

We heard an idea at a talk on Saturday and tried it on Monday night. It mostly worked.

Claude Opus rewrote the instructions of Gemma, a small free model on our office Mac, four times overnight. Judged side by side with Opus on screens it had never seen, Gemma went from 0.59 points out of 10 behind to **0.16 behind**.

The talk

On Saturday 3 October we went to hear **Prof. Sayan Ranu of IIT Delhi** at Spotify Unlock × Pint of View in Delhi. He talked about how systems like Google DeepMind's FunSearch and AlphaEvolve find new solutions, and how to make that cheaper. **One result stuck:** in his lab's tests, the most expensive model didn't find the best answer even once. His advice, as we noted it: start with a cheaper model and step up only when it disappoints you.

Then he showed a loop. Attempts sit in a pool with their scores, a builder asks an AI model for a better one, and a tester scores that and puts it back in the pool. A small note in the middle said what you write yourself, and everything else goes round on its own.

The job we gave it

We keep a map of what our software does, screen by screen. A **crawler walks the software** on test servers with sample data and saves every screen; then an AI writes a plain description of each one: what it's for, who sees it, every field and button. It's about 1,300 screens of repetitive work, and it ran on Claude Opus, the strongest model we use.

On Monday we tried a free open model, Google's Gemma 4, on the office Mac's own chip. **Our first try used Opus's instructions** word for word: 8.6 out of 10 where Opus scored 9.9, so 1.3 points behind, and 5 of 60 descriptions wrong in a way that would mislead the map. **Our second try, by hand**, added clearer wording and code that checks every recorded button made it in and sends the description back once if not. That got Gemma to 0.65 behind. Its smaller 12B version did as well and left the Mac more memory to spare, so that one went into the loop.

On the slide, "Inside FunSearch and AlphaEvolve"

"You write: the problem, a scorer, a first rule."

We built exactly that, for one job, and ran it overnight.

What we wrote

The problem: describe one screen from its screenshot and the crawler's record of it.

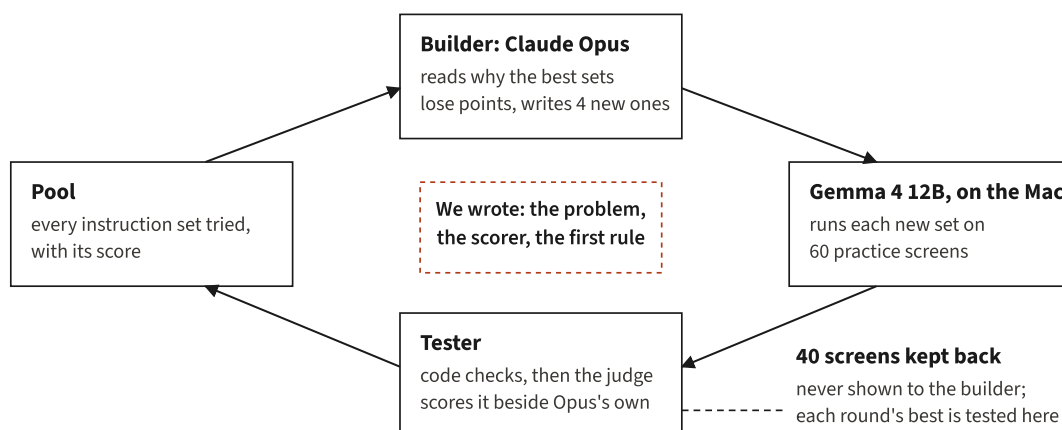
The scorer: code checks (is every recorded field and button there?), then a judge, Claude Sonnet, scores out of 10 against the screenshot: right purpose, nothing missed, nothing invented, readable. It doesn't know whose description is whose.

The first rule: our hand-tightened instructions.

What "self-improving" means here

One AI rewrote the other model's instructions, round after round. **Gemma itself never changed.** Around it we wrote a handful of short scripts, and that's all.

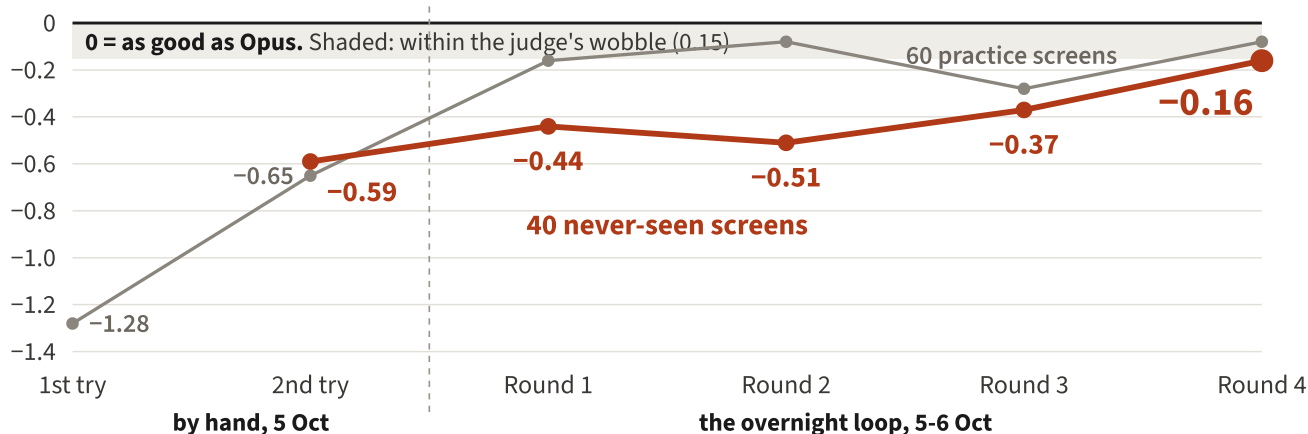
What we set up: the talk's loop, for real



The loop: four new instruction sets a round. It stops after two rounds with no gain beyond the judge's own wobble,

The result: how far behind the top model, try by try

How far behind Opus, in points out of 10, judged side by side. Each point is the best instruction set of that try or round.



Why a gap and not a score: the judge gave Opus's same, unchanged descriptions anything from 9.45 to 9.93 from run to run, so we judge both side by side and count the difference. **Round 2 got worse before 3 and 4 got better.** On practice screens it was almost level with Opus, but it had learned those screens' own mistakes, and the 40 held back showed it. The loop also docks points for what the code checks catch, so by its count the second try starts at 0.81 behind, not 0.65.

₹0

for the small model's work on the Mac's chip. The builder and judge ran on our Claude plan.

8 hours

of the small model's time, Monday evening to about 3:30 on Tuesday morning

18

instruction sets tried over four rounds

1 in 40

new screens still described wrongly, the same as before the loop

What we learned

Facts from code beat imitation. The changes that helped most handed Gemma what it couldn't work out for itself. The crawler's record of whether a click had opened a pop-up took the gap from 0.51 to 0.37, and Gemma's own score rose with it. The text the Mac's built-in reader (OCR) takes off the screenshot stopped misreads like ₹23,51,250 as ₹23,51,501, and the gap reached 0.16, though in that run most of the step came from the judge scoring Opus lower.

Copying the big model made it invent. Asked to name icons by shape as Opus does, Gemma called a key icon a search icon; asked to explain every button, it made up what they do. None of six such habits helped.

It does what the words say. Our first rule said "Never guess; say 'unclear'". Gemma wrote "Unclear: unclear" on 38 of 60 screens. A code check that sends it back once closed more than a third of the gap on the practice screens.

Keep some screens back. Without the 40 the builder never saw, we would have believed round 2.

What this doesn't show

One task, one small model, 100 screens from our own software on test servers with sample data.

The judge is an AI too. Differences under about 0.15 are noise; 0.16 is only just clear of it. **On never-seen screens Gemma's own score hardly moved**, 9.35 to 9.30. Opus's moved, 9.93 to 9.45, and in that last run the judge marked 2 of Opus's 40 wrong over a rule not on the screen. Side by side is the fair comparison, but this is one job, 100 screens, and an AI judge, so treat it as a strong early result, not proof. On practice screens Gemma's own score did rise, 9.18 to 9.63.

We set ourselves 9.5 with at most 2% wrong. We didn't reach it: the one wrong screen left was a list Gemma titled after a small alert box on top of it.

Try it yourself

1. **A job you repeat often**, and a strong model's answers to compare against.
2. **A scorer.** Code for what can be counted, a judge for the rest. Run the judge twice on the same answers first, to see how much it wobbles.
3. **A held-back set** the builder never sees.
4. **A first instruction**, the best you can write by hand. Give the model every fact code can pull out for it; for us that was the text on the screenshot.
5. **The loop, with a stop rule.** Ours ran on a desk Mac with 24 GB of memory, overnight.