Why you should learn deep learning

It's a powerful tool for the incremental automation of intelligence.

From the beginning of time, humans have been building better and better tools to understand and control the environment around us. Deep learning is today's chapter in this story of innovation.

Perhaps what makes this chapter so compelling is that this field is more of a *mental* innovation than a *mechanical one*. Much like its sister fields in machine learning, deep learning seeks to *automate intelligence* bit by bit. In the past few years, it has achieved enormous success and progress in this endeavor, exceeding previous records in computer vision, speech recognition, machine translation, and many other tasks.

This is particularly extraordinary given that deep learning seems to use *largely the same brain-inspired algorithm* (neural networks) for achieving these accomplishments across a vast number of fields. Even though deep learning is still an actively developing field with many challenges, recent developments have lead to tremendous excitement: perhaps we've discovered not just a great tool, but a window into our own minds.

Deep learning has the potential for significant automation of skilled labor.

There's a substantial amount of hype around the potential impacts of deep learning if the current trend of progress is extrapolated at varying speeds. Although many of these predictions are overzealous, I believe one merits your consideration: job displacement. I think this claim stands out from the rest because even if deep learning's innovations stopped *today*, there would already be an incredible impact on skilled labor around the globe. Call-center operators, taxi drivers, and low-level business analysts are compelling examples where deep learning can provide a low-cost alternative.

Fortunately, the economy doesn't turn on a dime; but in many ways we're already past the point of concern, given the current power of the technology. It's my hope that you (and people you know) will be enabled by this book to transition from perhaps one of the industries facing disruption into an industry ripe with growth and prosperity: deep learning.

It's fun and creative. You'll discover much about what it is to be human by trying to simulate intelligence and creativity.

Personally, I got into deep learning because it's fascinating. It's an amazing intersection between human and machine. Unpacking exactly what it means to think, to reason, and to create is enlightening, engaging, and, for me, inspiring. Consider having a dataset filled with every painting ever painted, and then using that to teach a machine how to paint like Monet. Insanely, it's possible, and it's mind-bogglingly cool to see how it works.

Will this be difficult to learn?

How hard will you have to work before there's a "fun" payoff?

This is my favorite question. My definition of a "fun" payoff is the experience of witnessing something that I built *learning*. There's something amazing about seeing a creation of your hands do something like that. If you also feel this way, then the answer is simple. A few pages into chapter 3, you'll create your first neural network. The only work involved until then is reading the pages between here and there.

After chapter 3, you may be interested to know that the *next* fun payoff occurs after you've memorized a small snippet of code and proceeded to read to the midway of chapter 4. Each chapter will work this way: memorize a small code segment from the previous chapter, read the next chapter, and then experience the payoff of a new learning neural network.

Why you should read this book

It has a uniquely low barrier to entry.

The reason you should read this book is the same reason I'm writing it. I don't know of another resource (book, course, large blog series) that teaches deep learning *without assuming advanced knowledge of mathematics* (a college degree in a mathy field).

Don't get me wrong: there are really good reasons for teaching it using math. Math is, after all, a language. It's certainly more *efficient* to teach deep learning using this language, but I don't think it's absolutely necessary to assume advanced knowledge of math in order to become a skilled, knowledgeable practitioner who has a firm understanding of the "how" behind deep learning.

So, why should you learn deep learning using this book? Because I'm going to assume you have a high school–level background in math (and that it's rusty) and *explain everything else you need to know as we go along*. Remember multiplication? Remember x-y graphs (the squares with lines on them)? Awesome! You'll be fine.

It will help you understand what's *inside* a framework (Torch, TensorFlow, and so on).

There are two major groups of deep learning educational material (such as books and courses). One group is focused around how to use popular frameworks and code libraries like Torch, TensorFlow, Keras, and others. The other group is focused around teaching deep learning itself, otherwise known as the *science under the hood* of these major frameworks.

Ultimately, learning about *both* is important. It's like if you want to be a NASCAR driver: you need to learn both about the particular model of car you're driving (the framework) and about driving (the science/skill). But just learning about a framework is like learning about the pros and cons of a Generation 6 Chevrolet SS before you know what a stick shift is. This book is about teaching you what deep learning is so you can then be prepared to learn a framework.

All math-related material will be backed by intuitive analogies.

Whenever I encounter a math formula in the wild, I take a two-step approach. The first is to translate its methods into an intuitive *analogy* to the real world. I almost never take a formula at face value: I break it into *parts*, each with a story of its own. That will be the approach of this book, as well. Anytime we encounter a math concept, I'll offer an alternative analogy for what the formula is actually doing.

> "Everything should be made as simple as possible, but not simpler."
> —Attributed to Albert Einstein

Everything after the introduction chapters is "project" based.

If there's one thing I hate when learning something new, it's having to question whether what I'm learning is useful or relevant. If someone is teaching me everything there is to know about a hammer without actually taking my hand and helping me drive in a nail, then they're not really teaching me how to use a hammer. I know there will be dots that aren't connected, and if I'm thrown out into the real world with a hammer, a box of nails, and a bunch of two-by-fours, I'll have to do some guesswork.

This book is about giving you the wood, nails, and hammer *before* telling you what they do. Each lesson is about picking up the tools and building stuff with them, explaining how things work as we go. This way, you won't leave with a list of facts about the various deep learning tools you'll work with; you'll have the ability to use them to solve problems. Furthermore, you'll understand the most important part: when and why each tool is appropriate for each problem you want to solve. It is with this knowledge that you'll be empowered to pursue a career in research and/or industry.

What you need to get started

Install Jupyter Notebook and the NumPy Python library.

My absolute favorite place to work is in Jupyter Notebook. One of the most important parts of learning deep learning (for me) is the ability to stop a network while it's training and tear apart absolutely every piece to see what it looks like. This is something Jupyter Notebook is incredibly useful for.

As for NumPy, perhaps the most compelling case for why this book leaves nothing out is that we'll be using only a single matrix library. In this way, you'll understand *how* everything works, not just how to call a framework. This book teaches deep learning from absolute scratch, soup to nuts.

Installation instructions for these two tools can be found at http://jupyter.org for Jupyter and http://numpy.org for NumPy. I'll build the examples in Python 2.7, but I've tested them for Python 3 as well. For easy installation, I also recommend the Anaconda framework: https://docs.continuum.io/anaconda/install.

Pass high school mathematics.

Some mathematical assumptions are out of depth for this book, but my goal is to teach deep learning assuming that you understand only basic algebra.

Find a personal problem you're interested in.

This might seem like an optional "need" to get started. I guess it could be, but seriously, I highly, highly recommend finding one. Everyone I know who has become successful at this stuff had some sort of problem they were trying to solve. Learning deep learning was just a "dependency" to solving some other interesting task.

For me, it was using Twitter to predict the stock market. It's just something I thought was really fascinating. It's what drove me to sit down and read the next chapter and build the next prototype.

And as it turns out, this field is *so new*, and is changing *so fast*, that if you spend the next couple of years chasing one project with these tools, you'll find yourself becoming one of the leading experts in that *particular problem* faster than you might think. For me, chasing this idea took me from barely knowing anything about programming to a research grant at a hedge fund applying what I learned, in around 18 months! For deep learning, having a problem you're fascinated with that involves using one dataset to predict another is the key catalyst! Go find one!

You'll probably need some Python knowledge

Python is my teaching library of choice, but I'll provide a few others online.

Python is an amazingly intuitive language. I think it just might be the most widely adopted and intuitively readable language yet constructed. Furthermore, the Python community has a passion for simplicity that can't be beat. For these reasons, I want to stick with Python for all the examples (Python 2.7 is what I'm working in). In the book's downloadable source code, available at www.manning.com/books/grokking-deep-learning and also at https://github.com/iamtrask/Grokking-Deep-Learning, I provide all the examples in a variety of other languages online.

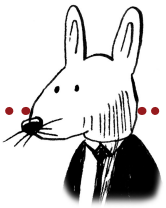
How much coding experience should you have?

Scan through the Python Codecademy course (www.codecademy.com/learn/python). If you can read the table of contents and feel comfortable with the terms mentioned, you're all set! If not, then take the course and come back when you're done. It's designed to be a beginner course, and it's very well crafted.

Summary

If you've got a Jupyter notebook in hand and feel comfortable with the basics of Python, you're ready for the next chapter! As a heads-up, chapter 2 is the last chapter that will be mostly dialogue based (without building something). It's designed to give you an awareness of the high-level vocabulary, concepts, and fields in artificial intelligence, machine learning, and, most important, deep learning.

fundamental concepts: how do machines learn? 2



In this chapter

- What are deep learning, machine learning, and artificial intelligence?
- What are parametric models and nonparametric models?
- What are supervised learning and unsupervised learning?
- How can machines learn?

> "Machine learning will cause every successful IPO win in five years."
>
> —Eric Schmidt, Google executive chairman, keynote speech, Cloud Computing Platform conference, 2016