SNIA | Data, Storage & Networking

Post Training and Fine Tuning for LLMs in the Enterprise

Live Webinar
April 23, 2026
10:00 am PT / 1:00 pm ET

Today's Presenters

Rohan Mehta
Systems Performance Architect
Micron Technology
Moderator

Erik Smith
Distinguished Engineer
Dell Technologies

Ulf Hanebutte
Distinguished Engineer
Marvell

The SNIA Community

200
industry leading organizations

2,000
active contributing members

50,000
IT end users & storage pros worldwide

SNIA | DSN A SNIA Community

SNIA | Data, Storage & Networking

What We Do

Drive the awareness and adoption of a broad set of technologies, including:

- Storage Protocols (Block, File, Object)
- Traditional and software-defined storage
- Disaggregated, virtualized and hyperconverged
- AI, including storage and networking considerations
- Edge implementation opportunities and factors
- Storage and networking security
- Acceleration and offloads
- Programming frameworks
- Sustainability

How We Do It

By delivering:

- Expert webinars and podcasts
- White papers
- Articles in trade journals
- Blogs
- Social Media
- Presentations at industry events

Logistics

- The slides are available under the attachments tab at the bottom of your console.
- Questions are welcome!
- Please rate the session and provide feedback!
- Want more sessions like this or other topics, let us know!
  - JOIN US! We meet on Thursday mornings at 11:00 AM eastern.
  - Email dsn-chair@snia.com if you have questions.

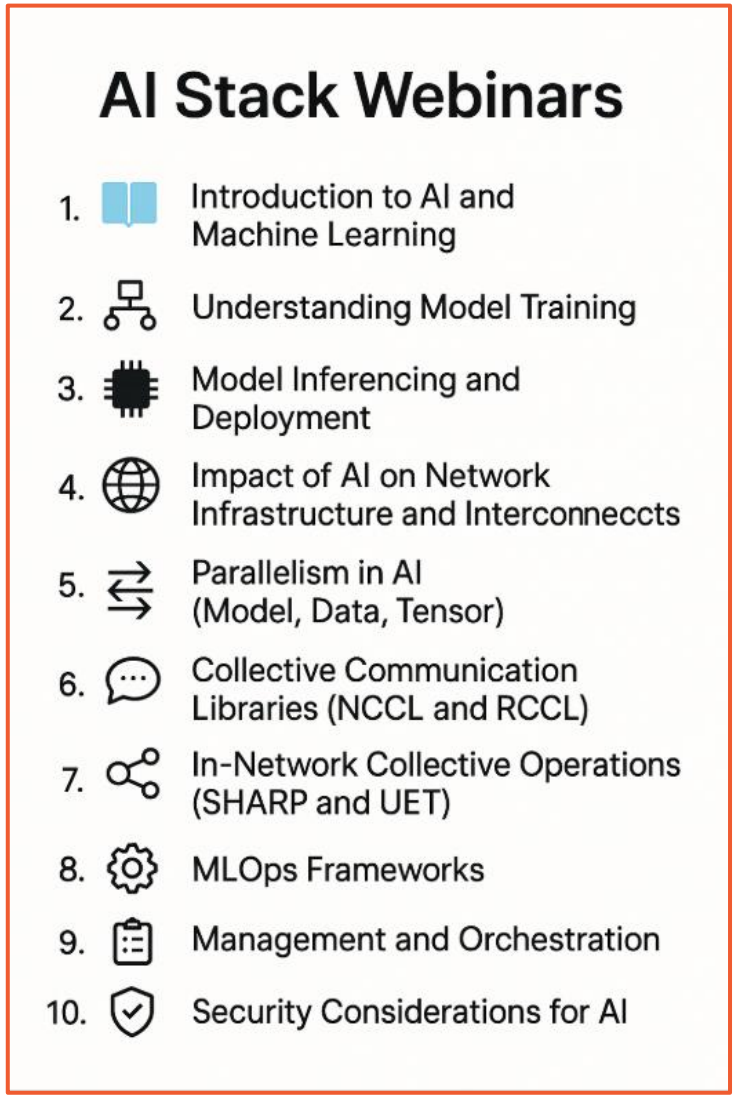
SNIA Legal Notice

- The material contained in this presentation is copyrighted by SNIA unless otherwise noted.
- Member companies and individual members may use this material in presentations and literature under the following conditions:
  - Any slide or slides used must be reproduced in their entirety without modification
  - SNIA must be acknowledged as the source of any material used in the body of any document containing material from these presentations.
- This presentation is a project of SNIA.
- Neither the author nor the presenter is an attorney and nothing in this presentation is intended to be, or should be construed as legal advice or an opinion of counsel. If you need legal advice or a legal opinion please contact your attorney.
- The information presented herein represents the author's personal opinion and current understanding of the relevant issues involved. The author, the presenter, and the SNIA do not assume any responsibility or liability for damages arising out of any reliance on or use of this information.

  NO WARRANTIES, EXPRESS OR IMPLIED. USE AT YOUR OWN RISK.

The “AI Stack” Webinar Series

- Building a Strong Foundation for All Experience Levels:
  - Starting from the basics
  - Building steps-by-step
  - Connecting theory to practice
  - Demonstrations
  - Preparing for real-world challenges



Agenda

1

PART 1

The Foundations

The quadrant map, training pipeline, prompt engineering, RAG, SFT, RLHF, LoRA • Live Demo: Post-Training Explorer • Traditional ML surprise, live app walkthrough, debugging stories

2

PART 2

Evolution of Post-Training

PPO, GRPO, DPO, RLVR, hybrid approaches

3

PART 3

Infrastructure Implications

Checkpointing, RL storage implications, quantization

4

PART 4

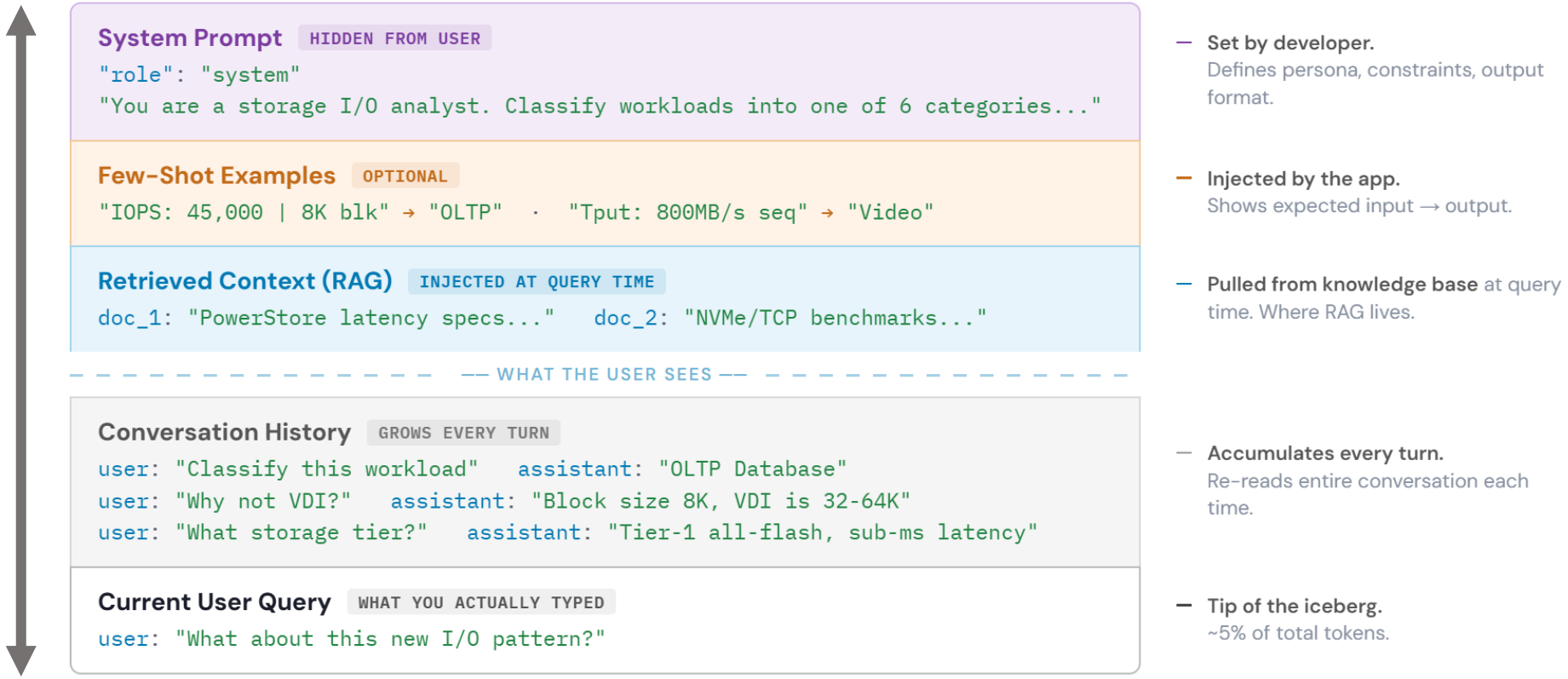
Wrap up & Q&A

SNIA | DSN
A SNIA Community

Part 1: The Foundations

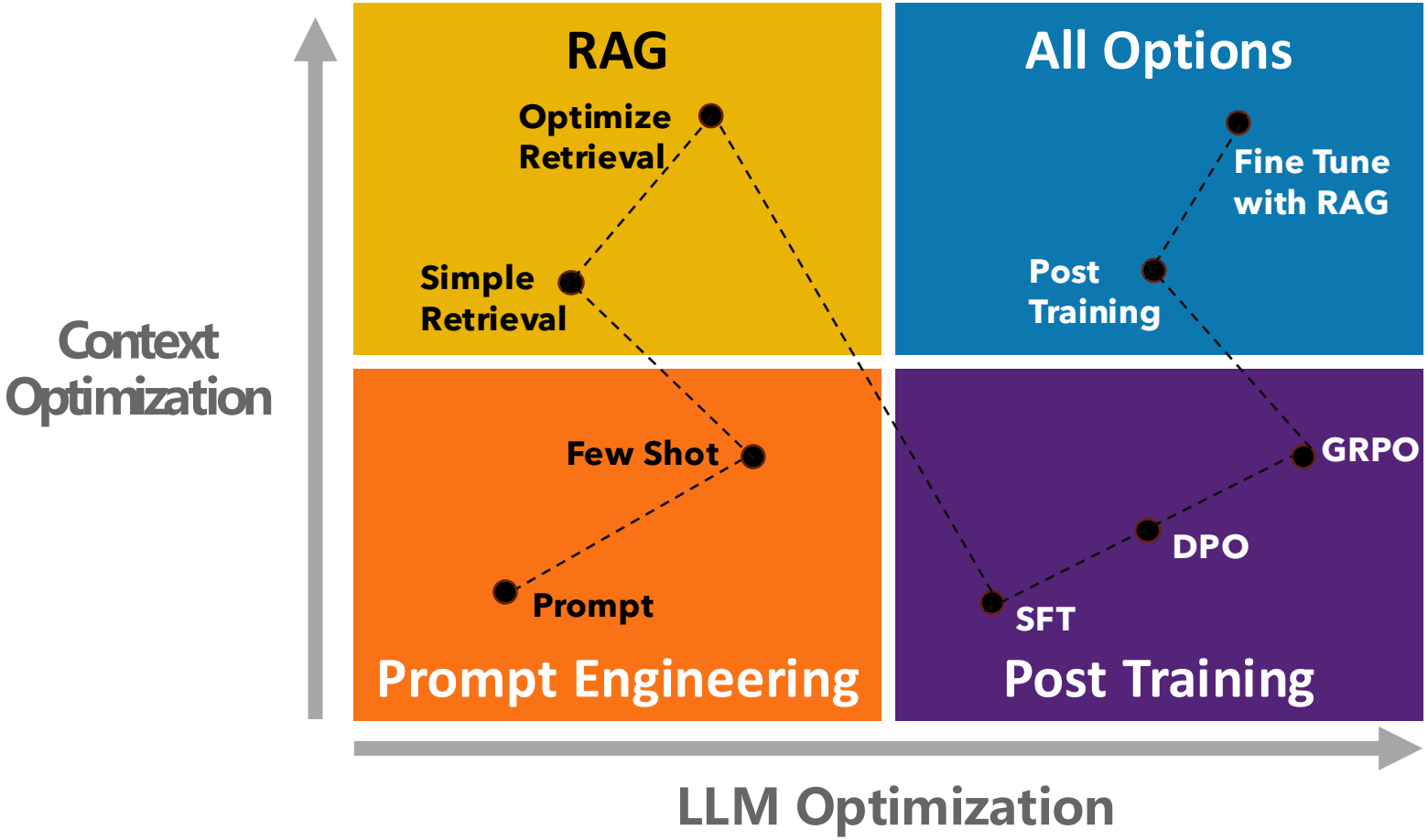
What is a prompt?

"Context"
The full prompt sent to the LLM on each turn.



SNIA | DSN A SNIA Community

The Quadrant Map



The original four quadrant diagram pictured above can be found here:
https://developers.openai.com/api/docs/guides/optimizing-llm-accuracy

The AI Model Pipeline

You don't need to train the model.
You choose one and shape it to your needs.

Post-Training: Where You Have Agency

Techniques that change how the model behaves

Different objectives, different algorithms, different infrastructure

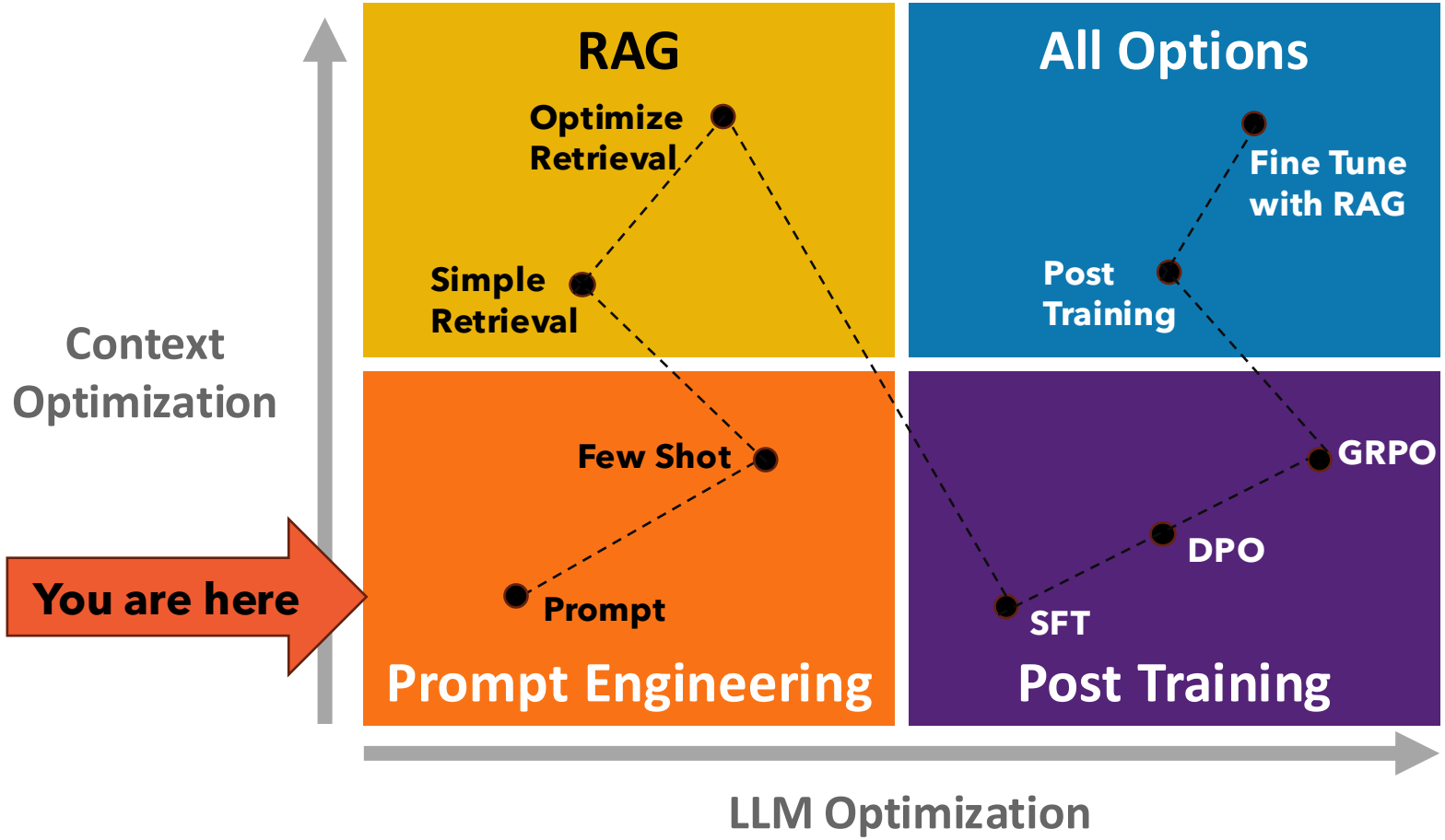
Open-weights vs closed-model is a first-order decision

- **Open-weights:** you have access to model weights, can fine-tune directly
- **Closed-model (API):** limited to prompting, RAG, and API-level fine-tuning

Post-training gives you agency over model behavior

- Domain adaptation, output formatting, behavioral consistency
- Alignment, safety, reasoning capabilities

The Quadrant Map



The original four quadrant diagram pictured above can be found here:
https://developers.openai.com/api/docs/guides/optimizing-llm-accuracy

Prompt Engineering

The simplest approach

No training required

- Basic prompts: natural language instructions to the model
- Few-shot learning: provide examples in the prompt
- Fast iteration - change the prompt, get different results

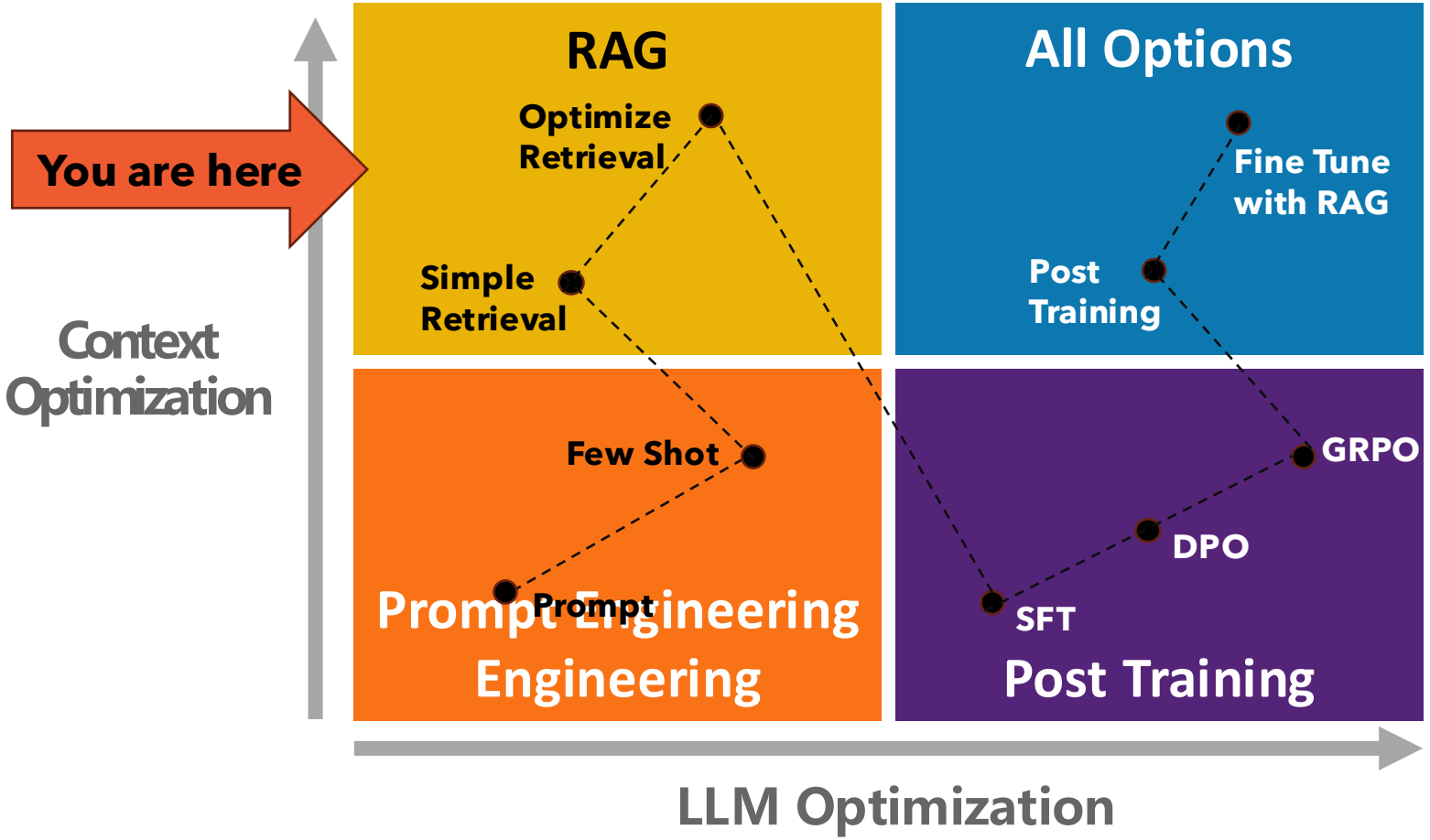
The ceiling

Where prompting breaks down

- Context window is finite - examples compete for space
- No permanent learning - every session starts from scratch
- Behavioral consistency is not guaranteed at scale

Takeaway: When reliability matters more than getting the job done quickly, you need to move up or right on the map.

The Quadrant Map



The original four quadrant diagram pictured above can be found here:
https://developers.openai.com/api/docs/guides/optimizing-llm-accuracy

RAG & Context Engineering

Augmenting at inference time

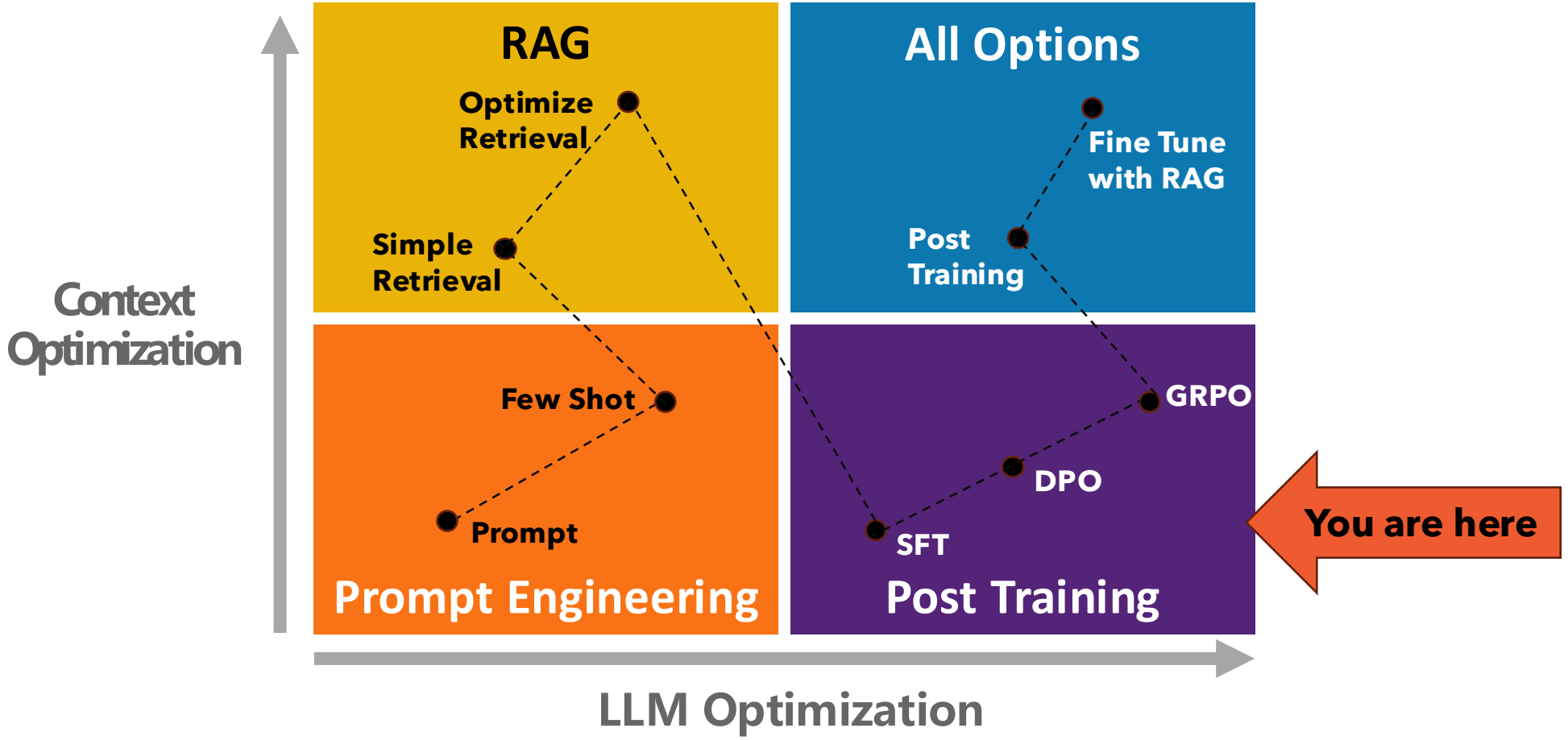
- Retrieve relevant context from external knowledge bases
- Vector search, semantic similarity, document retrieval
- New information available immediately - no retraining required

The context window problem

- Model responses degrade as context grows
- "Lost in the middle" - information in the middle is often ignored
- More context ≠ better answers

Takeaway: RAG keeps the model current without retraining.

The Quadrant Map



The original four quadrant diagram pictured above can be found here:
https://developers.openai.com/api/docs/guides/optimizing-llm-accuracy

Supervised Fine-Tuning (SFT)

A note on terminology

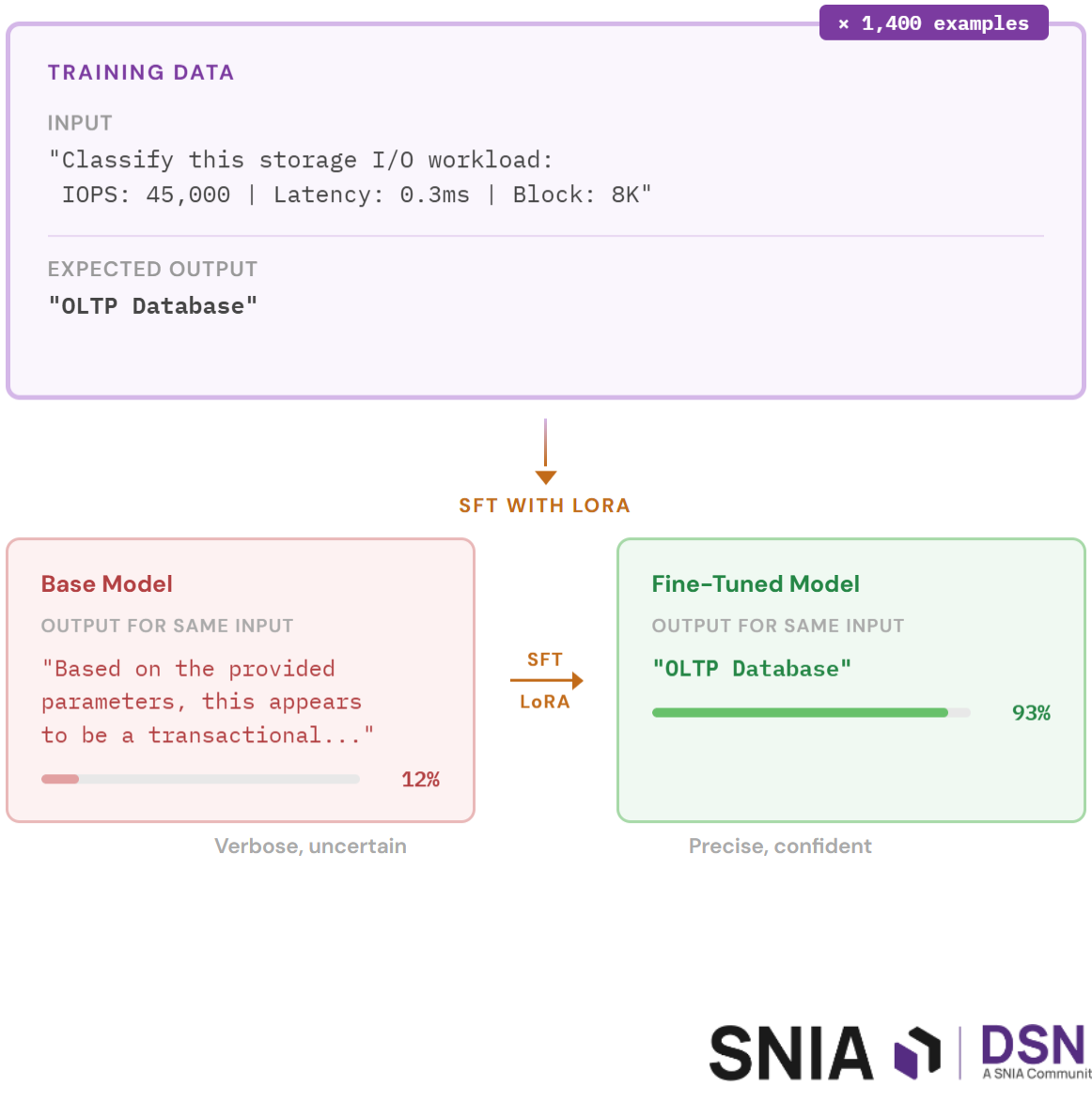
- “Fine-tuning” is often used loosely to mean any post-training adaptation
- SFT specifically means supervised learning on labeled input/output pairs

When you see this input, produce this output

- Instruction tuning — why modern models follow instructions at all
- This is the technique that turned base models into assistants

When to use SFT

- Domain adaptation (teach the model your terminology)
- Output formatting (structured JSON, specific templates)
- Behavioral consistency (reliable tone, style, approach)



SNIA | DSN A SNIA Community

RLHF: Learning from Human Preferences

The technique that made ChatGPT possible

- Reward model trained from human rankings of outputs
- Humans rank outputs A > B > C, model learns the preference pattern
- PPO (Proximal Policy Optimization) to optimize against the reward

The cost problem

- Requires large-scale human annotation campaigns
- PPO is complex and compute-intensive
- Difficult to scale and reproduce

RLAIF — The practical alternative

- Use a strong model to generate preferences instead of humans
- Same framework, dramatically lower cost
- Quality of the feedback model becomes the critical dependency

LoRA & Parameter-Efficient Fine-Tuning

The problem

- Fine-tuning all parameters requires massive GPU memory
- Full fine-tuning of 7B model: ~56 GB VRAM
- That's a $10,000+ GPU required to train

LoRA: freeze + adapt

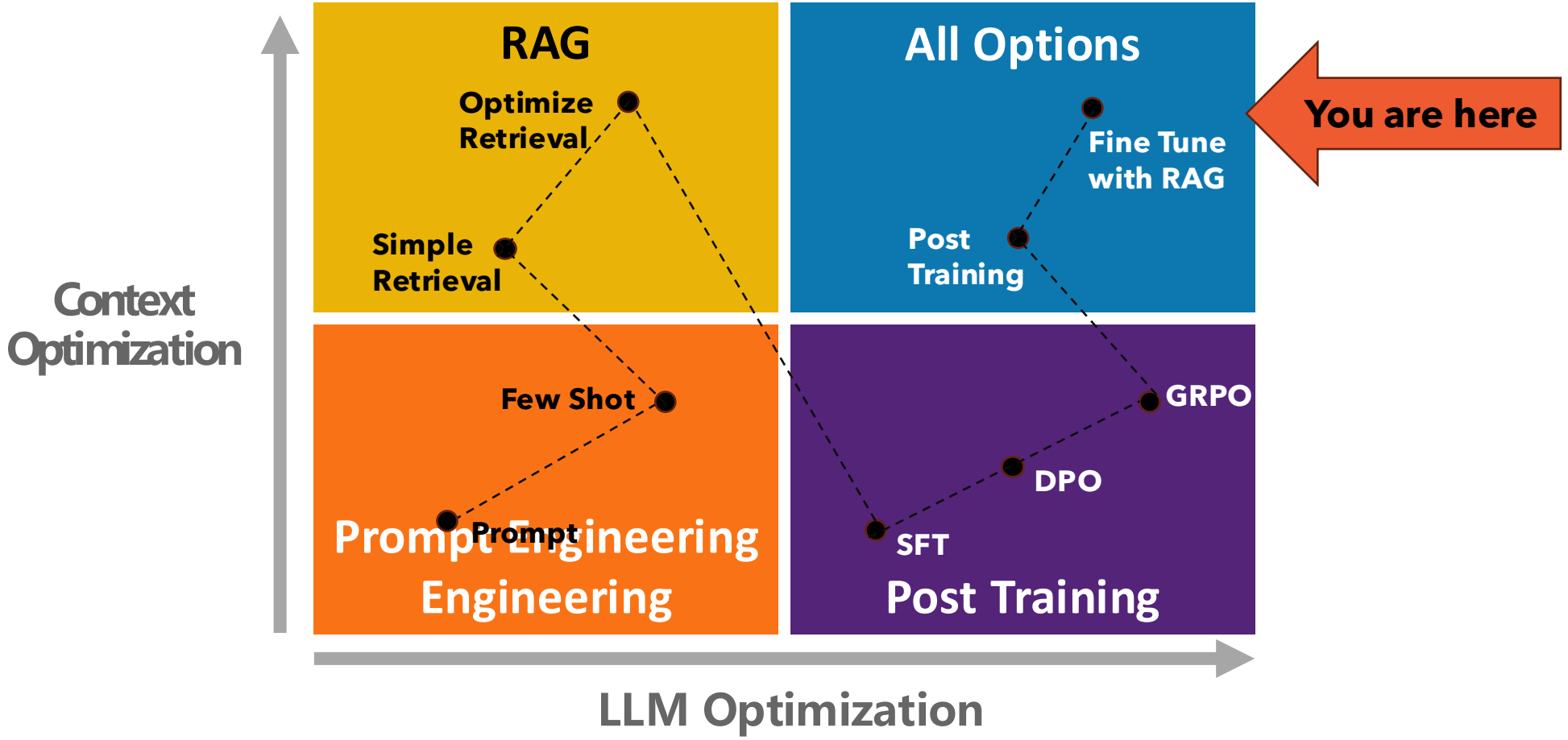
- Freeze original weights, train small adapters
- Original matrix (frozen) + low-rank adapter (trainable)
- Adapter is typically <1% of original parameters
- LoRA fine-tuning of 7B model: 14 GB VRAM

QLoRA: 4-bit + LoRA

- 4-bit quantized base + LoRA adapters
- 70B+ parameter models on a single consumer GPU
- The technique that democratized fine-tuning (2023–2024)

Note: memory savings are for training only; inference memory unchanged.

The Quadrant Map



The original four quadrant diagram pictured above can be found here:
https://developers.openai.com/api/docs/guides/optimizing-llm-accuracy

Combining Techniques

The progressive improvement path

1

Prompt engineering

Hours

Validate the use case

2

Add RAG

Days

Add your data (knowledge augmentation)

3

Fine-tune

Weeks

Change the behavior (domain adaptation)

4

Add RL

Weeks–months

Refine the edges (behavioral optimization)

Each layer compounds on the others

- Fine-tuned model + RAG outperforms either alone
- Prompt-engineering skills transfer — you still write prompts for fine-tuned models
- You don't abandon earlier techniques, you build on them

This is where enterprise teams land.

The Memory Layer

Context compaction — summarize and compress retrieved context.

Memory abstraction

Different memory types for different needs

- Short-term session memory (conversation history)
- Persistent memory (facts that survive across sessions)

Where storage fits

- Vector databases for embedding search
- Embedding indexes for similarity matching
- Knowledge graphs for structured relationships

Storage implications

For agentic flows

- Long-running agent sessions create persistent I/O patterns
- Memory retrieval latency directly impacts response time

Let’s See This in Practice
Live Demo: Post-Training Explorer

DEMO

Post-Training Explorer

Interactive guide to post-training techniques
for large language models

https://provandal.github.io/post-training-explorer/

Demo: Key Takeaway

Right tool for the right job

Structured numeric data → Traditional ML

- Random Forest / XGBoost: 97% accuracy, 0.3 seconds, CPU only
- Faster, smaller, more interpretable, more accurate

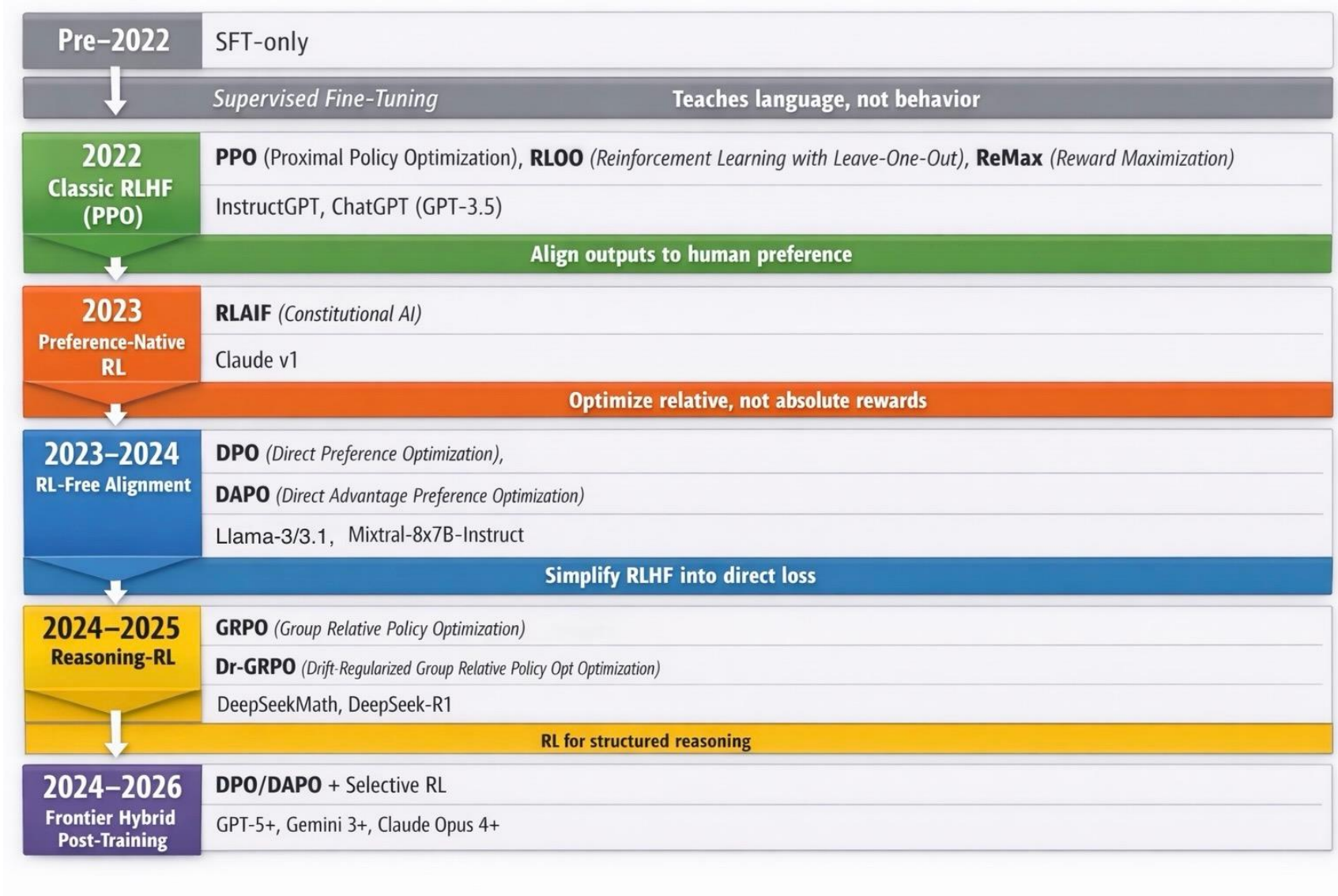
Unstructured text → LLMs

- Error logs, incident reports, support tickets
- LLMs understand context that bag-of-words models can't
- SFT gives 10–15% accuracy improvement over traditional approaches

The barrier is knowledge, not hardware

Part 2: Evolution of Post-Training

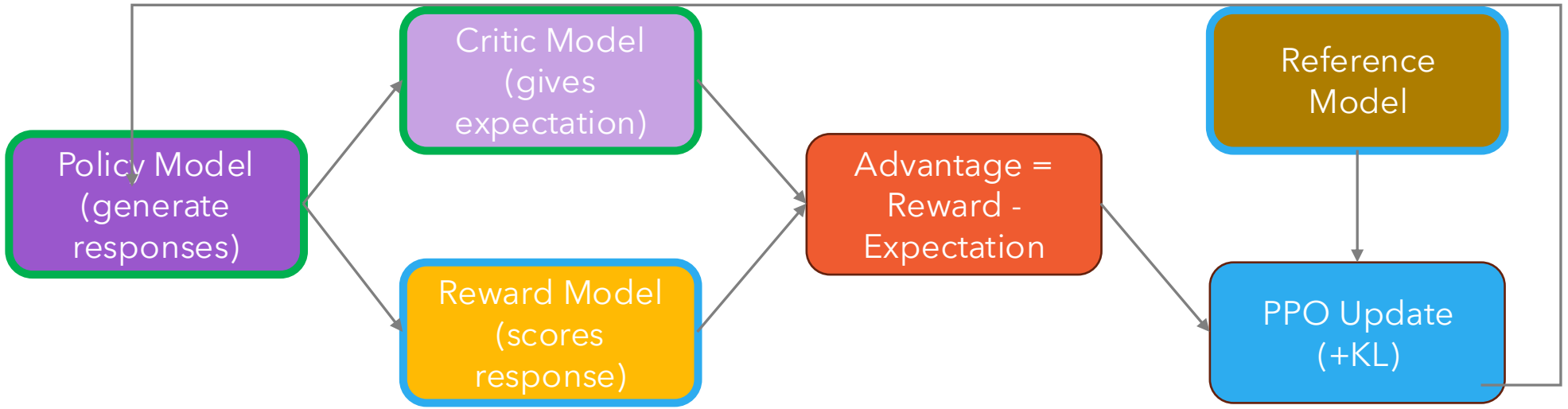
What we cover here



SNIA | DSN A SNIA Community

Proximal Policy Optimization (PPO) for LLM Training

- RL algorithm to fine-tune LLMs after SFT using scalar rewards
- Goal: Increase reward-scored responses while maintaining fluency
- Why used: Stable, alignment, safety tuning, clearer, friendlier outputs
- Challenge: Multi-model VRAM overhead (Policy, Reference, Reward, Critic)



For large scale reasoning: Methods with RL loop (1/2)

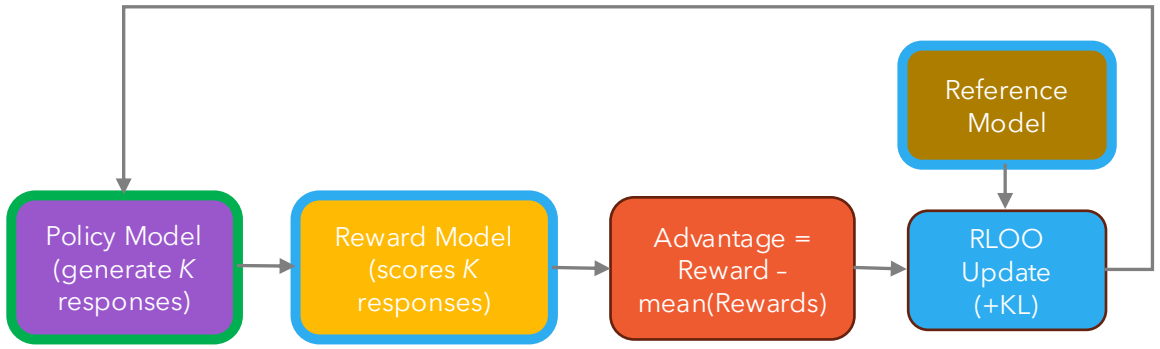
RLOO

REINFORCE Leave-One-Out

Core idea: Compute advantage by comparing each sample's reward to the mean reward of the other samples

Benefit: Lower variance than vanilla REINFORCE, simpler than PPO (no critic model)

Use case: Verifier-based or binary-reward tasks (math, code, reasoning)



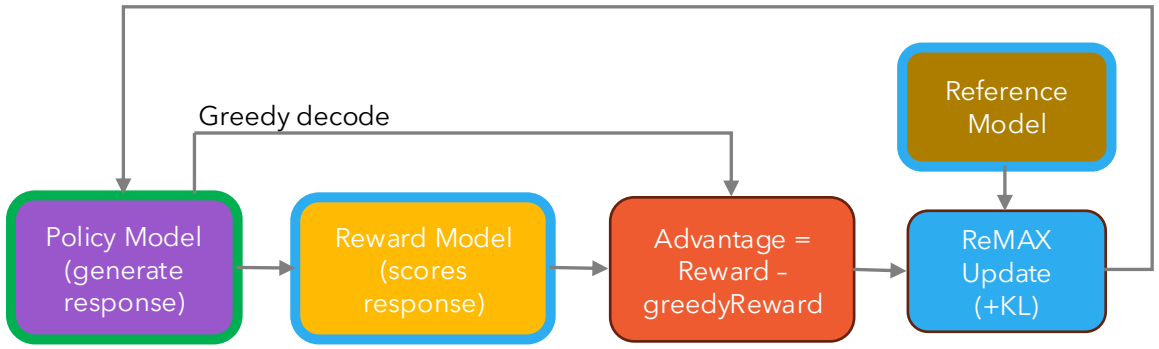
ReMax

Reward-Maximization

Core idea: Convert rewards into weights on log-likelihood, turning RL into weighted LM training

Benefit: Simpler and more stable than PPO (no advantage estimation – just calculation, no critic)

Use case: RLHF or verifier-based tuning where rewards are reliable and low-variance



SNIA DSN A SNIA Community

For large scale reasoning: Methods with RL loop (2/2)

GRPO

Group Relative Policy Optimization

Core idea: Computes advantages purely from relative rewards within a group of responses to the same prompt

No critic network — 50% compute reduction vs PPO

DeepSeek R1 (January 2025) introduced GRPO

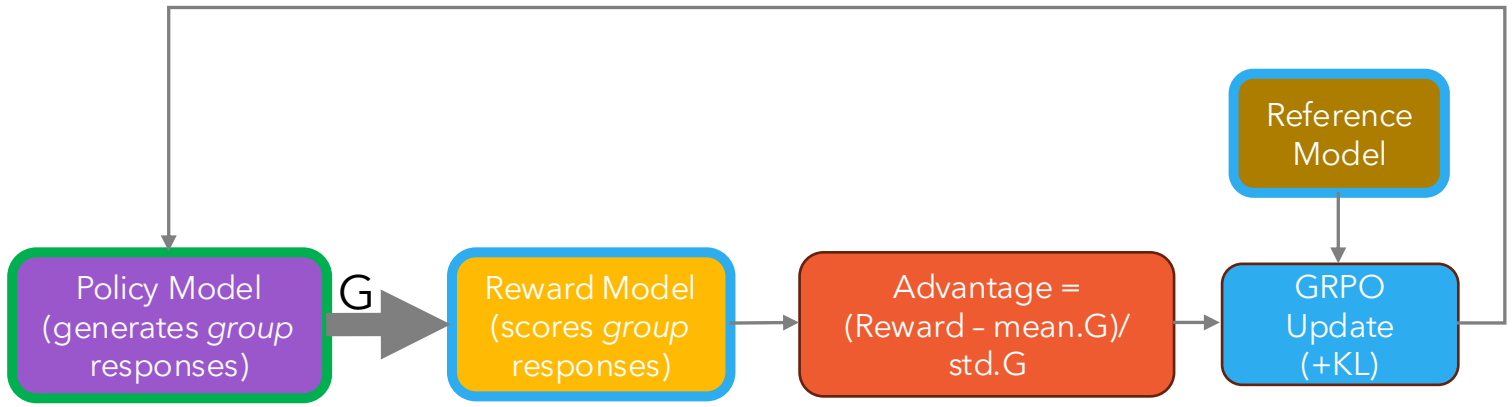
Dr-GRPO

Decoupled Regularized GRPO

Core idea: Optimize policy using group-relative advantages without absolute rewards

Adds decoupled regularization (developed in academia after analysis of GRPO)

Benefit: Improves stability and credit assignment for reasoning-heavy tasks



SNIA DSN A SNIA Community

For general purpose alignment: Methods without RL loop

DPO

Direct Preference Optimization

Skip the reward model entirely — learn directly from preferences

Simpler to implement, more stable training, compute-efficient alternative to PPO

Input: Pairs of responses (preferred vs. rejected) for the same prompt

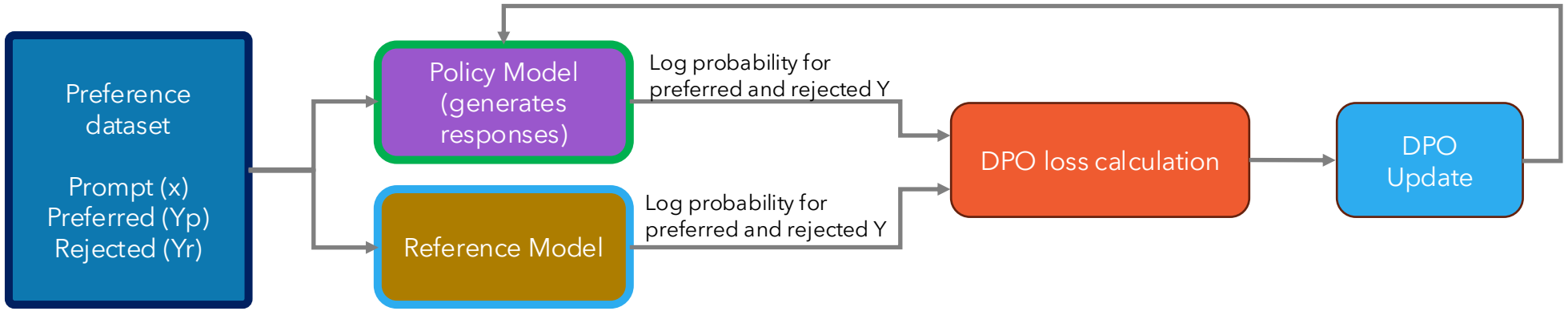
DAPO

Direct Advantage Policy Optimization

Core idea: Apply advantage-weighted log-likelihood directly within a DPO-style objective

Simplification: No explicit trust-region or clipping; relies on KL regularization for stability

Use case: RLHF or verifier-based training when advantages are well-behaved and low-variance



RLVR: Reinforcement Learning with Verifiable Rewards

Binary rewards for correctness	Math and code answers are scored 1 (correct) or 0 (incorrect) — no human preference labels needed
Multi-path exploration	Model generates 16–64 candidate solutions, scores them, and reinforces the correct ones
Learning signal	Correct solutions receive positive reward; incorrect paths receive none
Reasoning emerges via selection pressure	Training shifts from next-token prediction to discovering effective reasoning via trial and error
Best-fit domains	Most effective where answers are verifiable — mathematics, code generation, logic puzzles

Hybrid Approach: Knowledge Distillation + RL (KDRL)

Combine teacher–student learning with RL optimization

Most enterprise-relevant hybrid approach today

Example: Support agent with tools

KD: student imitates best-practice resolutions from teacher (tone, steps, tool usage)

RL: reward for “issue resolved”, penalty for policy violations, reward for correct categorization, bonus for successful tool calls, penalty for long responses

Outcome: cheaper model with higher resolution rate and consistent compliance

What's Coming

Process reward models

Reward each step of reasoning, not just the final answer

Multi-Agent RL (MARL)

Multiple models training together, negotiating, competing

- Examples: Self-play (AlphaStar, OpenAI Five); multi-robot coordination

RLVR expanding beyond math and code

Verifiable rewards for science, engineering, legal reasoning

On-device post-training

Fine-tuning on edge devices, personalization without the cloud

SNIA | DSN
A SNIA Community

Part 3: Infrastructure Implications

What we cover here

It is not just GPU compute that matters

CPU

Orchestration, data prep, and evaluation all run off-GPU

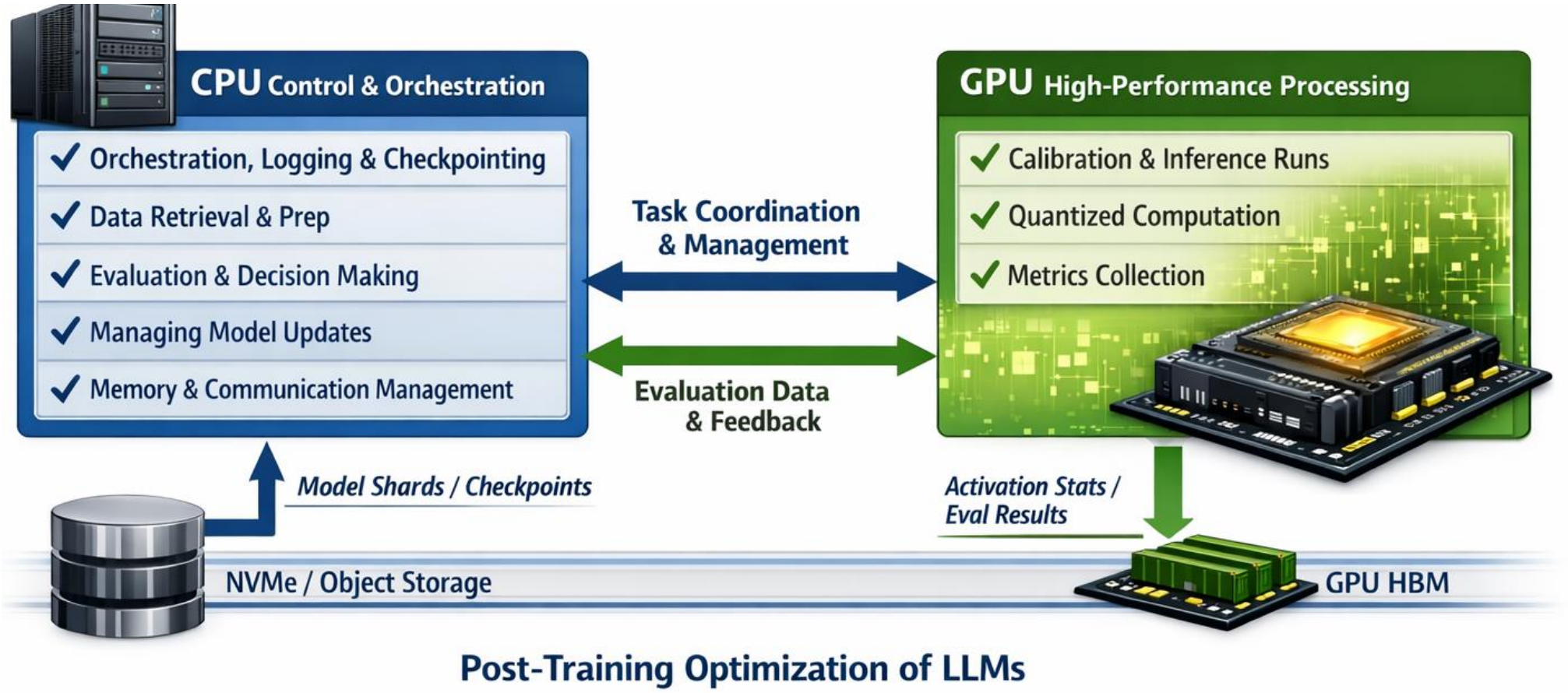
Quantization

Reduce storage and compute needs without retraining

Memory & Storage

Memory hierarchies and storage throughput shape real-world training

CPU’s role in the post training and fine tuning



Quantization: Reduces numerical precision

Quantization is a way to reduces model size and data movement

FP32 → FP16 → INT8 → INT4

each step halves storage

Scaling factors are needed when going to Integer representations

Smaller ≠ faster without all three layers aligned

A 4-bit model on hardware that doesn't natively support 4-bit = slower

Three-layer alignment required

1. **Hardware format support** (GPU native data types)
2. **Software stack end-to-end** (frameworks, kernels, runtimes)
3. **Model accuracy under quantization** (quality degradation)

Runtime implications

Inference speed depends on memory bandwidth, not just compute

Quantization is primarily a memory optimization, especially if a hardware does not support it.

Microscaling formats (MX) provides block wise scaling

Checkpointing & Storage

Long-running RL loops

Need reliable, low-latency storage

- Training runs of hours to days with frequent saves

Checkpoint recovery

Difference between failed and resumable runs

- RL training instability means checkpoints are safety nets
- Rolling checkpoints with configurable retention

Post-training = storage-intensive

More than inference

- Model weights + optimizer states + gradients + trajectory data
- A 7B parameter model: ~14 GB weights, ~56 GB total training state

The RL Storage Tax

RL training generates **10–100×** more I/O than SFT

Storage implications are significant

- Trajectory samples: storing rollouts for RL training
- Frequent checkpointing: RL is unstable, need rollback capability
- Reference models: kept in memory alongside the training model

Why the 10–100× multiplier

- Multiple candidate generations per training step
- Checkpoint frequency measured in minutes, not epochs

Part 4: Wrap up

Key Takeaways

1. **You don't train models, you shape them.** Post-training is your domain.
2. **Right tool for the right job:** structured data → ML, unstructured → LLMs.
3. **LoRA/QLoRA** made fine-tuning accessible on consumer hardware.
4. **Reasoning revolution** (GRPO/RLVR) is changing what models can do.
5. **The barrier to entry is knowledge, not hardware.** Start with the demos.
6. **Post-training storage I/O is 10–100x more intensive than SFT;** checkpointing and storage throughput are make-or-break for RL workloads.

Resources

- Post-Training Explorer (interactive app + guided tour)
  - https://provandal.github.io/post-training-explorer/
- Google Colab notebooks (SFT, DPO, GRPO, ONNX export)
  - Linked from the app — free T4 GPU, ~1 hour total
- DeepSeek R1 paper
  - "DeepSeek-R1: Incentivizing Reasoning Capability in LLMs" (2025)
- HuggingFace SmolLM Playbook
  - End-to-end guide for small model post-training
- Previous SNIA webinar {AI Storage: The Critical Role of Storage in Optimizing AI Training Workloads" https://youtu.be/vycUUhlRmVs
- SNIA DSN AI Stack series — complementary content
  - https://youtube.com/playlist?list=PLH_ag5Km-YUZaWla60wr-s3vqX0M40-t-&si=7htRMH8Ya_zJUGnb

Q & A

After this Webinar

- Please rate this webinar and provide us with your feedback
- This webinar and a copy of the slides are available at the SNIA Educational Library snia.org/educational-library
- A Q&A from this webinar, including answers to questions we couldn't get to today, will be posted on our blog at sniablog.org
- Follow us on LinkedIn and X @SNIA

SNIA | DSN
A SNIA Community

Thank You