DADS7305: MLOPs

Northeastern University

Instructor: Ramin Mohammadi

September 7, 2025

These materials have been prepared and sourced for the course **MLOPs** at Northeastern University. Every effort has been made to provide proper citations and credit for all referenced works.

If you believe any material has been inadequately cited or requires correction, please contact me at:

r.mohammadi@northeastern.edu

Thank you for your understanding and collaboration.

LLMs and Model Development

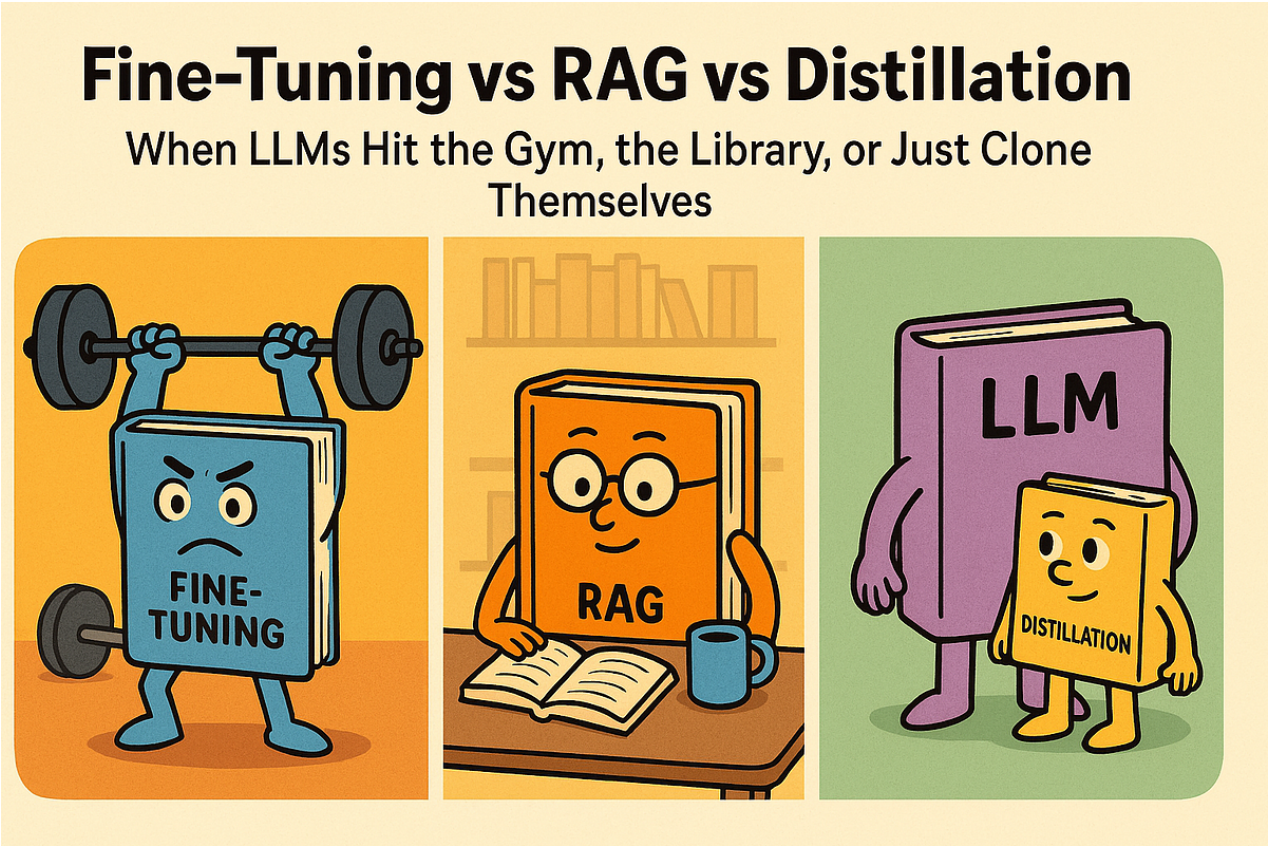
Key Considerations

From Model Development to Adaptation (Control Surface)

What enterprises actually control

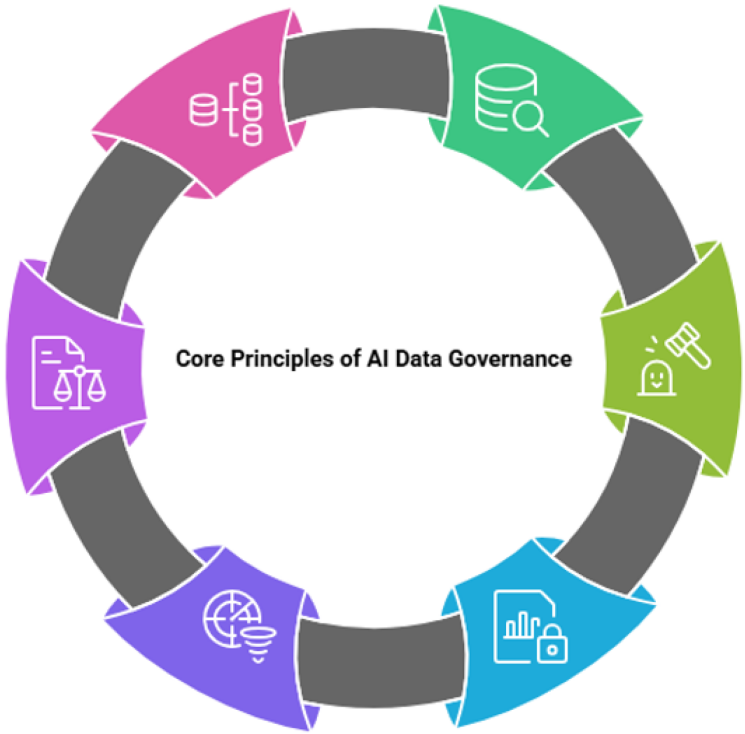
- **Model choice:** size (7B/8B/13B/70B), license, context length, tokenizer, eval profile.
- **Adaptation path:** Prompting $\rightarrow$ RAG $\rightarrow$ PEFT/QLoRA $\rightarrow$ Full FT.
- **Data & eval:** curation, contamination control, domain eval suites, human review.
- **Systems:** serving stack (vLLM/TGI/TRT-LLM), quantization, KV cache policy, autoscaling.
- **Governance:** safety gates, red-teaming, audit trails, lineage, rollback strategy.

Simple decision rule



- Dynamic or sensitive knowledge → **RAG-first**.
- Stable behavior/style or tool-use policy → **SFT/PEFT**.
- Hard latency/offline devices → **distillation** + small student model.

Data Governance & Contamination Control



1. Data Quality and Integrity

(Ensures data is accurate, reliable, fine tunning of data, data transparency)

2. Ethical AI Standards and Fairness

(Promotes fairness, trustworthy output and ethical consideration in AI systems)

3. Data Privacy and Security

(Protects sensitive data and well encrypted from unauthorized access and attacks such as adversarial, inversion, data poisoning)

4. Model Deployment and Monitoring

(Deployments of model requires safeguards, and continuous monitoring of model)

5. Regulatory and Compliance

(Ensures adherence to laws and model adapts global regulation such has GDPR, CCPA, EU AI Act)

6. Data Lineage and Traceability

(Tracks source of truth of data and monitoring on data over time for transparency)

Data Governance & Contamination Control

Curation

- Dedup (exact + near-dup via MinHash/LSH); profanity/toxicity filters; PII scrubbing.
- License hygiene: track source, permissions, and redistribution constraints.

Decontaminated evaluation

- Block train/eval overlap with fuzzy matching, URL hashing, n-gram Jaccard checks.
- Freeze eval *snapshots* with checksums; record prompts and scoring code.

Data contracts

- Schema for prompts/outputs/annotations; reviewer rubrics; inter-rater agreement tracking.

BPE (Byte-Pair Encoding)

- Deterministic subword segmentation built by iteratively merging frequent byte/pair sequences.
- Pros: compact vocab, robust to OOV via byte fallback; Cons: can fragment rare domain terms.

Unigram Language Model Tokenizer

- Probabilistic piece inventory optimized with EM; inference picks best segmentation (e.g., Viterbi).
- Pros: flexible segmentations, good for multilingual; Cons: slightly heavier build/inference.

Extending Vocabulary for Domain Tokens

- Add single tokens for units/part IDs/code symbols to reduce sequence length.
- Each new token adds an embedding row that *must be trained* (init from average/random).
- Keep tokenizer/version in the model registry; serving must use the exact same tokenizer.

Special/Control Tokens

- Reserve IDs for *system/tool/function-call* roles; exclude from normal merges/normalization.
- Document formatting contracts (e.g., how tools/calls are delimited) to prevent accidental splitting.

RoPE Scaling (NTK/YaRN)

- **RoPE**: rotary positional embeddings rotate Q, K by position-dependent angles (relational bias).
- **NTK-aware scaling**: adjusts RoPE base to extend usable context with minimal drift.
- **YaRN-style scaling**: resamples/interpolates positions to trade near-vs-far position fidelity.

Sliding-Window / Streaming Attention

- Attend only to the most recent W tokens during decode; evict or compress older KV.
- Lowers compute from $O(L^2)$ toward $O(L \cdot W)$; choose W by task retention needs.

Retrieval Windowing (32k–256k contexts)

- Schedule which retrieved spans enter the prompt at large lengths; refresh across turns.
- Use compression (extractive summaries, citations) to fit budgets without losing grounding.

Long-Context Retention Curves & Stability

- Measure loss/perplexity vs token position (and QA EM vs span depth) to detect fading.
- Exceeding trained context can cause instability/repetition; mitigate via continued pretraining or long-context tuning.

Decoder-only Transformer

- Autoregressive LM with *causal* self-attention only (no encoder/cross-attention); predicts next token given left context.

Pre-LN (Pre–LayerNorm)

- Apply LayerNorm before attention/MLP blocks; stabilizes very deep training and permits higher learning rates via better gradient flow.

SwiGLU

- Gated MLP activation: $\mathrm{SwiGLU}(x_1, x_2) = \mathrm{SiLU}(x_1) \odot x_2$ with two linear projections; higher quality/compute than GeLU at similar or reduced width.

GQA / MQA (Grouped/Multi-Query Attention)

- Share K, V across groups of heads (GQA) or a single pair for all heads (MQA); slashes KV-cache size with minor quality trade-offs.

FlashAttention-2/3

- IO-aware attention kernels that tile in SRAM and fuse ops to cut memory traffic; boosts prefill & decode throughput and reduces memory.

Activation Checkpointing

- Drop most activations during forward and recompute them in backward; trades extra compute for large memory savings.

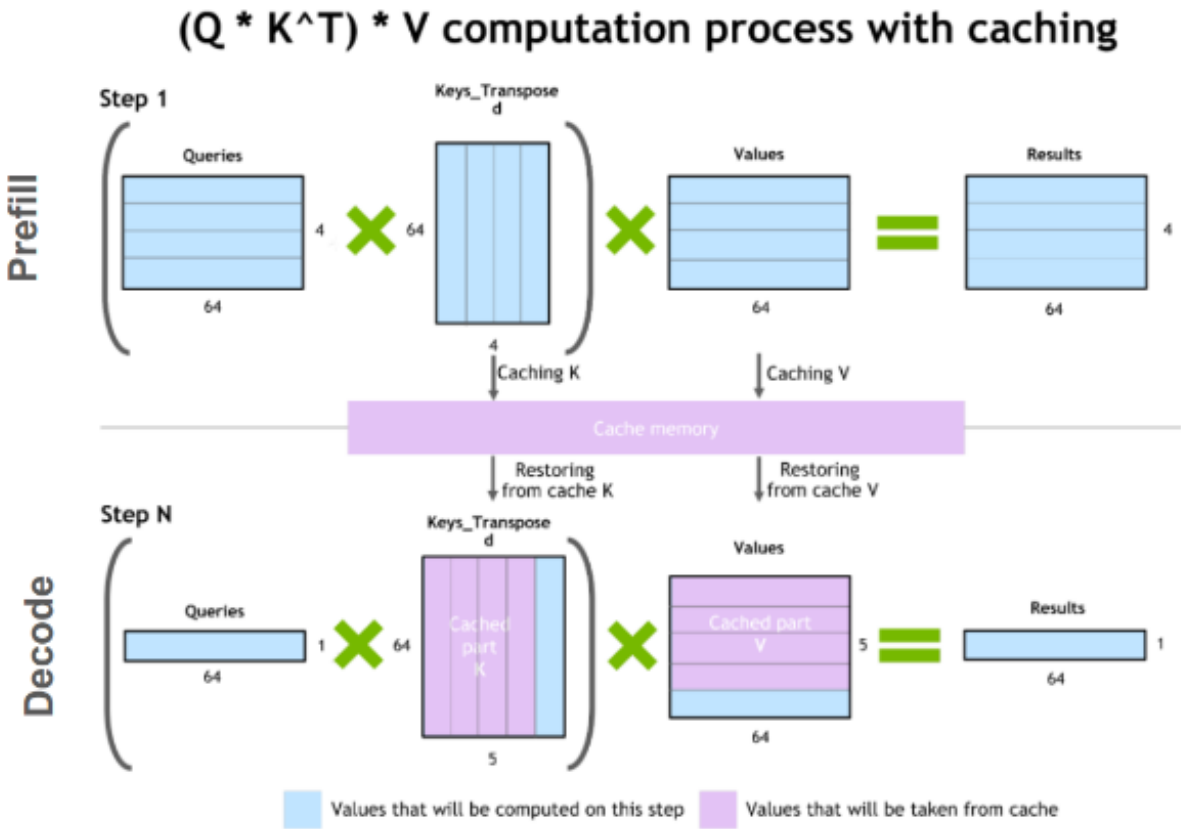
Fused Kernels

- Combine ops (e.g., bias+dropout+residual+LayerNorm) into single kernels to reduce launch overhead and memory bandwidth.

- In autoregressive decoding, the **KV cache** stores past attention *keys* (K) and *values* (V) per layer/head.
- Next step computes only the new query q_{t+1} and attends over cached $K_{1:t}$, $V_{1:t}$ (no recompute of the prefix).

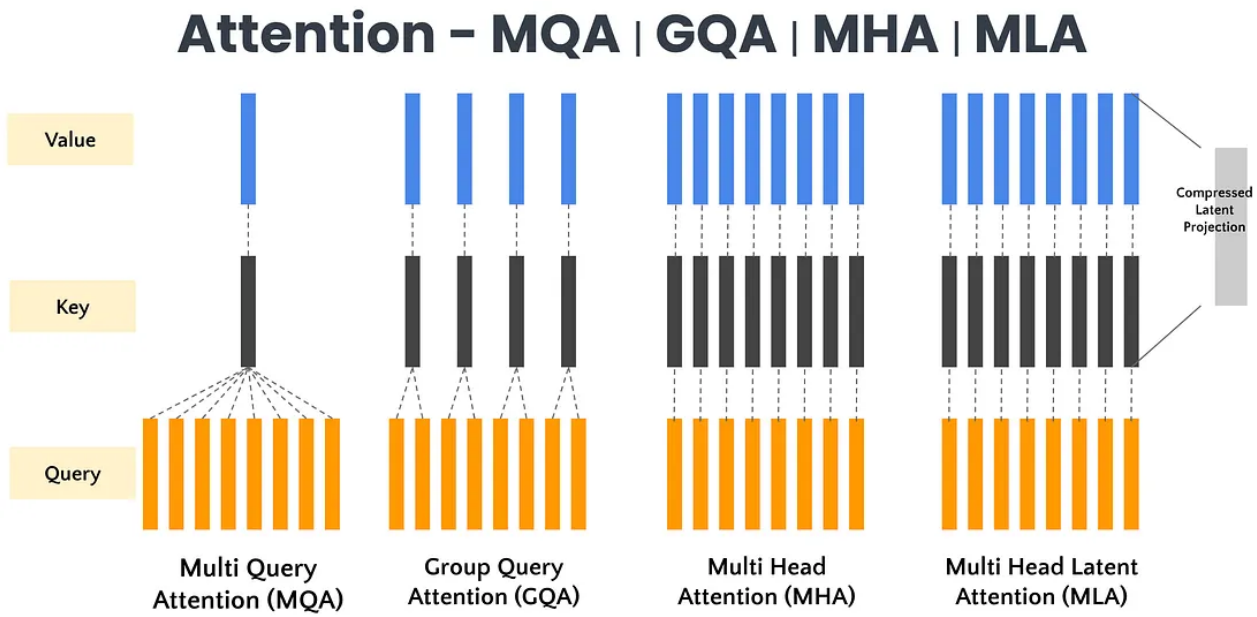
How it works

- **Prefill:** cache K, V once for the prompt of length T.
- **Decode:** append one row per new token to each layer's K, V.



GQA $\Rightarrow$ higher batch via smaller KV

- KV-cache memory $\propto$ `batch` $\times$ `layers` $\times$ `KV_heads` $\times$ `seq_len` $\times$ `head_dim`; sharing K,V reduces `KV_heads` $\Rightarrow$ more concurrent requests per GPU.



RoPE Scaling and Long-Context Quality

- Adjust RoPE base (e.g., NTK/YaRN) to extend usable context; preserves far-position alignment but can slightly alter near-position inductive bias-validate retention curves at target lengths.

Shared Input/Output Embeddings (Weight Tying)

- Tie token embedding and LM head weights to shrink parameters and improve consistency; requires identical tokenizer/vocab at train & serve time.

Tokenizer Compatibility at Serve Time

- Inference must use the exact tokenizer (merges, special tokens) used in training; any mismatch shifts tokenization, harming quality and invalidating KV-cache reuse.

Training Recipe: What Ships in Practice

Defaults

- BF16 mixed precision, grad clip 1.0, AdamW (β_1=0.9, β_2=0.95), weight decay 0.1.
- Warm-up (0.5–2% steps), cosine or linear decay; gradient accumulation + microbatching.
- ZeRO/FSDP for sharding (full FT); gradient checkpointing; FlashAttention.

Checkpointing & registry

- Save base + adapter deltas; keep optimizer states only when resuming.
- Register model, dataset snapshot, tokenizer, training code, hyperparams, eval suite.

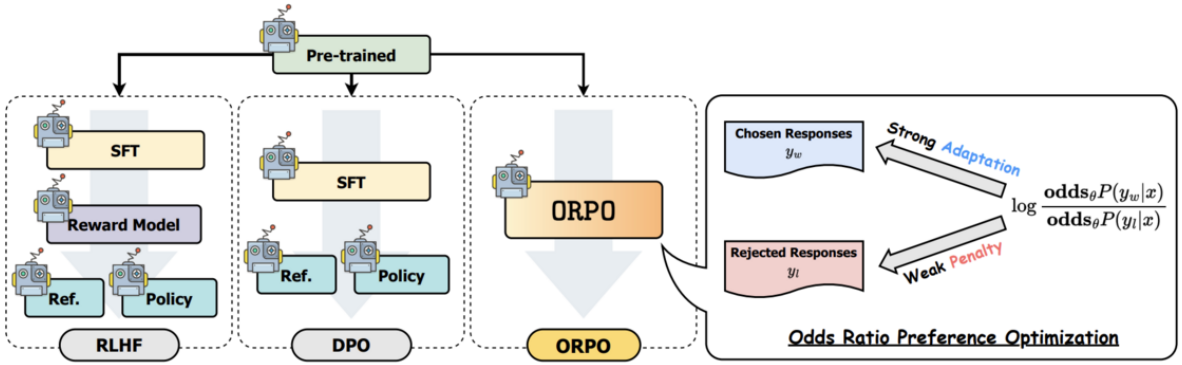


Figure 2: Comparison of model alignment techniques. ORPO aligns the language model *without a reference model* in a single-step manner by assigning a weak penalty to the rejected responses and a strong adaptation signal to the chosen responses with a simple log odds ratio term appended to the negative log-likelihood loss.

SFT (Instruction Tuning)

- High-quality instruction–response pairs; format/style control; low instability.

Preference-based Optimization

- DPO/ORPO/RRHF: use pairwise preferences without PPO; simpler infra than RLHF.
- RLHF: reward model + PPO; powerful but costlier and more unstable to tune.

Safety

- Red-team datasets; refusal policies; safety-eval gates; jailbreak rate before promotion.

Evaluation Matrix & Harnesses

Intrinsic

- Perplexity, next-token loss, long-context retention.

Task metrics

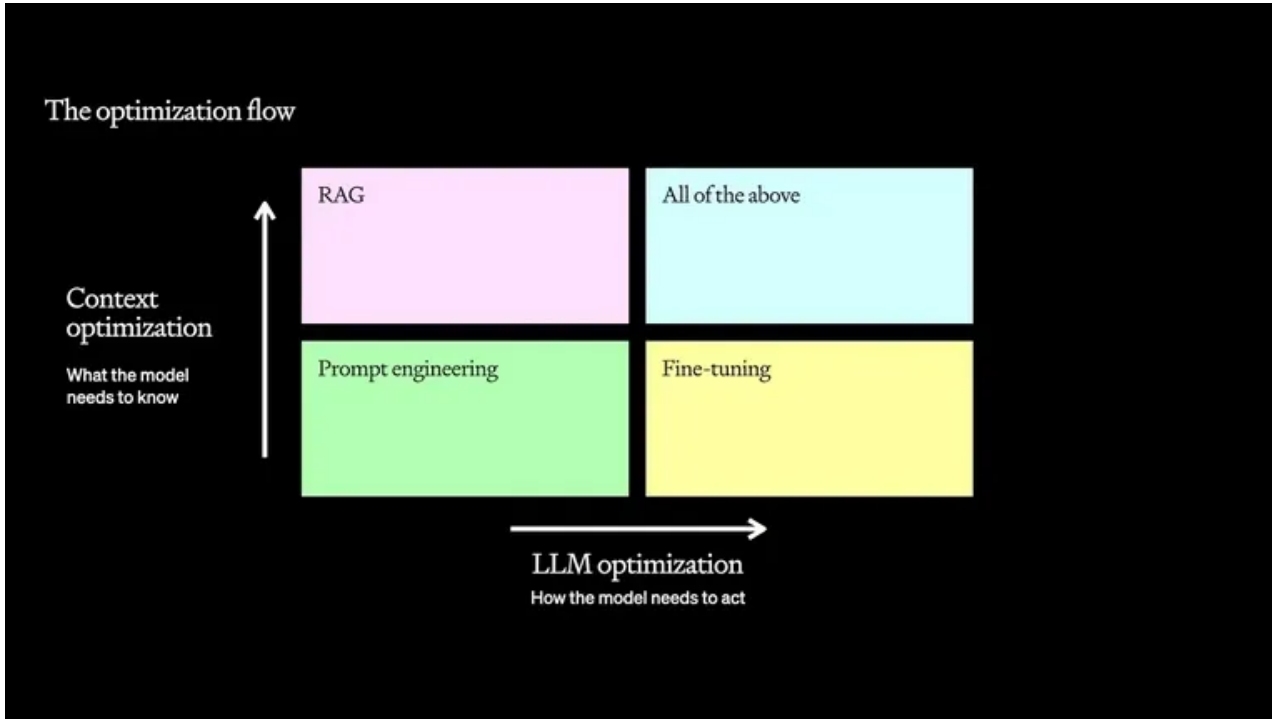
- EM/F1 (QA), BLEU/ROUGE (summarization), Pass@k (code), grounded accuracy (RAG).

Operational & safety

- Latency p95, tokens/s, cache hit rate, refusal/harmful rate, prompt-injection susceptibility.

Tooling

- Harnesses (lm-eval style), human eval with calibrated rubrics; blind review queues.



Prefer RAG when

- Knowledge changes frequently, is large/sensitive, or requires citations.

Prefer PEFT/QLoRA when

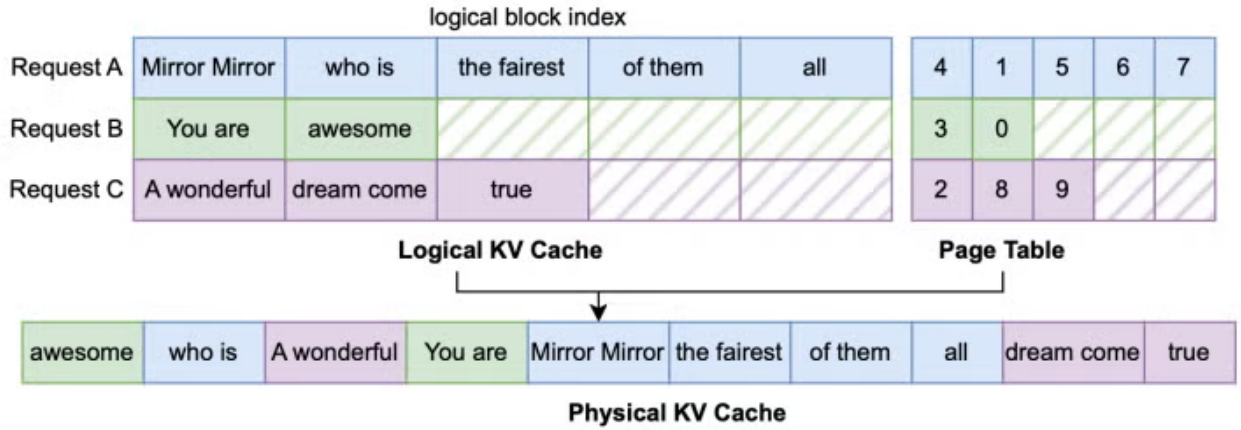
- You need behavior/style/tool-use to change consistently across tasks.

Hybrid

- PEFT for interface/format + RAG for fresh facts; optimize end-to-end KPIs (accuracy, latency, cost).

vLLM (paged attention)

- High-throughput inference server implementing *PagedAttention* to page KV-cache memory, enabling longer context and larger batches with stable latency; OpenAI-compatible API.
- PagedAttention attempts to optimize memory use by partitioning the KV cache into blocks that are accessed through a lookup table. Thus, the KV cache does not need to be stored in contiguous memory, and blocks are allocated as needed.



TGI / TEI

- **TGI** (Text Generation Inference): Production LLM server (tensor-parallel, continuous batching, KV-cache, speculative decoding) with REST/gRPC; integrates well with HF tooling.
- **TEI** (Text Embeddings Inference): Optimized embeddings service focusing on large-batch vector encoding and low-latency throughput.

KV cache policy

- Reuse of past attention *key/value* tensors across decoding steps; defines paging/eviction (e.g., per-session quotas) and prefill vs decode handling to bound memory while maximizing reuse.

Batching (dynamic/continuous)

- Aggregating concurrent requests into larger GPU batches; trades a small queueing delay for much higher tokens/s; continuous batching admits requests between steps.

Quantization (8/4-bit)

- Lower-precision weights/activations (INT8/FP8/INT4/NF4) to reduce memory and increase batch size; slight throughput gains with careful calibration and minimal accuracy impact.

GQA (Grouped Query Attention)

- Share K/V across groups of heads to shrink KV-cache size (roughly by head-grouping factor), raising throughput and lowering memory with minor quality trade-offs.

Concurrency plan

- Target number of simultaneous sessions and tokens/s capacity per replica; used for capacity planning (batch size, context limits) and cost forecasting.

p95 latency budget

- End-to-end time bound for 95% of requests; allocate budget across prefill, decode, queueing, network; tie to user SLAs and autoscaling triggers.

Autoscaling rules

- Policies that add/remove replicas based on queue depth, tokens/s, GPU utilization, or p95 targets; include cooldowns and max burst safeguards to control spend.

Prompt/KV caching

- Persist prefill computation (system + few-shot prompts) or session KV states to avoid recomputation; reduces cold-start latency and cost for repeated or multi-turn requests.

Pipeline

- Feedback → curate → (RAG/PEFT) retrain → evaluate → deploy.

MLOps glue

- Data/versioning (DVC/lakeFS), experiment tracking (W&B/MLflow), registry, prompt store.

Monitoring

- Drift, hallucination rate, jailbreak attempts, refusal patterns; human-in-the-loop review.

LLMs and Model Development

LLM Development

LLM Model Development Lifecycle (1/3)

Stage 1: Pretraining (Foundation Model Creation)

- **Data Collection:** trillions of tokens (web, code, books, papers); dedup, toxicity filtering, language balancing.
- **Objective:** causal LM loss $p_\theta(x_t|x_{<t})$; curriculum/mixing by domain.
- **Compute:** 1,000–10,000+ GPUs; data/tensor/pipeline parallel; ZeRO sharding.
- **Output:** 7B–70B checkpoints (100GB+); tokenizer + config + eval report.

LLM Model Development Lifecycle (2/3)

Stage 2: Post-Training Alignment

- **Instruction Tuning:** curated instruction–response sets (FLAN/Dolly/OpenAssistant + proprietary).
- **Preference Optimization:** DPO/ORPO/RRHF; RLHF (reward model + PPO).
- **Constitutional/Synthetic:** reduce human cost; encode constraints/policies explicitly.

Goal

- Helpful, harmless, honest; consistent style and tool-use; safe deployment-ready.

LLM Model Development Lifecycle (3/3)

Stage 3: Adaptation & Deployment

- **Adaptation:** full FT (rare); PEFT (LoRA/QLoRA/prefix/adapters); RAG for freshness.
- **Deployment:** quantization (8/4-bit), distillation for latency; batching, KV cache, tensor-parallel.
- **Monitoring:** hallucinations, drift, prompt-injection; human-in-the-loop pipelines.

Reality

- Frontier labs: Stage 1+2. Enterprises: primarily Stage 3.

Challenges in Developing with LLMs

- **Data-centric:** bias, staleness, licensing; enforce lineage and redaction policies.
- **Evaluation:** beyond accuracy—BLEU/ROUGE/EM/F1, grounded accuracy, human eval.
- **Business alignment:** map outputs to KPIs (accuracy, time-to-resolution, CSAT, cost/call).

Establishing Baselines for LLMs

- **Zero-shot** $\rightarrow$ **few-shot** $\rightarrow$ **tool-augmented** prompting ladder.
- **RAG baseline** with retrieval precision@k, citation correctness, latency budget.
- Compare **open-source checkpoints** vs hosted APIs; record cost/latency/accuracy tradeoffs.

LLMs and Model Development

LLM Fine-Tuning

Fine-Tuning LLMs: Foundations (1/4)

Definition

- Continue training a pre-trained LLM on new data to specialize without retraining from scratch.
- Optimize $\mathcal{L} = -\sum \log p_\theta(y|x)$ on domain data.

Why in LLMOps

- Efficient adaptation; enterprise privacy/compliance; continuous improvement cycles.

Fine-Tuning Paradigms (2/4)

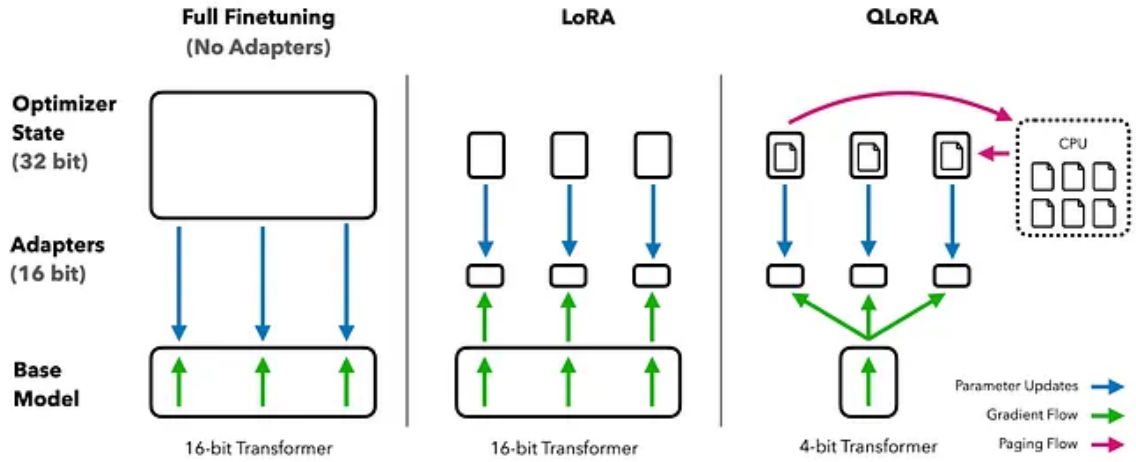


Figure 1: Different finetuning methods and their memory requirements. QLoRA improves over LoRA by quantizing the transformer model to 4-bit precision and using paged optimizers to handle memory spikes.

Full Fine-Tuning

- Update all weights; highest capacity; requires distributed training and large GPU memory.

PEFT

- Train a small parameter set: LoRA, QLoRA, prefix/adapters; 90–99% performance with 1–5% parameters.

Technical Challenges in Fine-Tuning (3/4)

Catastrophic Forgetting

- Mitigate with small LR, freeze lower layers, interleave old+new data.

Resources

- Optimizer states $\sim$2–3$\times$ model size (AdamW); use Adafactor for memory when possible.

Data

- Quality $\rightarrow$ bias/reliability; small data overfits; noisy data degrades outputs.

LoRA and QLoRA: State of the Art (4/4)

LoRA

- $\Delta W \approx AB$ with rank $r \ll d, k$; train A, B only; large reduction in trainable params.

QLoRA

- Base in 4-bit (NF4) + LoRA in FP16/FP32; $\sim$33% memory savings vs 16-bit LoRA; near-parity accuracy.

Operational

- Enables single-GPU adaptation of large models; faster/cheaper iteration cycles.

Hyperparameter Tuning in LLM Fine-Tuning (1/4)

Key Sensitivities

- **LR:** too high $\rightarrow$ divergence/forgetting; too low $\rightarrow$ underfit. Typical 10^{-5}–10^{-4} (full FT); slightly higher for LoRA.
- **Warm-up & schedule:** 50–500 warm-up steps; cosine/linear decay.

Optimizer

- AdamW default; Adafactor (memory-efficient); Lion/Sophia emerging.

Hyperparameter Tuning in LLM Fine-Tuning (2/4)

Batch & Sequence

- Larger batch $\rightarrow$ smoother gradients; attention cost is $O(L^2)$.
- Use gradient accumulation to simulate large batch under memory limits.

Regularization

- Dropout, weight decay; label smoothing for classification heads; LoRA rank acts as implicit regularizer.

Hyperparameter Tuning in LLM Fine-Tuning (3/4)

LoRA/QLoRA-specific

- Rank r (4–64) vs capacity; $\alpha \approx 2r$; adapter dropout 0.05–0.1 for small data.

Quantization (QLoRA)

- NF4 + double quantization; consider CPU offload; watch optimizer state memory.

Hyperparameter Tuning in LLM Fine-Tuning (4/4)

Search strategies

- Random & Bayesian (Optuna/Ray Tune); population-based (adapt LR/batch on the fly).

Ops best practices

- Low-fidelity trials, early stopping, MLflow/W&B logging; track cost/run and p95 latency.

PEFT Target Modules & Recipes

Attention targets

- Apply LoRA to `q_proj`, `k_proj`, `v_proj`, `o_proj`; start with Q/V or Q/K/V for stability.

MLP targets

- `gate_proj`, `up_proj`, `down_proj` for style/format shifts; increase r moderately.

Heuristics

- Small data: lower r, higher dropout; Large data: higher r, lower dropout.
- Freeze lower layers first; unfreeze top-k if needed.

Tuning by Data Regime

Tiny (<50k tokens)

- LoRA $r \in [4, 8]$, dropout 0.1, LR 2×10^{-5}; few epochs; heavy regularization.

Mid (50k–5M tokens)

- LoRA $r \in [8, 32]$, LR $1\text{–}3\times10^{-5}$; cosine schedule; validation every N steps.

Large (>5M tokens)

- Consider full FT or QLoRA with lower dropout; careful curriculum/mixing; checkpoint every epoch.

Cost Modeling & Capacity Planning

Before training

- Estimate tokens $\times$ epochs, tokens/s/GPU, optimizer overhead, checkpoint cadence.

During training

- Track loss vs tokens, cost/run (GPU-hrs), utilization, throughput; abort criteria for bad runs.

Serving

- Model size, quant level, KV cache per user, target QPS/p95, autoscaling policy.

LLMs and Model Development

Case Study: Tuning QLoRA on LLaMA-7B

Scenario

- Goal: adapt LLaMA-7B to a domain dataset (e.g., legal documents).
- Constraint: single 24GB GPU.
- Method: QLoRA (4-bit NF4 + low-rank adapters).

Key hyperparameters

- LR sweep $[1\times10^{-5}, 5\times10^{-5}]$; linear decay; 100 warm-up steps.
- LoRA rank $r \in \{8, 16, 32\}$; α=2r; adapter dropout 0.05.
- Effective batch 64 via grad accumulation; NF4 + double quantization.

Outcomes

- Peak memory $\sim$18GB; validation perplexity -45% (3 epochs).
- $< 5\%$ drop on general tasks; $\sim$30 GPU-hrs vs $>$500 for full FT.

Full Fine-Tuning vs. LoRA vs. QLoRA

	Full Fine-Tuning	**LoRA**	**QLoRA**
Parameters Updated	100% (billions)	∼0.1–1%	∼0.1–1%
GPU Memory (7B model)	∼80GB+	∼24GB	∼18GB
GPU Hours (7B model)	500+	30–50	30–60
Accuracy vs. Full FT	100%	95–99%	95–99%
Hardware Needs	Multi-GPU	Single 24GB GPU	Single 24GB GPU
Deployment	Full weights	Base + adapters	4-bit base + adapters
Use Case	Frontier labs	Domain adaptation	Large models, limited HW

Takeaway: PEFT (LoRA/QLoRA) democratizes adaptation of billion-parameter models on commodity GPUs.

Labs for This Week

Objective

Briefly describe the learning goal for this week's lab(s).

Lab Activities:

- Lab 9: [LLMs] - [Fine-tuning]

Submission Deadline: [Before the next class]

- Assignment 9: [LLMs] - [Fine-tune a model of your choice]

Reading Materials

This Week's Theme

Topic focus: [People + AI Guidebook - Data Collection + Evaluation.pdf]
You should use the worksheet related to this pdf to your project and submit it when its requested.

Required Readings:

- [On the Reliable Detection of Concept Drift from Streaming Unlabeled Data]

Be prepared to discuss highlights and open questions in class.

Sources

 DeepLearning.AI

 The People + AI Guidebook