
Dive into
Deep Learning
LANGUAGE MODELS
Alexander J Smola

Table of contents

Language Models 1

18 Working with Pretrained Language Models 3
18.1 Pretrained Models with Hugging Face . . . 3
18.2 A Released Checkpoint in the Book's Own Code . . . 3
18.3 Prompting and In-Context Learning . . . 3
18.4 Scoring Language Models . . . 3
18.5 Evaluation Suites, Judges and Error Bars . . . 3
18.6 Interpretability . . . 3
18.7 Models behind an API . . . 3

19 Encoders, Embeddings and Retrieval 5
19.1 Word Embeddings . . . 5
19.2 Pretrained Encoders . . . 5
19.3 Text Embeddings from Contrastive Training . . . 5
19.4 Sparse, Dense and Approximate Retrieval . . . 5
19.5 Retrieval-Augmented Generation . . . 5
19.6 Further Reading . . . 5

20 Data for Language Models 7
20.1 Document Extraction from Web Crawls . . . 7
20.2 Quality Filtering . . . 7
20.3 Deduplication and Decontamination . . . 7
20.4 Data Mixtures . . . 7
20.5 Synthetic and Rephrased Data . . . 7
20.6 Tokenizers at Scale . . . 7

21 Pretraining 9
21.1 The Pretraining Data Path . . . 9
21.2 The Reference Pretraining Run . . . 9
21.3 Low-Precision Training . . . 9
21.4 Continued Pretraining and Mid-Training . . . 9
21.5 Long-Context Training . . . 9
21.6 Training Budgets beyond the Compute Optimum . . . 9

22 Hardware for Training and Inference 11
22.1 Accelerator Design . . . 11
22.2 Kernels, Tiles and Fusion . . . 11

22.3 Training, Prefill and Decode Workloads . . . 11
22.4 Memory and Storage Tiers . . . 11
22.5 Data Paths inside a Server . . . 11
22.6 Scale-Up Domains and Scale-Out Networks . . . 11
22.7 Collective Communication at Scale . . . 11
22.8 Energy, Power and Cost . . . 11

23 Parallelism Beyond One Node **13**
23.1 Multi-Node Jobs and Hang Diagnosis . . . 13
23.2 Sharded Data Parallelism at Scale . . . 13
23.3 Tensor and Sequence Parallelism . . . 13
23.4 Pipeline Parallelism . . . 13
23.5 Context Parallelism . . . 13
23.6 Expert Parallelism . . . 13
23.7 Composed Parallelism and Topology . . . 13
23.8 Profiling and Performance Debugging of Distributed Runs . . . 13
23.9 Checkpoints at Scale . . . 13
23.10 Failures, Elasticity and Goodput . . . 13

24 Fine-Tuning **15**
24.1 Full Fine-Tuning . . . 15
24.2 Supervised Instruction Tuning . . . 15
24.3 Instruction Data . . . 15
24.4 Low-Rank Adaptation . . . 15
24.5 Quantized Fine-Tuning . . . 15
24.6 Fine-Tuning at Scale and Toolchains . . . 15
24.7 Merging and Retaining Capabilities . . . 15

25 Preference Optimization and Reinforcement Learning **17**
25.1 Reward Models . . . 17
25.2 RLHF with PPO at Language-Model Scale . . . 17
25.3 Direct Preference Optimization and Its Variants . . . 17
25.4 RL with Verifiable Rewards . . . 17
25.5 RL Training Systems . . . 17
25.6 Reward Overoptimization . . . 17
25.7 Safety Evaluation and Red-Teaming . . . 17
25.8 Further Reading . . . 17

26 Distillation and Reasoning **19**
26.1 Knowledge Distillation . . . 19
26.2 On-Policy Distillation . . . 19
26.3 Training Reasoning Models . . . 19
26.4 Verifiers and Process Supervision . . . 19
26.5 Parallel Test-Time Compute . . . 19
26.6 Sequential and Adaptive Test-Time Compute . . . 19

27 Agents and Tool Use **21**
27.1 Tool Calling . . . 21
27.2 Structured Outputs and Constrained Decoding . . . 21
27.3 Agent Loops and Harnesses . . . 21
27.4 Training Agents with Reinforcement Learning . . . 21

27.5 Evaluating and Securing Agents 21

28 Model Compression **23**
28.1 Quantization for Inference 23
28.2 Activation Outliers and Equivalent Rescaling 23
28.3 Reconstruction-Based Quantization 23
28.4 Rotations and Incoherence 23
28.5 Quantization-Aware Training and Native Low-Bit Formats 23
28.6 Pruning and Sparsity 23
28.7 Further Reading 23

29 Efficient Inference **25**
29.1 A Fast Decoder on One GPU 25
29.2 Decode Attention Kernels 25
29.3 Deterministic and Batch-Invariant Inference 25
29.4 Speculative Decoding 25
29.5 Drafters and Multi-Token Prediction 25
29.6 KV-Cache Quantization 25
29.7 KV-Cache Eviction and Selective Reading 25
29.8 Inference on CPUs and Devices 25

30 Serving Systems **27**
30.1 Workloads and Service Objectives 27
30.2 Continuous Batching and Chunked Prefill 27
30.3 Paged KV Memory 27
30.4 Prefix Caching and Shared-Prefix Workloads 27
30.5 Scheduling under Overload 27
30.6 KV-Cache Offload, Stores and Prefix-Aware Routing 27
30.7 Tensor and Expert Parallelism for Serving 27
30.8 Prefill and Decode Disaggregation 27
30.9 Serving Many Adapters 27

31 Advanced Architectures **29**
31.1 Architecture Configurations 29
31.2 Mixture-of-Experts Design at Scale 29
31.3 Latent Attention and Cross-Layer KV Sharing 29
31.4 Softmax Attention Refinements 29
31.5 Learned Sparse Attention 29
31.6 Hybrid Models at Scale 29
31.7 The Residual Stream across Depth 29
31.8 Further Reading 29

Language Models

The part teaches how modern language models are built, scaled across machines, adapted, aligned, extended to tools and multi-step tasks, compressed and served, and how current architectures depart from the transformer built in Advanced. It starts where Advanced stops.

Chapter 18

Working with Pretrained Language Models

The reader has trained the GPT of Section 11.2 on a few hundred megabytes of text and has seen released models answer questions that model cannot. This chapter loads such a model, runs the same tensors in the book's own code, scores it under explicit conventions, opens it, and reaches the models that no local machine can hold.

18.1 Pretrained Models with Hugging Face

18.2 A Released Checkpoint in the Book's Own Code

18.3 Prompting and In-Context Learning

18.4 Scoring Language Models

18.5 Evaluation Suites, Judges and Error Bars

18.6 Interpretability

18.7 Models behind an API

Chapter 19

Encoders, Embeddings and Retrieval

The reader has a decoder that answers questions but no way to find, among a million passages, the one that holds the answer, and no cheap way to label ten thousand sentences an hour.

19.1 Word Embeddings

19.2 Pretrained Encoders

19.3 Text Embeddings from Contrastive Training

19.4 Sparse, Dense and Approximate Retrieval

19.5 Retrieval-Augmented Generation

19.6 Further Reading

Chapter 20

Data for Language Models

The reader has trained on a text file someone else prepared. Released models start from billions of pages, and every choice made before the first batch (what to extract, what to drop, what to repeat, how to cut it into tokens) changes what the model can ever learn.

20.1 Document Extraction from Web Crawls

20.2 Quality Filtering

20.3 Deduplication and Decontamination

20.4 Data Mixtures

20.5 Synthetic and Rephrased Data

20.6 Tokenizers at Scale

Chapter 21

Pretraining

The reader has trained the GPT of Section 11.2 for an afternoon on one GPU and read its loss curve. A real pretraining run lasts days, is interrupted and resumed, spikes at hour six, runs its matrix multiplications in eight bits, is continued on new data after it finishes, and is read almost entirely from its logs. This chapter builds the parts of that run that an afternoon never exercises, and trains the reference model that later chapters shard, fine-tune, quantize and serve.

21.1 The Pretraining Data Path

21.2 The Reference Pretraining Run

21.3 Low-Precision Training

21.4 Continued Pretraining and Mid-Training

21.5 Long-Context Training

21.6 Training Budgets beyond the Compute Optimum

Chapter 22

Hardware for Training and Inference

The reader knows one GPU from Section 13.2 and Section 13.1: a roofline, a memory hierarchy, a prefill that is compute-bound and a decode that is not. A released model is trained on thousands of chips and served from racks, and the chips, the kernels that run on them, the links between them and the power they draw decide what a token costs.

22.1 Accelerator Design

22.2 Kernels, Tiles and Fusion

22.3 Training, Prefill and Decode Workloads

22.4 Memory and Storage Tiers

22.5 Data Paths inside a Server

22.6 Scale-Up Domains and Scale-Out Networks

22.7 Collective Communication at Scale

22.8 Energy, Power and Cost

Chapter 23

Parallelism Beyond One Node

The reader ran DDP on four GPUs in Section 13.6 and derived the ZeRO stages without executing them; an 8B model's 128 GB of training state fits no card, so the state is cut up and the pieces must talk, across machines, for weeks, while the hardware fails under them.

23.1 Multi-Node Jobs and Hang Diagnosis

23.2 Sharded Data Parallelism at Scale

23.3 Tensor and Sequence Parallelism

23.4 Pipeline Parallelism

23.5 Context Parallelism

23.6 Expert Parallelism

23.7 Composed Parallelism and Topology

23.8 Profiling and Performance Debugging of Distributed Runs

23.9 Checkpoints at Scale

23.10 Failures, Elasticity and Goodput

Chapter 24

Fine-Tuning

The reader has a pretrained model that continues text and a task it should do instead: follow an instruction, keep to a format, stop when the answer is complete, classify a message. Changing some weights on a few thousand examples is the oldest route, and the questions that remain are which weights may move, in what precision, on which examples, and what the base model forgets while they move.

24.1 Full Fine-Tuning

24.2 Supervised Instruction Tuning

24.3 Instruction Data

24.4 Low-Rank Adaptation

24.5 Quantized Fine-Tuning

24.6 Fine-Tuning at Scale and Toolchains

24.7 Merging and Retaining Capabilities

Chapter 25

Preference Optimization and Reinforcement Learning

The reader derived GRPO and DPO on a toy verifier task in Advanced and saw the KL-regularized optimum in closed form; on a real language model the reward is a second network that can be wrong, sampling takes most of every step, four copies of the model compete for one GPU, and the policy learns whatever the reward left unsaid. This chapter runs the methods the reader already knows on Qwen3-0.6B and measures the symptoms that decide which variant is usable.

25.1 Reward Models

25.2 RLHF with PPO at Language-Model Scale

25.3 Direct Preference Optimization and Its Variants

25.4 RL with Verifiable Rewards

25.5 RL Training Systems

25.6 Reward Overoptimization

25.7 Safety Evaluation and Red-Teaming

25.8 Further Reading

Chapter 26

Distillation and Reasoning

The reader has a 1.7B model that solves the arithmetic family after the reinforcement learning of Chapter 8, and a 0.6B model that does not. This chapter moves the capability into the smaller model, and then shows how to buy accuracy at the moment of answering with tokens instead of parameters.

26.1 Knowledge Distillation

26.2 On-Policy Distillation

26.3 Training Reasoning Models

26.4 Verifiers and Process Supervision

26.5 Parallel Test-Time Compute

26.6 Sequential and Adaptive Test-Time Compute

Chapter 27

Agents and Tool Use

The reader has prompted a model, fine-tuned it and trained it with a verifier; a model that only emits text can be asked a question, while a model that emits actions can be given a job. This chapter turns the models of Chapters 1 to 9 into agents: systems that call a tool, read what comes back, run code and keep going over many turns.

27.1 Tool Calling

27.2 Structured Outputs and Constrained Decoding

27.3 Agent Loops and Harnesses

27.4 Training Agents with Reinforcement Learning

27.5 Evaluating and Securing Agents

Chapter 28

Model Compression

In Section 24.5 the reader stored a frozen model in four bits so that a fine-tuning run would fit, and the model ran no faster. A served model has to be fast, its activations enter the arithmetic, and it is judged on held-out quality at a stated speed. This chapter builds, one mechanism at a time, the methods that compress a trained model for that purpose, and checks each against brute force, an exact identity, or a paired interval.

28.1 Quantization for Inference

28.2 Activation Outliers and Equivalent Rescaling

28.3 Reconstruction-Based Quantization

28.4 Rotations and Incoherence

28.5 Quantization-Aware Training and Native Low-Bit Formats

28.6 Pruning and Sparsity

28.7 Further Reading

Chapter 29

Efficient Inference

The reader has a compressed model from Chapter 11 and, from Section 11.3 and Section 13.2, a bound on how fast one GPU can decode it; the plain generation loop reaches a small fraction of that bound. This chapter takes one request, or a handful, from the eager loop to the bound and then past its per-token form: the kernels that read the weights and the cache at full bandwidth, the reduction orders that make outputs reproducible, the verification trick that checks several tokens per weight read, and the cache compressions that a trained model tolerates. It ends on a laptop, where the same bound holds with a tenth of the bandwidth.

29.1 A Fast Decoder on One GPU

29.2 Decode Attention Kernels

29.3 Deterministic and Batch-Invariant Inference

29.4 Speculative Decoding

29.5 Drafters and Multi-Token Prediction

29.6 KV-Cache Quantization

29.7 KV-Cache Eviction and Selective Reading

29.8 Inference on CPUs and Devices

Chapter 30

Serving Systems

The reader has Section 29.1's decoder, which answers one request near the bandwidth bound. A service never sees one request: it sees an arrival process, a mix of prompt and output lengths, a latency contract, and a hundred users who are each waiting. This chapter builds the mechanisms inside a serving engine, each with an invariant a test can check and a measurement against a reference server, from the contract a service makes to the layouts that spread one model over a node.

30.1 Workloads and Service Objectives

30.2 Continuous Batching and Chunked Prefill

30.3 Paged KV Memory

30.4 Prefix Caching and Shared-Prefix Workloads

30.5 Scheduling under Overload

30.6 KV-Cache Offload, Stores and Prefix-Aware Routing

30.7 Tensor and Expert Parallelism for Serving

30.8 Prefill and Decode Disaggregation

30.9 Serving Many Adapters

Chapter 31

Advanced Architectures

The reader has the transformer of Advanced (RoPE, GQA, RMSNorm, SwiGLU, a mixture of experts, sliding windows and sinks, DeltaNet and hybrid stacks) and a configuration file from Section 18.1 with fields that match none of it: a latent rank, an indexer that keeps 512 of a million keys, a layer type called `linear_attention`, an `hc_mult` of 4, an embedding table with 384 million rows. This chapter teaches those fields, one design axis per section, and for each change asks what it saves, what it costs, and which training and serving setup makes the trade worthwhile.

31.1 Architecture Configurations

31.2 Mixture-of-Experts Design at Scale

31.3 Latent Attention and Cross-Layer KV Sharing

31.4 Softmax Attention Refinements

31.5 Learned Sparse Attention

31.6 Hybrid Models at Scale

31.7 The Residual Stream across Depth

31.8 Further Reading