Preface

In 2006, Geoffrey Hinton et al. published a paper[1] showing how to train a deep neural network capable of recognizing handwritten digits with state-of-the-art precision (>98%). They branded this technique "deep learning". A deep neural network is a (very) simplified model of our cerebral cortex, composed of a stack of layers of artificial neurons. Training a deep neural net was widely considered impossible at the time,[2] and most researchers had abandoned the idea in the late 1990s. This paper revived the interest of the scientific community, and before long many new papers demonstrated that deep learning was not only possible, but capable of mind-blowing achievements that no other machine learning (ML) technique could hope to match (with the help of tremendous computing power and great amounts of data). This enthusiasm soon extended to many other areas of machine learning.

A decade later, machine learning had already conquered many industries, ranking web results, recommending videos to watch and products to buy, sorting items on production lines, sometimes even driving cars. Machine learning often made the headlines, for example when DeepMind's AlphaFold machine learning system solved a long-standing protein-folding problem that had stomped researchers for decades. But most of the time, machine learning was just working discretely in the background. However, another decade later came the rise of AI assistants: from ChatGPT in 2022, Gemini, Claude, and Grok in 2023, and many others since then. AI has now truly taken off and it is rapidly transforming every single industry: what used to be sci-fi is now very real.[3]

Machine Learning in Your Projects

So, naturally you are excited about machine learning and would love to join the party! Perhaps you would like to give your homemade robot a brain of its own? Make it recognize faces? Or learn to walk around?

Or maybe your company has tons of data (user logs, financial data, production data, machine sensor data, hotline stats, HR reports, etc.), and more than likely you could unearth some hidden gems if you just knew where to look. With machine learning, you could accomplish the following and much more:

- Segment customers and find the best marketing strategy for each group.
- Recommend products for each client based on what similar clients bought.
- Detect which transactions are likely to be fraudulent.
- Forecast next year's revenue.
- Predict peak workloads and suggest optimal staffing levels.
- Build a chatbot to assist your customers.

Whatever the reason, you have decided to learn machine learning and implement it in your projects. Great idea!

Objective and Approach

This book assumes that you know close to nothing about machine learning. Its goal is to give you the concepts, tools, and intuition you need to implement programs capable of *learning from data*.

We will cover a large number of techniques, from the simplest and most commonly used (such as linear regression) to some of the deep learning techniques that regularly win competitions. For this, we will be using Python—the leading language for data science and machine learning—as well as open source and production-ready Python frameworks:

- Scikit-Learn is very easy to use, yet it implements many machine learning algorithms efficiently, so it makes for a great entry point to learning machine learning. It was created by David Cournapeau in 2007, then led by a team of researchers at the French Institute for Research in Computer Science and Automation (Inria), and recently Probabl.ai.
- PyTorch is a powerful and flexible library for deep learning. It makes it possible to train and run all sorts of neural networks efficiently, and it can distribute the computations across multiple GPUs (graphics processing units). PyTorch (PT) was developed by Facebook's AI Research lab (FAIR) and first released in 2016. It evolved from Torch, an older framework coded in Lua. In 2022, PyTorch was transitioned to the PyTorch Foundation, under the Linux Foundation, to promote community-driven development.

We will also use these open source machine learning libraries along the way:

- XGBoost in Chapter 6 to implement a powerful technique called *gradient boosting*.
- Hugging Face libraries in Chapters 13 and 15 to download datasets and pretrained models, including transformer models. Transformers are incredibly powerful and versatile, and they are the main building block of virtually all AI assistants today.
- Gymnasium in Chapter 19 for reinforcement learning (i.e., training autonomous agents).

The book favors a hands-on approach, growing an intuitive understanding of machine learning through concrete working examples and just a little bit of theory.

Tip

While you can read this book without picking up your laptop, I highly recommend you experiment with the code examples.

Code Examples

All the code examples in this book are open source and available online at *https://github.com/ageron/handson-mlp*, as Jupyter notebooks. These are interactive documents containing text, images, and executable code snippets (Python in our case). The easiest and quickest way to get started is to run these notebooks using Google Colab: this is a free service that allows you to run any Jupyter notebook directly online without having to install anything on your machine. All you need is a web browser and a Google account.

Note

In this book, I will assume that you are using Google Colab, but I have also tested the notebooks on other online platforms such as Kaggle and Binder, so you can use those if you prefer. Alternatively, you can install the required libraries and tools (or the Docker image for this book) and run the notebooks directly on your own machine. See the instructions at *https://homl.info/install-p*.

Prerequisites

This book assumes that you have some Python programming experience. If you don't know Python yet, *https://learnpython.org* is a great place to start. The official tutorial on Python.org is also quite good.

This book also assumes that you are familiar with Python's main scientific libraries—in particular, NumPy, pandas, and Matplotlib. If you have never used these libraries, don't worry; they're easy to learn, and I've created a tutorial for each of them. You can access them online at *https://homl.info/tutorials-p*.

Moreover, if you want to fully understand how the machine learning algorithms work (not just how to use them), then you should have at least a basic understanding of a few math concepts, especially linear algebra. Specifically, you should know what vectors and matrices are, and how to perform some simple operations like adding vectors, or transposing and multiplying matrices. If you need a quick introduction to linear algebra (it's really not rocket science!), I provide a tutorial at *https://homl.info/tutorials-p*. You will also find a tutorial on differential calculus, which may be helpful to understand how neural networks are trained, but it's not entirely essential to grasp the important concepts. This book also uses other mathematical concepts occasionally, such as exponentials and logarithms, a bit of probability theory, and some basic concepts from statistics, but nothing too advanced. If you need help on any of these, please check out *https://khanacademy.org*, which offers many excellent and free math courses online.

Roadmap

This book is organized in two parts. Part I, "The Fundamentals of Machine Learning", covers the following topics:

- What machine learning is, what problems it tries to solve, and the main categories and fundamental concepts of its systems
- The steps in a typical machine learning project
- Learning by fitting a model to data
- Minimizing a cost function (i.e., a measure of prediction errors)
- Handling, cleaning, and preparing data
- Selecting and engineering features (i.e., data fields)
- Selecting a model and tuning hyperparameters using cross-validation (e.g., training many model variants and choosing the one that performs best on data it didn't see during training)
- The challenges of machine learning, in particular underfitting and overfitting (the bias/variance trade-off)
- The most common learning algorithms: linear and polynomial regression, logistic regression, *k*-nearest neighbors, decision trees, random forests, and ensemble methods

- Reducing the dimensionality of the training data to fight the “curse of dimensionality”
- Other unsupervised learning techniques, including clustering, density estimation, and anomaly detection

Part II, “Neural Networks and Deep Learning”, covers the following topics:

- What neural nets are and what they’re good for
- Building and training deep neural nets using PyTorch
- The most important neural net architectures: feedforward neural nets for tabular data; convolutional nets for computer vision; recurrent nets and long short-term memory (LSTM) nets for sequence processing; encoder-decoders, transformers, state space models (SSMs), and hybrid architectures for natural language processing, vision, and more; autoencoders, generative adversarial networks (GANs), and diffusion models for generative learning
- How to build an agent (e.g., a bot in a game) that can learn good strategies through trial and error, using reinforcement learning
- Loading and preprocessing large amounts of data efficiently

The first part is based mostly on Scikit-Learn; the second part uses mostly PyTorch.

Caution

Don’t jump into deep waters too hastily: deep learning is no doubt one of the most exciting areas in machine learning, but you should master the fundamentals first. Moreover, most problems can be solved quite well using simpler techniques such as random forests and ensemble methods (discussed in Part I). Deep learning is best suited for complex problems such as image recognition, speech recognition, or natural language processing, and it often requires a lot of data, computing power, and patience (unless you can leverage a pretrained neural network, as you will see).

If you are particularly interested in one topic and want to reach it as quickly as possible, Figure P-1 will show you which chapters you must read first, and which ones you can safely skip.

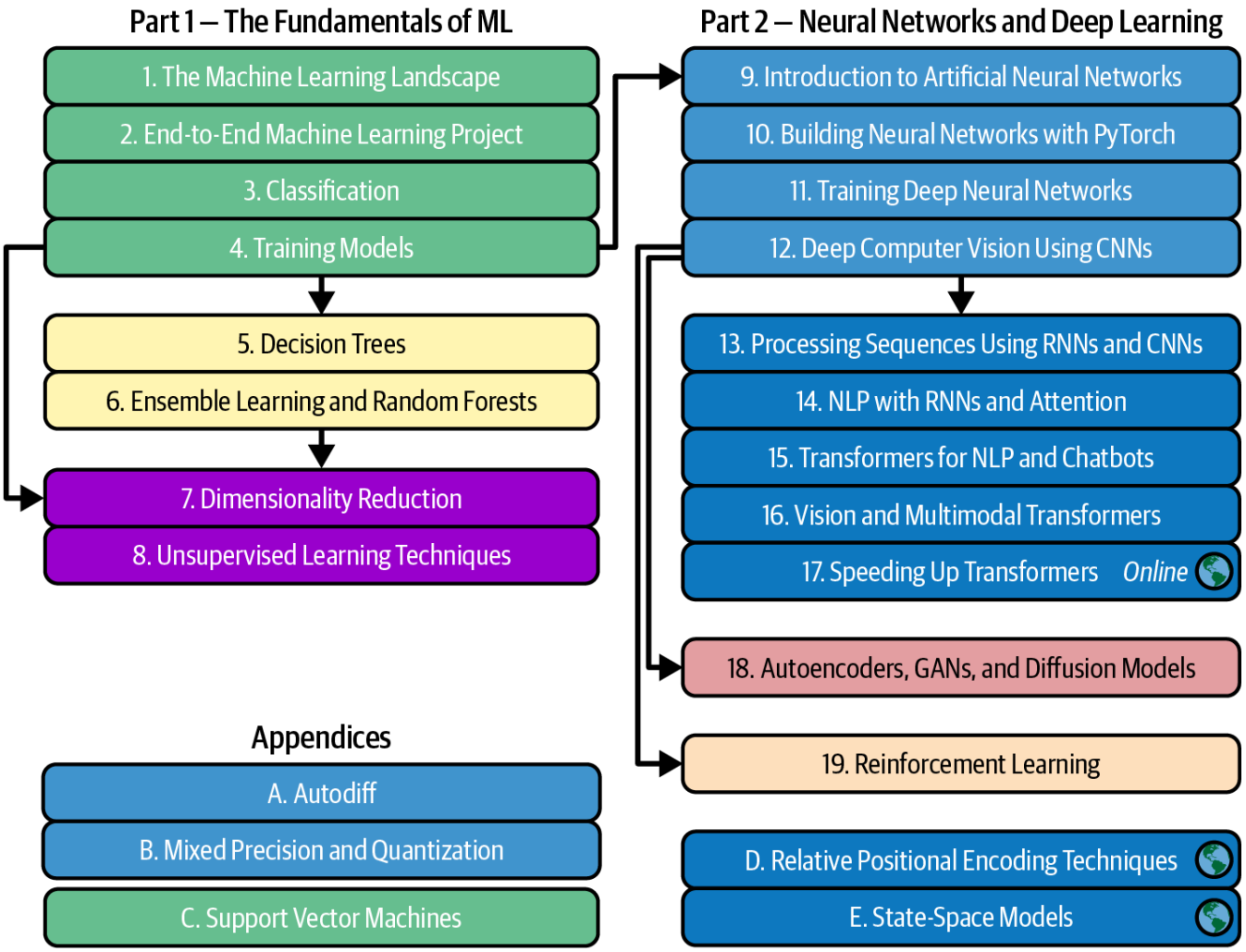

Figure P-1. Chapter dependencies

Changes Between the TensorFlow and PyTorch Versions

I wrote three TensorFlow (TF) editions of this book, published by O’Reilly in 2017, 2019, and 2022. TF was the leading deep learning library for many years, used internally by Google and therefore optimized for production at scale. But PyTorch has gradually taken the lead, owing to its simplicity, flexibility and openness: it now dominates research papers and open source projects, which means that most new models are available in PyTorch first. As a result, the industry has also gradually shifted toward PyTorch.

In recent years, Google has reduced its investments in TensorFlow, and focused more on JAX, another excellent deep learning library with a great mix of qualities for both research and production. However, its adoption is still low compared to PyTorch.

This is why I chose to use PyTorch this time around! O’Reilly and I decided to make it the first edition of a new PyTorch series rather than the fourth edition of the original series. This leaves the door open for a JAX series or perhaps a new edition for the TF series (time will tell if either are needed).

If you have already read the latest TensorFlow version of this book, here are the main changes you will find in this book (see *https://homl.info/changes-p* for more details):

- All the code in the book was updated to recent library versions.
- All the code in Part II was migrated from TensorFlow and Keras to PyTorch. There were significant changes in all of these chapters.
- TensorFlow-specific content was removed, including former Chapters 12 and 13, and former Appendices C and D.
- Chapter 10 now introduces PyTorch.
- I also added three new chapters on transformers:
  - Chapter 15 covers transformers for natural language processing, including how to build a chatbot.
  - Chapter 16 presents vision transformers and multimodal transformers.
  - Chapter 17, available online at *https://homl.info/*, discusses several advanced techniques to speed up and scale up transformers. This includes FlashAttention, mixture of experts (MoE), low-rank adaptation (LoRA),

and many more.

- There are also three new appendices: Appendix B explains how to shrink models so they can run faster and fit on smaller devices, “Relative Positional Encoding” discusses advanced positional encoding techniques for transformers, and “State-Space Models (SSMs)” presents state-space models.
- To make room for the newer content, the chapter on support vector machines (SVMs) was moved online and renamed “Support Vector Machines”; the last two appendices are also online at the same URL, and the deployment chapter was partially merged into Chapter 10.

The three editions of the TensorFlow/Keras version of this book are nicknamed homl1, homl2, and homl3. This book, which is the first edition of the PyTorch version, is nicknamed homlp. Try saying that three times in a row.

Note

Most of the changes compared to the latest TensorFlow edition are in the second part of the book. If you have read homl3, then don’t expect big changes in the first part of the book: the fundamental concepts of machine learning haven’t changed since 2022.

Other Resources

Many excellent resources are available to learn about machine learning. For example, Andrew Ng’s ML course on Coursera is amazing, although it requires a significant time investment.

There are also many interesting websites about machine learning, including Scikit-Learn’s exceptional User Guide. You may also enjoy Dataquest, which provides very nice interactive tutorials, and countless ML blogs and YouTube channels.

There are many other introductory books about machine learning. In particular:

- Joel Grus’s *Data Science from Scratch*, 2nd edition (O’Reilly), presents the fundamentals of machine learning and implements some of the main algorithms in pure Python (from scratch, as the name suggests).
- Stephen Marsland’s *Machine Learning: An Algorithmic Perspective*, 2nd edition (Chapman & Hall), is a great introduction to machine learning, covering a wide range of topics in depth with code examples in Python (also from scratch, but using NumPy).
- Sebastian Raschka’s *Machine Learning with PyTorch and Scikit-Learn*, 1st edition (Packt Publishing), is also a great introduction to machine learning using Scikit-Learn and PyTorch.
- François Chollet’s *Deep Learning with Python*, 3rd edition (Manning), is a very practical book that covers a large range of topics in a clear and concise way, as you might expect from the author of the excellent Keras library. It favors code examples over mathematical theory.
- Andriy Burkov’s *The Hundred-Page Machine Learning Book* (self-published) is very short but covers an impressive range of topics, introducing them in approachable terms without shying away from the math equations.
- Yaser S. Abu-Mostafa, Malik Magdon-Ismail, and Hsuan-Tien Lin’s *Learning from Data* (AMLBook) is a more theoretical approach to ML that provides deep insights, in particular on the bias/variance trade-off (see Chapter 4).
- Stuart Russell and Peter Norvig’s *Artificial Intelligence: A Modern Approach*, 4th edition (Pearson), is a great (and huge) book covering an incredible amount of topics, including machine learning. It helps put ML into perspective.
- Jeremy Howard and Sylvain Gugger’s *Deep Learning for Coders with fastai and PyTorch* (O’Reilly) provides a wonderfully clear and practical introduction to deep learning using the fastai and PyTorch libraries.
- Andrew Ng’s *Machine Learning Yearning* is a free ebook that provides a thoughtful exploration of machine learning, focusing on the practical considerations of building and deploying models, including data quality and long-term maintenance.
- Lewis Tunstall, Leandro von Werra, and Thomas Wolf’s *Natural Language Processing with Transformers: Building Language Applications with Hugging Face* (O’Reilly) is a great practical dive into transformers using popular libraries by Hugging Face.
- Jay Alammar and Maarten Grootendorst’s *Hands-On Large Language Models* is a beautifully illustrated book on LLMs, covering everything you need to know to understand, train, fine-tune, and use LLMs across a wide variety of tasks.

Finally, joining ML competition websites such as Kaggle.com will allow you to practice your skills on real-world problems, with help and insights from some of the best ML professionals out there.

Conventions Used in This Book

The following typographical conventions are used in this book:

Italic

Indicates new terms, URLs, email addresses, filenames, and file extensions.

`Constant width`

Used for program listings, as well as within paragraphs to refer to program elements such as variable or function names, databases, data types, environment variables, statements, and keywords.

`Constant width bold`

Shows commands or other text that should be typed literally by the user.

`Constant width italic`

Shows text that should be replaced with user-supplied values or by values determined by context.

Punctuation

To avoid any confusion, punctuation appears outside of quotes throughout the book. My apologies to the purists.

Tip

This element signifies a tip or suggestion.

Note

This element signifies a general note.

Warning

This element indicates a warning or caution.

O’Reilly Online Learning

Note

For more than 40 years, *O’Reilly Media* has provided technology and business training, knowledge, and insight to help companies succeed.

Our unique network of experts and innovators share their knowledge and expertise through books, articles, and our online learning platform. O’Reilly’s online learning platform gives you on-demand access to live training courses, in-depth learning paths, interactive coding environments, and a vast collection of text and video from O’Reilly and 200+ other publishers. For more information, visit *https://oreilly.com*.

Acknowledgments

Never in my wildest dreams did I imagine that the three TensorFlow editions of this book would reach such a large audience. I received so many messages from readers, many asking questions, some kindly pointing out errata, and most sending me encouraging words. I cannot express how grateful I am to all these readers for their tremendous support. Thank you all so very much! Please do not hesitate to file issues on GitHub if you find errors in the code examples (or just to ask questions), or to submit errata if you find errors in the text. Some readers also shared how this book helped them get their first job, or how it helped them solve a concrete problem they were working on. I find such feedback incredibly motivating. If you find this book helpful, I would love it if you could share your story with me, either privately (e.g., via LinkedIn) or publicly (e.g., tweet me at @aureliengeron or write an Amazon review).

Huge thanks as well to all the generous people who offered their time and expertise to review this book, correcting errors and making countless suggestions. They made this book so much better: Jeremy Howard, Haesun Park, Omar Sanseviero, Lewis Tunstall, Leandro Von Werra, and Sam Witteveen reviewed the table of contents and helped me refine the scope of the book. Hesam Hassani, Ashu Jha, Meetu Malhotra, and Ammar Mohanna reviewed the first part, while Ulf Bissbort, Louis-Francois Bouchard, Luba Elliot, Thomas Lacombe, Tarun Narayanan, Marco Tabor, and my dear brother Sylvain reviewed the second part. Special thanks to Haesun Park, who reviewed every single chapter. You are all amazing!

Of course, this book would not exist without the fantastic staff at O’Reilly. I am especially indebted to Michele Cronin, who reviewed every chapter and supported me weekly for a whole year. I am also deeply grateful to Nicole Butterfield for leading this project and helping refine the book’s scope, and to the production team—particularly Beth Kelly and Kristen Brown—who did a remarkable job. I want to acknowledge Sonia Saruba for her countless careful copyedits, Kate Dullea for making my diagrams much prettier, and Susan Thompson for the beautiful orangutan on the cover.

Last but not least, I am infinitely grateful to my beloved wife, Emmanuelle, and to our three wonderful children—Alexandre, Rémi, and Gabrielle—for encouraging me to work so hard on this book. Their insatiable curiosity was priceless: explaining some of the most difficult concepts in this book to my wife and children helped me clarify my own thoughts and directly improved many parts of it. Plus, they kept bringing me cookies and coffee. Who could ask for more?

[1] Geoffrey E. Hinton et al., "A Fast Learning Algorithm for Deep Belief Nets", *Neural Computation* 18 (2006): 1527–1554.

[2] Despite the fact that Yann LeCun's deep convolutional neural networks had worked well for image recognition since the 1990s, although they were not as general-purpose.

[3] Geoffrey Hinton was awarded the 2018 Turing Award (with Yann LeCun and Yoshua Bengio) and the 2024 Nobel Prize in Physics (with John Hopfield) for early work on neural networks back in the 1980s. DeepMind's founder and CEO Demis Hassabis and director John Jumper were awarded the 2024 Nobel Prize in Chemistry for their work on AlphaFold. They shared this Nobel Prize with another protein researcher, David Baker.