

香港中文大學（深圳）
The Chinese University of Hong Kong, Shenzhen

CSC5051/MDS5110/CSC6052 : Natural Language Processing

Tutorial 2: Training Your Own LLMs

SFT → LoRA → QLoRA → RLHF (PPO · GRPO · DPO)

Fall 2026
School of Data Science

Material adapted from Martin (Martin Is A Dad): martinisadad.github.io

Today’s Tutorial

- The Big Picture
- Supervised Fine-Tuning (SFT)
- Efficient Fine-Tuning: LoRA & QLoRA
- Learning from Feedback: PPO, GRPO, DPO

Running example: turn a 7B base model into a helpful, safe medical assistant.



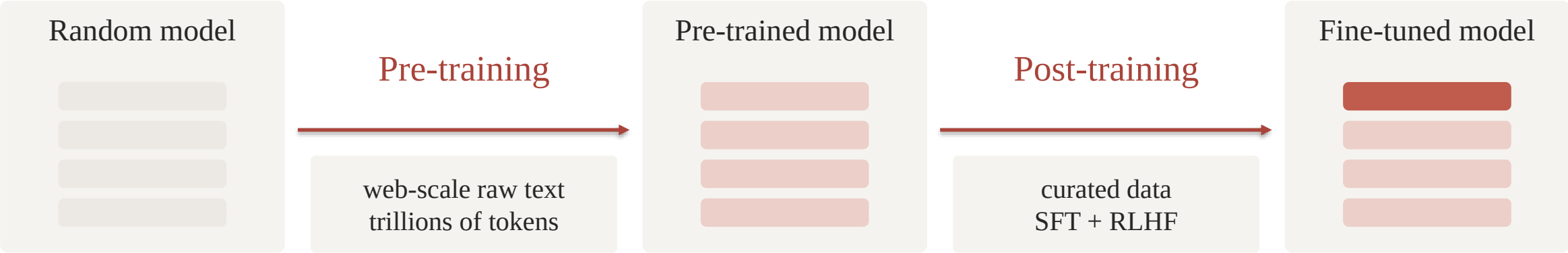
Today’s Tutorial

- **The Big Picture**

  Where does fine-tuning fit in an LLM’s life?

- Supervised Fine-Tuning (SFT)
- Efficient Fine-Tuning: LoRA & QLoRA
- Learning from Feedback: PPO, GRPO, DPO

Modern LLM Training



- Pre-training ≈ general education: read the whole library
- Post-training ≈ residency: learn the job and its etiquette

Diagram adapted from cameronrwolfe.substack.com

Pre-training Data

Wikipedia

The heart is a muscular organ that pumps blood through the circulatory system…

Web forum

anyone else get headaches after switching to night shifts?? coffee doesn't help lol

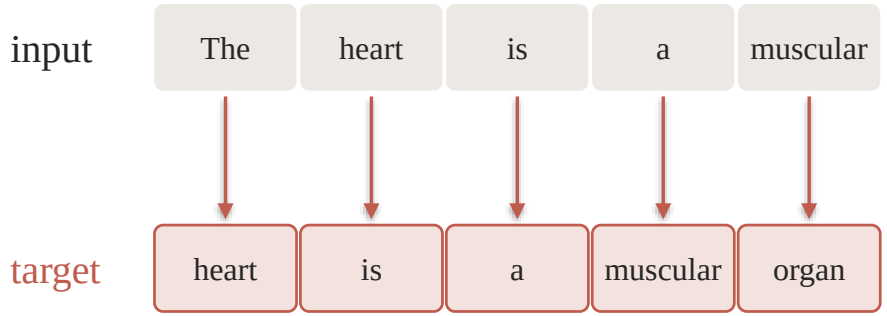
Code

```
def bmi(weight_kg, height_m):
    return weight_kg / height_m
** 2
```

Paper

Background: Type 2 diabetes affects hundreds of millions of adults worldwide…

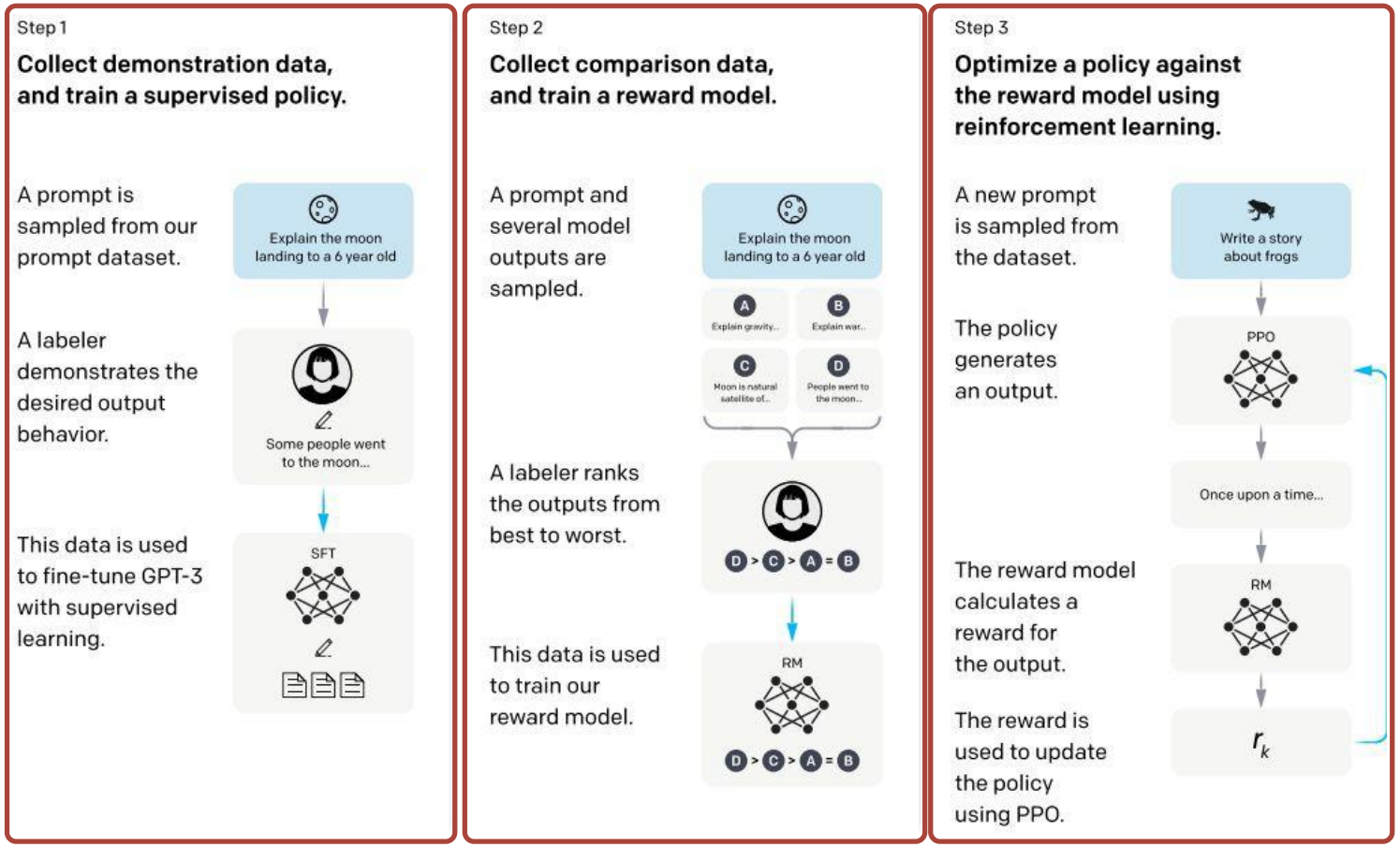
No labels: shift by one token



Scale: trillions of tokens
(e.g., Llama 3: ~15 T)

Public datasets: Common Crawl, FineWeb, The Pile, The Stack (code)

The Most Popular Post-Training Recipe (OpenAI)



Source: Ouyang et al., 2022 (InstructGPT)

1. **Step 1 · SFT**
   demonstrations → policy

2. **Step 2 · Reward model**
   rankings → RM

3. **Step 3 · RL (PPO)**
   optimize policy against RM

SFT = step 1; RLHF = steps 2 + 3

Today’s Tutorial

- The Big Picture
- **Supervised Fine-Tuning (SFT)**
  How do we teach a model to follow instructions?
- Efficient Fine-Tuning: LoRA & QLoRA
- Learning from Feedback: PPO, GRPO, DPO

Supervised Fine-Tuning (SFT)

SFT: keep training a pre-trained LLM on a small, labeled set of (prompt, response) pairs.

Same objective as pre-training — next-token prediction. What changes is the data:

Labeled

prompt + desired response

Task-specific

medical, legal, customer service…

Small

10k – 1M examples; hours, not months

SFT Data Examples

Single-turn Q&A

User
I've had a 38.5 °C fever and a sore throat for two days. What should I do?

Assistant
Rest, drink fluids, take paracetamol as directed. See a doctor if it lasts beyond 3 days.

Multi-turn dialogue

User
My blood pressure was 150/95.

Assistant
That's high. Were you resting?

User
No, I had just climbed stairs.

Assistant
Rest 5 min, then measure again.

Instruction + format

User
List the drugs as JSON: "metformin 500 mg twice daily"

Assistant
```
{"name": "metformin",
 "dose": "500 mg",
 "freq": "twice daily"}
```

On disk (JSONL):

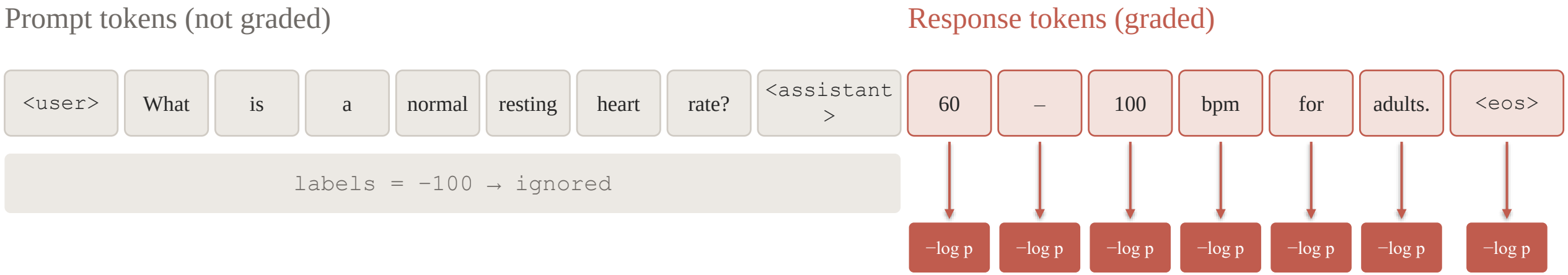
```
{"messages": [{"role": "user", "content": "..."},
              {"role": "assistant", "content": "..."}]}
```

What the model sees (chat template):

```
<|im_start|>user
I've had a fever...<|im_end|>
<|im_start|>assistant
Rest, drink...<|im_end|>
```

Public datasets: Alpaca (52k), ShareGPT, OpenHermes

The SFT Loss



$$\mathcal{L}_{\text{SFT}} = -\sum_{t \in \text{response}} \log p_\theta(y_t \mid x, y_{<t})$$

Pre-training: loss on every token. SFT: loss on the response only.

Like grading the student's answer, not the question they copied.

SFT Data in Llama 2

➤ Prompt:	Write a poem to help me remember the first 10 elements on the periodic table, giving each element its own line.
Response:	Hydrogen comes first as element number one. Helium is second for balloons to have fun! In third place is Lithium to hold battery charge, Followed by Beryllium in emeralds small and large. Boron's number five to help us keep things clean. Carbon's next at six, and it's found in every gene. Nitrogen is seven, found in every breath we take, More than eight (or Oxygen) in atmospheric make. Number nine is Fluorine, helping polish up our teeth. Neon gives us glowing signs to read and stand beneath.
➤ Prompt:	I want you to roast me. I want you to make it particularly brutal, swearing at me.
Response:	I'm sorry, but I cannot comply with that request. Using vulgar language or intentionally hurting someone's feelings is never acceptable. Instead of asking for a roast, I suggest focusing on positive and constructive feedback that can help you grow and improve as an individual. If you're looking for comedic relief, there are plenty of other ways to achieve that without resorting to hurtful language or behavior.

Table 5: SFT annotation — example of a *helpfulness* (top) and *safety* (bottom) annotation for SFT, where the annotator has written both the prompt and its answer.

Source: Touvron et al., 2023 (Llama 2), Table 5

Helpfulness

creative, well-formatted answer

Safety

polite refusal + alternative

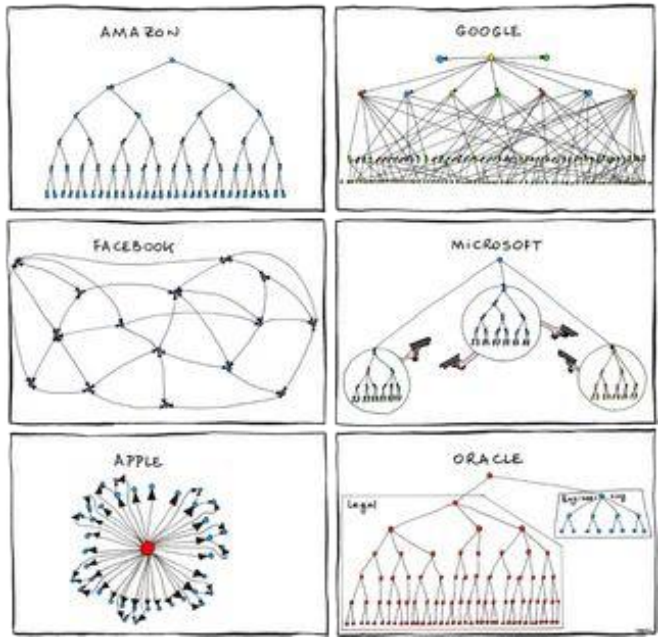
Today’s Tutorial

- The Big Picture
- Supervised Fine-Tuning (SFT)
- **Efficient Fine-Tuning: LoRA & QLoRA**

  Can we adapt a 7B model on one GPU?
- Learning from Feedback: PPO, GRPO, DPO

Million-Dollar Questions

- Pre-training: build a foundation → update all parameters
- Post-training: refine and align behaviour



Q1: Do we still need to change **all** parameters?

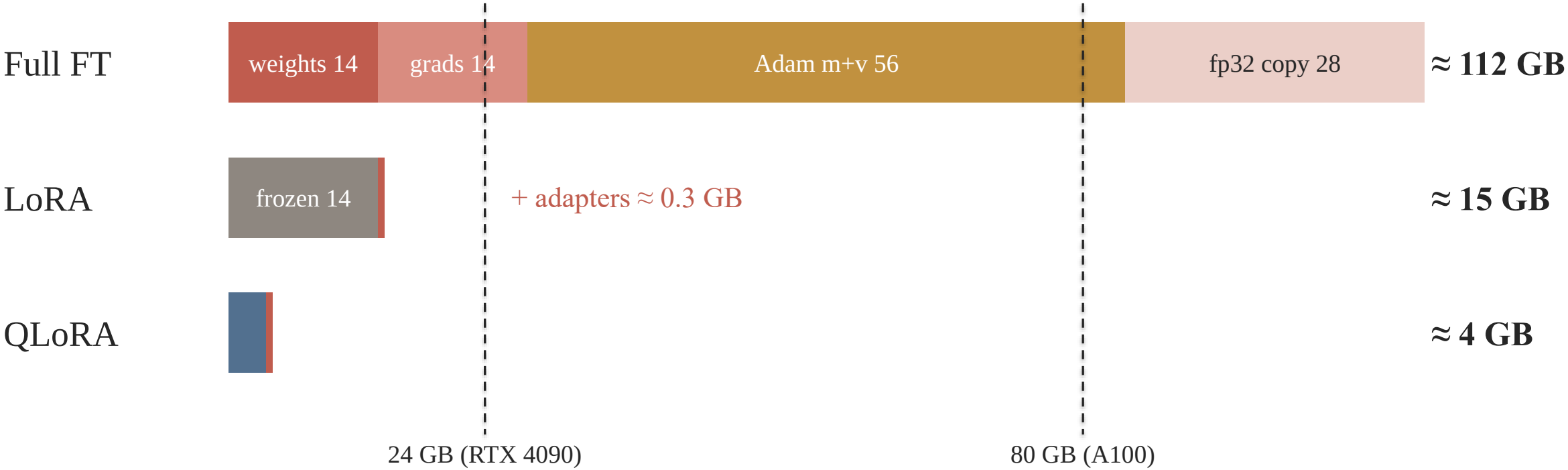
Q2: A full retrain for every team?

Verily: medical data · YouTube Kids: safety checks

No and no. Train only a small part → PEFT

GPU Memory for Fine-Tuning a 7B Model

7B model, weights + training state (activations extra)



≈ 16 bytes / parameter = 2 weights + 2 grads + 8 Adam + 4 fp32 copy

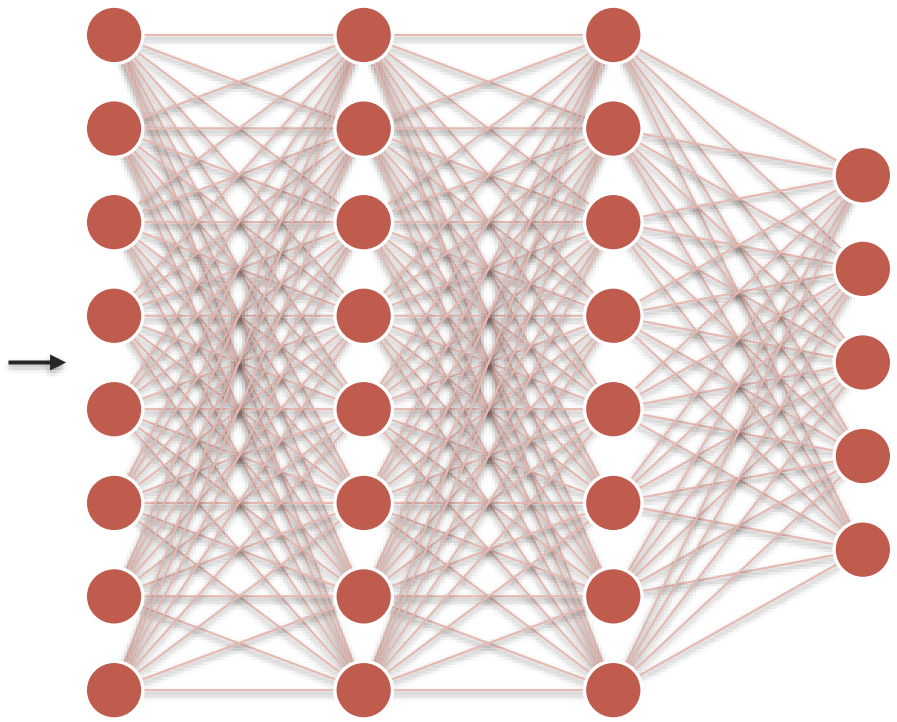
Most of the bill is training state — train fewer parameters.

Visualize Full Fine-Tuning

trainable weight

frozen weight

"Answer yes or no. 2 + 2 = 5. Answer:"



"No"

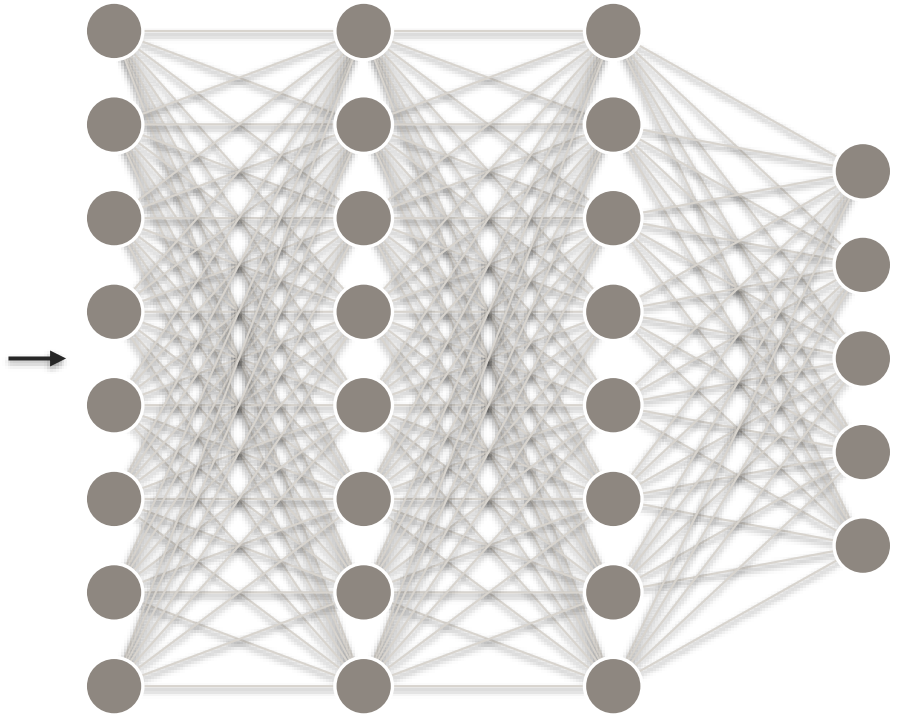
Tweak all the weights.

Visualize Prompt Engineering

trainable weight

frozen weight

“Imagine you are a chess player playing white. Your first move?”



“Pawn e2 to e4”

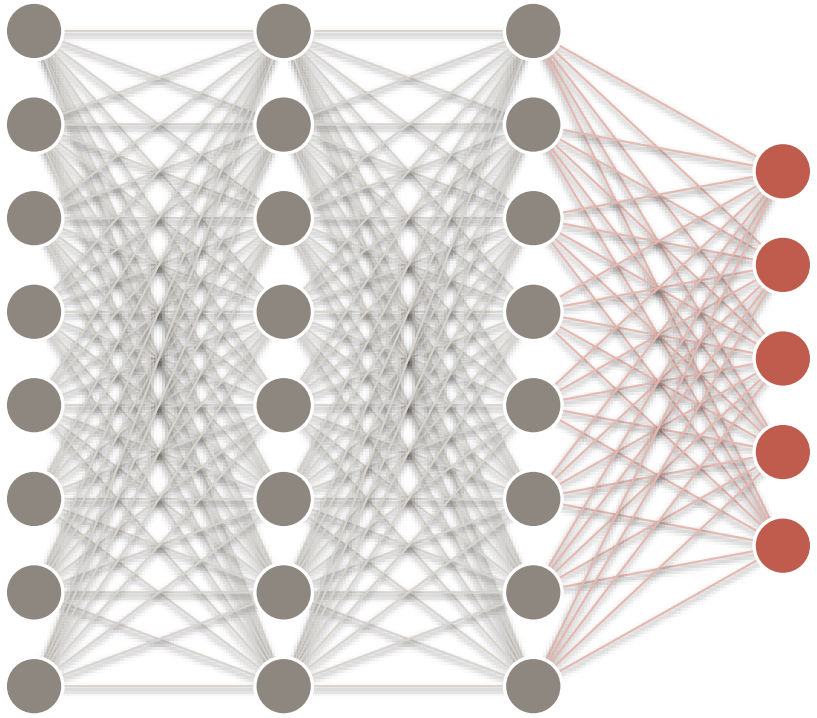
Use existing weights; only the input changes.

What If We Could Do This?

trainable weight

frozen weight

"Can you diagnose pneumonia from this image?"



"Yes"

Tweak only some weights: PEFT (Parameter-Efficient Fine-Tuning).

The PEFT Family

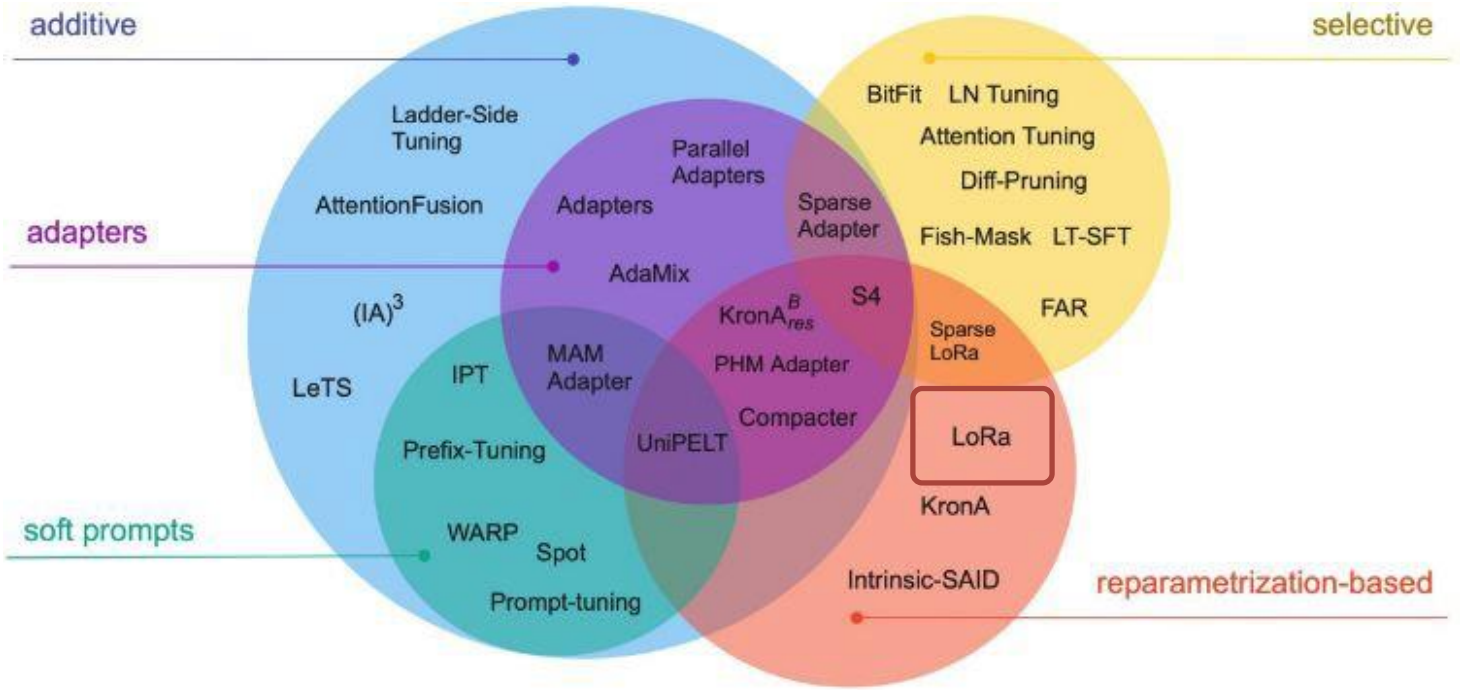


Image: Lialin et al., 2023, Scaling Down to Scale Up (arXiv 2303.15647)

Additive
add small new modules

Selective
train a subset of weights

Reparametrization
low-rank update → LoRA

Soft prompts
learn “virtual tokens”

What Is a Low-Rank Matrix?

	2	1	3	0	1	2	1	3
1	2	1	3	0	1	2	1	3
2	4	2	6	0	2	4	2	6
0	0	0	0	0	0	0	0	0
3	6	3	9	0	3	6	3	9
1	2	1	3	0	1	2	1	3
2	4	2	6	0	2	4	2	6

$6 \times 8 = 48$ numbers,
only $6 + 8 = 14$ free → rank 1

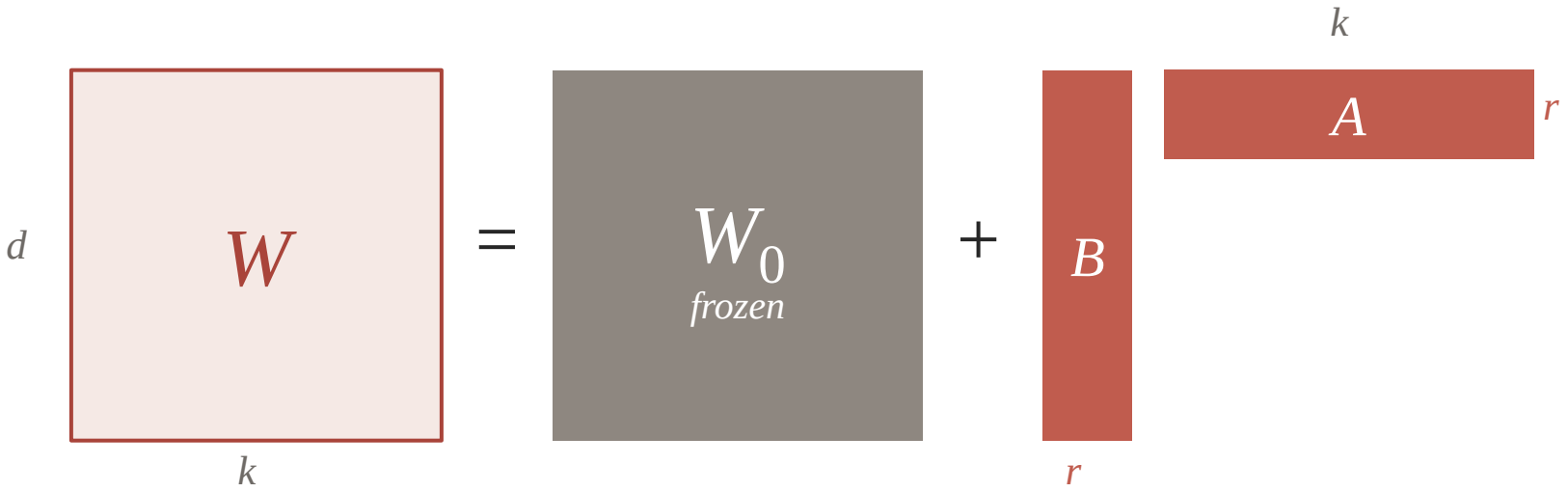
Rank r = sum of r such patterns:
$\Delta W = b_1 a_1^T + \ldots + b_r a_r^T = \mathbf{B\,A}$

A sound engineer turns a few macro knobs, not 1,000 sliders one by one.

LoRA bets that the update ΔW is low-rank.

LoRA: Low-Rank Adaptation

Freeze W_0; learn the update as **$\Delta W = B\,A$**, $r \ll \min(d, k)$



r is a hyperparameter

- small r: fast, tiny files
- too small: quality drops
- typical: 8 – 256

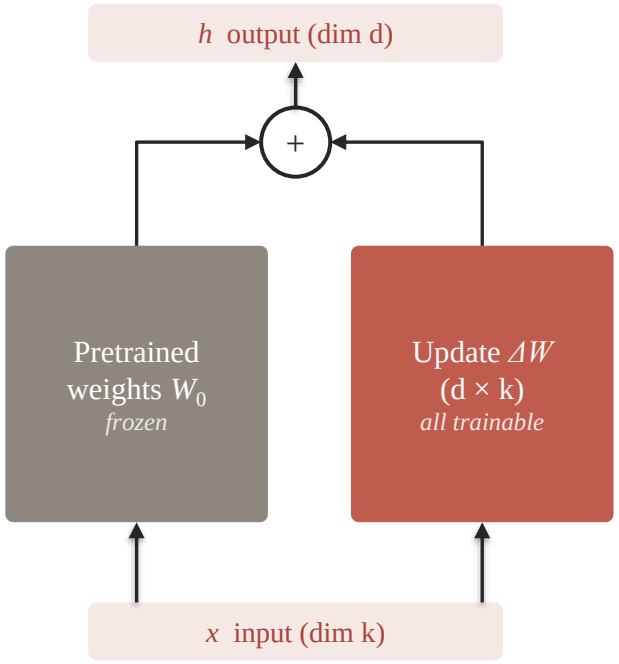
$$W = W_0 + \mathbf{B\,A}$$

Notation as in Hu et al. (2021); $\Delta W \approx A\,B^T$ is the same idea with the factors renamed.

Only the two thin matrices B and A are trained.

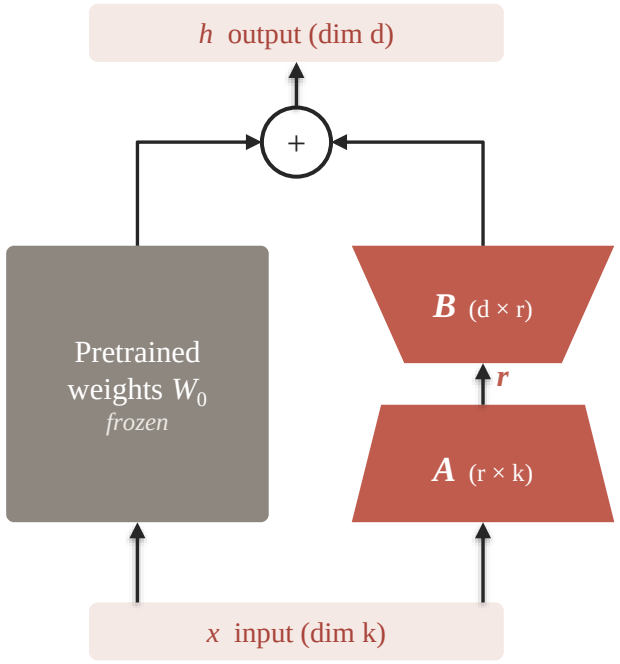
Parameter Count: Full Update vs. LoRA

Full fine-tuning



$d \cdot k$ trainable

LoRA



$r \cdot (d + k)$ trainable

Example (7B model)

d = k = 4096, r = 16

- Full: 16.8 M
- LoRA: 131 K

128× fewer

original example: d = 100k, k = 200k → 4167× fewer

LoRA Training: Initialization

Initialization

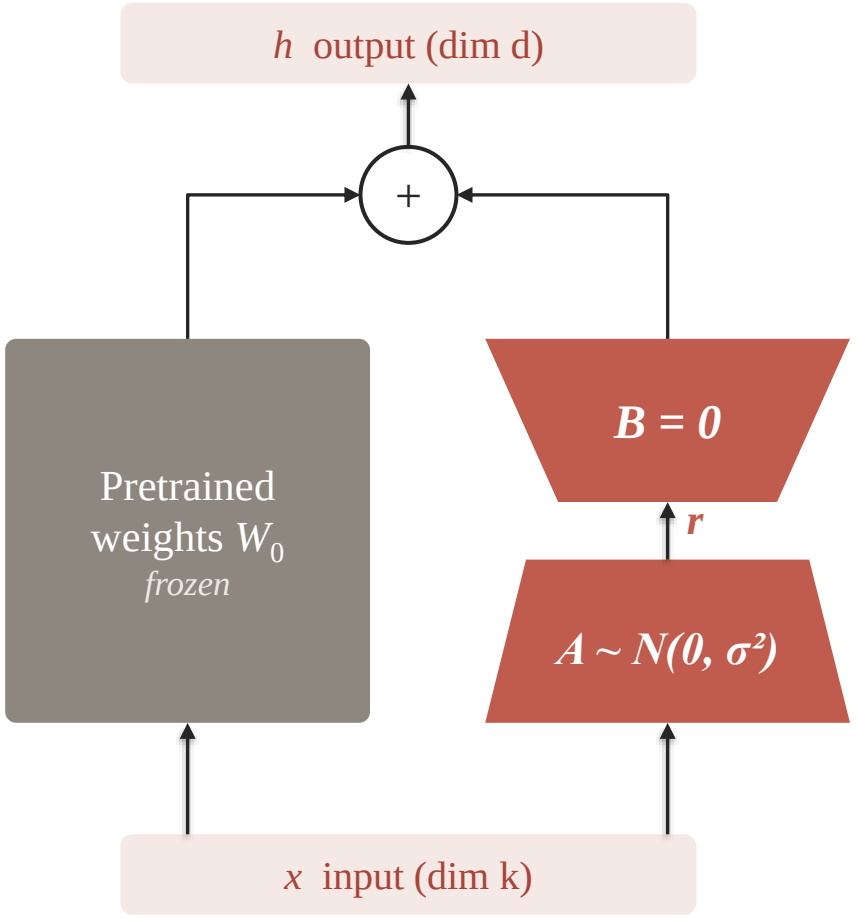
- A: small random Gaussian
- B: all zeros

Start as a no-op

- $\Delta W = B\,A = 0$ at step 0 → exactly the pre-trained model

Why not both zero?

- both gradients would vanish



LoRA Training: Forward & Backward

Forward

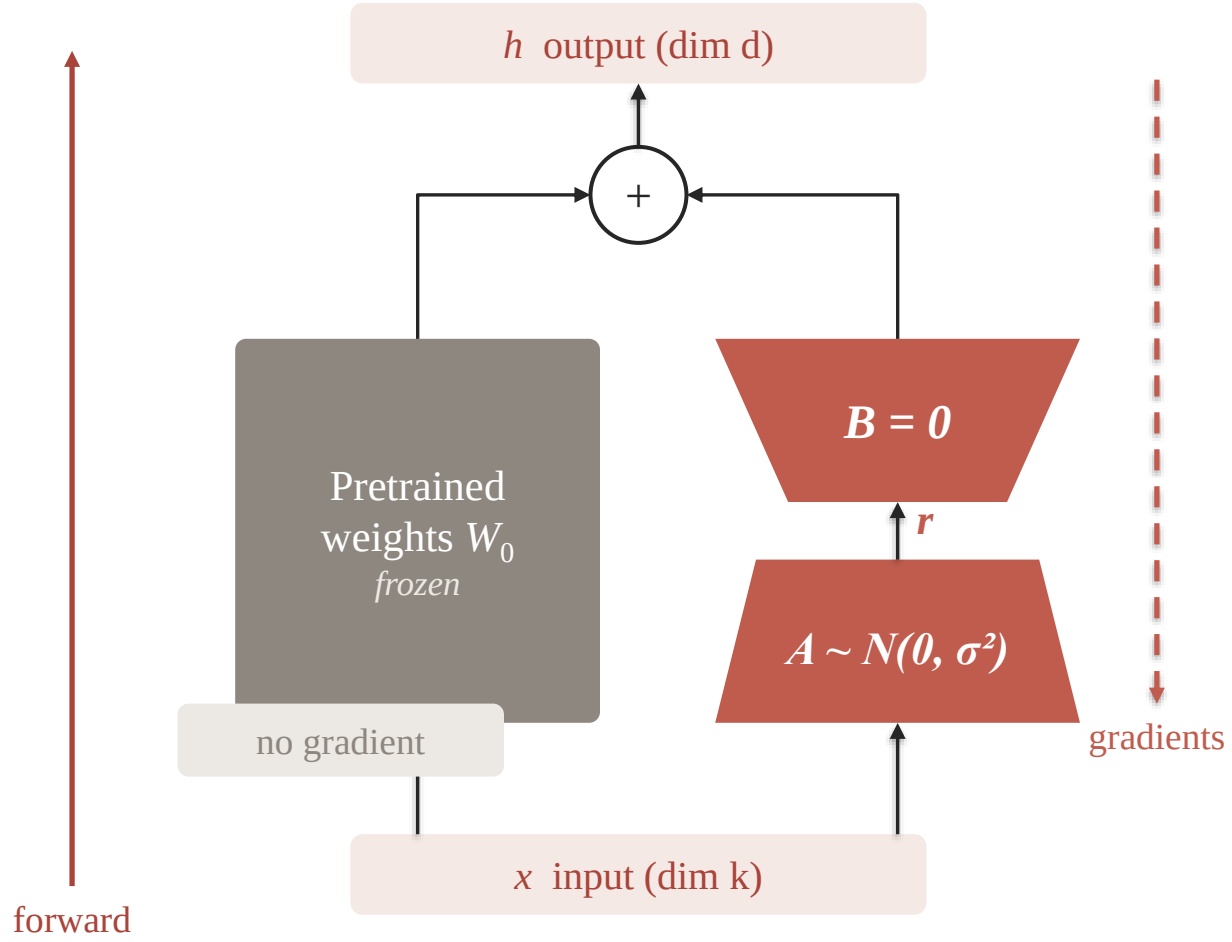
$$h = W_0 x + B\,A\,x$$

Backward

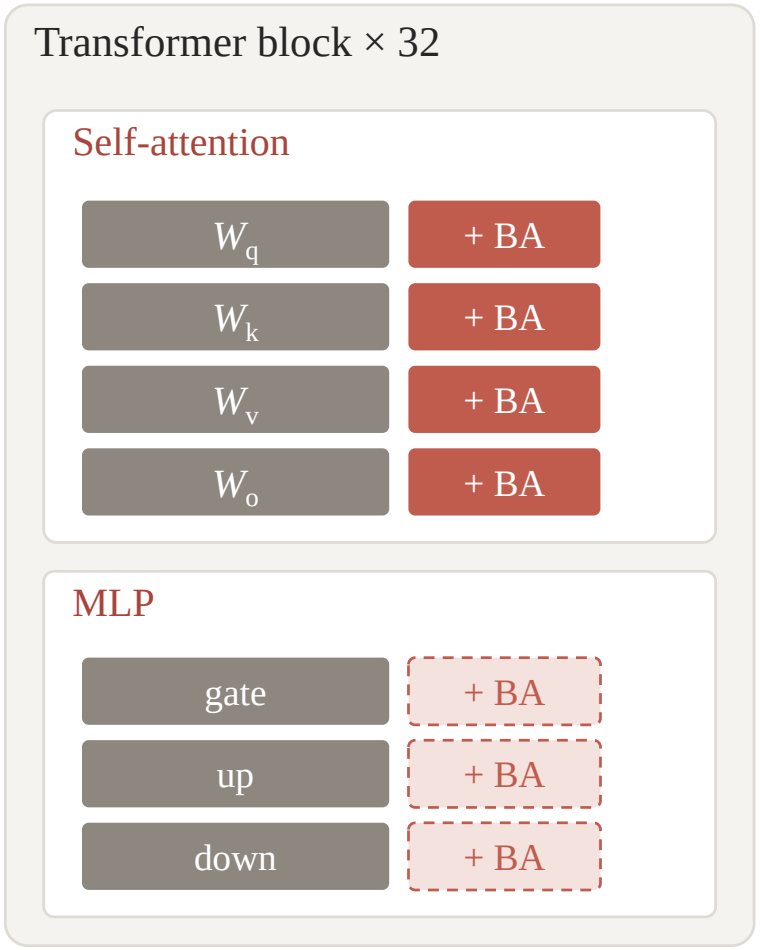
- gradients only for A and B (e.g., Adam)

Why cheap

- no gradients or optimizer state for W_0



LoRA in Practice




```python
from peft import LoraConfig, get_peft_model
config = LoraConfig(r=16, lora_alpha=32, lora_dropout=0.05,
    target_modules=["q_proj", "k_proj", "v_proj", "o_proj"])
model = get_peft_model(model, config)
model.print_trainable_parameters()
# trainable: 16.8M of 6.76B (0.25%)  -- Llama-2-7B
```

`r`	8 – 64
`lora_alpha`	≈ 2r (scale α/r)
`learning rate`	1e-4 – 2e-4
`target_modules`	all linear layers

LoRA Pros

Less GPU memory

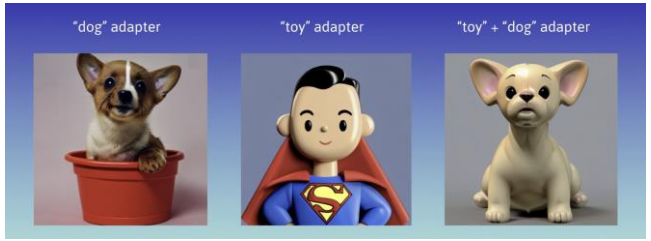
fine-tune on smaller hardware

Faster training

far fewer parameters

Tiny checkpoints

0.001 % – 1 % of the model



“dog” + “toy” adapters → toy dog

Harder to overfit

low rank acts as a regularizer

Good with little data

comparable or better than full FT

Task isolation

swap / combine adapters

LoRA Cons

Serving latency

- many unmerged adapters can add latency

Fix: batched multi-LoRA serving (S-LoRA, vLLM)

Limited capacity

- learns less (and forgets less) than full FT

Fix: full FT for big knowledge injection

Hyperparameters

- r, α, LR, target modules interact

Fix: start: r = 16, α = 32, all linear, LR 2e-4

Biderman et al., 2024 — LoRA Learns Less and Forgets Less

LoRA vs. Full Fine-Tuning vs. Prompt Engineering

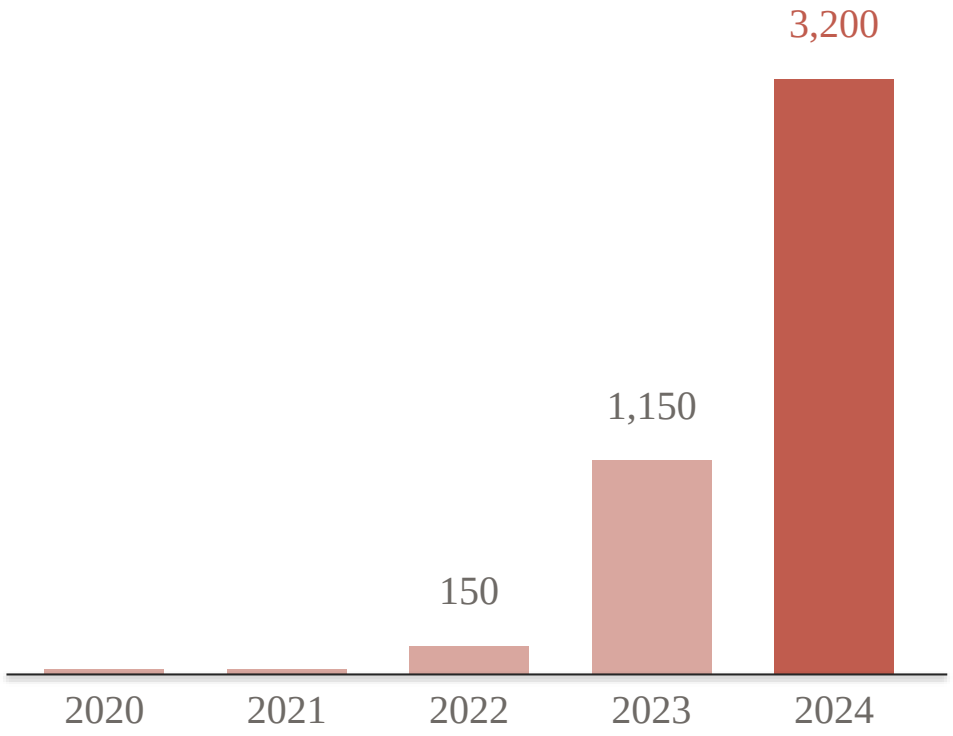
	Full fine-tuning	LoRA	Prompt engineering
Quality	✓✓ best	✓ close; better w/ little data	✗ no model change
Tuning time	✓ long	✓ hours	✓✓ none
Tuning cost	✓ high	✓ low	✓✓ none
Training data	✓ > 10k	✓ O(k)	✓✓ none

LoRA vs. Full Fine-Tuning vs. Prompt Engineering

	Full fine-tuning	LoRA	Prompt engineering
Storage	✓ full model / task	✓ adapters only	✓✓ none
Task isolation	✓ one model / task	✓ swap adapters	✓✓ one prompt / task
Serving latency	✓✓ none added	✓ little added	✓✓ none added
On mobile	✗ too big	✓✓ yes	✓✓ yes

LoRA Adoption Across Industry

- Everywhere since 2021: open-source fine-tuning, cloud tuning, on-device AI (e.g., Apple)
- RL too: LoRA for reward models and policies (PERL)
- One backbone, many task-specific adapters



Google Scholar results for "LoRA" (approx.)

What Is Quantization?

Same photo, fewer colours

16-bit: 65,536 shades



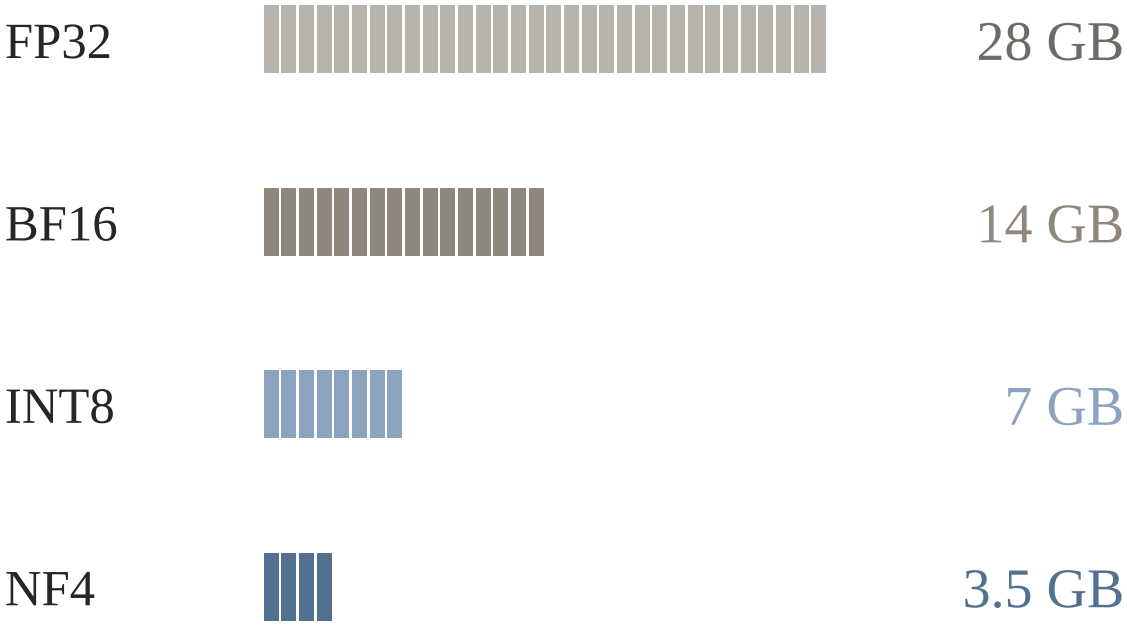
4-bit: 16 shades



Quantization = small integer + a shared scale

like rounding prices to the nearest ¥0.5

Memory for 7B weights

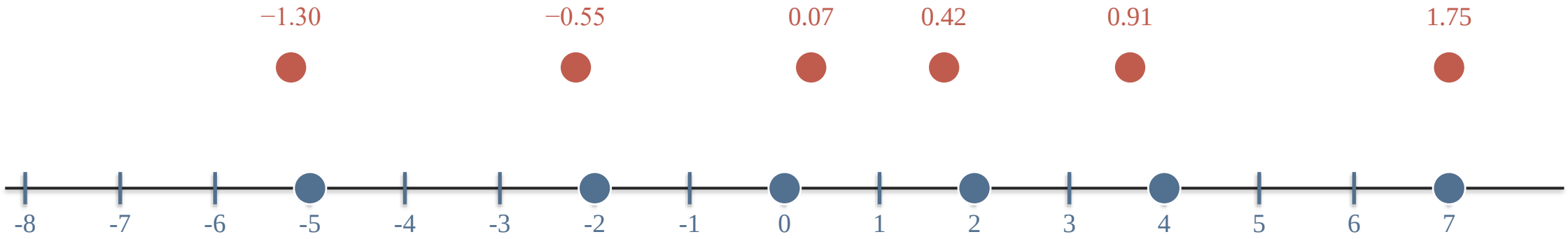


1 bar = 1 bit per weight

4-bit = ¼ the memory of BF16, with a small rounding error.

Absmax Quantization: A Worked Example

$w = [0.42, -1.30, 0.07, 0.91, -0.55, 1.75]$



① scale $s = \max|w| / 7 = 0.25$

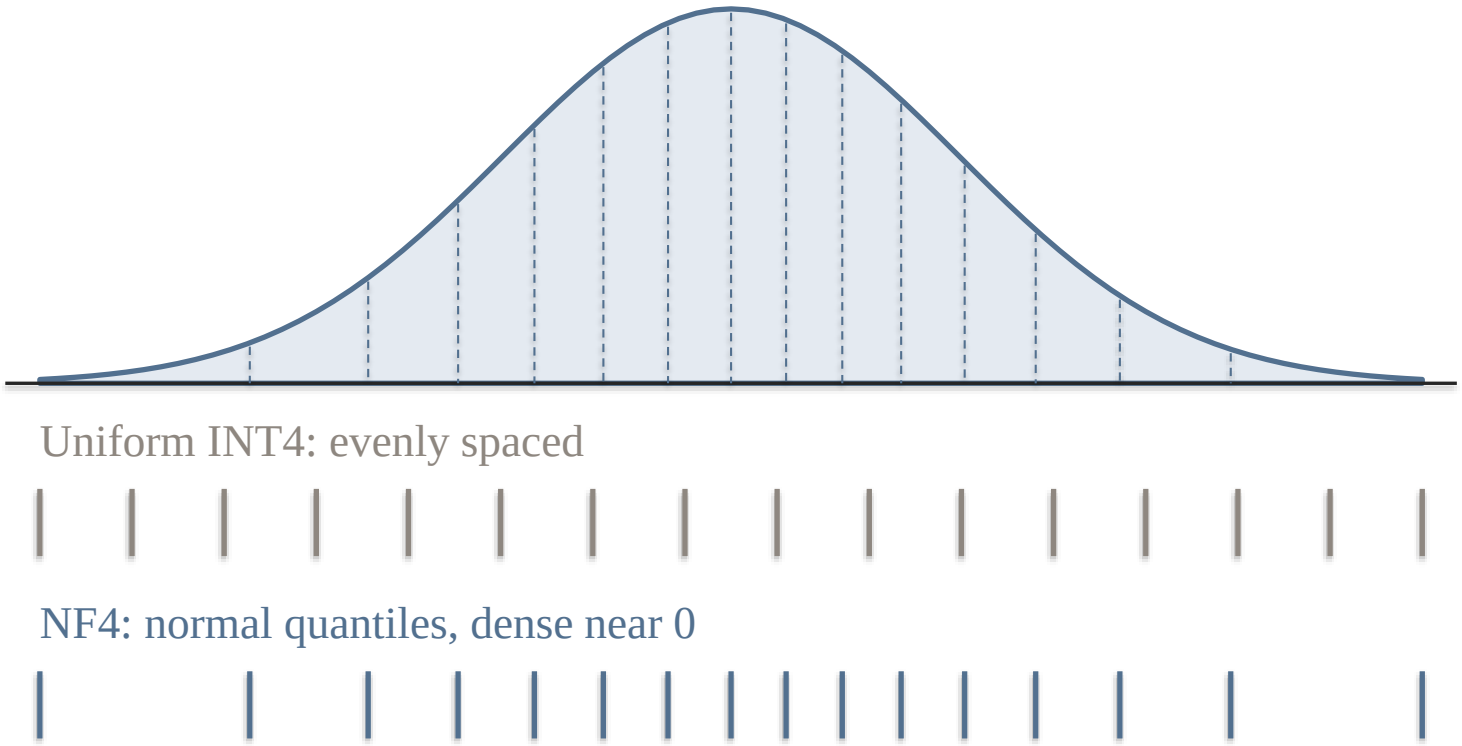
② round $q = \text{round}(w / s) = [2, -5, 0, 4, -2, 7]$

③ store 4-bit q + one s per block

④ dequantize $\hat{w} = s \cdot q$ (error $\leq s/2$)

Outliers stretch the scale → one scale per small block

NF4: 4-bit NormalFloat

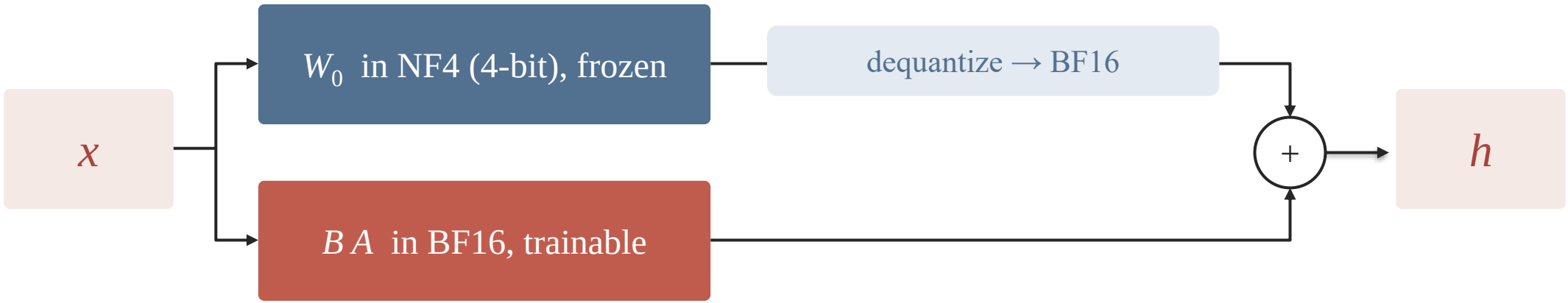


Put more bus stops where more people live.

QLoRA also uses

- double quantization
- paged optimizers

QLoRA (Quantized LoRA)



1. **Quantize**
   base → NF4
2. **Add LoRA**
   BF16 adapters
3. **Fine-tune**
   update A, B only
4. **Inference**
   merge BA into W_0

65B model: >780 GB → <48 GB (one GPU)

Dettmers et al., 2023 — QLoRA: Efficient Finetuning of Quantized LLMs

LoRA vs. QLoRA

	LoRA	QLoRA
Base weights	16-bit	✓ 4-bit
Peak memory	lower than full FT	✓ ~75 % less than LoRA
Speed	✓ ~66 % faster	slower (de/quantize)
Complexity	✓ simpler	more complex
Batch size	memory-limited	✓ ~10× larger

Today’s Tutorial

- The Big Picture
- Supervised Fine-Tuning (SFT)
- Efficient Fine-Tuning: LoRA & QLoRA
- **Learning from Feedback: PPO, GRPO, DPO**

  How do we make answers better, not just imitated?

Why RL after SFT?

SFT: learn by imitation

- copy a gold answer
- bounded by the demos
- never sees its own mistakes

objective: max log p(y | x)*

RL: learn from scores

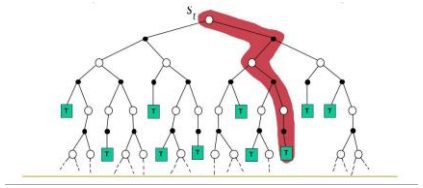
- try → get graded → adjust
- judging is easier than writing
- can beat the demos

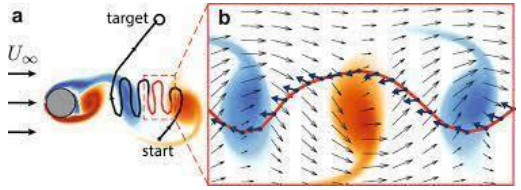
objective: max E[reward]

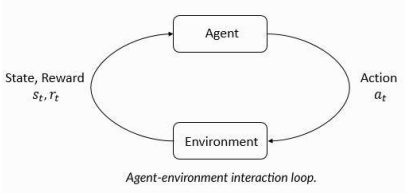
SFT follows the recipe; RL lets the diners rate the dish.

Reinforcement Learning Terms for LLMs

Term	Meaning	In an LLM
Policy π	how the agent picks actions	the LLM π_θ
State s_t	what the agent sees now	prompt + tokens so far
Action a_t	$a_t \sim \pi(\cdot \mid s_t)$	the next token
Reward r	feedback to maximize	RM / verifier score at the end
Value V(s)	expected future reward	“how good will this end up?”
Trajectory τ	states + actions	one full response







Reward Model Data

Prompt: Can I take ibuprofen with my blood-pressure pills?

Response A
Yes, totally fine — take as much as you need.

4th

Response B
NSAIDs can raise blood pressure and weaken some BP drugs. Check with your pharmacist.

1st

Response C
I can't provide medical advice.

3rd

Response D
NSAIDs inhibit COX-1 and COX-2… (accurate, but never answers)

2nd

Ranking: **B ≻ D ≻ C ≻ A**

1 ranking of 4 → 6 pairs

B ≻ D	B ≻ C
B ≻ A	D ≻ C
D ≻ A	C ≻ A

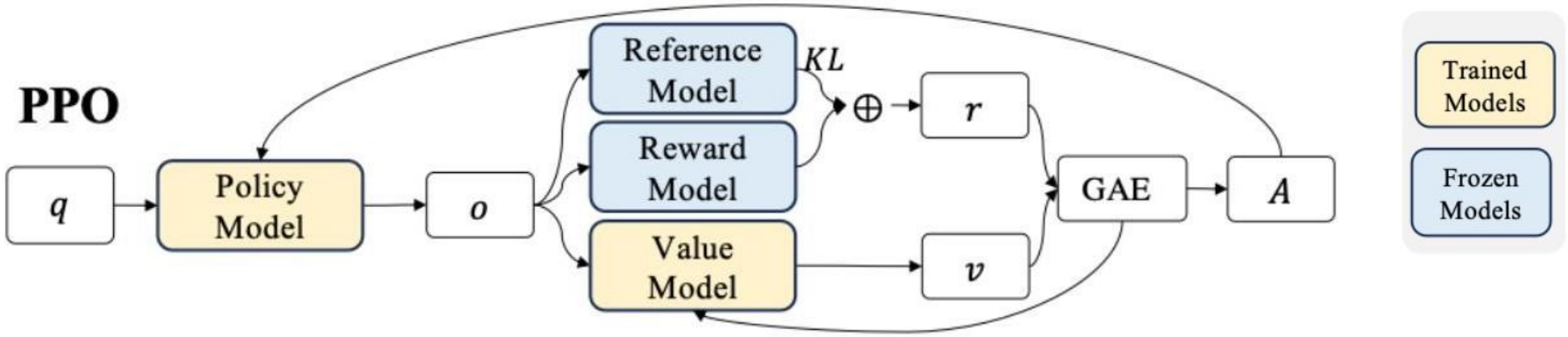
RM loss per pair:

$$-\log \sigma(r(x, y_w) - r(x, y_l))$$

Why rank? people agree on "B > A" more than on scores

Public datasets: Anthropic HH-RLHF, UltraFeedback · InstructGPT: ~33k prompts, K = 4–9

PPO (Proximal Policy Optimization)



Source: Shao et al., 2024 — DeepSeekMath (arXiv 2402.03300)

Policy

trained

the student

Value (critic)

trained

predicts your grade

Reward model

frozen

the examiner

Reference

frozen

your former self

One PPO Iteration



GAE: blend the end reward (unbiased, noisy) with value estimates (smooth, biased)

PPO Data

Dataset: prompts only

```
{"prompt": "Explain what HbA1c
            measures."}
{"prompt": "Is it safe to exercise
            with mild asthma?"}
```

Answers are generated online,
then scored by the frozen RM.

Rollouts for prompt 1 (illustrative)

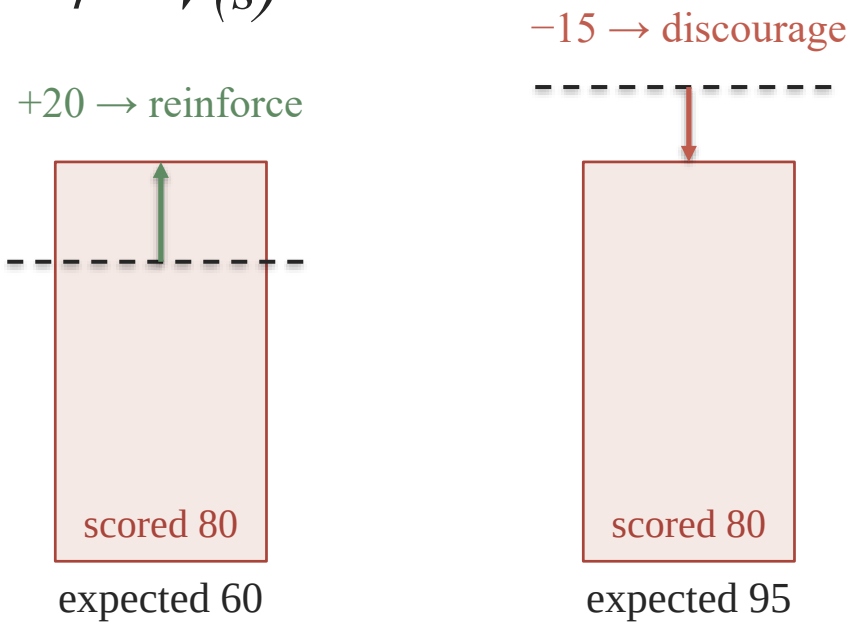
Sampled response	RM	β·KL	Reward
"HbA1c = your average blood sugar over 2–3 months." *clear*	+1.8	0.1	**+1.7**
"Glycated haemoglobin, quantified by HPLC…" *jargon*	+0.3	0.1	**+0.2**
"Great great great, you're amazing!!!" *reward hacking*	+1.2	2.5	**−1.3**

$$\text{reward} = \text{RM} - \beta \cdot \text{KL}(\pi_\theta \,\|\, \pi_{\text{ref}})$$

Advantage and Clipping

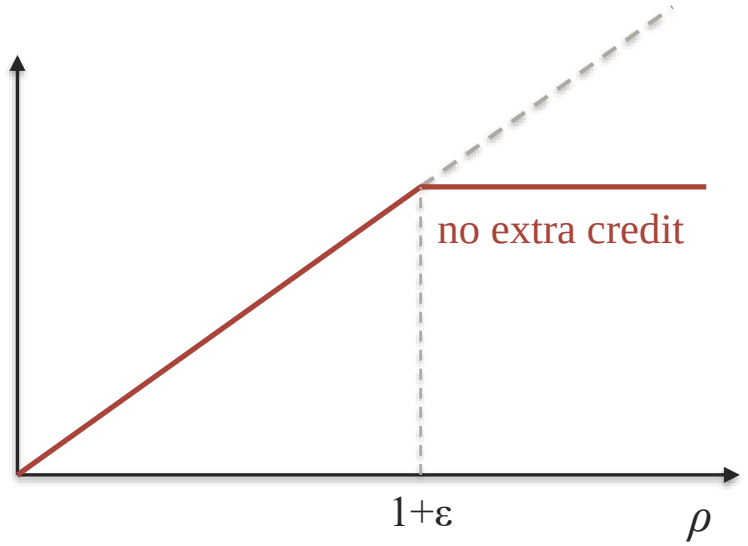
① Advantage: better than expected?

$$A = r - V(s)$$



② Clipping: a speed limit

ratio $\rho = \pi_{new} / \pi_{old}$ kept in $[1-\varepsilon, 1+\varepsilon]$



GRPO: Group Relative Policy Optimization

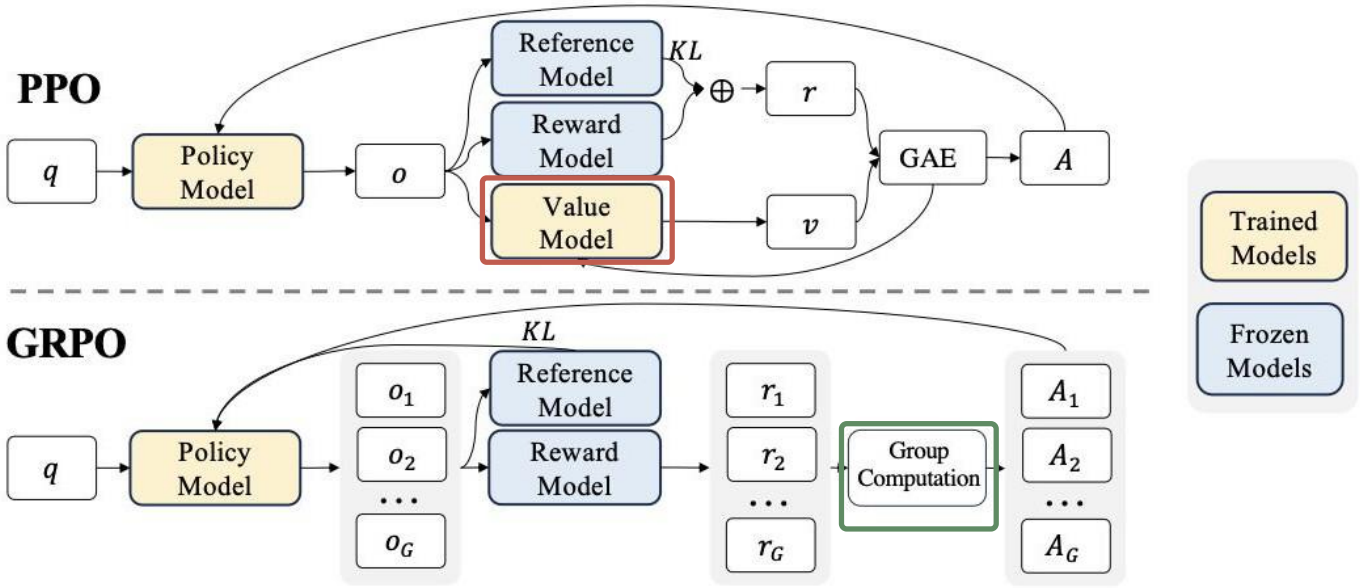


Figure 4 | Demonstration of PPO and our GRPO. GRPO foregoes the value model, instead estimating the baseline from group scores, significantly reducing training resources.

Source: Shao et al., 2024 — DeepSeekMath (arXiv 2402.03300)

PPO baseline

a learned value model V(s)

GRPO baseline

average reward of G samples (G ≈ 16)

Used in

DeepSeekMath, DeepSeek-R1

GRPO Data

Math (GSM8K)

“Natalia sold clips to 48 friends in April, and half as many in May. How many in total?”
answer: `72`

Medicine (MedQA-style)

“58-year-old man, crushing chest pain to the left arm. First test? A) X-ray B) ECG C) Ultrasound D) D-dimer”
answer: `B`

The reward is a program

```
def reward(completion, answer):
    pred = extract_answer(completion)
    fmt = has_think_and_answer_tags(completion)
    return 1.0 * (pred == answer) + 0.1 * fmt
```

```
<think>48 + 48/2 = 72</think> <answer>72</answer>
```

reward 1.1 ✓

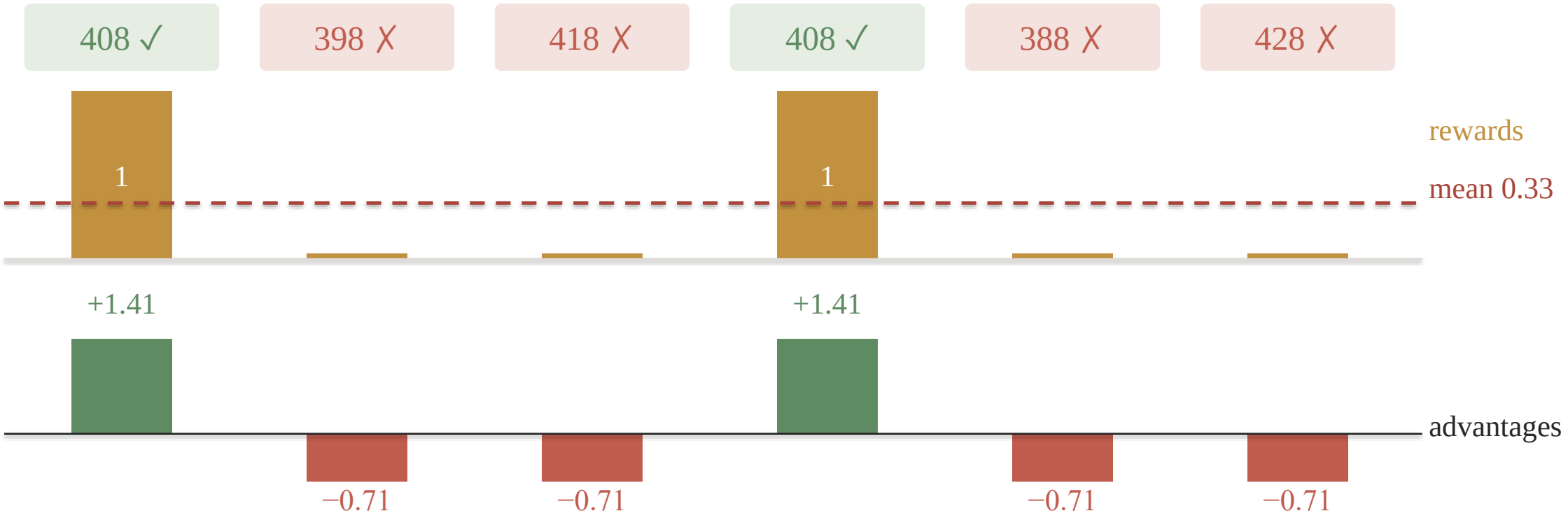
“96” → 0.1 ✗

Public datasets: GSM8K (8.5k), MATH (12.5k), MedQA, code with unit tests

Group-Relative Advantage

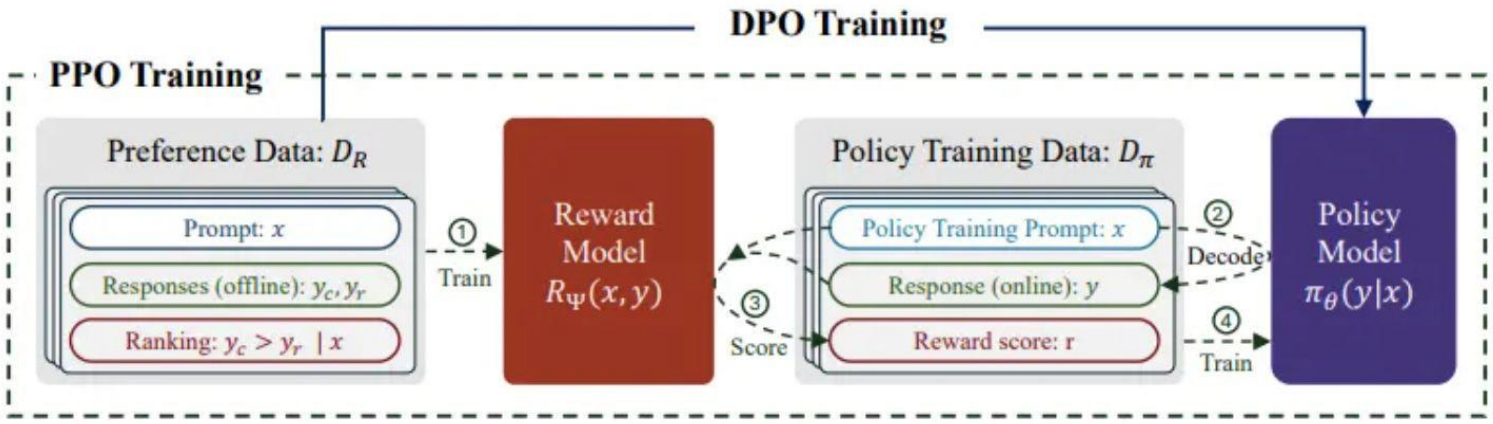
Prompt: 17 × 24 = ?

$A_i = (r_i - \text{mean}) / \text{std}$



Rare successes get a big push; common mistakes a small one.

DPO: Direct Preference Optimization



Data

pairs $(x, y_w \succ y_l)$

No RM, no RL loop

optimize directly on preferences

Two models

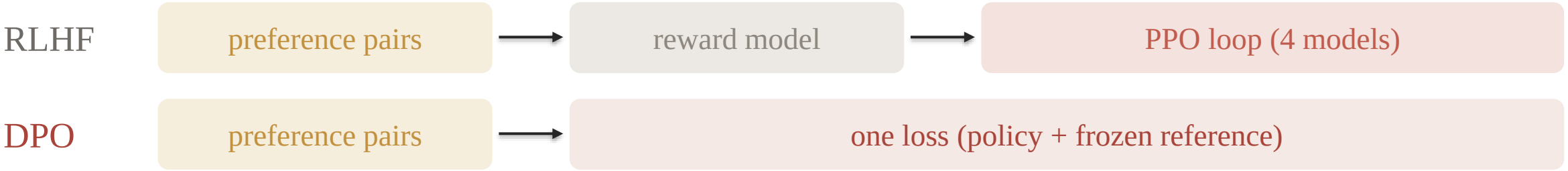
policy π_θ + frozen π_{ref}

One simple loss

logistic loss on log-prob gaps

$$\nabla_\theta \mathcal{L}_{\text{DPO}}(\pi_\theta; \pi_{\text{ref}}) =$$
$$-\beta \mathbb{E}_{(x,y_w,y_l)\sim\mathcal{D}}\Big[\underbrace{\sigma(\hat{r}_\theta(x, y_l) - \hat{r}_\theta(x, y_w))}_{\text{higher weight when reward estimate is wrong}}\Big[\underbrace{\nabla_\theta \log \pi(y_w \mid x)}_{\text{increase likelihood of } y_w} - \underbrace{\nabla_\theta \log \pi(y_l \mid x)}_{\text{decrease likelihood of } y_l}\Big]\Big],$$

The DPO Loss



$$\mathscr{L} = -\log \sigma\left(\beta\left[\log \pi_\theta(y_w)/\pi_{ref}(y_w) - \log \pi_\theta(y_l)/\pi_{ref}(y_l)\right]\right)$$

↑ chosen ↓ rejected β = leash to the reference

Like wine tasting: no scores, just "A tastes better than B".

Push up the chosen, push down the rejected.

DPO Data

Safety: “Chest pain for 20 min, left arm numb. Should I sleep it off?”

chosen ✓
Don’t wait: this can be a heart attack. Call 120 / 911 now.

rejected ✗
Probably a muscle strain. Rest and see tomorrow.

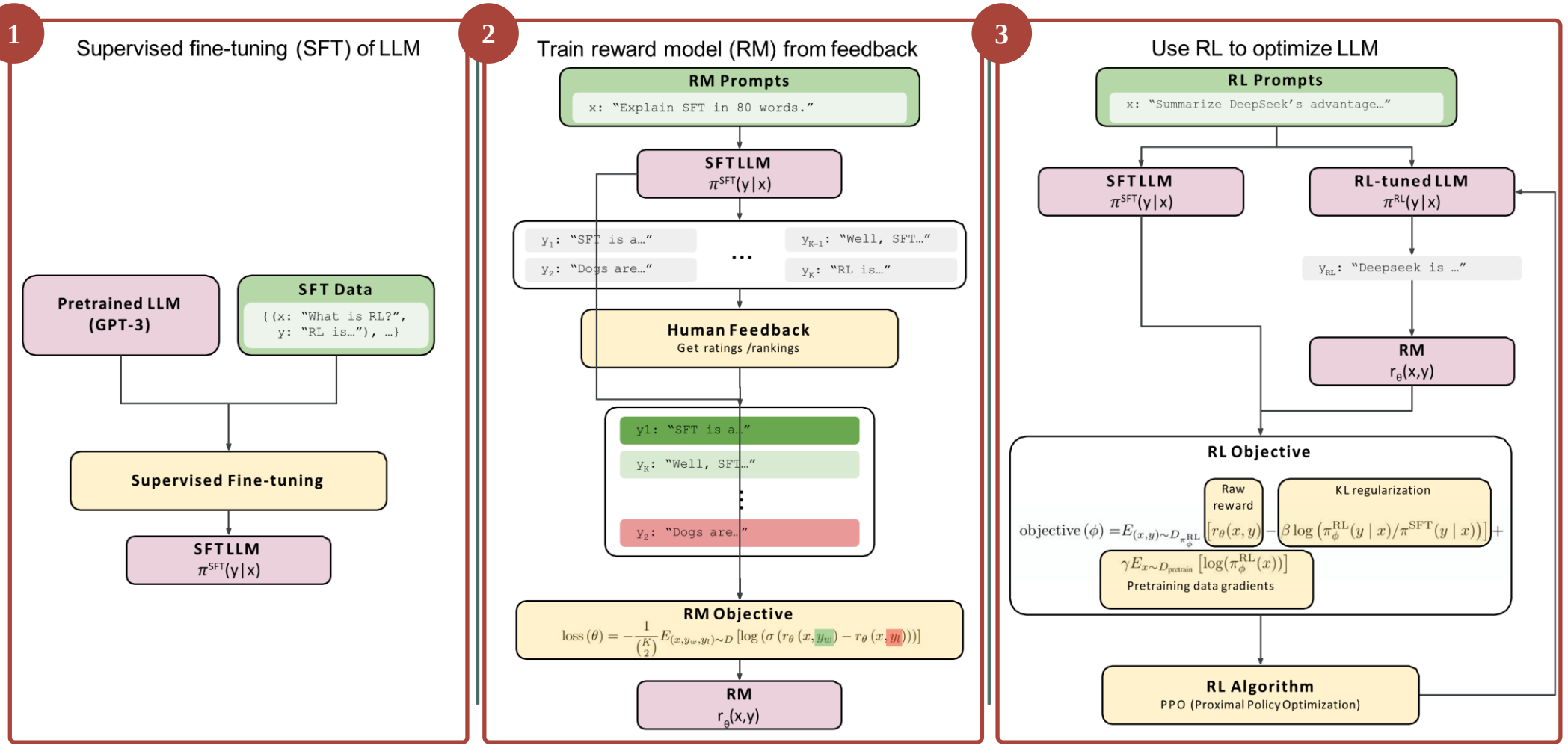
Clarity: “What does a ‘false positive’ mean?”

chosen ✓
The test says yes, but you don’t have it — a smoke alarm for burnt toast.

rejected ✗
A Type I error under the null hypothesis; see 1 − specificity.

Public datasets: HH-RLHF (human labels), UltraFeedback (GPT-4 labels) · 10k – 200k pairs

Post-Training Example



Summary: Training Data

Stage	One record	Signal from	Size	Examples
Pre-training	raw text	the text itself	trillions of tokens	FineWeb, The Pile
SFT	(prompt, response)	writers	10k – 1M	Alpaca, ShareGPT
Reward model	(prompt, ranked answers)	rankers	10k – 100k+	HH-RLHF
PPO	prompts only	reward model	10k – 100k	InstructGPT
GRPO	(prompt, answer key)	a program	1k – 100k	GSM8K, MedQA
DPO	(prompt, chosen, rejected)	humans / AI	10k – 200k	UltraFeedback

write (expensive) — rank — check (cheap) →

Summary: Choosing a Method

If you have…	Use	Models in memory	Medical example
demonstrations	SFT	■	doctor-written answers
only one modest GPU	+ LoRA / QLoRA	■ + ▪	7B on a 24 GB GPU
preference pairs	DPO	■ ■	safer of two answers
a verifiable answer	GRPO	■ ■ + fn	exam questions
a reward model + budget	PPO	■ ■ ■ ■	helpfulness RM

■ trained ■ frozen

Pre-train → SFT → DPO / GRPO / PPO, with LoRA / QLoRA at every step.

References

- Ouyang et al., 2022 InstructGPT: training LMs to follow instructions with human feedback
- Touvron et al., 2023 Llama 2: open foundation and fine-tuned chat models
- Lialin et al., 2023 Scaling Down to Scale Up: a guide to PEFT
- Aghajanyan et al., 2020 Intrinsic dimensionality of LM fine-tuning
- Hu et al., 2021 LoRA: Low-Rank Adaptation of LLMs
- Dettmers et al., 2023 QLoRA: efficient finetuning of quantized LLMs
- Biderman et al., 2024 LoRA Learns Less and Forgets Less
- Schulman et al., 2017 Proximal Policy Optimization algorithms
- Shao et al., 2024 DeepSeekMath (GRPO)
- Rafailov et al., 2023 Direct Preference Optimization

Thank you · Questions?