SkillOpt-Lite *and* HarnessOpt — Better and Faster Agent Self-evolution *with One Line of Vibe*

Yifei Shen **Bo Li**[1,2] **Xinjie Zhang**[3] [1]LMMs-Lab [2]NTU MMLab [3]Microsoft

GitHub `github.com/EvolvingLMMs-Lab/SkillOpt-Lite`

Code & Paper

WeChat Group
Scan to join us

TL;DR Treat rollout trajectories as **flat files** and let a coding agent debug your agent's **skills**—and its **harness**—with **one slash command**. This minimal-viable pipeline is **simpler, faster, and stronger** than heavily engineered baselines.

1 Motivation

An agent's real capability is set by **base model** × **harness** × **skills**. Since the model stays **frozen**, agent engineering reduces to editing skill documents.

Yet skill-optimization pipelines keep growing in **architectural complexity**—tree merging, textual learning-rate schedules, rejection buffers—without asking:

> *What is the **minimal viable pipeline** where every component is justified by **theory or empirical necessity**?*

2 Skill Opt ↔ Zeroth-Order Optimization

The reward **f(s)=$\mathbb{E}_z$[R(H(M,z,s))]** has an intractable gradient. We show a tight **connection** to zeroth-order (ZO) optimization—yet a fundamental **divergence**.

Connection | ⚡ Divergence

ZO operator	Agent realization
1-point estimate	Single-trace reflection (Reflexion)
Mini-batch	Batch consensus (SkillOpt, Trace2Skill)
Central diff.	Success–failure contrast (SkillCat)
Coord. descent	Fault-isolated edit (SkillAdapter)
Trust region	Bounded edit budget (SkillForge)
Control variate	Rejected-edit buffer (SkillOpt)

The divergence. Classical ZO perturbs *blindly* over unobservable states; agentic rollouts emit **readable traces** → skill optimization is **language-mediated program compilation**, with trajectories as debug feedback.

3 Three Design Principles

PAC 1 **Consensus mining.** Overfitting single-trial anomalies inflates the stability coefficient β_{exp}; compress cross-task invariants instead.

PAC 2 **Independent validation gating.** A disjoint held-out set *removes* β_{exp} from the generalization bound.

Bitter **A bitter lesson.** As base LLMs scale, give the agent **primitive file tools over raw logs**—not bespoke damping topologies.

4 PAC-Learning Perspective

Skill optimization is still a statistical learning problem. Via **algorithmic stability**, the generalization error of a learned skill library 𝕊 obeys:

Training bound (anchored to stability)

$$\varepsilon(S) \le \hat{\varepsilon}_D(S) + \mathcal{O}\left(\beta_{\text{exp}} + \sqrt{\frac{\ln(1/\delta)}{N}}\right)$$

β_{exp} is the expected on-average stability: hardcoding a single failed trace inflates it and collapses generalization. **Consensus mining** across trajectories shrinks β_{exp} (Principle 1).

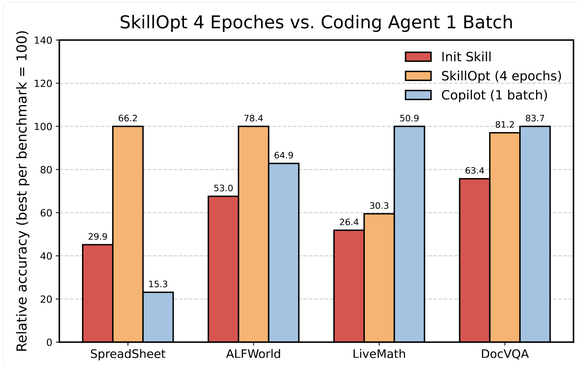
Validation bound (β_{exp} eliminated)

$$\varepsilon(S_{\text{val}}) \le \hat{\varepsilon}_{\text{val}}(S_{\text{val}}) + \mathcal{O}\left(\sqrt{\frac{\ln(1/\delta)}{m}}\right)$$

Why gating works. A *disjoint* validation set of size **m** removes β_{exp} from the bound entirely—so gates must never reuse training failures (Principle 2).

5 Pilot: Coding Agents Write Skills

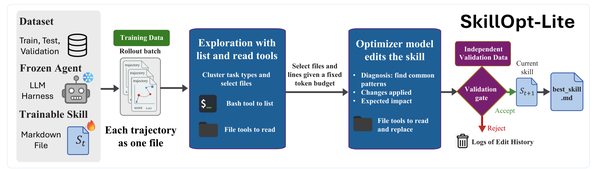
Stage one batch of raw GPT-5.4-nano trajectories as files; let a coding agent explore the directory with **only file-system tools** and patch the skill **—no validation loop**.



Bitter-lesson evidence. A single-batch, unvalidated exploration **matches or beats 4-epoch SkillOpt** on LiveMath & DocVQA—but degrades on Spreadsheet, motivating the closed-loop **validation gate**.

6 SkillOpt-Lite Pipeline

A minimal loop **directly on disk**. We remove mini-batch pooling, slow-update damping, and rejection buffers—each trajectory is an **independent file**.

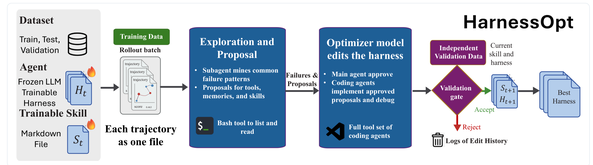


1. **Staging.** Dump each rollout (plans, states, score) as a text file.
2. **Exploration.** `list`/`read` to cluster failures under a token budget.
3. **Consensus + minimal edit.** Diagnose shared patterns; emit a compact diff.
4. **Validation gating.** Score on an *independent* set; overwrite `best_skill.md` on gain.

```
/skillopt-loop rounds=10 batchsize=40 target=gpt5.4-nano
```

7 HarnessOpt — Everything is a File

Because all artifacts are flat files, the **harness reduces to editable code**. Lifting path restrictions lets the same loop refine the **execution scaffolding**.

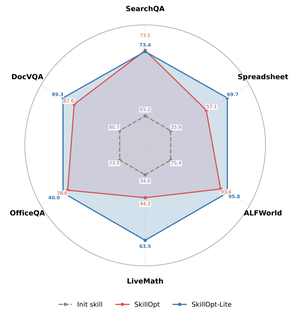


- **Round-0 human gate:** a subagent proposes tool / memory / control-flow changes for approval in VS Code.
- **Sandboxed evolution:** compile + smoke test (N=5) before full validation; `git reset`-reversible behind toggles.
- **Allowlist:** only scaffolding is mutable; evaluators & configs stay read-only.

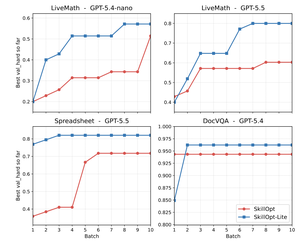
```
/harnessopt-loop rounds=2 target=gpt5.4-nano
skill=best_skill_nano.md
```

8 Results — Skill Optimization

Across **6 benchmarks** and 5 model scales, SkillOpt-Lite matches or beats the fully-engineered SkillOpt baseline.



Frontier expands on every axis (avg over scales).



Faster convergence: SkillOpt-Lite (blue) vs SkillOpt (red).

+37.0 LiveMath, GPT-5.5 (36.6→73.6)

79.4 Spreadsheet 5.4 vs 61.5 SkillOpt

55.7 nano > full GPT-5.4 SkillOpt (54.0)

+12.6 avg Spreadsheet gain (69.7 vs 57.1)

9 HarnessOpt — SpreadsheetBench

Method	nano	mini	5.4	5.5
Init skill	.299	.282	.399	.374
SkillOpt	.516	.475	.615	.762
SkillOpt-Lite	.662	.733	.794	.797
HarnessOpt w/o skill	.765	.811	.836	.836
HarnessOpt w/ skill	**.776**	**.826**	**.851**	**.858**

Capability inversion. Lightweight **nano reaches 0.776**, surpassing GPT-5.5 under a standard harness + full SkillOpt (0.762).

10 Takeaways

- A **minimal, file-centric** pipeline beats engineered baselines—and converges faster.
- The same loop generalizes **skills → harness**; small models overtake frontier ones.
- Shipped as a **VS Code extension**: evolve your agent with one line of vibe.

GitHub & Paper

Join WeChat Group

Scan to get started

github.com/EvolvingLMMs-Lab/SkillOpt-Lite · open-source · six benchmarks · VS Code extension

Correspondence: `yshenaw@connect.ust.hk`

Presented at
ICML 2026
LMMs-Lab · NTU · Microsoft