“Andriy's long-awaited sequel in his "The Hundred-Page" series of machine learning textbooks is a masterpiece of concision.”

— **Bob van Luij**t, CEO and Co-Founder of Weaviate

“Andriy has this almost supernatural talent for shrinking epic AI concepts down to bite-sized, ‘Ah, now I get it!’ moments.”

— **Jorge Torres**, CEO at MindsDB

“Andriy paints for us, in 100 marvelous strokes, the journey from linear algebra basics to the implementation of transformers.”

— **Florian Douetteau**, Co-founder and CEO at Dataiku

“Andriy's book is an incredibly concise, clear, and accessible introduction to machine learning.”

— **Andre Zayarni**, Co-founder and CEO at Qdrant

“This is one of the most comprehensive yet concise handbooks out there for truly understanding how LLMs work under the hood.”

— **Jerry Liu**, Co-founder and CEO at LlamaIndex

Featuring a foreword by **Tomáš Mikolov** and back cover text by **Vint Cerf**

The Hundred-Page Language Models Book

Andriy Burkov

ISBN 978-1-7780427-2-0

Publisher: True Positive Inc.

To my family, with love

“Language is the source of misunderstandings.”
—***Antoine de Saint-Exupéry***, *The Little Prince*

“In mathematics you don't understand things. You just get used to them.”
—John von Neumann

“Computers are useless. They can only give you answers.”
— Pablo Picasso

The book is distributed on the “read first, buy later” principle

Contents

Chapter 6. Further Reading

You've learned the core concepts of language modeling throughout this book. There are many advanced topics to explore on your own, and this final chapter provides pointers for further study. I've chosen topics that represent important current developments in the field, from architectural innovations to security considerations.

6.1. Mixture of Experts

Mixture of experts (**MoE**) is an architectural pattern designed to increase model capacity without a proportional rise in cost. Instead of a single position-wise **MLP** processing all tokens in a decoder block, MoE uses multiple specialized sub-networks called **experts**. A **router network** (or **gate network**) decides which tokens are processed by which experts.

The core idea is activating only a subset of experts for each token. This **sparse** activation reduces active computations while enabling larger overall parameter counts. **Sparse MoE layers** replace traditional MLP layers, using techniques like **top-k routing** and **load balancing** to efficiently assign tokens to experts.

This concept gained attention with the **Switch Transformer** and has been applied in models such as **Mixtral 8x7B**, which has 47B total parameters but only activates about 13B during inference.

6.2. Model Merging

Model merging combines multiple pretrained models to make use of their complementary strengths. Techniques include **model soups**, **SLERP** (spherical interpolation that maintains parameter norms), and **task vector algorithms** such as **TIES-Merging** and **DARE**.

These methods generally rely on some architectural similarity or compatibility between models. The **passthrough** method stands out by concatenating layers from different LLMs. This approach can create models with unconventional parameter counts (e.g., 13B by merging two 7B models). Such models are often called **frankenmerges**.

mergekit is a popular open-source tool for merging and combining language models that implements many of these techniques. It provides a flexible configuration system for experimenting with different merging strategies and architectures.

6.3. Model Compression

Model compression addresses deploying LLMs in resource-limited environments by reducing size and computation needs without greatly sacrificing performance. Neural networks are often **over-parameterized**, containing redundant units that can be optimized.

Key methods include **post-training quantization**, which lowers parameter precision (e.g., 32-bit floats to 8-bit integers), **quantization-aware training**, training models at lower precision, such as **QLoRA** (quantized low-rank adaptation), **unstructured pruning**, removing individual weights by importance, **structured pruning**, removing components like layers or attention heads, and **knowledge distillation**, where a smaller "student" model learns from a larger "teacher" model.

6.4. Preference-Based Alignment

Preference-based alignment methods help align LLMs with user values and intent, so they produce helpful and safe outputs. A widely used approach is **reinforcement learning from human feedback** (**RLHF**), where humans rank model responses, a **reward model** is trained on these rankings, and then the LLM is finetuned to optimize for higher reward.

Another approach is **constitutional AI** (CAI), which uses a set of guiding principles or a "constitution" that the model refers to when producing its output; the model can **self-critique** and revise its responses based on these principles. Both strategies address the problem that LLMs, when trained on vast internet text, may generate harmful or **misaligned** responses, but they differ in how they incorporate human oversight and explicit guidelines.

6.5. Advanced Reasoning

Advanced reasoning techniques enable large language models to handle complex tasks by (1) training them to generate an explicit **chain of thought** (**CoT**) for step-by-step reasoning and (2) equipping them with **function calling** capabilities to invoke external APIs or tools, thereby addressing limitations of simple prompt-response patterns. Chain-of-thought reasoning can significantly improve performance on tasks such as multi-step mathematics and logical inference, while function calling allows offloading specialized computations to external frameworks.

Additionally, **tree of thought** (*ToT*) extends CoT by exploring multiple reasoning paths in a tree-like structure. **Self-consistency** further refines reasoning by aggregating multiple CoT outputs for the most consistent answer. **ReAct** (**reasoning+act**) integrates reasoning with action-taking, allowing models to interact with environments dynamically. **Program-aided language models** (**PAL**) leverage interpreters (e.g., Python) to execute code for precise calculations.

6.6. Language Model Security

Jailbreak attacks and **prompt injection** are major security vulnerabilities in LLMs. Jailbreaks bypass the model's safety controls by crafting specific inputs that trick the model into producing restricted content, often using techniques like roleplaying as a different character or setting up hypothetical scenarios. For example, an attacker might prompt the model to act as a pirate to obtain instructions on illegal activities.

In contrast, prompt injection attacks manipulate how LLM applications combine **system prompts** with user input, allowing attackers to alter the application's behavior. For instance, an attacker could insert commands that make the application execute unauthorized actions. While jailbreaks primarily risk exposing harmful or restricted content, prompt injection presents more severe security implications for applications with privileged access, such as those that read emails or execute system commands.

6.7. Vision Language Model

Vision language models (**VLMs**) integrate an LLM with a **vision encoder** to handle both text and images. Unlike traditional models that process modalities in isolation, VLMs excel at **multimodal reasoning**, enabling them to perform a variety of vision tasks by following natural language instructions without task-specific retraining. The architecture includes three main components: a **CLIP**-based (contrastive language-image pretraining) **vision encoder** trained on millions of image-text pairs to understand visual content, a **cross-attention** mechanism that allows the VLM to integrate and reason about visual and textual information, and the language model itself that generates and interprets text. VLMs are developed through multiple training stages, starting with pretraining to align the visual and language components, followed by supervised finetuning to improve their ability to understand and respond to user prompts.

6.8. Preventing Overfitting

Techniques for preventing **overfitting** are essential for achieving model **generalization**, ensuring that models perform well not just on training data but also on new, unseen examples. The primary defense against overfitting is **regularization**, which includes methods like **L1** and **L2**. These techniques add specific penalty terms—such as the sum of absolute or squared weights—to the loss function, limiting the size of model parameters and encouraging simpler models.

Dropout is a regularization method for neural networks. It works by randomly deactivating some units during each training step. This encourages the network to develop multiple independent pathways, reducing reliance on specific features. **Early stopping** prevents overfitting by monitoring validation performance. Training stops when validation accuracy stops improving or starts to decline, avoiding the memorization of random noise happening at later epochs.

A **validation set** is similar to the **test set** in that it is used to evaluate the model's performance on unseen data; however, the key difference is that the validation set is used during the training process to tune hyperparameters and make decisions such as early stopping, while the test set is reserved for final evaluation to measure the model's performance after training is complete.

6.9. Concluding Remarks

You've come a long way in understanding language models, from the basic building blocks of machine learning to the inner workings of transformers and the practical aspects of working with large language models. You now have a solid technical foundation that lets you not only understand how these models work but also implement and adapt them for your own purposes.

New architectures, training methods, and applications of language models are emerging. You now have the tools to read research papers, follow technical discussions, and evaluate new developments critically. Whether you aim to train models or build systems using them, you have the core concepts to proceed confidently.

I encourage you to stay curious and hands-on—implement the concepts you've learned, experiment with different approaches, and keep up with the latest developments. Consider starting with some of the advanced topics covered in this chapter, but remember that the fundamentals you've learned here will serve as your compass in navigating future innovations.

A good way of keeping up with the latest developments is to subscribe to the book's newsletter.

The book ends here. Remember to check the companion wiki from time to time for updates on developments in various language modeling areas. Please don't forget that the book is shared under the *read first, buy later* principle. So, if you're reading this as a PDF and don't recall paying for it, you are probably the right person to purchase the book.

6.10. More From the Author

If you're still reading, it likely means you enjoyed the book and are wondering what else you can read from this author. I have two more books that will definitely enhance your understanding of machine learning and build on the knowledge and intuition you've gained about language models:

- **The Hundred-Page Machine Learning Book** offers a concise yet thorough overview of core machine learning concepts, ranging from fundamental statistics to advanced algorithms. It's an excellent companion to the language modeling material covered here.
- **Machine Learning Engineering** covers the practical aspects of designing, deploying, and maintaining ML systems at scale. If you're looking to move beyond experimentation and create robust, real-world machine learning applications, this book will guide you through every stage of the machine learning engineering lifecycle.