appendix B References and further reading

Chapter 1

Custom-built LLMs are able to outperform general-purpose LLMs as a team at Bloomberg showed via a version of GPT pretrained on finance data from scratch. The custom LLM outperformed ChatGPT on financial tasks while maintaining good performance on general LLM benchmarks:

- "BloombergGPT: A Large Language Model for Finance" (2023) by Wu et al., https://arxiv.org/abs/2303.17564

Existing LLMs can be adapted and fine-tuned to outperform general LLMs as well, which teams from Google Research and Google DeepMind showed in a medical context:

- "Towards Expert-Level Medical Question Answering with Large Language Models" (2023) by Singhal et al., https://arxiv.org/abs/2305.09617

The following paper proposed the original transformer architecture:

- "Attention Is All You Need" (2017) by Vaswani et al., https://arxiv.org/abs/1706.03762

On the original encoder-style transformer, called BERT, see

- "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding" (2018) by Devlin et al., https://arxiv.org/abs/1810.04805

The paper describing the decoder-style GPT-3 model, which inspired modern LLMs and will be used as a template for implementing an LLM from scratch in this book, is

- "Language Models are Few-Shot Learners" (2020) by Brown et al., https://arxiv.org/abs/2005.14165

The following covers the original vision transformer for classifying images, which illustrates that transformer architectures are not only restricted to text inputs:

- "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale" (2020) by Dosovitskiy et al., https://arxiv.org/abs/2010.11929

The following experimental (but less popular) LLM architectures serve as examples that not all LLMs need to be based on the transformer architecture:

- "RWKV: Reinventing RNNs for the Transformer Era" (2023) by Peng et al., https://arxiv.org/abs/2305.13048
- "Hyena Hierarchy: Towards Larger Convolutional Language Models" (2023) by Poli et al., https://arxiv.org/abs/2302.10866
- "Mamba: Linear-Time Sequence Modeling with Selective State Spaces" (2023) by Gu and Dao, https://arxiv.org/abs/2312.00752

Meta AI's model is a popular implementation of a GPT-like model that is openly available in contrast to GPT-3 and ChatGPT:

- "Llama 2: Open Foundation and Fine-Tuned Chat Models" (2023) by Touvron et al., https://arxiv.org/abs/2307.092881

For readers interested in additional details about the dataset references in section 1.5, this paper describes the publicly available *The Pile* dataset curated by Eleuther AI:

- "The Pile: An 800GB Dataset of Diverse Text for Language Modeling" (2020) by Gao et al., https://arxiv.org/abs/2101.00027

The following paper provides the reference for InstructGPT for fine-tuning GPT-3, which was mentioned in section 1.6 and will be discussed in more detail in chapter 7:

- "Training Language Models to Follow Instructions with Human Feedback" (2022) by Ouyang et al., https://arxiv.org/abs/2203.02155

Chapter 2

Readers who are interested in discussion and comparison of embedding spaces with latent spaces and the general notion of vector representations can find more information in the first chapter of my book:

- *Machine Learning Q and AI* (2023) by Sebastian Raschka, https://leanpub.com/machine-learning-q-and-ai

The following paper provides more in-depth discussions of how byte pair encoding is used as a tokenization method:

- "Neural Machine Translation of Rare Words with Subword Units" (2015) by Sennrich et al., https://arxiv.org/abs/1508.07909

The code for the byte pair encoding tokenizer used to train GPT-2 was open-sourced by OpenAI:

- https://github.com/openai/gpt-2/blob/master/src/encoder.py

OpenAI provides an interactive web UI to illustrate how the byte pair tokenizer in GPT models works:

- https://platform.openai.com/tokenizer

For readers interested in coding and training a BPE tokenizer from the ground up, Andrej Karpathy's GitHub repository `minbpe` offers a minimal and readable implementation:

- "A Minimal Implementation of a BPE Tokenizer," https://github.com/karpathy/minbpe

Readers who are interested in studying alternative tokenization schemes that are used by some other popular LLMs can find more information in the SentencePiece and WordPiece papers:

- "SentencePiece: A Simple and Language Independent Subword Tokenizer and Detokenizer for Neural Text Processing" (2018) by Kudo and Richardson, https://aclanthology.org/D18-2012/
- "Fast WordPiece Tokenization" (2020) by Song et al., https://arxiv.org/abs/2012.15524

Chapter 3

Readers interested in learning more about Bahdanau attention for RNN and language translation can find detailed insights in the following paper:

- "Neural Machine Translation by Jointly Learning to Align and Translate" (2014) by Bahdanau, Cho, and Bengio, https://arxiv.org/abs/1409.0473

The concept of self-attention as scaled dot-product attention was introduced in the original transformer paper:

- "Attention Is All You Need" (2017) by Vaswani et al., https://arxiv.org/abs/1706.03762

FlashAttention is a highly efficient implementation of a self-attention mechanism, which accelerates the computation process by optimizing memory access patterns. FlashAttention is mathematically the same as the standard self-attention mechanism but optimizes the computational process for efficiency:

- "FlashAttention: Fast and Memory-Efficient Exact Attention with IO-Awareness" (2022) by Dao et al., https://arxiv.org/abs/2205.14135
- "FlashAttention-2: Faster Attention with Better Parallelism and Work Partitioning" (2023) by Dao, https://arxiv.org/abs/2307.08691