Deep learning has revolutionized the technology industry. Modern machine translation, search engines, and computer assistants are all powered by deep learning. This trend will only continue as deep learning expands its reach into robotics, pharmaceuticals, energy, and all other fields of contemporary technology. It is rapidly becoming essential for the modern software professional to develop a working knowledge of the principles of deep learning.

In this chapter, we will introduce you to the history of deep learning, and to the broader impact deep learning has had on the research and commercial communities. We will next cover some of the most famous applications of deep learning. This will include both prominent machine learning architectures and fundamental deep learning primitives. We will end by giving a brief perspective of where deep learning is heading over the next few years before we dive into TensorFlow in the next few chapters.

Machine Learning Eats Computer Science

Until recently, software engineers went to school to learn a number of basic algorithms (graph search, sorting, database queries, and so on). After school, these engineers would go out into the real world to apply these algorithms to systems. Most of today's digital economy is built on intricate chains of basic algorithms laboriously glued together by generations of engineers. Most of these systems are not capable of adapting. All configurations and reconfigurations have to be performed by highly trained engineers, rendering systems brittle.

Machine learning promises to change the field of software development by enabling systems to adapt dynamically. Deployed machine learning systems are capable of learning desired behaviors from databases of examples. Furthermore, such systems

can be regularly retrained as new data comes in. Very sophisticated software systems, powered by machine learning, are capable of dramatically changing their behavior without major changes to their code (just to their training data). This trend is only likely to accelerate as machine learning tools and deployment become easier and easier.

As the behavior of software-engineered systems changes, the roles of software engineers will change as well. In some ways, this transformation will be analogous to the transformation following the development of programming languages. The first computers were painstakingly programmed. Networks of wires were connected and interconnected. Then punchcards were set up to enable the creation of new programs without hardware changes to computers. Following the punchcard era, the first assembly languages were created. Then higher-level languages like Fortran or Lisp. Succeeding layers of development have created very high-level languages like Python, with intricate ecosystems of precoded algorithms. Much modern computer science even relies on autogenerated code. Modern app developers use tools like Android Studio to autogenerate much of the code they'd like to make. Each successive wave of simplification has broadened the scope of computer science by lowering barriers to entry.

Machine learning promises to lower barriers even further; programmers will soon be able to change the behavior of systems by altering training data, possibly without writing a single line of code. On the user side, systems built on spoken language and natural language understanding such as Alexa and Siri will allow nonprogrammers to perform complex computations. Furthermore, ML powered systems are likely to become more *robust* against errors. The capacity to retrain models will mean that codebases can shrink and that maintainability will increase. In short, machine learning is likely to completely upend the role of software engineers. Today's programmers will need to understand how machine learning systems learn, and will need to understand the classes of errors that arise in common machine learning systems. Furthermore, they will need to understand the design patterns that underlie machine learning systems (very different in style and form from classical software design patterns). And, they will need to know enough tensor calculus to understand why a sophisticated deep architecture may be misbehaving during learning. It's not an understatement to say that understanding machine learning (theory and practice) will become a fundamental skill that every computer scientist and software engineer will need to understand for the coming decade.

In the remainder of this chapter, we will provide a whirlwind tour of the basics of modern deep learning. The remainder of this book will go into much greater depth on all the topics we touch on here.

Deep Learning Primitives

Most deep architectures are built by combining and recombining a limited set of architectural primitives. Such primitives, typically called neural network layers, are the foundational building blocks of deep networks. In the rest of this book, we will provide in-depth introductions to such layers. However, in this section, we will provide a brief overview of the common modules that are found in many deep networks. This section is not meant to provide a thorough introduction to these modules. Rather, we aim to provide a rapid overview of the building blocks of sophisticated deep architectures to whet your appetite. The art of deep learning consists of combining and recombining such modules and we want to show you the alphabet of the language to start you on the path to deep learning expertise.

Fully Connected Layer

A fully connected network transforms a list of inputs into a list of outputs. The transformation is called fully connected since any input value can affect any output value. These layers will have many learnable parameters, even for relatively small inputs, but they have the large advantage of assuming no structure in the inputs. This concept is illustrated in Figure 1-1.

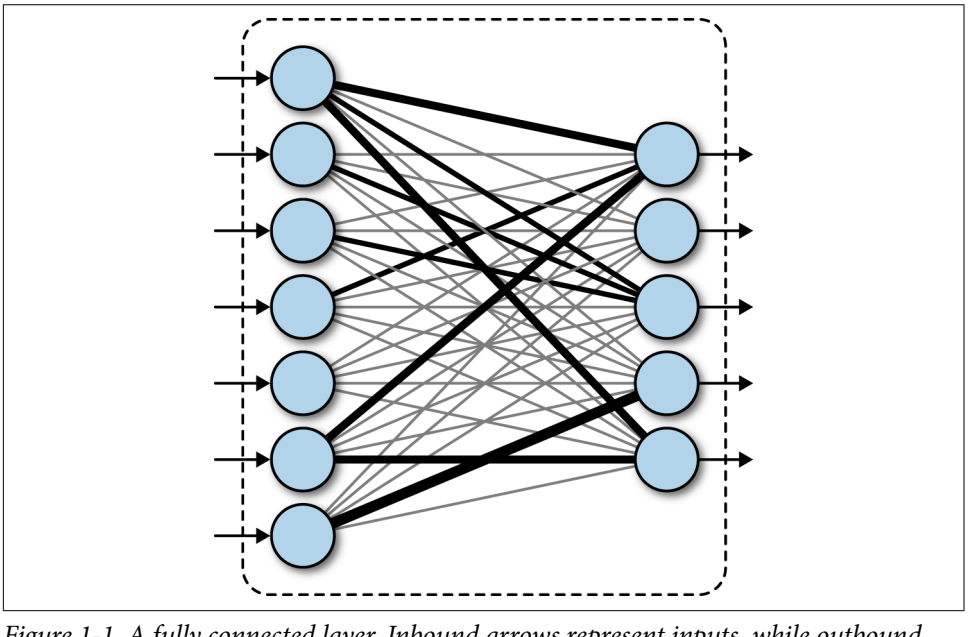

Figure 1-1. A fully connected layer. Inbound arrows represent inputs, while outbound arrows represent outputs. The thickness of interconnecting lines represents the magnitude of learned weights. The fully connected layer transforms inputs into outputs via the learned rule.