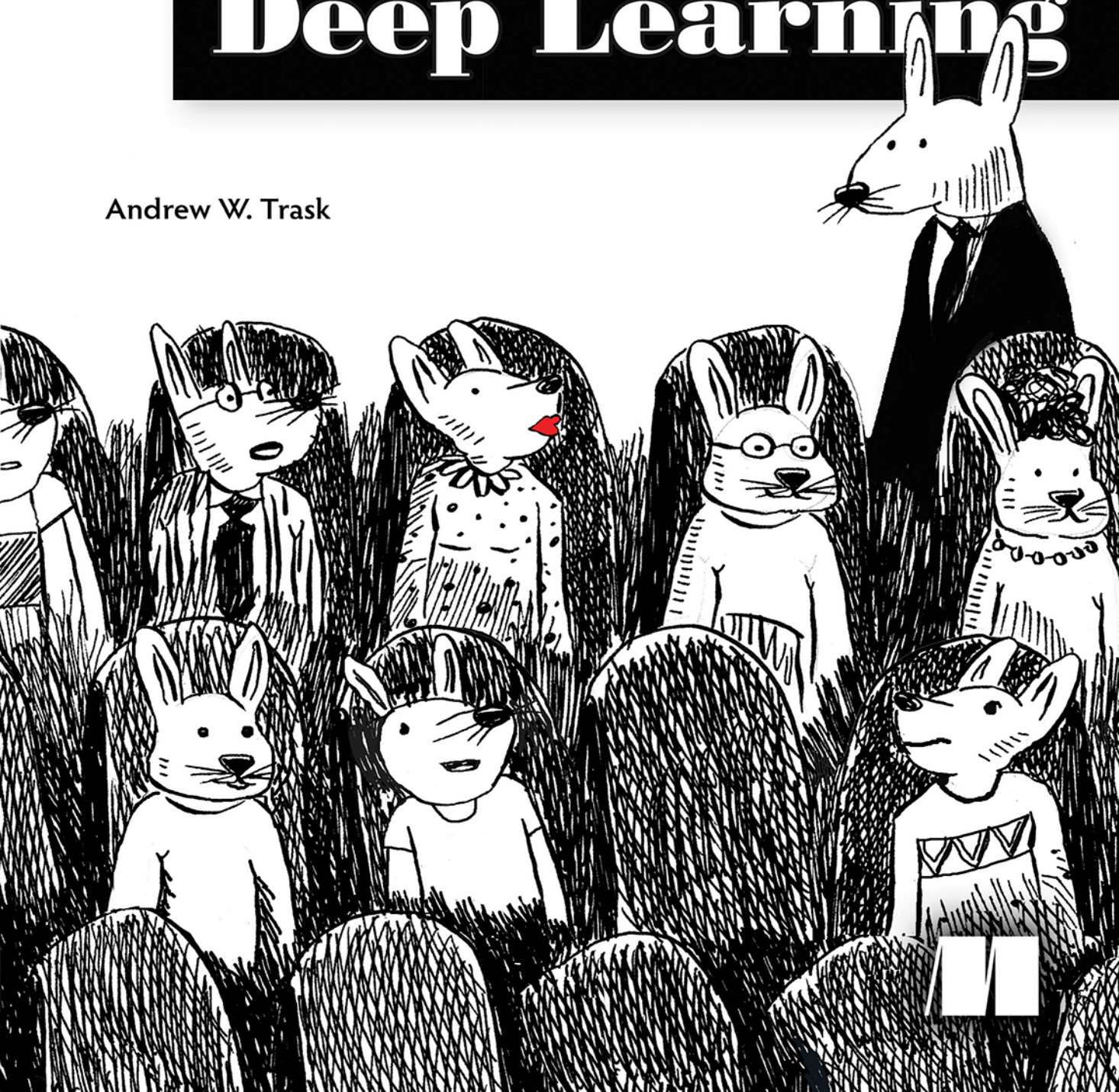


Sample Chapter

grokking

Deep Learning

Andrew W. Trask



grokking
Deep Learning

Andrew W. Trask



MANNING
SHELTER ISLAND

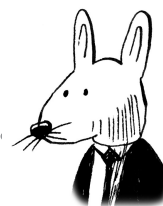


grokking Deep Learning

by Andrew W. Trask

Chapter 1

Copyright 2019 Manning Publications



contents

preface	xv
acknowledgments	xvi
about this book	xvii
about the author	xx
1 Introducing deep learning: why you should learn it	3
Welcome to <i>Grokking Deep Learning</i>	3
Why you should learn deep learning	4
Will this be difficult to learn?	5
Why you should read this book	5
What you need to get started	7
You'll probably need some Python knowledge	8
Summary	8
2 Fundamental concepts: how do machines learn?	9
What is deep learning?	10
What is machine learning?	11
Supervised machine learning	12
Unsupervised machine learning	13
Parametric vs. nonparametric learning	14
Supervised parametric learning	15
Unsupervised parametric learning	17
Nonparametric learning	18
Summary	19

3 Introduction to neural prediction: forward propagation	21
Step 1: Predict	22
A simple neural network making a prediction	24
What is a neural network?	25
What does this neural network do?	26
Making a prediction with multiple inputs	28
Multiple inputs: What does this neural network do?	30
Multiple inputs: Complete runnable code	35
Making a prediction with multiple outputs	36
Predicting with multiple inputs and outputs	38
Multiple inputs and outputs: How does it work?	40
Predicting on predictions	42
A quick primer on NumPy	44
Summary	46
4 Introduction to neural learning: gradient descent	47
Predict, compare, and learn	48
Compare	48
Learn	49
Compare: Does your network make good predictions?	50
Why measure error?	51
What's the simplest form of neural learning?	52
Hot and cold learning	54
Characteristics of hot and cold learning	55
Calculating both direction and amount from error	56
One iteration of gradient descent	58
Learning is just reducing error	60
Let's watch several steps of learning	62
Why does this work? What is weight_delta, really?	64
Tunnel vision on one concept	66
A box with rods poking out of it	67
Derivatives: Take two	68
What you really need to know	69
What you don't really need to know	69
How to use a derivative to learn	70
Look familiar?	71

Breaking gradient descent	72
Visualizing the overcorrections	73
Divergence	74
Introducing alpha	75
Alpha in code	76
Memorizing	77
5 Learning multiple weights at a time: generalizing gradient descent	79
Gradient descent learning with multiple inputs	80
Gradient descent with multiple inputs explained	82
Let's watch several steps of learning	86
Freezing one weight: What does it do?	88
Gradient descent learning with multiple outputs	90
Gradient descent with multiple inputs and outputs	92
What do these weights learn?	94
Visualizing weight values	96
Visualizing dot products (weighted sums)	97
Summary	98
6 Building your first deep neural network: introduction to backpropagation	99
The streetlight problem	100
Preparing the data	102
Matrices and the matrix relationship	103
Creating a matrix or two in Python	106
Building a neural network	107
Learning the whole dataset	108
Full, batch, and stochastic gradient descent	109
Neural networks learn correlation	110
Up and down pressure	111
Edge case: Overfitting	113
Edge case: Conflicting pressure	114
Learning indirect correlation	116
Creating correlation	117
Stacking neural networks: A review	118
Backpropagation: Long-distance error attribution	119

Backpropagation: Why does this work?	120
Linear vs. nonlinear	121
Why the neural network still doesn't work	122
The secret to sometimes correlation	123
A quick break	124
Your first deep neural network	125
Backpropagation in code	126
One iteration of backpropagation	128
Putting it all together	130
Why do deep networks matter?	131
7 How to picture neural networks: in your head and on paper	133
It's time to simplify	134
Correlation summarization	135
The previously overcomplicated visualization	136
The simplified visualization	137
Simplifying even further	138
Let's see this network predict	139
Visualizing using letters instead of pictures	140
Linking the variables	141
Everything side by side	142
The importance of visualization tools	143
8 Learning signal and ignoring noise: introduction to regularization and batching	145
Three-layer network on MNIST	146
Well, that was easy	148
Memorization vs. generalization	149
Overfitting in neural networks	150
Where overfitting comes from	151
The simplest regularization: Early stopping	152
Industry standard regularization: Dropout	153
Why dropout works: Ensembling works	154
Dropout in code	155
Dropout evaluated on MNIST	157
Batch gradient descent	158
Summary	160

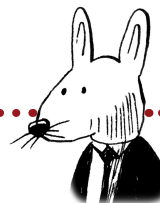
9 Modeling probabilities and nonlinearities: activation functions	161
What is an activation function?	162
Standard hidden-layer activation functions	165
Standard output layer activation functions	166
The core issue: Inputs have similarity	168
softmax computation	169
Activation installation instructions	170
Multiplying delta by the slope	172
Converting output to slope (derivative)	173
Upgrading the MNIST network	174
10 Neural learning about edges and corners: intro to convolutional neural networks	177
Reusing weights in multiple places	178
The convolutional layer	179
A simple implementation in NumPy	181
Summary	185
11 Neural networks that understand language: king – man + woman == ?	187
What does it mean to understand language?	188
Natural language processing (NLP)	189
Supervised NLP	190
IMDB movie reviews dataset	191
Capturing word correlation in input data	192
Predicting movie reviews	193
Intro to an embedding layer	194
Interpreting the output	196
Neural architecture	197
Comparing word embeddings	199
What is the meaning of a neuron?	200
Filling in the blank	201
Meaning is derived from loss	203
King – Man + Woman $\approx$ Queen	206
Word analogies	207
Summary	208

12 Neural networks that write like Shakespeare: recurrent layers for variable-length data	209
The challenge of arbitrary length	210
Do comparisons really matter?	211
The surprising power of averaged word vectors	212
How is information stored in these embeddings?	213
How does a neural network use embeddings?	214
The limitations of bag-of-words vectors	215
Using identity vectors to sum word embeddings	216
Matrices that change absolutely nothing	217
Learning the transition matrices	218
Learning to create useful sentence vectors	219
Forward propagation in Python	220
How do you backpropagate into this?	221
Let's train it!	222
Setting things up	223
Forward propagation with arbitrary length	224
Backpropagation with arbitrary length	225
Weight update with arbitrary length	226
Execution and output analysis	227
Summary	229
13 Introducing automatic optimization: let's build a deep learning framework	231
What is a deep learning framework?	232
Introduction to tensors	233
Introduction to automatic gradient computation (autograd)	234
A quick checkpoint	236
Tensors that are used multiple times	237
Upgrading autograd to support multiuse tensors	238
How does addition backpropagation work?	240
Adding support for negation	241
Adding support for additional functions	242
Using autograd to train a neural network	246
Adding automatic optimization	248
Adding support for layer types	249

Layers that contain layers	250
Loss-function layers	251
How to learn a framework	252
Nonlinearity layers	253
The embedding layer	255
Adding indexing to autograd	256
The embedding layer (revisited)	257
The cross-entropy layer	258
The recurrent neural network layer	260
Summary	263
14 Learning to write like Shakespeare: long short-term memory	265
Character language modeling	266
The need for truncated backpropagation	267
Truncated backpropagation	268
A sample of the output	271
Vanishing and exploding gradients	272
A toy example of RNN backpropagation	273
Long short-term memory (LSTM) cells	274
Some intuition about LSTM gates	275
The long short-term memory layer	276
Upgrading the character language model	277
Training the LSTM character language model	278
Tuning the LSTM character language model	279
Summary	280
15 Deep learning on unseen data: introducing federated learning	281
The problem of privacy in deep learning	282
Federated learning	283
Learning to detect spam	284
Let's make it federated	286
Hacking into federated learning	287
Secure aggregation	288
Homomorphic encryption	289

Homomorphically encrypted federated learning	290
Summary	291
16 Where to go from here: a brief guide	293
Congratulations!	294
Step 1: Start learning PyTorch	294
Step 2: Start another deep learning course	295
Step 3: Grab a mathy deep learning textbook	295
Step 4: Start a blog, and teach deep learning	296
Step 5: Twitter	297
Step 6: Implement academic papers	297
Step 7: Acquire access to a GPU (or many)	297
Step 8: Get paid to practice	298
Step 9: Join an open source project	298
Step 10: Develop your local community	299
 index	 301

introducing deep learning: why you should learn it | 1



In this chapter

- Why you should learn deep learning
- Why you should read this book
- What you need to get started

“ Do not worry about your difficulties in Mathematics. ”
I can assure you mine are still greater.

—Albert Einstein

Welcome to *Grokking Deep Learning*

You're about to learn some of the most valuable skills of the century!

I'm very excited that you're here! You should be, too! Deep learning represents an exciting intersection of machine learning and artificial intelligence, and a very significant disruption to society and industry. The methods discussed in this book are changing the world all around you. From optimizing the engine of your car to deciding which content you view on social media, it's everywhere, it's powerful, and, fortunately, it's fun!

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.

grokking Deep Learning

Andrew W. Trask

Deep learning, a branch of artificial intelligence, teaches computers to learn by using neural networks, technology inspired by the human brain. Online text translation, self-driving cars, personalized product recommendations, and virtual voice assistants are just a few of the exciting modern advancements possible thanks to deep learning.

Grokking Deep Learning teaches you to build deep learning neural networks from scratch! In his engaging style, seasoned deep learning expert Andrew Trask shows you the science under the hood, so you grok for yourself every detail of training neural networks. Using only Python and its math-supporting library, NumPy, you'll train your own neural networks to see and understand images, translate text into different languages, and even write like Shakespeare! When you're done, you'll be fully prepared to move on to mastering deep learning frameworks.

What's Inside

- The science behind deep learning
- Building and training your own neural networks
- Privacy concepts, including federated learning
- Tips for continuing your pursuit of deep learning

For readers with high school-level math and intermediate programming skills.

Andrew Trask is a PhD student at Oxford University and a research scientist at DeepMind. Previously, Andrew was a researcher and analytics product manager at Digital Reasoning, where he trained the world's largest artificial neural network and helped guide the analytics roadmap for the Synthesys cognitive computing platform.



“An excellent introduction and overview of deep learning by a masterful teacher who guides, illuminates, and encourages you along the way.”

—Kelvin D. Meeks
International Technology
Ventures

“All concepts are clearly explained with excellent visualizations and examples.”

—Kalyan Reddy, ArisGlobal

“Excels at navigating the reader through the introductory details of deep learning in a simple and intuitive way.”

—Eremey Valetov
Lancaster University/Fermilab

“Your step-by-step guide to learning AI.”

—Ian Stirk, I and K Consulting

“A complex topic simplified!”

—Vipul Gupta, Microsoft

To download their free eBook in PDF, ePub, and Kindle formats, owners of this book should visit manning.com/books/grokking-deep-learning

ISBN-13: 978-1-61729-370-2
ISBN-10: 1-61729-370-9



MANNING US \$49.99 | Can \$65.99 (including eBook)