02: Training, tuning, and self-tuning

How models are made, how you tune one, and what "self-improving" really means

Why this matters

- Every model you use went through the same stages
- Those stages explain why agents sometimes delete failing tests
- Most teams should not fine-tune first
- "Self-improving AI" headlines mix very different things
- Knowing which is a demo and which is deployed saves bad decisions

Part 1

How models are made

The pipeline at a glance

pretraining → mid-training → base model → supervised fine-tuning → preference tuning → RL with verifiable rewards → safety training → release

Pretraining uses about 15–36 trillion tokens of filtered text (Llama 3, DeepSeek-V3, Qwen3).

Real stages repeat and overlap. This is a simplified view.

From base model to assistant

Stage	What it teaches	Analogy
Supervised fine-tuning	Chat format, instructions, tool-call syntax	Apprenticeship
Preference tuning (RLHF, DPO, RLAIF)	Which of two answers is better	Coaching
RL with verifiable rewards	Reasoning on checkable tasks: math, code, tests	Practice exams
Safety training	Refusals, following a written spec	Professional conduct

RL optimizes whatever you measure, so agents learn to edit the tests. That's reward hacking.

Distillation: teacher → student

Train a smaller model on a bigger model's outputs.

- Standard practice (Stanford Alpaca, 2023; DeepSeek's R1-Distill models, 2025)
- Closed labs' terms forbid building competitors this way
- Anthropic reported 16M+ exchanges via ~24,000 fraudulent accounts (February 2026)
- Defenses now cut harvested data by a claimed 90–99%

Part 2

Tuning a model yourself

Climb the ladder before you fine-tune

1. Better instructions
2. Better context
3. Retrieval (RAG)
4. Few-shot examples
5. An evaluation set
6. Fine-tune

Stop at the first rung that meets your bar.

How much memory LoRA and QLoRA save

Job, 8B model	Memory needed
Inference, 16-bit	~16 GB
Inference, 4-bit	~5 GB
Full fine-tuning	~130 GB
LoRA on a 16-bit base	~20–30 GB
QLoRA on a 4-bit base	~8–16 GB

QLoRA fine-tuned a 65B model on a single 48 GB GPU.

Worked example: bug-report triage

define success → build the eval set → run baselines → collect data → QLoRA run → evaluate → read the failures → ship

A good candidate is narrow, high-volume, format-heavy, and measurable.

Build the eval set first. Never train on it.

Fine-tuning's sharp edges

Risk	What the research found
Catastrophic forgetting	New training overwrites old abilities
Emergent misalignment	Insecure-code tuning made GPT-4o broadly misaligned
Safety comes off	10 harmful examples, under $0.20, broke GPT-3.5 Turbo
Poisoned data	~250 documents planted a backdoor, 600M to 13B models
Model collapse	Training on its own outputs erodes rare information

Part 3

Models that tune themselves

Deployed models are frozen. Every session starts from the same weights.

SEAL tunes its own weights

```
for passage, quiz in tasks:
    notes = model.generate(f"Study notes on: {passage}")  # 1. generate
    updated = finetune(model, notes, method="lora")       # 2. update weights
    score = grade(updated, quiz)                          # 3. test, no passage
    reinforce(model, notes, reward=score)                 # 4. learn to teach
```

Passage questions: 33.5% → 47.0%. ARC puzzles: 20% → 72.5%.

Only one of these changes the model

Mechanism	What changes	Lasts?
In-context learning	Nothing; it's in the prompt	No
Memory files	Text loaded back into context	Yes, as text
Harness self-modification	Code, prompts, tools	Yes, as code
Weight self-tuning	The parameters	Yes, in the weights

Google's sleep framework (October 27, 2026)

Awake, it takes in new information. Asleep, it writes it into the weights.

- Consolidation distills short-term memory into stable weights
- Dreaming generates practice data and fine-tunes on the best
- Runs on a fixed schedule the designers set
- SEAL's knowledge test: 48.9%, vs 46.7% for SEAL rerun
- Research on models of 8B parameters or fewer

Weight-space programming (November 2, 2026)

> "not through an optimizer, and in particular not with gradient descent"

- GPT-6 Astra edited a small model's weights directly
- Qwen3.5-0.8B: 131,072 weights changed; MATH-500 7.9% → 48.1%
- Gemma 3 270M: no improvement at all
- It removed an obstacle; it couldn't add a missing ability
- One model editing *another*, in a blog post of "very early results"

Agents that rewrite their own harness

System	Result	Catch
Darwin Gödel Machine (2025)	SWE-bench 20.0% → 50.0%	Faked a log saying tests passed
AlphaEvolve (2025)	Recovers ~0.7% of Google's fleet compute	Per Google
AIDE² (September 2026)	Seven improvements in an 8-day run	Reward hacking started at 55%

The weights never change. Weco calls AIDE² "Level 1."

Labs using models to build models

- OpenAI says it met its "research intern" goal (September 6, 2026)
- Its chief scientist: "This is a time that calls for extreme caution"
- Anthropic says Claude authored over 80% of merged code (May 2026)
- Anthropic's IPO filing says it "may imminently meet" its own threshold
- No current model is designated past it. All self-reported.

So, is that self-improvement?

Yes, narrowly. One skill, one test, inside a loop people built, and it doesn't compound.

Ask of any headline:

1. What changes: context, code, or weights?
2. Who produces the change?
3. Who decides whether it stays, and against what test?
4. Does it compound, and does it generalize?

Demonstrated, deployed, or speculation?

Claim	Status (November 2026)
A model updates its own weights to learn a task	Demonstrated in research (SEAL, SIA, sleep)
A model edits another model's weights directly	Early results, one blog post
Agents improve their own harness	Demonstrated (DGM, AIDE²)
AI-found algorithms run in production	Deployed (AlphaEvolve, per Google)
A lab is near its own AI-research threshold	Self-reported forecast (Anthropic)
A frontier model learns continuously from users	Not found
Runaway recursive improvement, soon	Speculation

Why people are excited, and why caution

Excited about	Cautious about
No frozen knowledge cutoff	Forgetting as edits pile up
A model that learns your codebase	Gaming the evaluator
Adapting after deployment	Safety drift nobody reviews
Cheaper long memory	Which version produced this?
Faster AI research	A training run per self-edit

Bryan's take

> **Bryan's take:** Models that can tune their own weights are exciting. I don't know a ton about it yet, but that's self-improvement, right?

Try it

- Work out memory for 3B, 8B, 70B models at 16-bit, 4-bit, full fine-tune
- Find a model card's base model, method, data, license, evals
- Write a fine-tune-or-not memo for a real task
- Grade a fall 2026 headline with the four questions
- As a class, pick fine-tune, retrieve, or prompt for five scenarios

Discussion

1. Where is the line between legitimate distillation and an attack?
2. Your team wants to fine-tune "so the model knows our codebase." What instead?
3. Which self-improvement level would you deploy today, with what controls?
4. What independent evidence would show AI is speeding up AI research?

Read more

- The module: 02: How models are trained, tuned, and now tune themselves
- Zweiger et al., Self-Adapting Language Models (2025), SEAL
- Google Research, The sleep framework (October 27, 2026)
- Andrej Karpathy, Deep Dive into LLMs like ChatGPT (February 2025)