

FROM
SCRATCH

BUILD A Reasoning Model

Sebastian Raschka



Build a Reasoning Model (From Scratch)

Build a Reasoning Model (From Scratch)

SEBASTIAN RASCHKA



MANNING
SHELTER ISLAND

For online information and ordering of this and other Manning books, please visit www.manning.com. The publisher offers discounts on this book when ordered in quantity.

For more information, please contact

Special Sales Department
Manning Publications Co.
20 Baldwin Road
PO Box 761
Shelter Island, NY 11964
Email: orders@manning.com

© 2026 Manning Publications Co. All rights reserved.

No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by means electronic, mechanical, photocopying, or otherwise, without prior written permission of the publisher.

Many of the designations used by manufacturers and sellers to distinguish their products are claimed as trademarks. Where those designations appear in the book, and Manning Publications was aware of a trademark claim, the designations have been printed in initial caps or all caps.

⊗ Recognizing the importance of preserving what has been written, it is Manning's policy to have the books we publish printed on acid-free paper, and we exert our best efforts to that end. Recognizing also our responsibility to conserve the resources of our planet, Manning books are printed on paper that is at least 15 percent recycled and processed without the use of elemental chlorine.

The author and publisher have made every effort to ensure that the information in this book was correct at press time. The author and publisher do not assume and hereby disclaim any liability to any party for any loss, damage, or disruption caused by errors or omissions, whether such errors or omissions result from negligence, accident, or any other cause, or from any usage of the information herein.



Manning Publications Co.
20 Baldwin Road
PO Box 761
Shelter Island, NY 11964

Development editor: Dustin Archibald
Technical editor: David Caswell
Review editor: Kishor Rit
Production editor: Kathy Rossland
Copy editor: Andy Carroll
Proofreader: Katie Tennant
Typesetter: Tamara Švelić Sabljic
Cover designer: Marija Tudor

ISBN 9781633434677

Printed in the United States of America

To Liza, my partner in every adventure; to my beloved family; and to the worldwide community of programmers and creators who have shaped my journey as an author

brief contents

1	■	Understanding reasoning models	1
2	■	Generating text with a pretrained LLM	18
3	■	Evaluating reasoning models	57
4	■	Improving reasoning with inference-time scaling	94
5	■	Inference-time scaling via self-refinement	136
6	■	Training reasoning models with reinforcement learning	179
7	■	Improving GRPO for reinforcement learning	225
8	■	Distilling reasoning models for efficient reasoning	267
<i>appendix A</i>	■	References and further reading	305
<i>appendix B</i>	■	Exercise solutions	314
<i>appendix C</i>	■	Qwen3 LLM source code	336
<i>appendix D</i>	■	Using larger LLMs	358
<i>appendix E</i>	■	Batching and throughput-oriented execution	365
<i>appendix F</i>	■	Common approaches to model evaluation	377
<i>appendix G</i>	■	Building a chat interface	397

contents

<i>preface</i>	<i>xiv</i>
<i>acknowledgments</i>	<i>xvi</i>
<i>about this book</i>	<i>xviii</i>
<i>about the author</i>	<i>xxii</i>
<i>about the cover illustration</i>	<i>xxiii</i>

1	<i>Understanding reasoning models</i>	1
1.1	Defining reasoning in the context of LLMs	2
1.2	Understanding the standard LLM training pipeline	5
1.3	Improving LLM reasoning with training and inference techniques	7
1.4	Pattern matching vs. logical reasoning	10
1.5	Simulating reasoning without explicit rules	11
1.6	Why build reasoning models from scratch?	13
1.7	A road map to building reasoning models from scratch	15
2	<i>Generating text with a pretrained LLM</i>	18
2.1	Introducing LLMs for text generation	19
2.2	Setting up the coding environment	20
2.3	Understanding hardware needs and recommendations	23

- 2.4 Preparing input texts for LLMs 25
- 2.5 Loading pretrained models 29
- 2.6 Understanding the sequential LLM text generation process 34
- 2.7 Coding a minimal text generation function 40
- 2.8 Faster inference via KV caching 46
- 2.9 Faster inference via PyTorch model compilation 50

3 *Evaluating reasoning models* 57

- 3.1 Building a math verifier 58
- 3.2 Loading a pretrained model to generate text 61
- 3.3 Implementing a wrapper for easier text generation 65
- 3.4 Extracting the final answer box 66
- 3.5 Normalizing the extracted answer 71
- 3.6 Verifying mathematical equivalence 74
- 3.7 Grading answers 77
- 3.8 Loading the evaluation dataset 81
- 3.9 Evaluating the model 84

4 *Improving reasoning with inference-time scaling* 94

- 4.1 Introduction to inference-time scaling 95
- 4.2 Loading a pretrained model 98
- 4.3 Generating better responses with chain-of-thought prompting 100
- 4.4 Controlling output diversity with temperature scaling 103
 - Understanding the process of selecting the next token* 104
 - Rescaling token scores (logits) via a temperature parameter* 107
 - Sampling the next token from a probability distribution* 110
 - Adding temperature scaling to the text generation function* 116
- 4.5 Balancing diversity and coherence with top-p sampling 118
 - Selecting a subset of top-p tokens* 119
 - *Adding a top-p filter to the text generation function* 123
- 4.6 Improving response accuracy with self-consistency 127

5	<i>Inference-time scaling via self-refinement</i>	136
5.1	Scoring and iteratively improving model responses	137
5.2	Loading a pretrained model	139
5.3	Scoring LLM responses with a rule-based score	142
5.4	Understanding token probability scores	146
5.5	From token probability scores to log probabilities	156
5.6	Scoring model confidence with log probabilities	162
5.7	Self-refinement through iterative feedback	167
5.8	Coding the self-refinement loop	170
6	<i>Training reasoning models with reinforcement learning</i>	179
6.1	Introduction to RL for LLMs	180
	<i>The original RL pipeline with human feedback (RLHF)</i>	182
	<i>From human feedback to verifiable rewards</i>	185
6.2	RLVR using GRPO	186
	<i>High-level GRPO intuition via a chef analogy</i>	188
	<i>The high-level GRPO procedure</i>	189
6.3	Loading a pretrained model	191
6.4	Loading a MATH training subset	193
6.5	Sampling rollouts	195
6.6	Calculating rewards	199
6.7	Preparing learning signals from rollouts via advantages	201
6.8	Scoring rollouts with sequence log probabilities	203
6.9	From advantages to policy updates via the GRPO loss	208
6.10	Putting everything together in a single GRPO function	211
6.11	Implementing the GRPO training loop	214
6.12	Loading and evaluating saved model checkpoints	220
7	<i>Improving GRPO for reinforcement learning</i>	225
7.1	Improving GRPO	226
7.2	Tracking GRPO performance metrics	226
	<i>Executing a GRPO training run</i>	227
	<i>Inspecting the GRPO training run</i>	229

- 7.3 Tracking more advanced GRPO performance metrics 233
 - Advantage tracking* 234 ■ *Entropy tracking* 236
 - Plotting additional GRPO metrics* 240
- 7.4 Stabilizing sequence-level GRPO using clipped policy ratios 242
 - Computing clipped policy ratios* 243 ■ *Training with clipped policy ratios* 247
- 7.5 Controlling how much the model changes with a KL term 249
 - Implementing the KL loss term* 250 ■ *Training with a KL loss term* 253
- 7.6 Adding an explicit format reward 256
 - Using <think> tokens* 257 ■ *Training a model to emit <think> tokens* 261
 - More GRPO modifications, tips, and tricks* 264

8 *Distilling reasoning models for efficient reasoning* 267

- 8.1 Introducing model distillation for reasoning tasks 268
- 8.2 Generating a dataset for reasoning distillation 271
- 8.3 Loading the MATH training dataset for distillation 273
- 8.4 Building training examples 276
 - Loading and understanding the tokenizer* 277 ■ *Formatting and tokenizing the dataset* 279
 - Filtering and splitting the dataset* 282
- 8.5 Loading a pretrained model 285
- 8.6 Computing the training and validation losses 286
- 8.7 Implementing the training loop for distillation 291
- 8.8 Evaluating the distilled model 296
- 8.9 Future directions for reasoning models 301
- 8.10 Conclusions 302
 - What's next* 302 ■ *Staying up to date in a fast-moving field* 302

appendix A *References and further reading* 305

appendix B *Exercise solutions* 314

appendix C *Qwen3 LLM source code* 336

appendix D *Using larger LLMs* 358

<i>appendix E</i>	<i>Batching and throughput-oriented execution</i>	365
<i>appendix F</i>	<i>Common approaches to model evaluation</i>	377
<i>appendix G</i>	<i>Building a chat interface</i>	397
	<i>index</i>	409

preface

More than a decade ago, my journey into AI began with statistical pattern classification, where I learned how far relatively simple models can go when they capture useful structure in data. Over time, that curiosity shifted from models that recognize patterns to models that can explain, plan, and solve multistep problems.

With the release of ChatGPT in 2022, large language models (LLMs) moved into the mainstream. A later shift, especially visible in late 2024 and throughout 2025, was the extension of LLMs to create so-called reasoning models, which can solve more complex problems, especially in technical domains such as math and code.

At the same time, reasoning models can feel opaque, and this book is my attempt to make reasoning models more approachable. Rather than training a giant model from scratch, we'll begin with a small pretrained LLM and apply reasoning techniques step by step. Along the way, you will implement text generation, evaluation with verifiers, inference-time scaling methods, and training methods, such as reinforcement learning with verifiable rewards and distillation from scratch. By the end, you will understand how the main reasoning methods work in practice and how they fit into a modern LLM development workflow.

Like my earlier book, *Build a Large Language Model (From Scratch)*, this one takes a code-first approach, but the focus here is different. Instead of explaining the Transformer architecture in depth or covering large-scale pretraining, here we'll concentrate on the methods that turn a conventional LLM into a reasoning model. If you have read the earlier book, this one should feel like a natural next step. If you haven't, you can still follow along here without needing all the underlying LLM details up front.

We'll use math examples throughout the book because they are practical for reasoning research and easy to verify automatically. That makes them a useful test bed for

understanding whether a model is actually solving a problem correctly. More broadly, my goal is not to teach one narrow benchmark but to give you the tools and intuition needed to build, evaluate, and improve reasoning systems in your own work.

When I first started working on reasoning models, I had to piece together ideas from papers, repositories, and scattered implementation notes. I hope this book saves you that effort and gives you a clearer path from first principles to working code. I strongly believe that the best way to understand reasoning models is to build one yourself.

Happy reading and coding!

acknowledgments

Writing a book is a significant undertaking, and I would like to express my sincere gratitude to my wife, Liza, for her patience and support throughout this process. Her unconditional love and constant encouragement have been absolutely essential.

I am incredibly grateful to Daniel Kleine and Dmitry Labazkin, whose invaluable feedback on the draft chapters and code went far above and beyond. Daniel's keen eye for detail and style, along with Dmitry's sharp insights into code performance, have undoubtedly made this book a smoother and more enjoyable read for all.

I would also like to thank the wonderful team at Manning Publications. Michael Stephens, thank you for the many productive conversations that helped shape the direction of this book. Dustin Archibald, I greatly appreciate your constructive guidance on adhering to Manning's guidelines, as well as your flexibility in accommodating the unique demands of this unconventional from-scratch approach. Special thanks also go to Aleksandar Dragosavljevic and Ivan Martinović for their work on layout and formatting. Finally, technical editor David Caswell, thank you for your thoughtful and generous feedback throughout the process. David is an industry-leading consultant focused on applying LLMs to journalism. He previously led AI innovation at the BBC, Tribune Publishing, and Yahoo!, he publishes peer-reviewed research on AI automation of information workflows, and he is a frequent speaker and writer on LLMs in the emerging AI-mediated information ecosystem.

Finally, I extend my thanks to the reviewers Alejandro Cuevas Rivero, Muhammad Ali Shafique, Arun Prakash A, Christopher Brousseau, David Curran, Fabio Montagna, Hamza Farooq, Hongming Zheng, Julien Thomazo, Lindo William Khoza, Michael Anthony Garcia, Naman Dwivedi, Pere Martra, Pradeep Dasigi, Salvatore Raieli, Sifal Klioui, Syed Baqir Ali, Thomas Viehmann, Tianrui Liu, Toni Ramchandani, and Viton

Vitanis for their thorough feedback on the drafts. Your keen eyes and insightful comments have been essential in improving the quality of this book.

To everyone who has contributed to this journey, I am sincerely grateful. Your support, expertise, and dedication have been instrumental in bringing this book to fruition. Thank you!

about this book

Build a Reasoning Model (From Scratch) is a hands-on guide to understanding how modern reasoning LLMs work by implementing their core techniques yourself. Rather than training a base model from the ground up, this book starts with a pretrained open source base LLM and shows you how to extend it step by step with reasoning capabilities. Along the way, you will learn how to generate text with a base model, evaluate answers with verifiers, improve performance through inference-time scaling, train with reinforcement learning, and distill reasoning behavior into smaller models. By the end of the book, you will understand how the main reasoning methods fit together in a practical LLM development workflow and how to build small but functional reasoning systems of your own.

The book takes a code-first approach. The “from scratch” part refers to implementing the reasoning methods yourself so that the underlying mechanics become transparent. If you are interested in how base LLMs are built and pretrained in detail, my earlier book, *Build a Large Language Model (From Scratch)* (<http://mng.bz/M96o>), provides that foundation, but it is not required for reading this book.

Who should read this book

Build a Reasoning Model (From Scratch) is for machine learning enthusiasts, engineers, researchers, students, and software developers who want a practical understanding of how reasoning models work and how to implement the main techniques behind them. It is written for readers who want to go beyond high-level descriptions and see how these systems are built, evaluated, and improved in code.

The most important prerequisite is solid Python experience. Prior exposure to machine learning, deep learning, or large language models is helpful, but you do not

need to be an expert. The book is intended to be approachable for motivated beginners while still offering enough technical depth for experienced practitioners.

The book uses math examples frequently because they are practical for reasoning research and easy to verify automatically. However, advanced mathematics is not required. Familiarity with basic algebra and a general comfort level with vectors and matrices is useful, though, to follow some of the code implementations.

Some prior PyTorch experience may help you move more quickly through the code, but it is not required. The focus of the book is on understanding reasoning methods and implementing them clearly, not on mastering every detail of a deep learning framework. Readers who have already read *Build a Large Language Model (From Scratch)* may recognize some broader LLM concepts, but this book is fully self-contained.

How this book is organized: A road map

This book is organized around a practical development pipeline for reasoning models. We'll begin with a conventional pretrained LLM, add evaluation so that we can measure progress, and then explore two main ways of improving reasoning: inference-time methods and additional training. Because later chapters build directly on the code and ideas introduced earlier, the book is best read in sequence.

Chapter 1 introduces the practical meaning of reasoning in the context of LLMs. It explains how reasoning models differ from conventional LLMs, it reviews the standard LLM training pipeline at a high level, and it introduces the main approaches used to improve reasoning.

Chapter 2 establishes the starting point by loading and using a pretrained base LLM. It covers the coding environment, tokenization, step-by-step text generation, and practical speedups such as key-value caching and model code compilation.

Chapter 3 adds evaluation. It shows how to extract final answers reliably, verify correctness against reference solutions, and build an evaluation pipeline for math-oriented reasoning tasks.

With that foundation in place, chapters 4 and 5 focus on inference-time scaling, improving reasoning without changing the model weights. Chapter 4 covers prompt-based reasoning, diverse decoding, and self-consistency. Chapter 5 extends this direction with self-refinement, simple scoring strategies, confidence estimation, and iterative answer improvement.

Chapters 6 through 8 implement training-time methods. Chapter 6 introduces reinforcement learning for reasoning models, with a focus on reinforcement learning with verifiable rewards (RLVR) and Group Relative Policy Optimization (GRPO). Chapter 7 builds on that implementation by covering practical monitoring, common training-failure modes, reward hacking, and useful extensions such as KL regularization and response-format rewards. Chapter 8 concludes the main text with distillation, showing how to create teacher-generated reasoning datasets, train a smaller student model, and evaluate the distilled result.