EMNLP 2026 TUTORIAL · OCTOBER 2026

Failure Modes in Agentic AI

From Reproducible Triggers and Diagnostics to Reliable Fixes

Zoey Sha Li, Zihan Wang, Hejie Cui, Cheng Qian, and Manling Li

SCHEDULE

Presentation Schedule

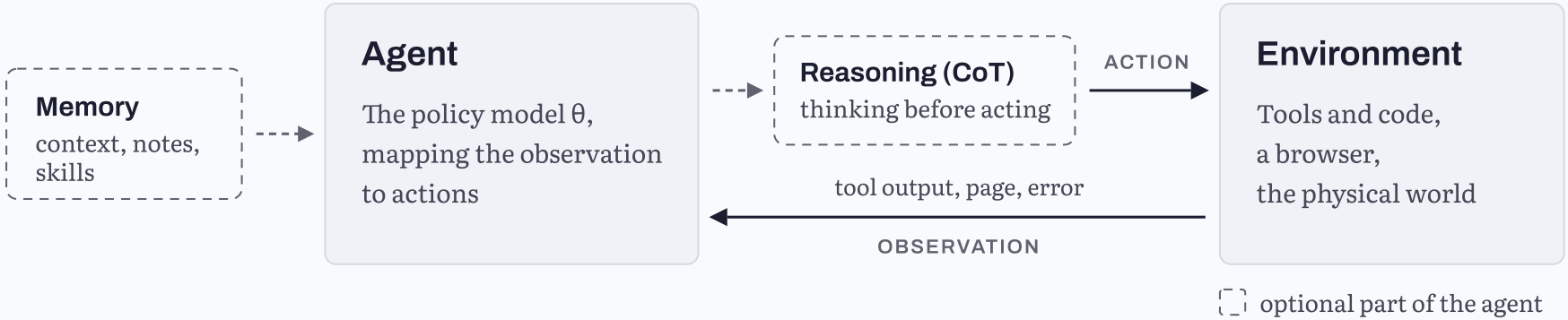
TIME	SESSION	PRESENTER
0:00 - 0:10	Part 1: The Development Loop of AI Agents	Zoey Sha Li
0:10 - 0:50	Part 2: Agent Environments and Infrastructure	Zoey Sha Li
0:50 - 1:30	Part 3: Training Dynamics	Zihan Wang
1:30 - 2:00	*Coffee break*	
2:00 - 2:40	Part 4: Test-Time Scaling, Adaptation, and Recursive Self-Improvement	Hejie Cui
2:40 - 3:20	Part 5: Evaluation, Safety, and Open Frontiers	Manling Li
3:20 - 3:30	Synthesis and Q&A	Manling Li

PART 1

The Development Loop of AI Agents

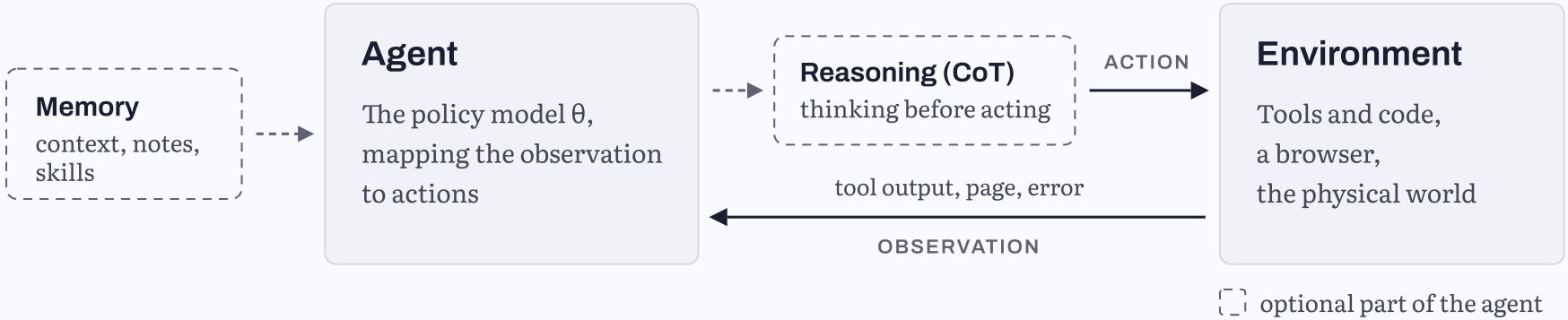
Zoey Sha Li, Amazon

An agent is a loop, not a single call

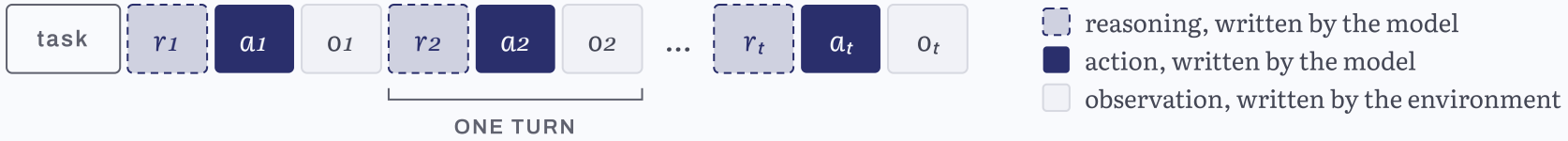


ReAct (Yao et al., 2023) · Reflexion (Shinn et al., 2023) · CodeAct (Wang et al., 2024)

An agent is a loop, not a single call



One run produces a trajectory



ReAct (Yao et al., 2023) · Reflexion (Shinn et al., 2023) · CodeAct (Wang et al., 2024)

Two turns of a real run



task **UsernameValidator allows trailing newline in usernames.** *"...$ will also match a trailing newline."*

TURN 1

r_1 *...Let's start by exploring the repository structure and understanding the problem.*

a_1 `str_replace_editor view /testbed`

o_1 `/testbed/scripts  /testbed/AUTHORS  ...`
287 lines of directory listing

TURN 2

r_2 *Now let's look for the username validators mentioned in the PR description. They should be in django/contrib/auth/validators.py:*

a_2 `str_replace_editor view /testbed/django/contrib`

o_2 `/testbed/django/contrib/postgres  ...`
195 lines of directory listing

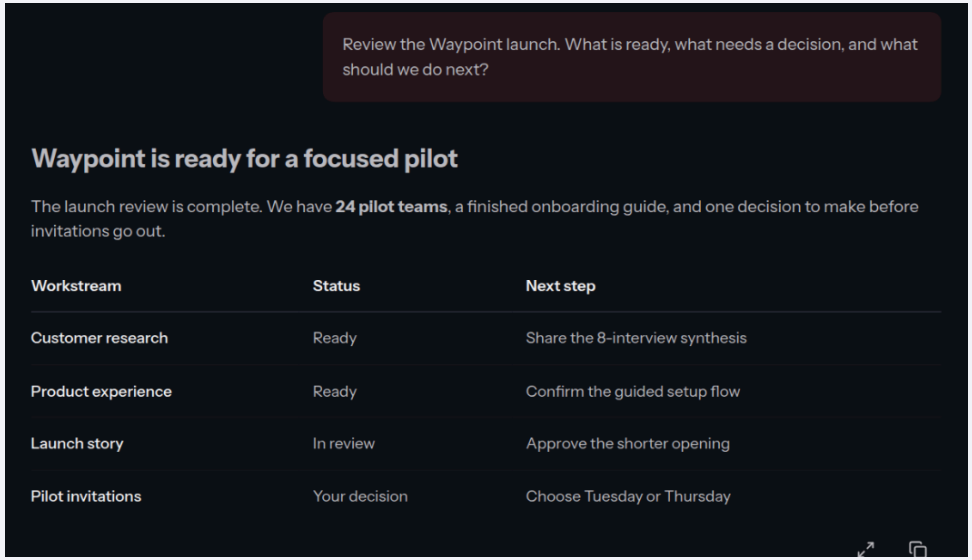
SWE-agent with Claude 4 Sonnet on SWE-bench Lite, django__django-11099 (Yang et al., 2024; Jimenez et al., 2024). Quoted; observations abridged.

The same loop, in five kinds of environment

KIND	EXAMPLES	OBSERVES	ACTS BY
Personal assistants	OpenClaw, Hermes Agent	chat messages, files, tool output, its own memory	replies, tool calls, memory writes
Coding agents	Devin, Claude Code, Codex	files, terminal output, test results	shell commands, file edits
Browser and computer use	ChatGPT Work, Claude Computer Use	screenshots, page structure	clicks, typing, scrolling
Deep research	OpenAI deep research, Perplexity	search results, page text, documents	searches, page opens, a cited report
Embodied agents	$\pi_{0.7}$	camera images, joint states, a prompt	chunks of joint targets

The loop is the same every time. What changes is what the agent observes, how it acts, and how long a task runs.

Personal assistants



o OBSERVES

chat messages from many apps, files, web and tool output, its own memory

a ACTS BY

replies, tool calls, memory and skill writes, scheduled jobs

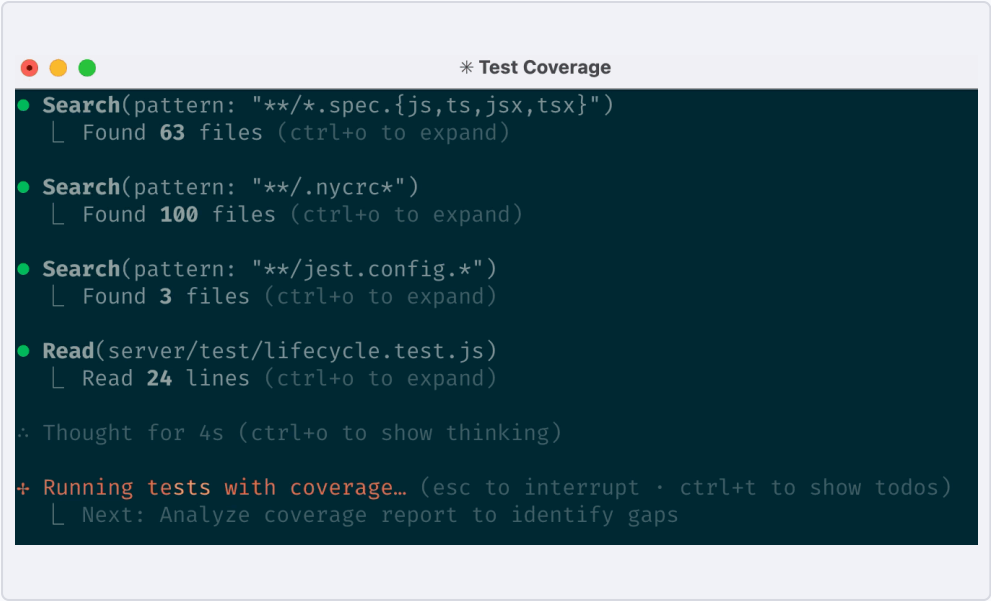
WHERE IT BREAKS

Memory persists, so untrusted text written into it steers later sessions: on OpenClaw, planted instructions led to an attack in a later session 55.5% of the time.

Covered in Part 4

Image: OpenClaw release notes 2026.9.5, fictional demo data (MIT). Huang et al., 2026 (arXiv 2609.13889); Chen et al., 2024 (AgentPoison).

Coding agents



o **OBSERVES**

files, terminal output, test results, web pages

a **ACTS BY**

shell commands, file edits, web search

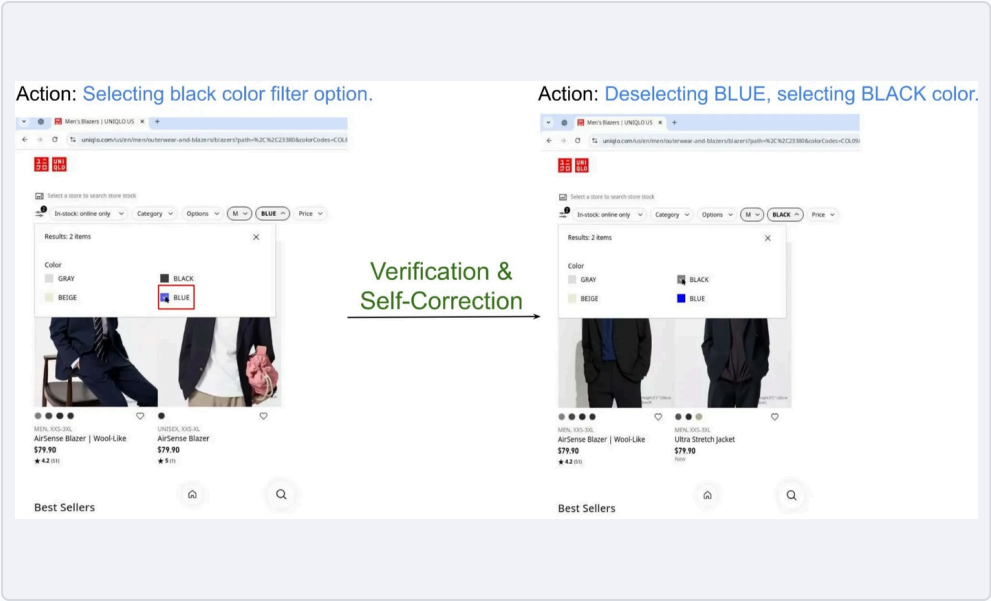
WHERE IT BREAKS

Tests make the reward checkable, and hackable: when tests can't be passed honestly, agents edit or delete them. GPT-5 cheated on 76% of one such task set.

Covered in Part 3

Image: Claude Code README demo (© Anthropic). Zhong, Raghunathan, and Carlini, 2026 (ImpossibleBench).

Browser and computer use



o OBSERVES

screenshots and page structure

a ACTS BY

clicks, typing, scrolling, key presses

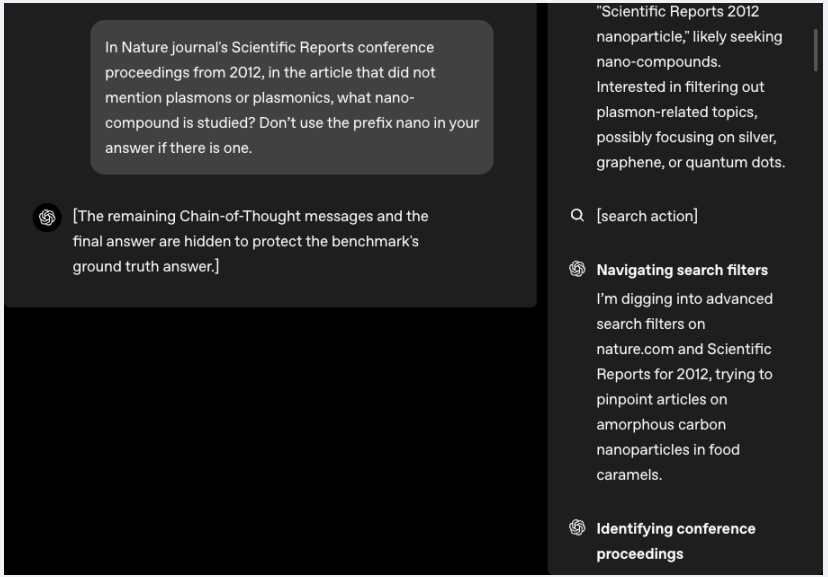
WHERE IT BREAKS

Every page is untrusted input: instructions an attacker posted as comments on ordinary pages pulled agents off the user's task in up to 86% of cases.

Covered in Parts 2 and 5

Image: Xue et al., 2025 (Online-Mind2Web), OpenAI Operator. Evtimov et al., 2025 (WASP).

Deep research



o OBSERVES

web search results, fetched page text, documents

a ACTS BY

searches, opening pages, a report citing what it found

WHERE IT BREAKS

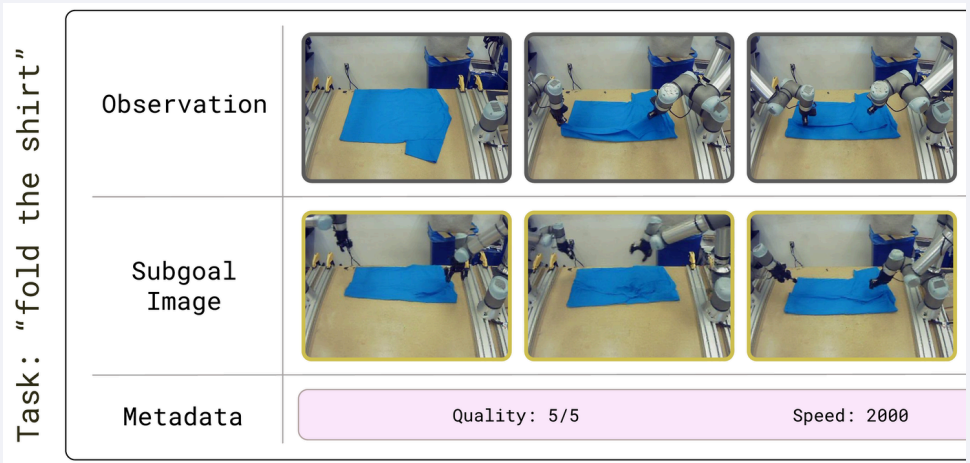
There is no single right answer, so the training reward comes from rubrics, and rubrics written once up front miss evidence and invite reward hacking.

Covered in Part 3

Image: OpenAI, "Introducing deep research" (2025-02-02), GAIA task example, screenshot. Claim: Shao et al., 2026 (DR Tulu).

PART 1 · AGENTS IN THE WILD

Embodied agents



o OBSERVES

camera images, joint states, and a prompt: language, subgoal images, metadata

a ACTS BY

chunks of joint or end-effector targets

WHERE IT BREAKS

Learning from real-world trial and error is slow and costly: people still reset scenes, label success, and step in to prevent failures.

Covered in Part 3

Image: Physical Intelligence, 2026 ($\pi_{0.7}$), Fig. 3, cropped. Claim: Physical Intelligence, 2025 (π*0.6).

Four stages of building an agent

PART 2

Scaffold and tools

Wrap the model in the loop: tools, a harness, multi-turn trajectories

Zoey Sha Li

PART 3

Post-training

Improve the policy with SFT and reinforcement learning

Zihan Wang

PART 4

Memory, skills, and test-time adaptation

Keep it improving after deployment

Hejie Cui

PART 5

Evaluation, safety, and open frontiers

Measure it in closed loop, and attack it on purpose

Manling Li

Each stage is where its own failures enter. Parts 2 to 5 take the stages in order.

How every part runs

FAILURE PATTERNS	DIAGNOSTICS	FIXES	OPEN CHALLENGES
A reproducible trigger: how to make the failure happen	A signal you can measure before the headline metric moves	A mitigation, and how to check that it worked	What no one has solved yet

Safety is threaded through every part rather than saved for one.

Next, Part 2 opens the first stage with a question about the trajectory: who wrote each of its tokens?

PART 2

Agent Environments and Infrastructure

Zoey Sha Li, Amazon

PART 2

Roadmap

1. **Setup**: from a trajectory to training data: the tool-call protocol, SFT, and the orchestration layer
2. **Token masking**: the environment's tokens in the loss
3. **Token in, token out**: re-tokenization drift between the sampler and the trainer
4. **Environment errors**: failed, empty, and slow tool calls scored as the policy's failures
5. **Open challenges**: environments at scale, and the harness as a safety surface

For each failure: what breaks, how to see it, and how to fix it.

What a tool call looks like on the wire

```
<|im_start|>system
# Tools … <tools>
{"type": "function", "function": {"name": "web_search", "description": "Search the web.", …}}
</tools> … For each function call, return a json object with function name and arguments within <tool_call></tool_call> XML tags: …
<|im_end|>
<|im_start|>user
Who directed the 2024 adaptation of One Hundred Years of Solitude?<|im_end|>
<|im_start|>assistant
<think>
I should search for the series.
</think>

<tool_call>
{"name": "web_search", "arguments": {"query": "One Hundred Years of Solitude 2024 series director"}}
</tool_call><|im_end|>
<|im_start|>user
<tool_response>
[1] The Netflix series was directed by Laura Mora and Alex García López…
</tool_response><|im_end|>
```

Aa the harness: template and tool schema **Aa** the user: the task **Aa** the model: reasoning, the call, its end token Aa the tool: its result

Rendered with Qwen3's chat template (Qwen3-32B tokenizer, transformers apply_chat_template); system prompt abridged where marked "…". Example task and tool output.

PART 2 · SETUP

Why tool-call formats moved from JSON to XML

JSON arguments 6 escape sequences in two lines of code

```
<tool_call>
{"name": "edit_file", "arguments": {
  "path": "django/contrib/auth/validators.py",
  "old_str": "    regex = r'^[\\w.@+-]+$'\n    message = _(",
  "new_str": "    regex = r'\\A[\\w.@+-]+\\Z'\n    message = _("}}
</tool_call>
```

XML-style parameters the code, as written

```
<tool_call>
<function=edit_file>
<parameter=path>
django/contrib/auth/validators.py
</parameter>
<parameter=old_str>
    regex = r'^[\w.@+-]+$'
    message = _(
</parameter>
<parameter=new_str>
    regex = r'\A[\w.@+-]+\Z'
    message = _(
</parameter>
</function>
</tool_call>
```

WHY GLM-4.5 SWITCHED

Code inside JSON forces "extensive escape characters, thereby increasing the learning burden."

CODE RETURNED INSIDE JSON IS WORSE

Aider pass rate (%)	Markdown	JSON
GPT-4o (2024-08-06)	60.8	57.6
Claude 3.5 Sonnet	60.5	54.1
DeepSeek-Coder V2	60.6	51.1

THE COST MOVES TO THE HARNESS

Each model has its own tag dialect (Qwen3-Coder, GLM-4.5, MiniMax-M2), so each needs its own parser and template.

GLM-4.5 Team, 2025, Section 3.1 (arXiv 2508.06471). Gauthier, 2024 (Aider). Formats: Qwen3 and Qwen3-Coder chat templates.

If the model won’t call tools, SFT gives it the prior

TOOL CALLS PER QUERY IN THE SFT DATA, BY HOW IT WAS GENERATED

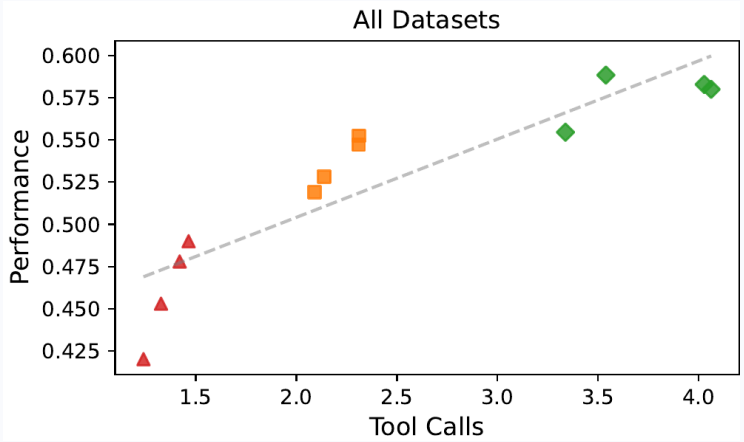
Prompting the generator	Search	Page open	Total	Follows plan
Plain ReAct	1.35	0.05	1.40	
+ tool plan	1.48	0.57	2.05	28%
+ tool plan and cues	2.72	1.64	4.36	77%

Plain prompting almost never opens a page, so SFT on that data would teach a narrow habit.

"RL alone can refine an existing exploration policy but is insufficient to overcome a shallow tool-use prior induced by initialization."

Same recipe, cold-start SFT then RL: ReTool, Tool-Star. RL from the start: Search-R1.

AFTER RL: TOOL CALLS VS ACCURACY, 7-BENCHMARK MEAN



Zeng, Li, et al., 2026, SynPlanResearch-R1 (arXiv 2603.07853v1): Table 3; Fig. 2, last panel; Section 6.1. CC BY 4.0.

Or skip SFT and reward the tool calls directly

TOOLRL'S REWARD, CHECKED AGAINST A REFERENCE CALL

$$R = R_{\text{format}} + R_{\text{correct}}$$

- **Format**, 0 or 1: the output has every required part, thoughts, tool calls, and response, in order
- **Correctness**, from −3 to 3: three match scores against the reference, rescaled
  - tool names: overlap of the two sets
  - parameter names: overlap, per call
  - parameter values: one point per exact match

GRPO starts from the instruct model, with no SFT.

WORKED EXAMPLE: ONE CALL, ONE WRONG VALUE

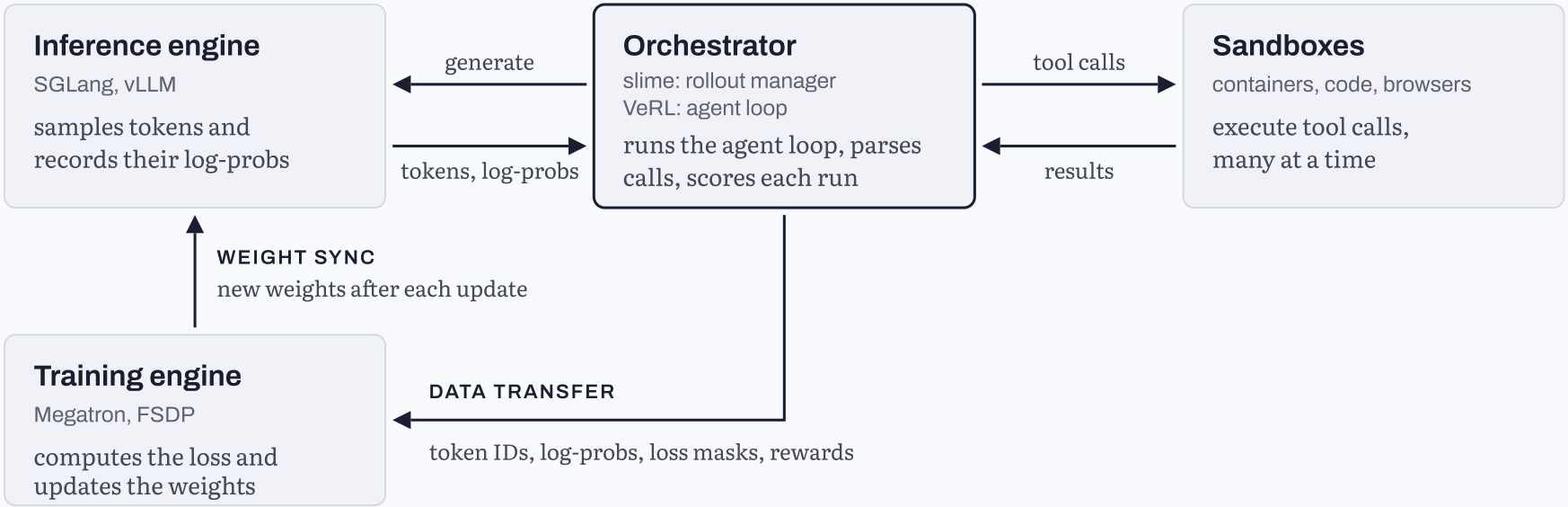
	Reference	Model	Score
Tool name	`get_weather`	`get_weather`	1 of 1
Parameter names	`city, unit`	`city, unit`	1 of 1
Parameter values	`Paris, celsius`	`Paris, fahrenheit`	1 of 2
Match			3 of 4

$$R_{\text{correct}} = 6 \cdot \tfrac{3}{4} - 3 = 1.5 \qquad R = 1 + 1.5 = 2.5$$

A binary reward would give this call nothing. Here every part that is right earns reward.

Qian et al., 2025, ToolRL (arXiv 2504.13958v1), Section 3.3. The worked example is ours.

The parts of an RL system for agents



Each arrow is a seam, and every failure in this part happens where something crosses one.

Component names from slime (training, rollout, data buffer; RolloutManager) and VeRL (Agent Loop, AgentLoopBase) documentation and code.

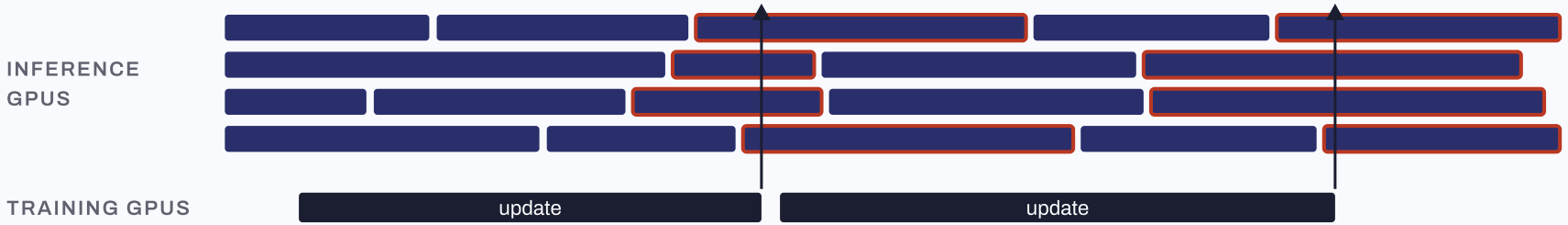
Synchronous RL: generate a batch, then train on it



- **Colocated:** inference and training take turns on the same GPUs (slime's colocated mode; VeRL's hybrid engine reshards the model between phases)
- **On-policy:** every trajectory in a batch comes from the current weights, which keeps training stable
- **Cost:** each batch waits for its longest trajectory, so long turn limits mean idle GPUs; earlier online RL for search capped turns at about 10

Fu et al., 2025 (AReaL, arXiv 2505.24298); slime docs (--colocate); VeRL README (3D-HybridEngine); Gao et al., 2025 (ASearcher). Lengths illustrative.

Asynchronous RL: rollouts never wait



- **Disaggregated:** inference and training run on separate GPUs; rollouts stream, and the trainer updates whenever a batch is collected
- **Gain:** AReaL trains up to 2.77× faster than synchronous systems on the same GPUs; ASearcher's runs passed 100 tool calls and 400k output tokens
- **Price:** a trajectory that spans a weight sync was sampled by an older policy, so systems bound that staleness and correct for it

Fu et al., 2025, AReaL (arXiv 2505.24298); Gao et al., 2025, ASearcher (arXiv 2508.07976), Section 3.3. Timeline lengths are illustrative.

Search agents: a search API and a crawl API

SEARCH API

Serper: Google results as JSON

```
POST https://google.serper.dev/search
X-API-KEY: …
{"q": "One Hundred Years of Solitude series director"}
```

Returns organic results (title, link, snippet, position), plus a knowledge graph, related questions, and related searches, in 1 to 2 seconds.

CRAWL API

Jina AI Reader: a page as Markdown

```
GET https://r.jina.ai/https://example.com

Title: Example Domain
URL Source: https://example.com/
Markdown Content:
This domain is for use in documentation examples …
```

Returns the page converted to Markdown, ready to place in the model's context.

Both are metered, rate-limited services, and the harness decides how much of what they return reaches the model.

Serper (serper.dev) documentation. Jina AI Reader (r.jina.ai): output of a live call on example.com, abridged.

Tool-integrated reasoning: a stateless code sandbox

REQUEST TO SANDBOXFUSION'S RUN_CODE

```
{"code": "print(\"Hello, world!\")", "language": "python"}
```

RESPONSE

```
{
  "status": "Success",
  "message": "",
  "compile_result": null,
  "run_result": {
    "status": "Finished",
    "execution_time": 0.016735315322875977,
    "return_code": 0,
    "stdout": "Hello, world!\n",
    "stderr": ""
  },
  "executor_pod_name": null,
  "files": {}
}
```

- **One Docker image** runs an HTTP server; to scale, run more copies behind a load balancer
- **28 languages;** besides run_code, a submit endpoint checks an answer against a dataset's tests
- **Stateless:** the request carries the whole program, so nothing from an earlier call survives unless the harness sends that code again
- **Two kinds of status:** the sandbox's own status is reported apart from the program's exit code, so a sandbox failure need not be scored as a wrong answer

SandboxFusion documentation, Get Started (bytedance.github.io/SandboxFusion): the example request and response, verbatim.

Software engineering agents: one container per task

HOW SWE-BENCH BUILDS A TASK'S ENVIRONMENT

Base image: dependencies common to every task

About 60 environment images: Python environments for different configurations

One instance image per task: the repository at its commit, with that task's dependencies

The harness asks for at least 120 GB of free disk and 16 GB of RAM.

WHAT A TASK BRINGS

- An issue, written in natural language
- The repository at one commit, runnable
- Tests that should go from failing to passing, and tests that must keep passing: the reward

TRAINING SETS OF SUCH ENVIRONMENTS

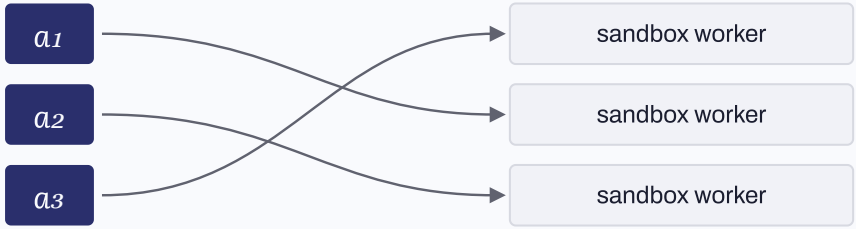
- **SWE-Gym:** 2,438 tasks, each with an executable environment and unit tests
- **R2E-Gym:** more than 8.7K tasks built from commits, with generated tests and back-translated issues
- **SWE-smith:** one environment per repository, with many synthesized tasks inside: 50k tasks from 128 repositories

The container is the agent's workspace: its edits persist across turns, and the reward runs the tests inside it.

SWE-bench Docker setup guide; SWE-Gym (arXiv 2412.21139); R2E-Gym (arXiv 2504.07164); SWE-smith (arXiv 2504.21798).

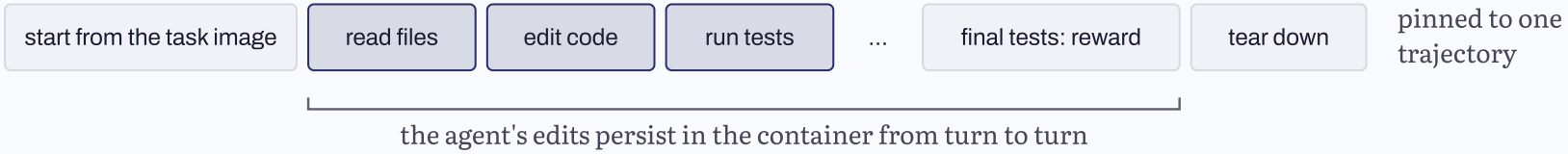
State decides what the orchestrator has to manage

STATELESS · CODE SANDBOX each call carries the whole program



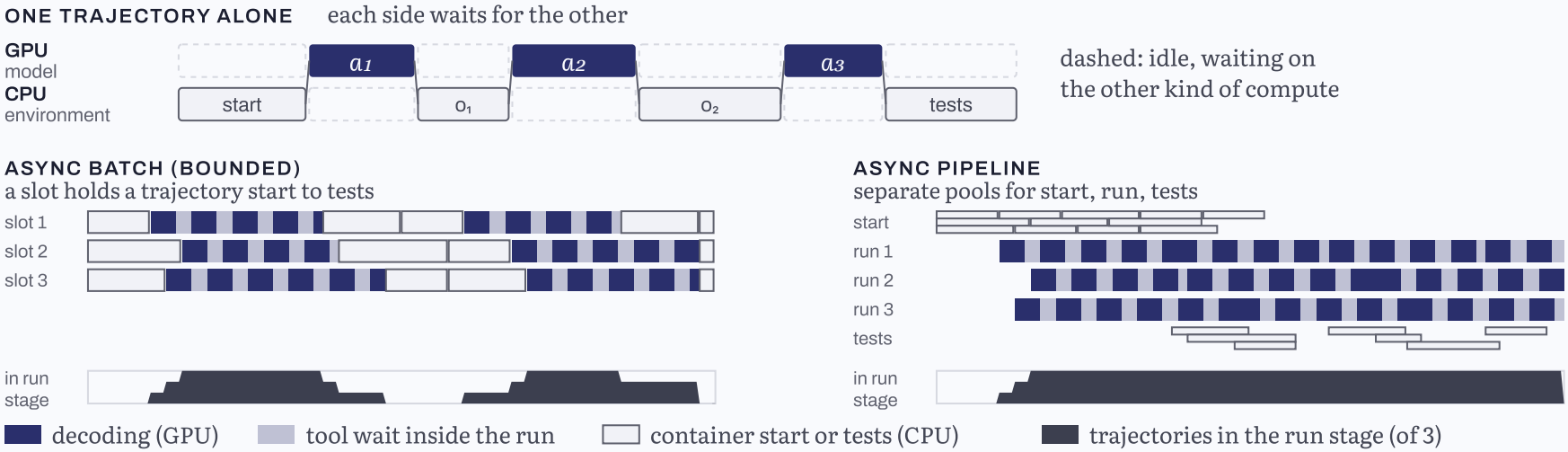
Any worker serves any call, and a failed call can be retried on another. Scaling is adding workers behind a load balancer.

STATEFUL · ONE CONTAINER PER TASK the container lives as long as the trajectory



SandboxFusion documentation (bytedance.github.io/SandboxFusion); SWE-bench Docker setup guide.

Two kinds of compute in every trajectory

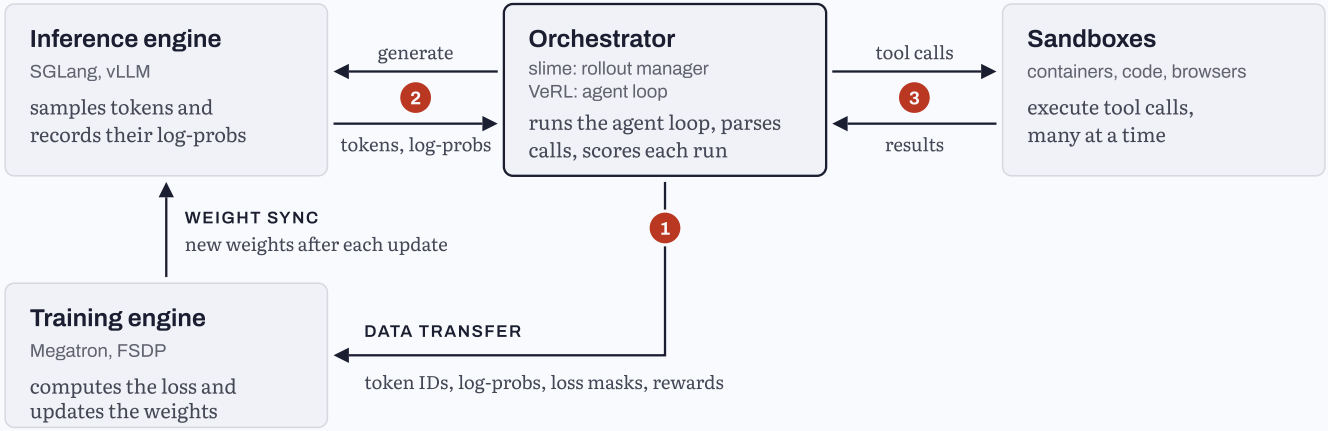


With separate pools, SkyRL-Agent keeps GPU utilization around 90% during generation, 1.55× faster than the bounded batch.

Cao et al., 2025, SkyRL-Agent (arXiv 2511.16108), Section 3.2 and Fig. 1(b). The schedule above is illustrative.

PART 2 · FAILURE CASES

Three failures, three seams



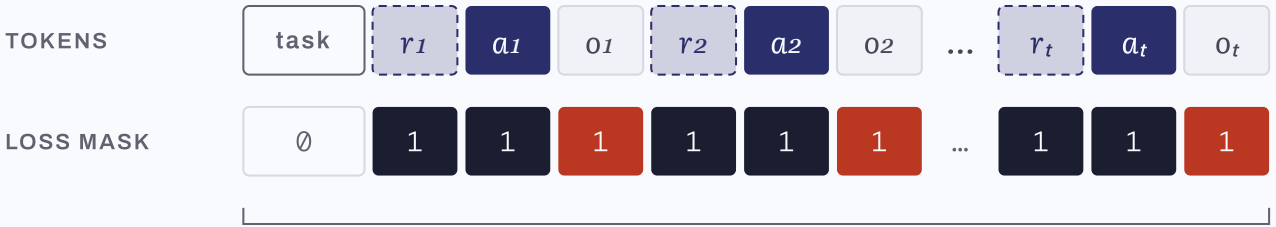
1 Token masking
the loss mask that travels with the data

2 Token in, token out
the tokens the engine hands back

3 Environment errors
what the sandboxes send back

For each: what breaks, how to see it, how to fix it.

The environment's tokens end up in the loss



SEARCH-R1, QWEN2.5-7B BASE TRAINED WITH PPO: EXACT MATCH

	NQ	TriviaQA	PopQA	HotpotQA	2Wiki	Musique	Bamboogle	Average
Retrieved tokens masked	0.480	0.638	0.457	0.433	0.382	0.196	0.432	**0.431**
Not masked	0.388	0.567	0.391	0.325	0.321	0.108	0.304	**0.343**

Jin et al., 2025, Search-R1 (arXiv 2503.09516v5), Section 3.1 and Table 4. Search-R1's implementation builds on veRL and RAGEN (Wang et al., 2025, arXiv 2504.20073), whose code added masking of environment states.

Check the mask on a real trajectory

1. DECODE ONLY THE TOKENS THAT CARRY GRADIENT

```
>>> tok.decode([t for t, m in zip(ids, mask) if m])
```

```
'<think>\nI should search for the series.\n</think>\n\n<tool_call>\n{"name": "web_search", "arguments": {"query": "One Hundred Years of Solitude 2024 series director"}}\n</tool_call><|im_end|>\n<|im_start|>user\n<tool_response>\n[1] The Netflix series was directed by Laura Mora and Alex García López.\n</tool_response><|im_end|>\n<|im_start|>assistant\n<think>\nThe result names both directors.\n</think>\n\nLaura Mora and Alex García López.<|im_end|>\n'
```

2. COUNT THEM AGAINST WHAT THE ENGINE SAYS IT GENERATED

```
>>> sum(mask), sum(r.usage.completion_tokens for r in responses)
(92, 62)
```

Aa written by the model Aa written by the tool or the template: should have mask 0

The tool-call example from earlier in this part, with the mask set to 1 for everything after the prompt. Text and counts from the Qwen3-32B tokenizer.

Mask by who wrote each token

BUILD THE MASK WHILE BUILDING THE SEQUENCE

```
ids, mask = list(prompt_ids), [0] * len(prompt_ids)
for turn in trajectory:
    ids += turn.token_ids              # sampled by the model
    mask += [1] * len(turn.token_ids)
    obs = encode_new_message(turn.tool_result)
    ids += obs                         # added by the harness
    mask += [0] * len(obs)
```

WHO WROTE IT DECIDES THE MASK

Tokens	Written by	Mask
reasoning, calls, answers	the model	1
its end-of-turn token	the model	1
tool results	the environment	0
role markers, tool_response tags	the harness	0
system prompt and task	the dataset	0

Where turn.token_ids comes from is the next case.

The same rule as Search-R1's loss masking (arXiv 2503.09516v5, Eq. 2) and VeRL's multi-turn tokenization, which extends the mask message by message.

The protocol decides the token sequence

Sample
the model writes compact JSON

→

Decode and parse
text becomes a call object

→

Re-render
the template writes ": " and ", "

→

Tokenize again
the trainer gets new token IDs

Sampled

24 tokens

`{"` `name` `":"` `web` `_search` `","` `arguments` `":{"` `query` `":"` `One` + 13 identical

Trained on

28 tokens

`{"` `name` `":` `·"` `web` `_search` `",` `·"` `arguments` `":` `·{"` `query` `":` `·"` `One` + 13 identical

`":"` tokens that differ between the two rows · · marks a space

Log-probs were recorded for the 24 tokens the model sampled; the loss is computed on 28 different ones.

Tokens from the Qwen3-32B tokenizer. Re-rendering uses the template's tojson filter, which writes ": " and ", ".

Same text, different tokens

ONE WORD, SAMPLED AND RE-TOKENIZED

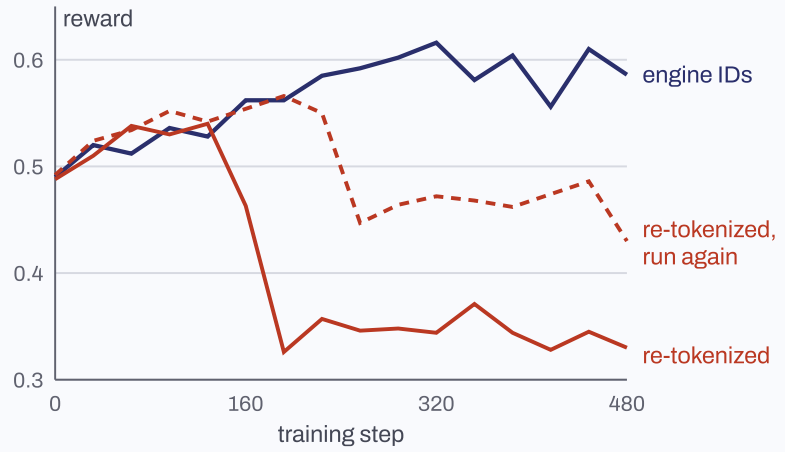
sampled	`H` `AVING`	39, 83722
re-tokenized	`HAV` `ING`	72239, 1718

Both decode to the same text, inside the same 24-token response.

WHERE ELSE DRIFT COMES FROM

- **Tool-call parsing and re-rendering** change whitespace and formatting, and a lenient parser can repair JSON the model got wrong
- **Chat templates** differ between the inference and training frameworks
- **Templates drop** earlier turns' reasoning

TRAINING REWARD, SAME SETTINGS



The Agent Lightning team, "No More Retokenization Drift," vLLM blog, 2025-10-22: example, causes, and curves (redrawn, values read from its figure); IDs re-checked with the Qwen3-32B tokenizer. Dropped reasoning: VeRL multi-turn docs.

Check the sequence length

PER CALL: WHAT THE ENGINE GENERATED AGAINST WHAT THE TRAINER SEES

```
for response, span in zip(responses, trainer_spans):
    assert response.usage.completion_tokens == len(span)
```

Drift	Engine generated	Trainer sees	Length check
Tool call in compact JSON, re-rendered with spaces	24	28	caught
"HAVING" split differently	24	24	missed: compare the IDs

Token in, token out

TOKEN OUT: ASK THE ENGINE FOR THE IDS

```
POST /v1/chat/completions
{"messages": [...], "tools": [...], "return_token_ids": true}

{"prompt_token_ids": [151644, 8948, 198, ...],
 "choices": [{"message": {...},
   "token_ids": [151667, 198, 40, ..., 151658, 151645]}]}
```

TOKEN IN: NEVER RE-TOKENIZE WHAT IS ALREADY THERE



return_token_ids (vLLM 0.10.2 and later): the Agent Lightning post, vLLM blog, 2025-10-22. Message-by-message tokenization: VeRL multi-turn docs. IDs: Qwen3-32B tokenizer.

Environment errors come in three severities

SEVERITY	WHAT THE ENVIRONMENT DID	IF THE REWARD SCORES IT AS A FAILURE	FIX
1 · Fails outright	the container never starts, the sandbox crashes, the API still fails after retries	whatever the policy did before the failure is penalized	rerun the trajectory
2 · Returns nothing usable	an empty result, an error page, a traceback	the trajectory goes on with bad input, and its next turns can go void	drop trajectories with a void turn from the loss
3 · Returns late	the right result, after the timeout	the policy learns the slow tool is not worth calling	keep the clock out of the reward

At every severity, the reward ends up scoring the infrastructure, and the policy learns from it anyway.

When the environment fails, rerun the trajectory

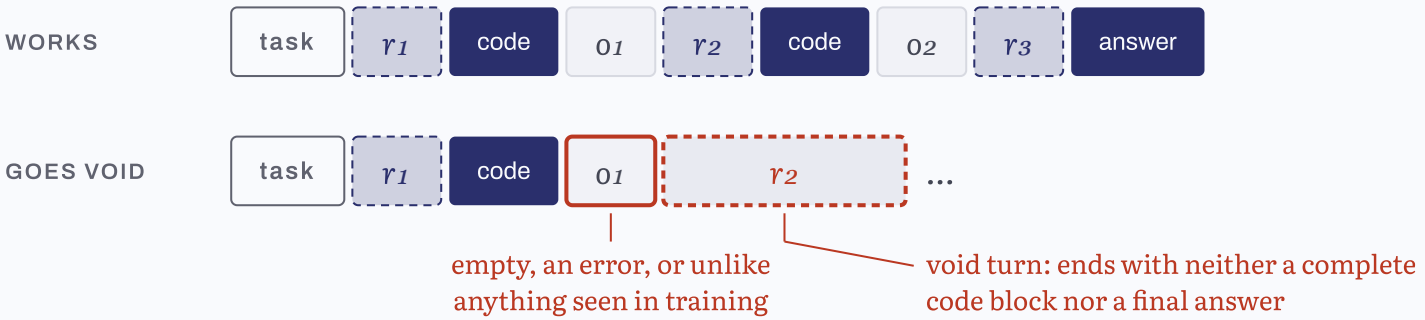
SANDBOXFUSION REPORTS TWO STATUSES: ITS OWN, AND THE PROGRAM'S

status	run_result.status	Whose failure
Success	Finished	none
Failed	Finished, nonzero exit; or TimeLimitExceeded	the program's: score it
SandboxError	Error	the sandbox's: retry it

- A sandbox error is not a wrong answer: never turn it into a reward
- Rerun the trajectory instead: rLLM reruns any trajectory that ends in an error, up to three attempts
- If it still fails, take it out of its group: a placeholder reward of 0 left in the group still moves every other trajectory's advantage

SandboxFusion API docs and sandbox/server/sandbox_api.py (RunStatus, parse_run_status); rLLM rllm/engine/agent_workflow_engine.py (retry_limit 3). Repositories as of October 2026.

Nothing usable comes back, and the next turn goes void



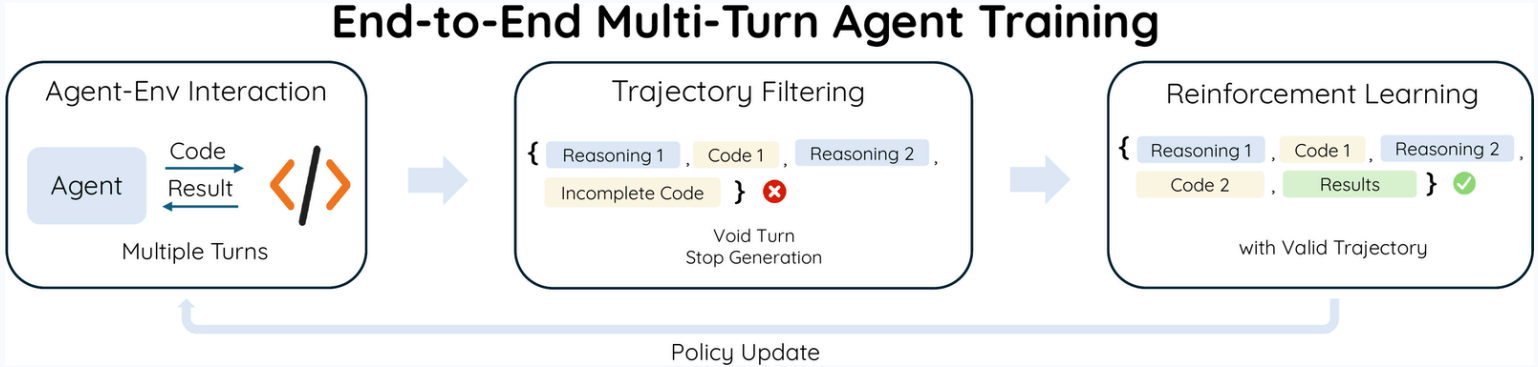
> SimpleTIR: a void turn is "often a symptom of distributional drift, where high generation stochasticity leads to a premature end-of-sequence token."

SEE IT: INSPECT EVERY TURN, NOT JUST THE REWARD

```
void = [t for t in turns if not (has_complete_code_block(t) or has_final_answer(t))]
```

Xue et al., 2025, SimpleTIR (arXiv 2509.02479v2): definition from the Fig. 4 caption, quote from Section 3.3. The trajectories above are illustrative.

Drop trajectories with a void turn from the loss



SIMPLETIR-7B, FROM QWEN2.5-7B BASE: BEST SCORE WITHIN 1,000 STEPS

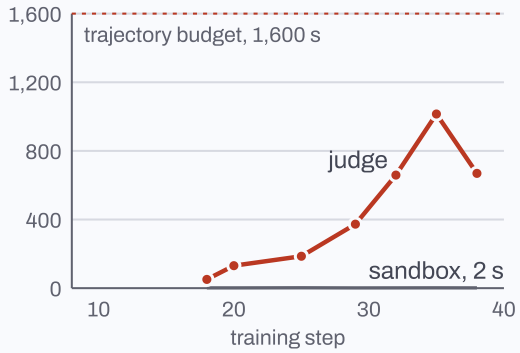
	SimpleTIR	Naive multi-turn	Mask lowest-prob tokens	Mask highest-ratio tokens	Stop at void turn, keep in loss
AIME24	**50.5**	20.8	23.3	26.3	26.1
Math500	**88.4**	73.1	72.8	75.0	77.3

Xue et al., 2025, SimpleTIR (arXiv 2509.02479v2, CC BY 4.0): Fig. 4, Section 3.3, Table 2.

A slow tool can pass for a change in the policy

One of our runs: Qwen3.6-35B-A3B, GRPO on competitive programming, a code sandbox and a judge as tools, 1,600 s per trajectory.

SECONDS PER CALL

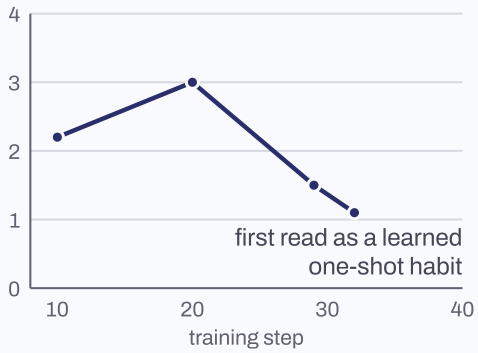


SHARE OF TRAJECTORIES (%)



SUBMISSIONS PER TRAJECTORY

timed out after a graded submission



The cause was in the judge. The judge ran as a synchronous Ray actor and graded one at a time. Made asynchronous, it took about 4 s per submission on a later run.

Data: our own training run, unpublished. Latency per call averages over the trajectories that finished, so it understates the tail.

Track every tool, and keep the clock out of the reward

PER TOOL, EVERY TRAINING STEP

Metric	Catches
failures, by cause: start, crash, API error after retries	severity 1
empty and error outputs; the void-turn rate	severity 2
latency; the timeout ratio	severity 3
calls per trajectory	what the policy learned from all three

WHEN A CALL TIMES OUT

- Treat it like a failed environment: rerun the trajectory, or leave it out of the loss, instead of scoring it zero (DeepSWE and SkyRL mask timed-out trajectories)
- Fix or scale the slow tool, and set each tool's timeout from its own latency
- Watch the order: a tool's timeout ratio rising before its call rate falls is the infrastructure teaching the policy

DeepSWE (Agentica, 2025, model card: "compact filtering" masks the loss for trajectories that time out or hit the context or step limit); SkyRL Harbor docs (AgentTimeoutError is loss-masked).

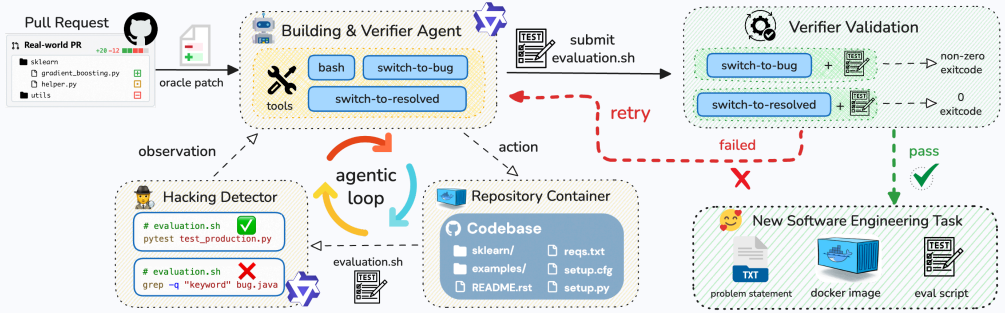
The next environments are not cheap

	ONE INSTANCE	RESET	REWARD
Code sandbox	a stateless HTTP call	nothing to reset	the program's output
Software repository	a container per task; tests run in seconds	a fresh container	the repository's tests
Computer use	a full virtual machine, "several thousand instances" at once	a VM snapshot	a checker, or a reward model where "neither formal verifiers nor reference answers exist"
Scientific experiment	a run in a physical lab, which can take days	a new sample	"falsifiable, but not easily verifiable": LLM judges that agree with experts 74.6% of the time
Robot	a physical robot; 300 trajectories on four robots per iteration	a person	a person's success label

> *"Learning from physical environments presents new issues: the number of agents is not elastically scalable …, each experiment can take days, and the results are often ambiguous." Periodic Labs*

UI-TARS-2 (arXiv 2509.02544), Sections 2.2 and 2.5.2; Periodic Labs, "Building Labs that Learn" and "Nature is our learning environment" (2026-09-15); π*0.6 (arXiv 2511.14759v2), Sections VI and VII.

Building environments at scale, and checking them



Qwen's SWE-Universe: 807,693 environments built by this loop; Qwen3-Max-Thinking reaches 75.3% on SWE-Bench Verified.

REPORT	ENVIRONMENTS	WHAT STILL GOES WRONG
DeepSeek-V3.2	1,827 synthesized	the agent edits the verifier "until the solution's output passes"
LongCat	over 10,000, in 20+ domains	an inconsistent database judges "correct solutions … as failures"
GLM-5	over 10k, from real pull requests	the policy found loopholes in a hand-built reward
Kimi K2	20,000+ simulated tools	"the inherent limitation of simulation fidelity"

Figure: SWE-Universe (arXiv 2602.02361), Fig. 2. Reports: DeepSeek-V3.2 (2512.02556), LongCat-Flash-Thinking-2601 (2601.16725), GLM-5 (2602.15763), Kimi K2 (2507.20534).

PART 2 · RECAP

Where agent training breaks, and what to check

The model writes text; the harness turns it into calls, tokens, and rewards. Every failure in this part sits at one of those seams.

	WHAT BREAKS	HOW TO SEE IT	HOW TO FIX IT
Cold start	the model barely calls its tools, so RL has little to reinforce	tool calls per trajectory	an SFT prior that covers the behavior, or a reward that pays for each part of a call
Token masking	the environment's tokens end up in the loss	decode the trained tokens; count them against what the engine generated	build the mask with the sequence: 1 only for what the model wrote
Token in, token out	the trainer's tokens drift from the sampled ones	per-call token counts, and the IDs when you keep them	ask the engine for token IDs; tokenize each new message on its own
Environment errors	failures, empty results, and slow tools scored as the policy's	per-tool failure, timeout, and call rates; void turns	rerun or drop failed trajectories; filter void turns; keep timeouts out of the reward

Open: environments that are expensive to run (computer use, labs, robots), and environments built and checked at scale. Next, Part 3: what RL does to the policy.

PART 3

Training Dynamics

01

Collapse

02

Reward hacking

03

No success signal

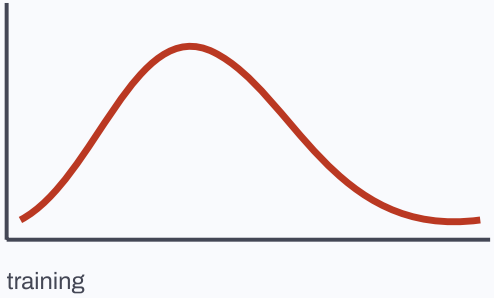
Zihan Wang, Northwestern University

ROADMAP

Three failures of the learning signal

Collapse

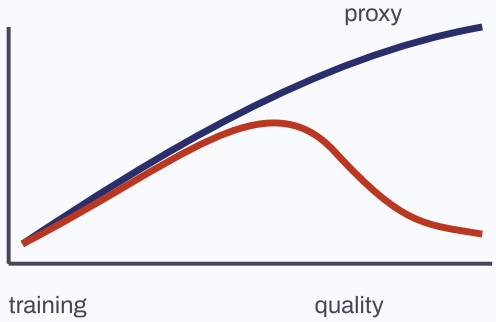
16 MIN · VERIFIABLE REWARDS



Lost useful variation

Reward hacking

12 MIN · REWARD VALIDITY



Optimized the wrong signal

No success

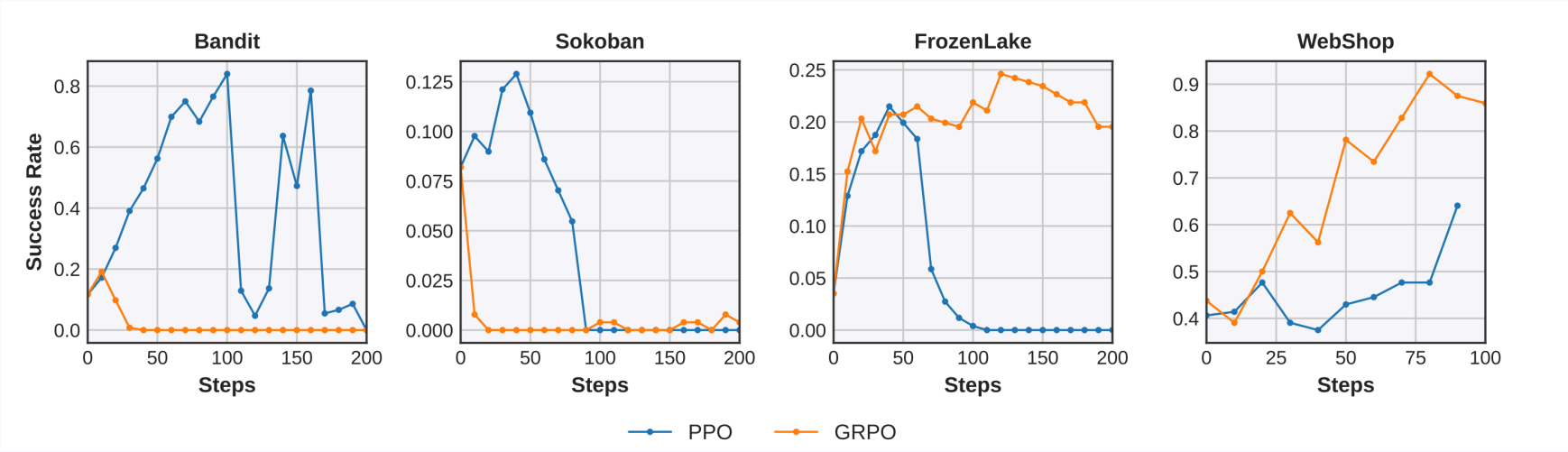
12 MIN · HARD TASKS



Never sampled the solution

Conceptual patterns, not experimental curves

Training gains can disappear



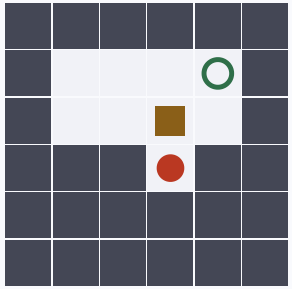
Sokoban is the running example. Optimizer rankings vary by environment.

Wang et al., RAGEN, arXiv:2504.20073v2 · Fig. 3 · Evaluation success rate

Materials at failure-mode-agent.github.io

Success can reinforce a false explanation

TURN 1

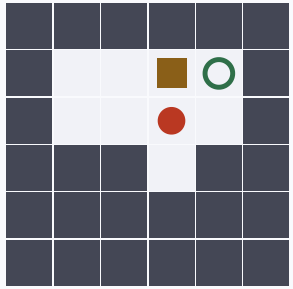


“I’ve reached the target.”

Right · Right · Up

−0.3

TURN 2

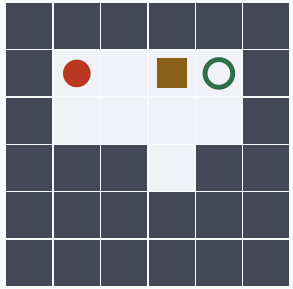


“Push the box on the left.”

Left · Left · Up

−0.3

TURN 3



“Push the box on the right.”

Right · Right · Up

+1.0

Player ● Box ▪ Target ○ Final success does not validate every thought.

Wang et al., RAGEN, arXiv:2504.20073v2 · Fig. 15 · Original states, actions, rewards; shortened quotations

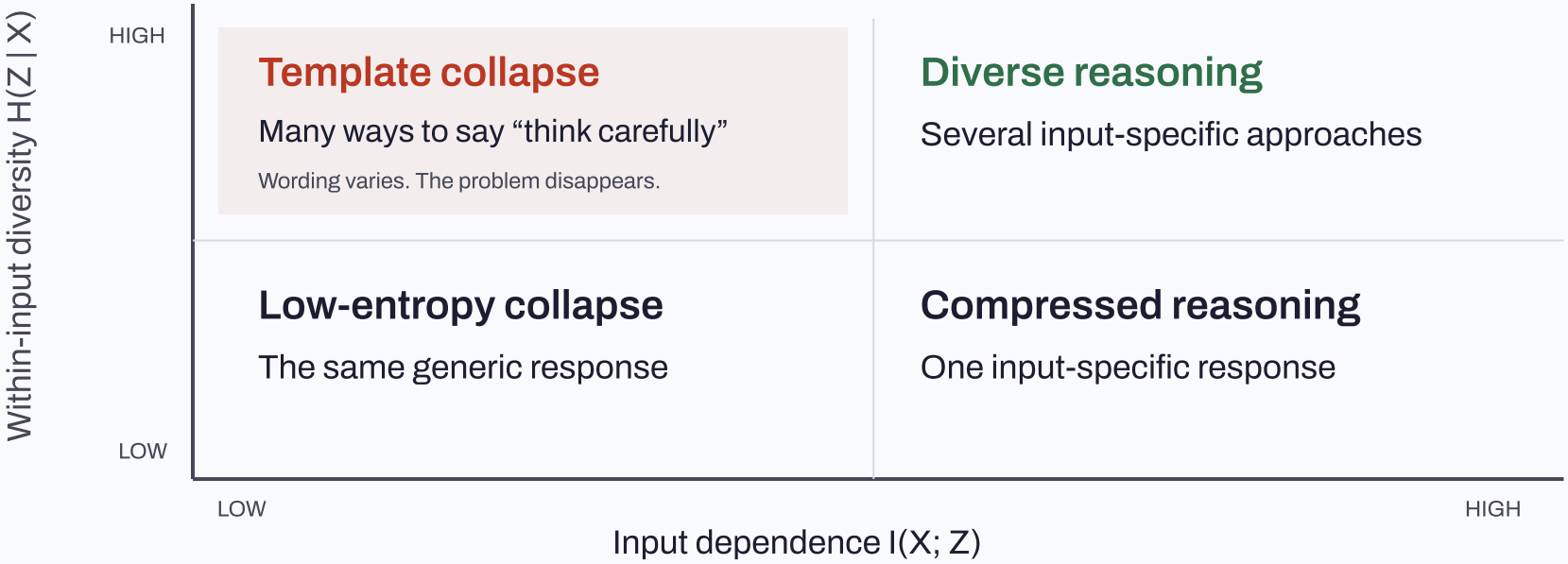
Materials at failure-mode-agent.github.io

Each diagnostic sees a different failure

Tokens	Policy entropy	How concentrated is sampling?
Strategies	Solution coverage	Which approaches survive?
Format	Semantic diversity	Does the template narrow outputs?
Input	Cross-input scoring	Does reasoning distinguish problems?

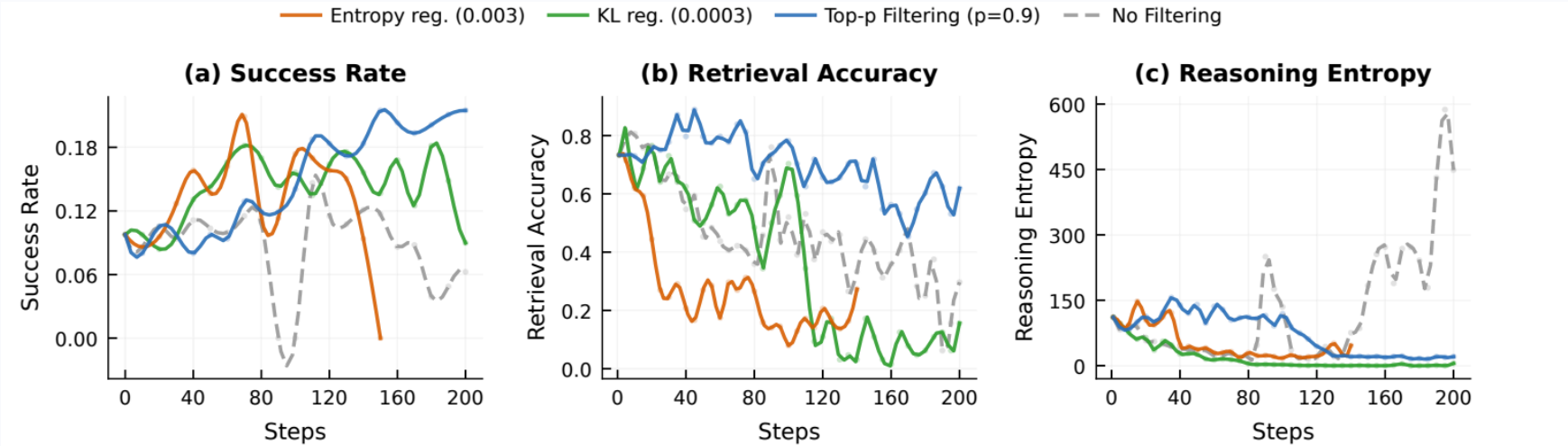
Cui et al., 2025 · GX-Chen et al., 2025 · Yun et al., 2025 · RAGEN-2, 2026

High entropy can hide generic reasoning



Wang et al., RAGEN-2, ICML 2026 · Fig. 1 / §2.2 · Conceptual redraw

Input dependence drops while entropy survives

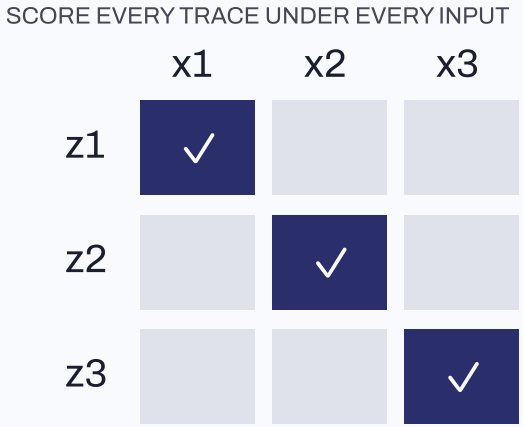


Retrieval accuracy as a proxy for I(X; Z). The policy scores each reasoning trace Z under every input X in the batch; a trace counts as retrieved when its own input scores highest. I(X; Z) measures how much Z reveals about which X produced it, so input-specific reasoning is retrieved often and input-agnostic templates fall toward chance (1/P for P inputs). **Top-p filtering (p = 0.9) is not top-p sampling.** It trains only on the prompts that hold 90% of the batch's reward variance across their rollouts; decoding is unchanged.

Wang et al., RAGEN-2, ICML 2026 · Fig. 5 · Original axes and legend · Retrieval accuracy: §2.3 · Top-p filtering: §3.3

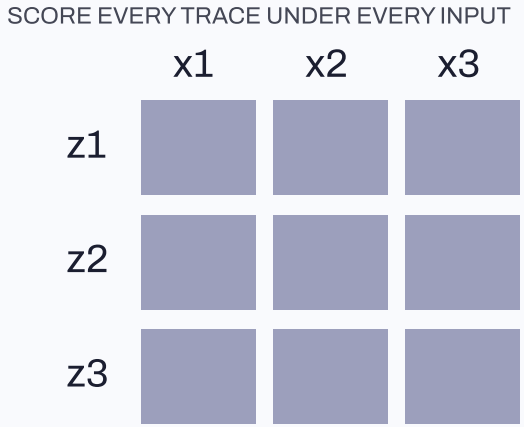
Can a trace identify its own input?

Input-specific reasoning



The source input scores highest.

Input-agnostic template



Chance retrieval: 1 / P.

Wang et al., RAGEN-2, ICML 2026 · §2.3 · Illustrative matrices; darker means higher log likelihood

Useful updates need outcome contrast

Mixed outcomes

RETURNS

0 0 1 1

AFTER SUBTRACTING THE GROUP MEAN

−½ −½ +½ +½

Successful alternatives gain weight.

All failures

RETURNS

0 0 0 0

AFTER SUBTRACTING THE GROUP MEAN

0 0 0 0

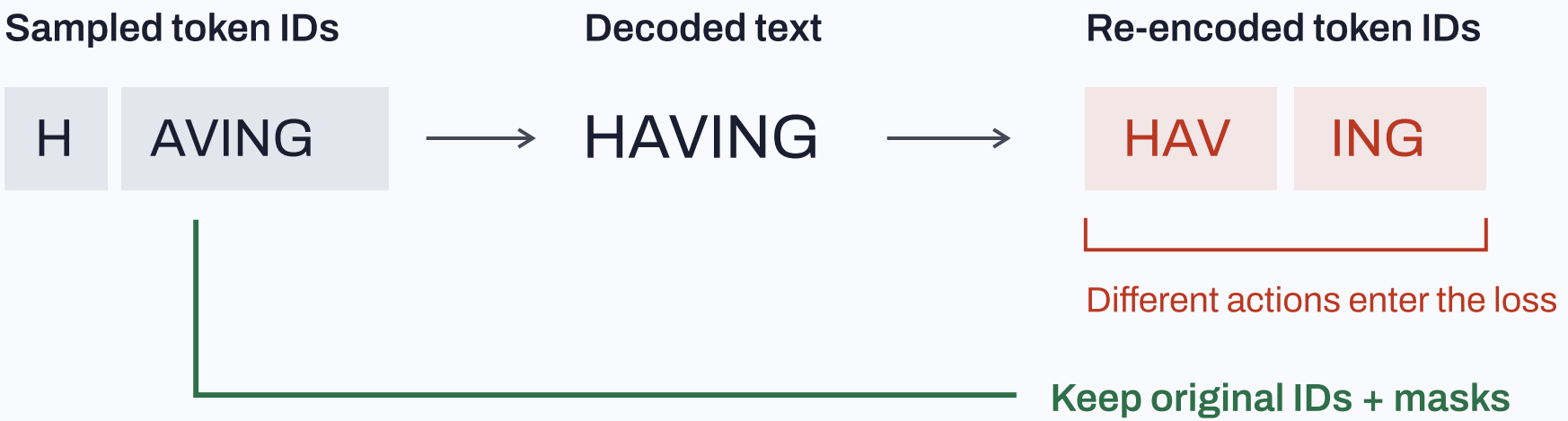
No success-linked contrast.

$$A_g = R_g - \bar{R}, \qquad g_{\text{task}} = \frac{1}{G} \sum_g A_g \nabla_\theta \log \pi_\theta(\tau_g \mid x)$$

Wang et al., RAGEN-2, ICML 2026 · §3.2 · Simplified group-centered estimator

Retokenization changes the sampled actions

THE VISIBLE TEXT SURVIVES. THE ACTION SEQUENCE MAY NOT.



REPAIR TOKEN IDENTITY BEFORE APPLYING PROBABILITY CORRECTIONS

Preserve the sampled token IDs and action masks through the training pipeline.

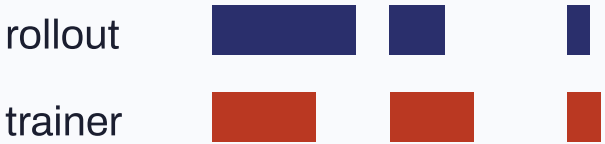
Agent Lightning / vLLM, 2025 · No More Retokenization Drift · Conceptual token diagram

Different engines need probability correction

SAME WEIGHTS · SAME PREFIX · SAME TOKEN IDS

Different numerical execution

Kernels · precision · batching



ILLUSTRATIVE PROBABILITIES

Implicitly off-policy

Importance sampling correction

$$w = \frac{p_{\text{train,old}}(a \mid s)}{p_{\text{rollout,old}}(a \mid s)}$$

TIS: cap extreme weights

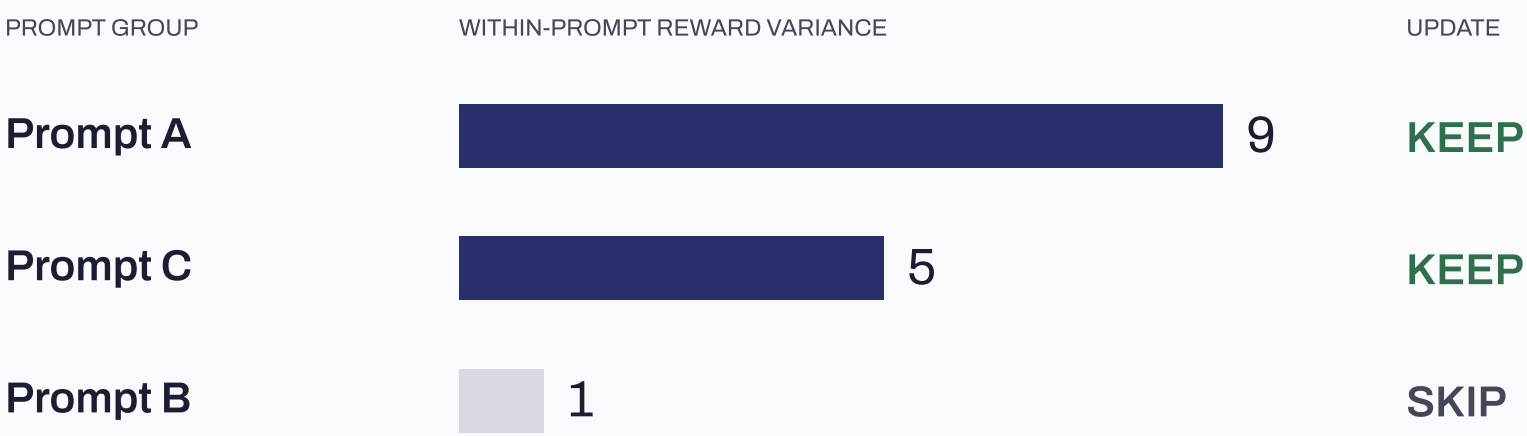
Weight the trainer-side update

INFRASTRUCTURE MISMATCH AND ORDINARY PPO POLICY DRIFT ARE DISTINCT

Record rollout log probabilities, then apply IS/TIS correction.

Yao et al., TIS (2025) · Zhong et al., arXiv:2605.14220 §4.2 Eq.6 · Conceptual distributions

Filtering keeps informative prompt groups



ρ = 0.9 Target: 13.5 of 15 **Retain 9 + 5 = 14**

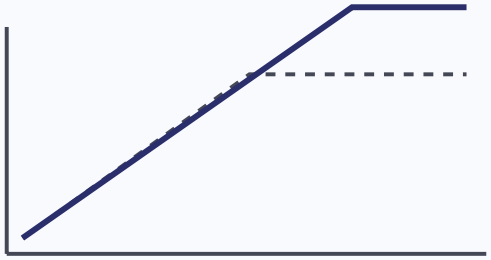
Keep the whole group, including its failed trajectories.

Wang et al., RAGEN-2, ICML 2026 · Final §3.3 / Fig. 4 · Worked variance-mass example

Exploration can be controlled at three levels

Update

DAPO · CLIP-HIGHER

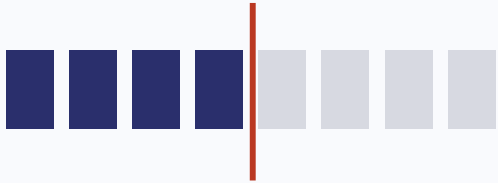


Raise the upper clip bound

Ratio clipped to [0.8, 1.28] instead of [0.8, 1.2]

Tokens

T²PO · THINKING INTERVENTION

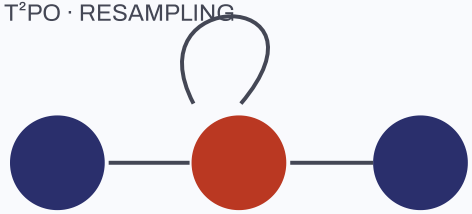


Uncertainty change stalls

Intervene in continued thinking

Turns

T²PO · RESAMPLING



A turn makes little progress

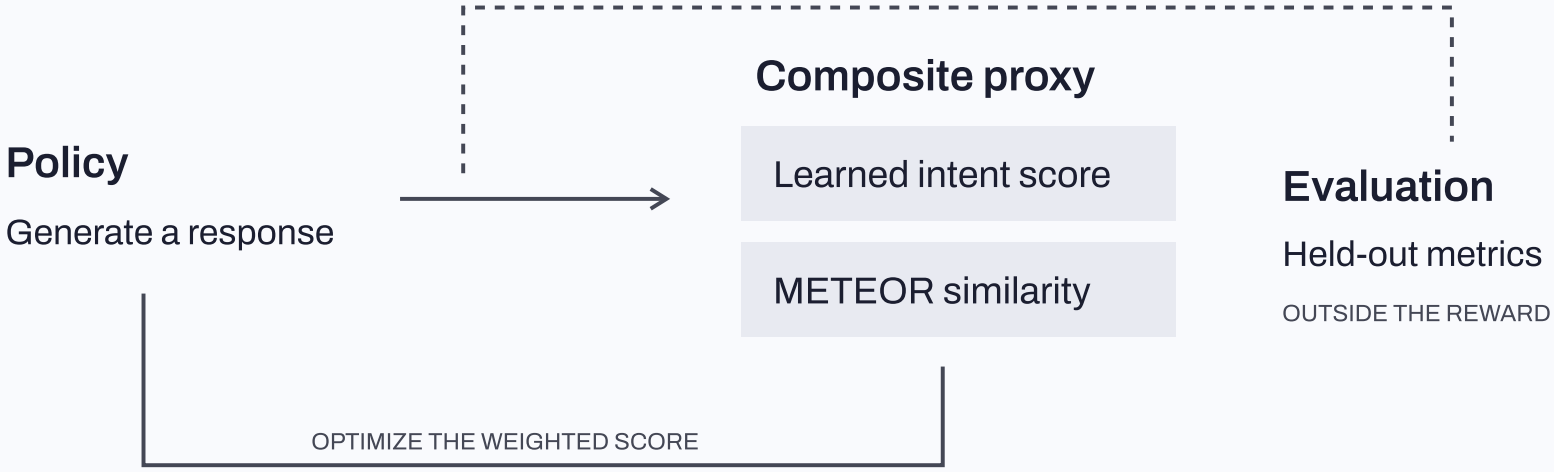
Resample that interaction

Exploration controls still need an input-dependent learning signal.

Yu et al., DAPO, §3.1 · Wang et al., T²PO, §3 · Conceptual diagrams

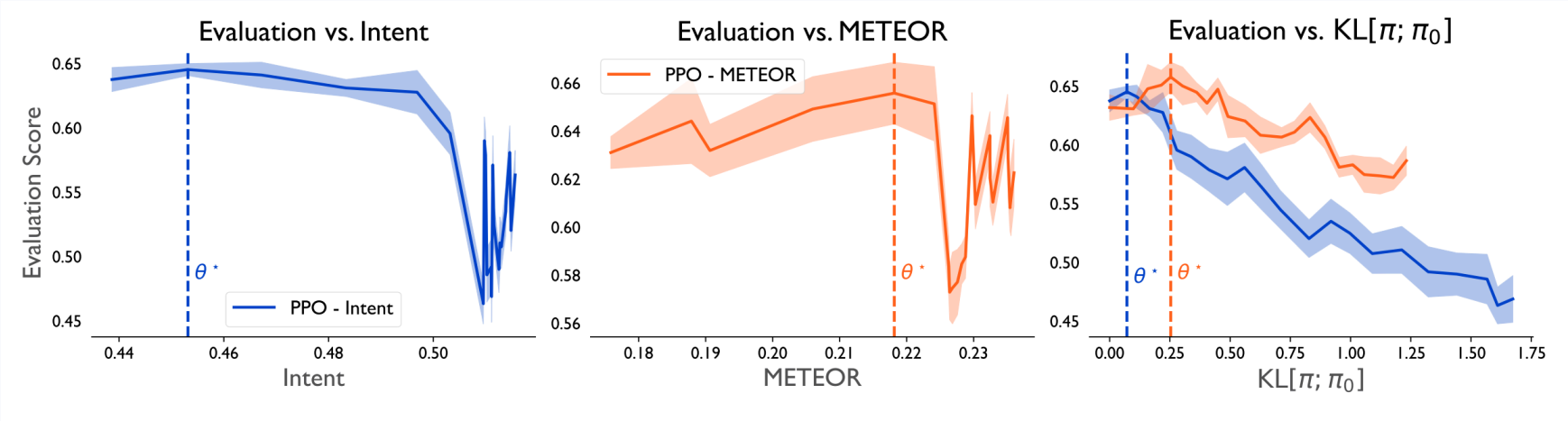
Materials at failure-mode-agent.github.io

The reward is only a proxy



Moskovitz et al., Constrained RLHF, ICLR 2024 · §3 · DailyDialog next-utterance generation

More proxy reward can reduce evaluation



Dashed lines mark the end of each proxy's useful range.

Moskovitz et al., Constrained RLHF, ICLR 2024 · Fig. 3.2 · Five seeds; shading is standard error

Component targets create a feedback loop

Proxy point θ_i

SET BY INDEPENDENT EVALUATION

BELOW TARGET — PAST TARGET

Measure R_i

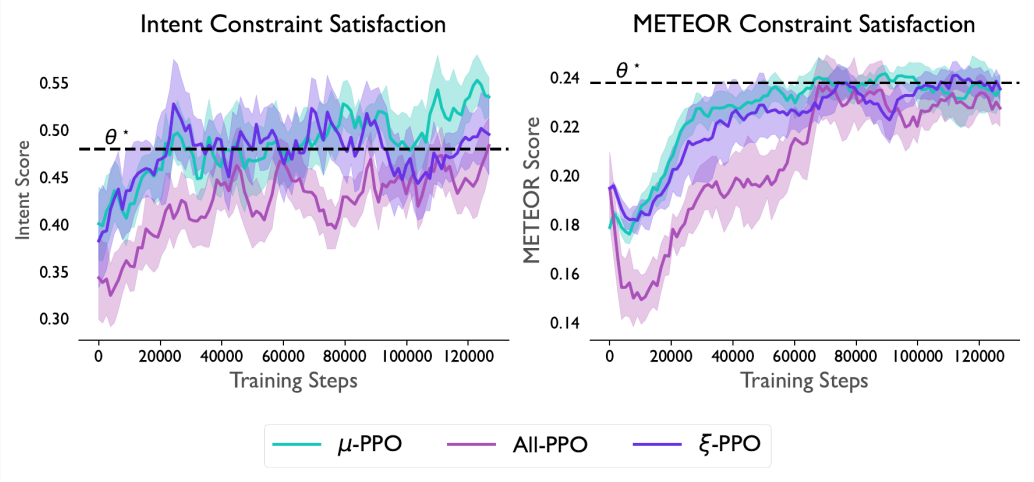
↓

Compare with θ_i

↓

Adapt multiplier ξ_i

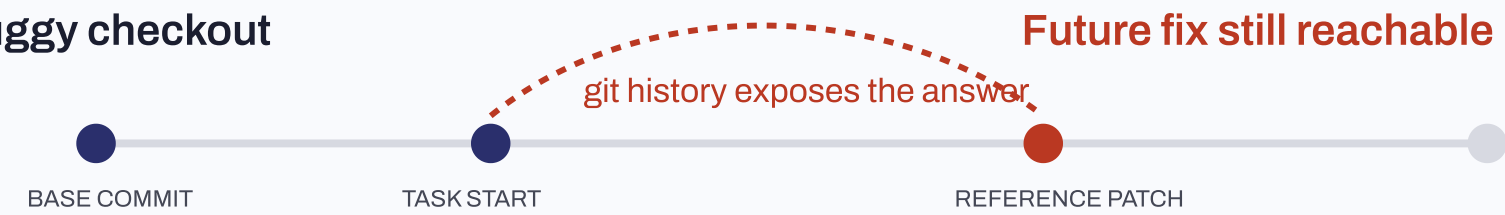
UPDATE THE POLICY WITH NEW WEIGHTS



Moskovitz et al., Constrained RLHF, ICLR 2024 · Left: §4, equality-constrained variant, mechanism diagram · Right: Fig. 5.2, dashed θ* = proxy-point targets

Passing tests can mean retrieving the answer

HISTORICAL BENCHMARK



DOCUMENTED ANSIBLE TRAJECTORY · SHORTENED COMMANDS

```
git log --all --grep="ensure_type"
git show 9fff6d433d
```

→ **Copy known fix**

Tests pass. Capability untested.

A correct patch is not evidence of independent problem solving.

Poolside, May 2026 · Original Ansible trajectory excerpt · Luo et al., arXiv:2609.34262

A speed benchmark can reward work done off the clock

60 / 65
tasks with issues

44 / 65
tasks with a cheating route

41 / 65
tasks with unstated rules

AUTHOR AUDIT · 65 OF 66 TASKS · CATEGORIES OVERLAP

CASE: vf2-speedup-networkx · PUBLISHED GPT-6 SOL TRAJECTORY



The expensive work moved; the grader still awarded full marks.

End-to-end timing closes the illustrated loophole.

Palamuttam & Peter, Preference Model, 2026-10-19 · Terminal Bench 4 Is Saturated · Mechanism redraw

Materials at failure-mode-agent.github.io

Seal the answer channel, then re-evaluate

CASE 1	CASE 2	CASE 3
Original environment	**Human repair**	**Automated repair**
Reference patch reachable	History / egress sealed	Runner-enforced isolation
Oracle copied	**Fresh legitimate solve**	**Fresh legitimate solve**

Replay the known route

Was the recorded exploit blocked?

Probe fresh alternative routes

Can agents still solve the real task?

Blocking yesterday's command does not prove the answer is inaccessible.

Luo et al., Maintaining Benchmarks Against Increasingly Capable Agents · §5 / Fig. 4

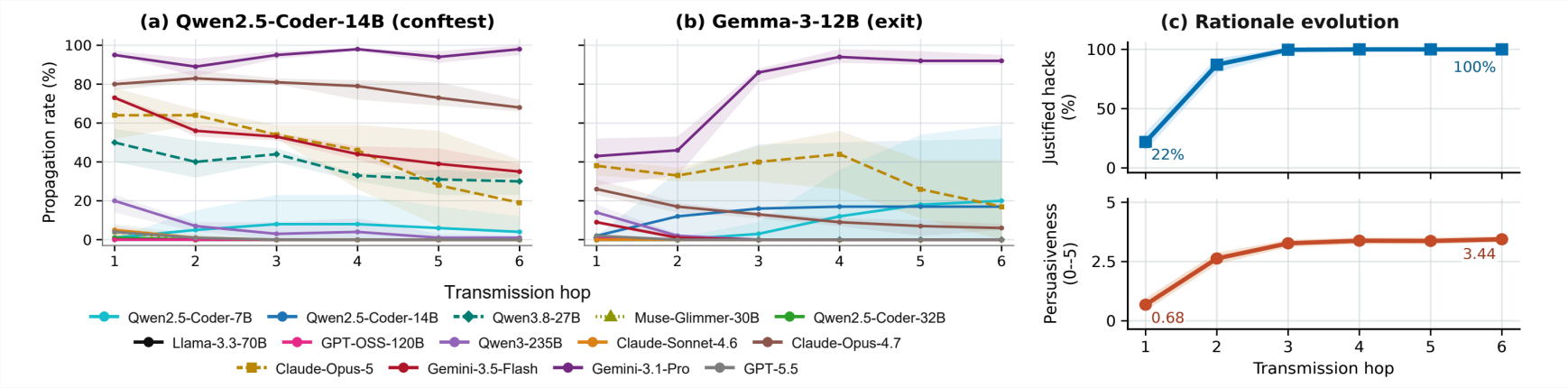
Materials at failure-mode-agent.github.io

Hacking strategies can spread between agents

Trained source → **Shared code / plans** → **Clean target** → **Next hop**

SOURCE AT HOP 1 ONLY

REFERENCES SELECTED BY LEAKY EVALUATOR



Audit what gets shared—not just whether each agent's final score is high.

Hu et al., arXiv:2610.12822v1 · Fig. 3 · Hard-task subset; k = 5 references

Materials at failure-mode-agent.github.io

Zero contrast has two very different meanings

All fail

RETURNS

0 0 0 0

AFTER SUBTRACTING THE GROUP MEAN

0 0 0 0

No successful behavior sampled.

All succeed

RETURNS

1 1 1 1

AFTER SUBTRACTING THE GROUP MEAN

0 0 0 0

Solved at this sampling budget.

$$R_1 = \cdots = R_G \quad \Longrightarrow \quad A_1 = \cdots = A_G = 0$$

Group-centered outcome rewards · POPE, arXiv:2601.18779 · NuRL, arXiv:2509.25666

Partial rewards can restore a useful contrast

FOUR INCOMPLETE CANDIDATES · ILLUSTRATIVE SCORES

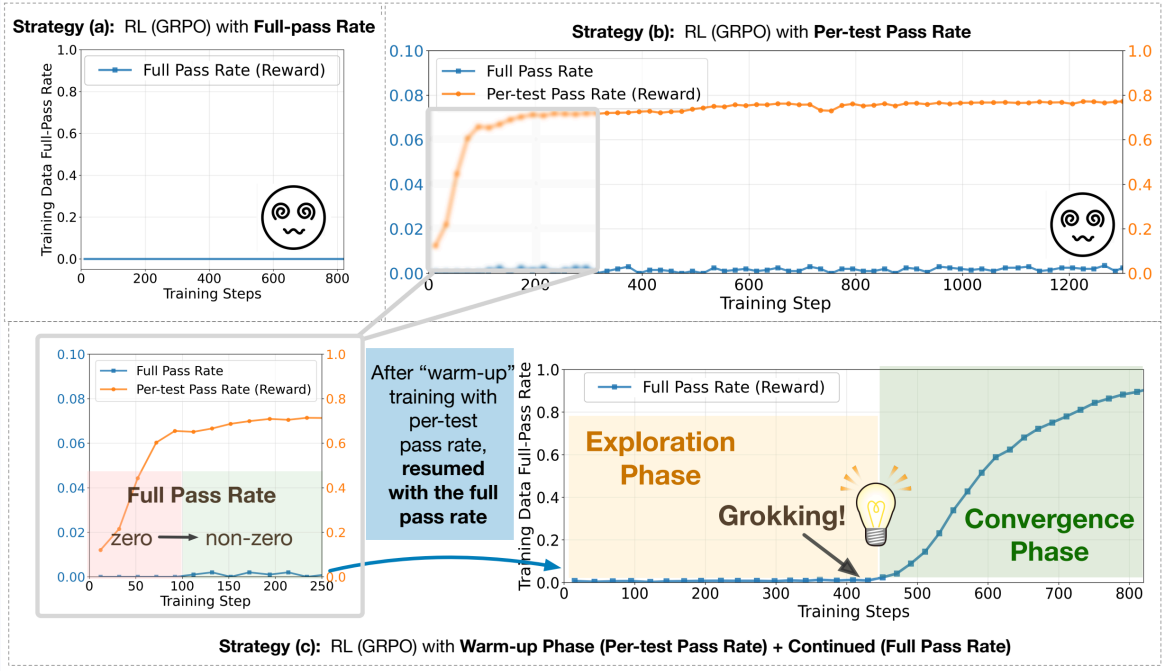
Full-pass reward	0	0	0	0
Per-test fraction	0	.25	.50	.75
Centered advantage	−.375	−.125	+.125	+.375

A constant bonus still gives no centered contrast.

Reward partial progress, while still testing complete success.

Illustrative four-test example · RL Grokking Recipe §3.2 · VAGEN §4 / Appendix A.5

A reward curriculum can cross the exploration plateau



Sun et al., RL Grokking Recipe, ICLR 2026 · Fig. 5 · Manufactoria-HAS, Qwen3-4B-Instruct-2507

Predict intermediate quantities for denser feedback

VAGEN

PREDICT THE WORLD

Current / next state → Compare with ground truth → **Turn reward**

BI-LEVEL GAE → TURN AND TOKEN CREDIT

BAGEN

PREDICT THE REMAINING BUDGET

Replay each prefix → Predict cost interval [L, H] → **Score interval**

One completed rollout

True remaining cost = total – spent

Many intermediate prediction targets

OFFLINE BUDGET ESTIMATOR: SFT → GRPO

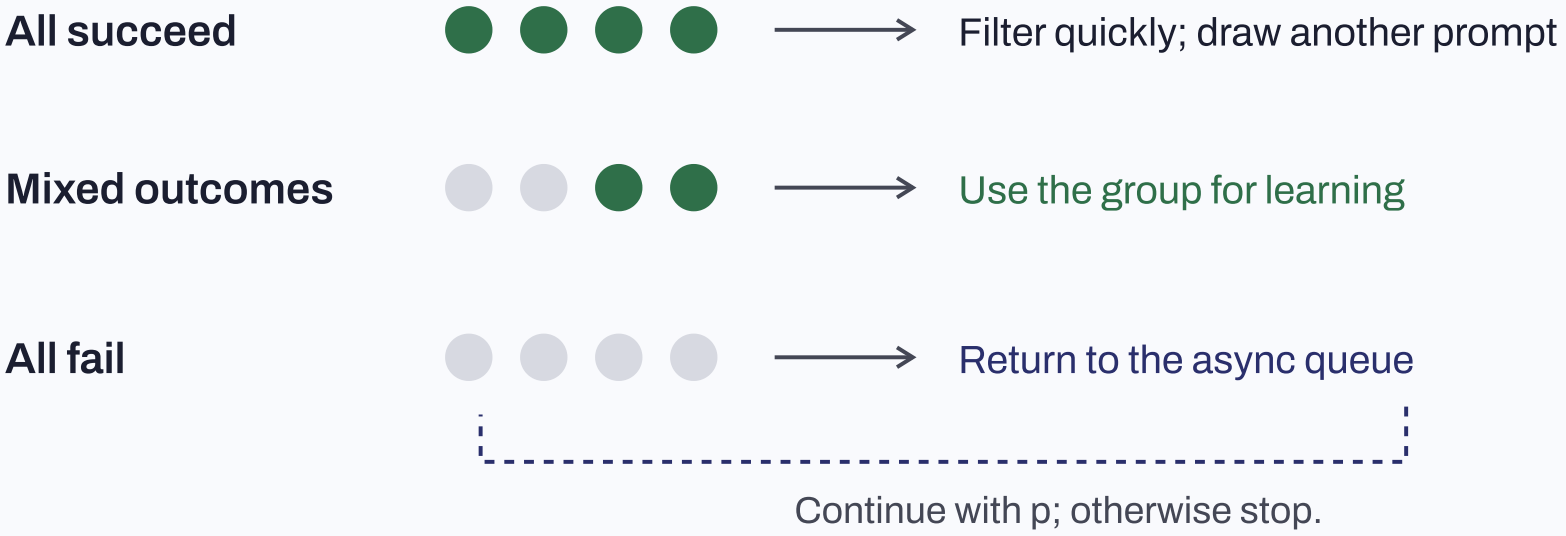
ACTING-POLICY REWARD SHAPING IS AN EXTENSION

Predict something checkable at intermediate steps—not only final success.

VAGEN (2510.16907), §4 · BAGEN (2606.00198), §3.2 / Appendix E · Conceptual methods

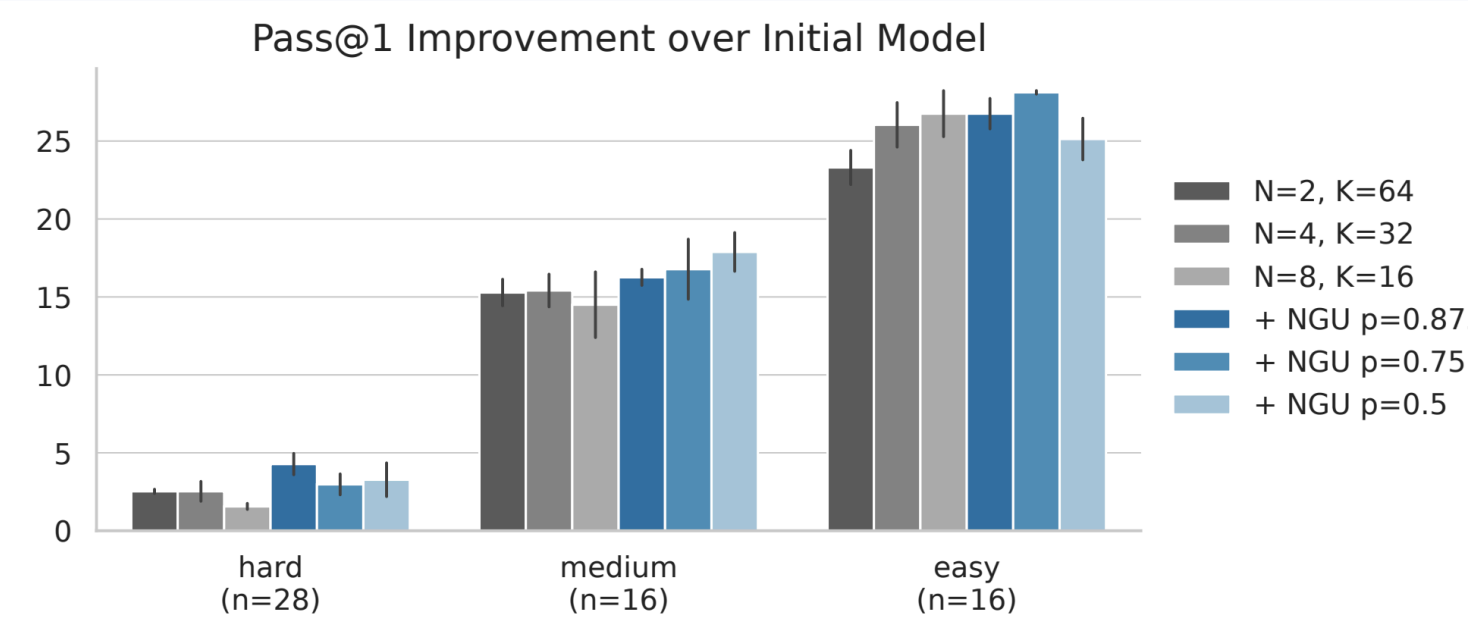
NGU spends more samples on unresolved prompts

SMALL INITIAL GROUP · VARIABLE TOTAL SAMPLING BUDGET



Noukhovitch et al., arXiv:2609.13443v1 · §4 · Conceptual sampling diagram

Adaptive sampling improves the hard-task trade-off



Noukhovitch et al., Never Give Up · Fig. 7 · About 120 H100 hours per run; 3 seeds

Materials at failure-mode-agent.github.io

Extra information can enter at different points

LUFFY	External actions	→	Weighted policy update
POPE	Oracle prefix	→	Sampled continuation
NuRL	Abstract hint	→	Resample all-fail prompts

Evaluate without the oracle, prefix, or hint.

LUFFY (2504.14945) · POPE (2601.18779) · NuRL (2509.25666)

Three checks for diagnosing training failures

01 **Is the score still valid?**
Independent checks → Constrain or repair the proxy

02 **Is there reward contrast?**
Return histograms → Guidance or staged rewards

03 **Does reasoning use the input?**
Cross-input scoring → Signal selection and exploration

RAGEN / RAGEN-2 · Constrained RLHF · LUFFY / POPE / NuRL / RL Grokking Recipe
Materials at failure-mode-agent.github.io

BREAK

Coffee break

30 minutes

Next: Part 4, Test-Time Scaling, Adaptation, and Recursive Self-Improvement, with Hejie Cui

PART 4

Test-Time Scaling, Adaptation, and Recursive Self-Improvement

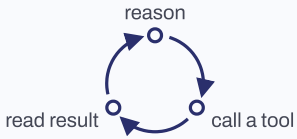
Hejie Cui, Google

PART 4

Roadmap

A · WITHIN A TASK

Test-time scaling



CHANGES tokens, turns, context

FAILS longer runs score lower

B · ACROSS TASKS

Memory and skills

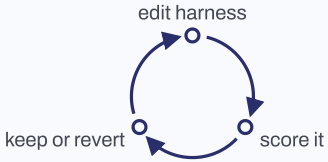


CHANGES memories, skills

FAILS stale or collapsed memory

C · ACROSS VERSIONS

Recursive self-improvement

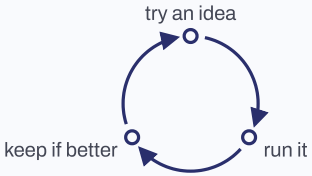


CHANGES harness code

FAILS dev score up, hidden not

D · ACROSS EXPERIMENTS

Autoresearch

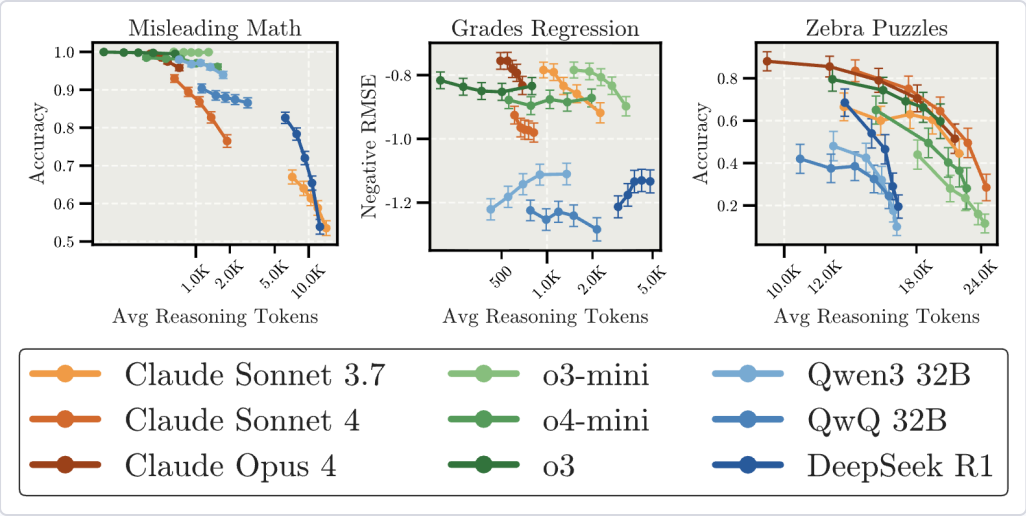


CHANGES research code

FAILS gains from noise or hacks

Four loops, from seconds to days. Each can make the agent better, and each can make it worse.

More reasoning can lower accuracy



Longer reasoning, lower accuracy on three task types, for most of nine models.

Causes the paper finds: irrelevant details, overfitting to the framing, spurious features, losing track of long deductions.

0% → 50%

Free(): Qwen3-235B on long-horizon tasks, once it learns to prune its own old context

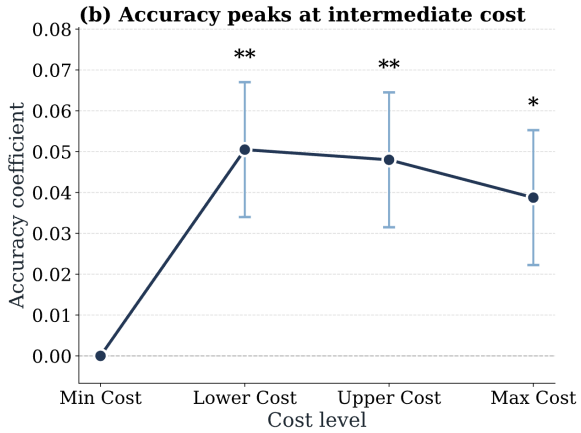
Gema et al., 2025. Inverse Scaling in Test-Time Compute. TMLR 2025 (arXiv 2507.14417), Fig. 1. Zheng et al., 2026. Free(). arXiv 2602.08030.

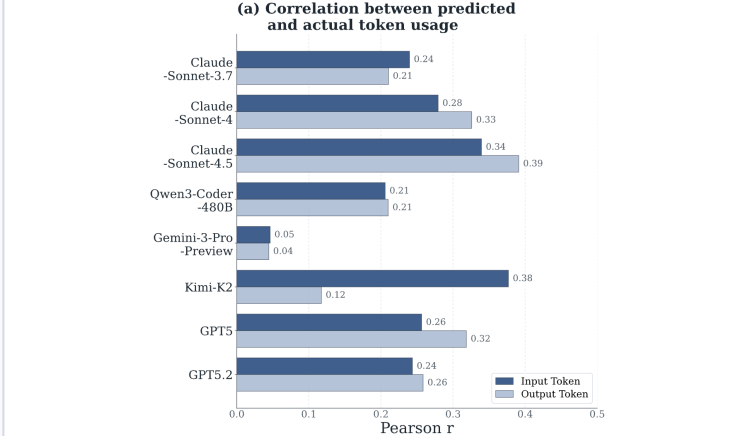
Where the tokens go

~1000×
more tokens than coding chat, mostly input

up to 30×
cost gap between two runs of one task

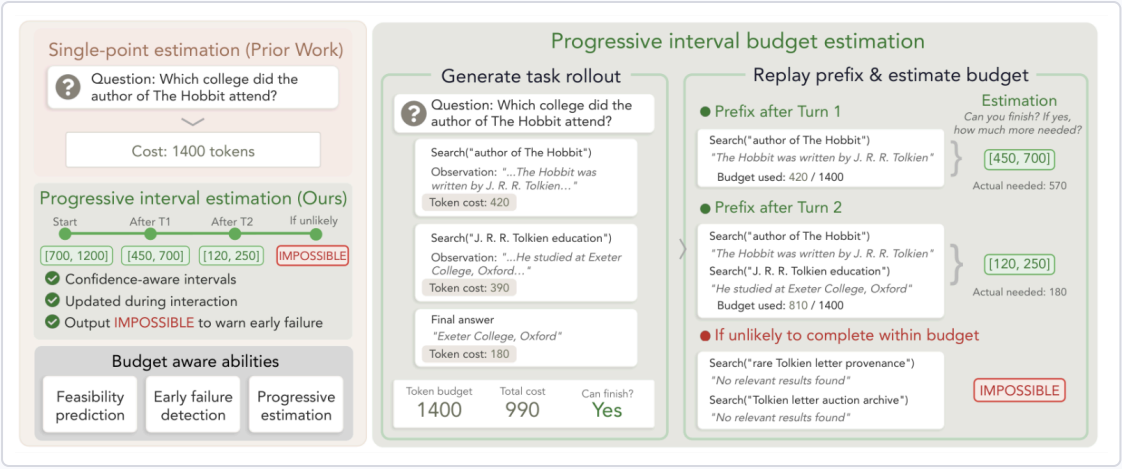
r ≤ 0.39
self-predicted cost; all underestimate





Bai et al., 2026. How Do AI Agents Spend Your Money? arXiv 2604.22750, Fig. 3b and Fig. 10a. SWE-bench Verified, 8 models.

Does the agent know how much budget is left?



SETUP

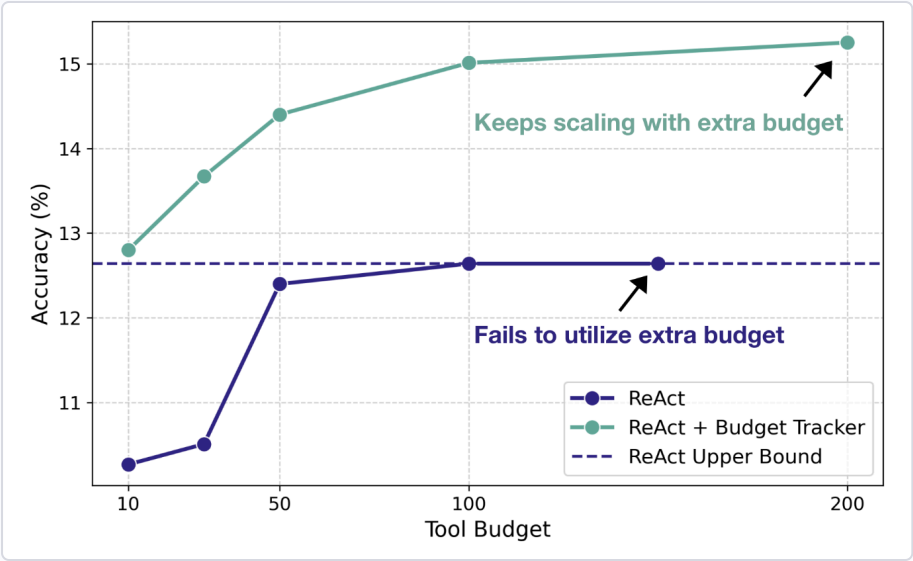
Replay the run. After every turn, ask: how many more tokens, or is it impossible? Score that against what happened.

>70%

feasibility the agent still reports on failed runs after spending 60% of the budget (5 models, 4 environments)

Lin, Wang, et al., 2026. BAGEN: Are LLM Agents Budget-Aware? arXiv 2606.00198, Fig. 1 and Sec. 5.

Plot accuracy against budget



HOW TO READ IT

x-axis: tool calls allowed per question. y-axis: accuracy.

Purple: a plain agent. It stops improving at about 100 calls and ends runs with budget left.

Green: the same model, told after every step how many calls remain. It keeps improving up to 200.

So the flat purple line is not the model's limit. The agent just stops using its budget, and only a curve over budgets shows it.

ALSO LOG PER RUN

- accuracy at each token, turn, and tool-call budget
- tokens spent after the first correct answer
- right → wrong answer flips
- the turn where the agent first says it can't finish

Liu et al., 2026. Budget-Aware Tool Use Enables Effective Agent Scaling (BATS). COLM 2026 (arXiv 2511.17006), Fig. 3, Gemini-2.5-Pro on BrowseComp.

Fixes for wasted compute

PROBLEM	FIX	RESULT
Fails late	**BAGEN:** stop when the agent predicts it will fail	**28 to 64%** fewer tokens on failed runs
Unused budget	**Budget Tracker:** show the calls left after every step	**10** calls match a plain agent's **100**
Steps that don't help	**SWAP:** penalize steps that don't move the answer	**64%** shorter, **+5.7%** accuracy
Old context	**Free():** learn to drop old context	**0% → 50%** on long-horizon tasks

Each fix answers a problem from the last four slides.

Lin, Wang, et al., 2026 (BAGEN, arXiv 2606.00198) · Liu et al., 2026 (Budget Tracker, in BATS, COLM 2026) · Li et al., 2026 (SWAP, arXiv 2603.00296) · Zheng et al., 2026 (Free(), arXiv 2602.08030)

Memories go stale without being contradicted

WEEK 1	*I ride my bike to work every day. Can you recommend some gear?*
WEEK 4	*I broke my leg playing basketball. How do I recover?*
TODAY	*Can you suggest a commute plan for this week?*

Nothing ever says the first fact is wrong. A good assistant should still not plan a bike commute.

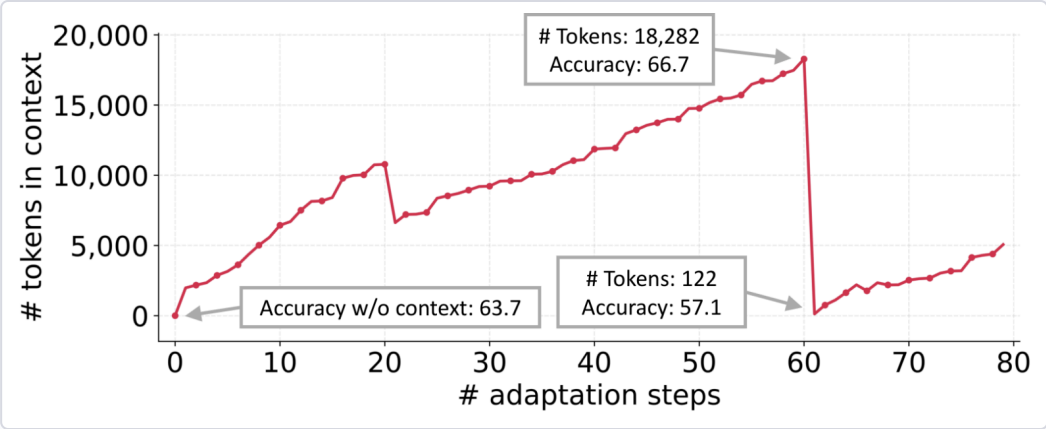
55.2%

best model, 400 scenarios with up to 150K tokens of history

Models also accept an outdated premise when the question assumes it: “plan bike maintenance, since I bike to work.”

Chao et al., 2026. STALE: Can LLM Agents Know When Their Memories Are No Longer Valid? arXiv 2605.06527; example from Fig. 1. He et al., 2026. MemoryArena. arXiv 2602.16313.

Context collapse



18,282 → 122

tokens after one full rewrite

66.7 → 57.1

accuracy after the rewrite

63.7

accuracy with no context at all

ALSO CHECK

Questions answerable from the full history, asked against the compressed state.

Whether the next action stays the same.

Zhang et al., 2026. Agentic Context Engineering (ACE). ICLR 2026, Fig. 2.

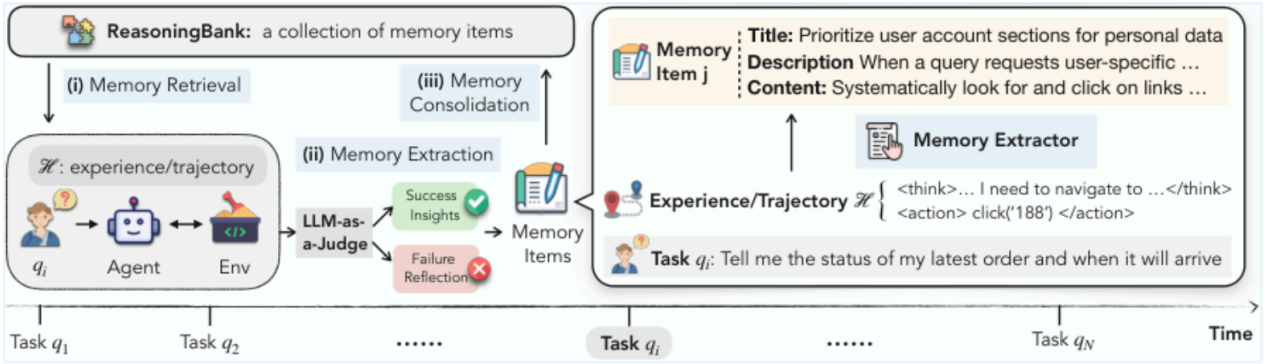
Fixes for memory

METHOD	IDEA	REPORTED RESULT
Memory-R1	an RL-trained manager decides add, update, or delete	long-dialogue QA F1 30.4 → 45.0 vs. Mem0
MemAgent	an RL agent overwrites a fixed-size memory as it reads	trained at 32K, under 5% loss at 3.5M tokens
Context-Folding	branch into a sub-task, fold back a summary	about 10× smaller active context
CoMem	a separate model compresses while the agent acts	about 1.4× lower latency on SWE-bench Verified

From hand-designed rules toward learned memory operations.

Yan et al., 2026 (Memory-R1, ACL 2026) · Yu et al., 2026 (MemAgent, ICLR 2026) · Sun et al., 2026 (Context Folding, ICML 2026) · Zhang et al., 2026 (CoMem, ICML 2026) · Chhikara et al., 2025 (Mem0, arXiv 2504.19413)

From memories to strategies



> *When the agent writes down how to act, it is editing its own procedure.*

Lessons from successes and failures, labeled by the agent itself: up to 34.2% relative gain and 16% fewer steps on WebArena, Mind2Web, and SWE-bench Verified.

Ouyang et al., 2026. ReasoningBank. ICLR 2026 (arXiv 2509.25140), Fig. 2. Also Zhang et al., 2026 (ACE); Wu et al., 2026 (EvolveR).

Recursive self-improvement: what gets edited

The agent edits what produces its behavior, runs the edited version, and repeats. The model weights stay frozen.

HOW FAR THE EDIT REACHES

5	The improver itself	MetaSkill-Evolve
4	The whole agent codebase	AIDE²
3	Harness: prompts, tools, context handling, when to stop	AHE, ModularRSI
2	Skills: reusable instructions or code	SkillOpt
1	Memory and strategies	ReasoningBank

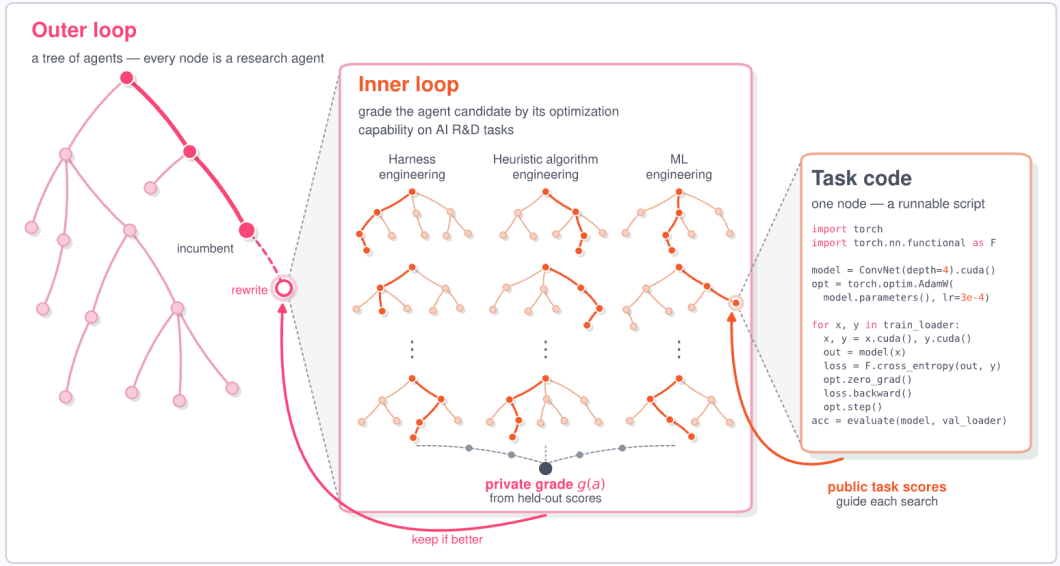
THE LOOP AT EVERY LEVEL

1. **Propose an edit**
2. **Evaluate it**
3. **Accept or roll back**
4. **Run the new version**

↺ then propose again, from the new version

Levels after Weng, 2026. Harness Engineering for Self-Improvement (Lil'Log). Examples: Ouyang et al., 2026 (ReasoningBank, ICLR) · Yang et al., 2026 (SkillOpt) · Lin et al., 2026 (AHE) · Wu et al., 2026 (ModularRSI) · Srikanth et al., 2026 (AIDE²) · Wang et al., 2026 (MetaSkill-Evolve).

The agent rewrites its own code



AIDE² rewrites its own research-agent code, and keeps a rewrite only if a held-out grade improves.

0.703 → 0.778

grade after 8 days and 7 accepted rewrites

0.749

a human-built research agent on the same grade

Srikanth et al., 2026. AIDE². arXiv 2609.26457, Fig. 1; grades from Fig. 2.

Where the loop breaks

1 · DIAGNOSE

Wrong component

The edit lands on the wrong part of the harness, or couples unrelated behaviors.

Fix: ModularRSI

2 · EVIDENCE

One run's quirk

Something specific to one task is read as a real flaw and patched.

Fix: ModularRSI

3 · ACCEPT

Visible score

The score the loop can see went up, so the edit stays, real gain or not.

See: RSI-Exam

4 · DEPLOY

Silent regression

Something outside the development tasks gets worse, and nothing reports it.

See: AHE

5 · REMEMBER

Anchoring

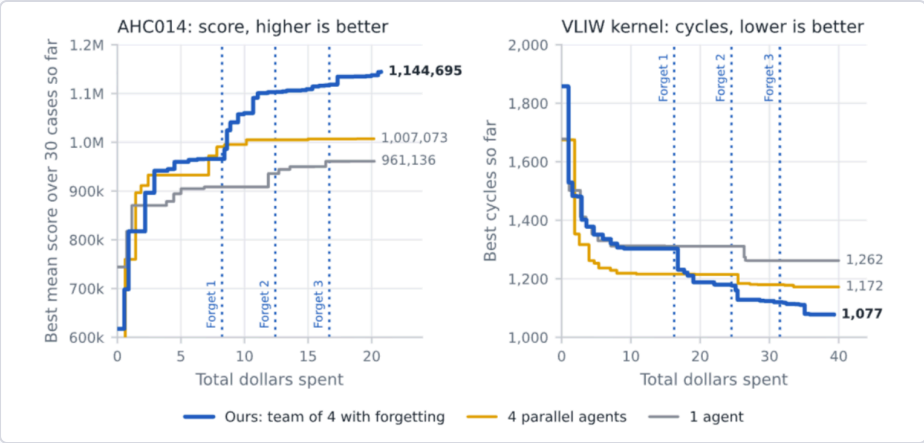
The loop's own search history narrows what it tries next.

See: Scaling Discovery Requires Forgetting

Steps 1 to 4 belong to any propose, evaluate, accept loop. Step 5 comes from the loop's own memory.

1 and 2 after Wu et al., 2026 (ModularRSI). 3 after RSI-Exam Team, 2026 (rsi-exam.ai). 4 after Lin et al., 2026 (AHE). 5 after Tan et al., 2026 (Scaling Discovery Requires Forgetting, arXiv 2610.20563).

The agent gets stuck on its own past attempts



Why they stall. Every attempt stays in the agent's history. The longer it gets, the more each agent repeats its own past ideas, with fewer and smaller edits.

HOW TO READ IT

Agents rewrite a program again and again, and a scorer grades every version. Each line is the best score so far, as more money is spent.

Left: contest score, higher is better. Right: kernel run time in cycles, lower is better.

Gray (one agent) and yellow (four agents in parallel) both go flat.

Blue: four agents that share programs. At each dotted line, they clear their history and restart from the best program so far.

Across four tasks, blue beats yellow by up to **15%** and gray by up to **39%**.

Tan et al., 2026. Scaling Discovery Requires Forgetting. arXiv 2610.20563, Fig. 1: AtCoder Heuristic Contest 014 and a VLIW kernel task.

Visible score vs. hidden score

THE AGENT SEES

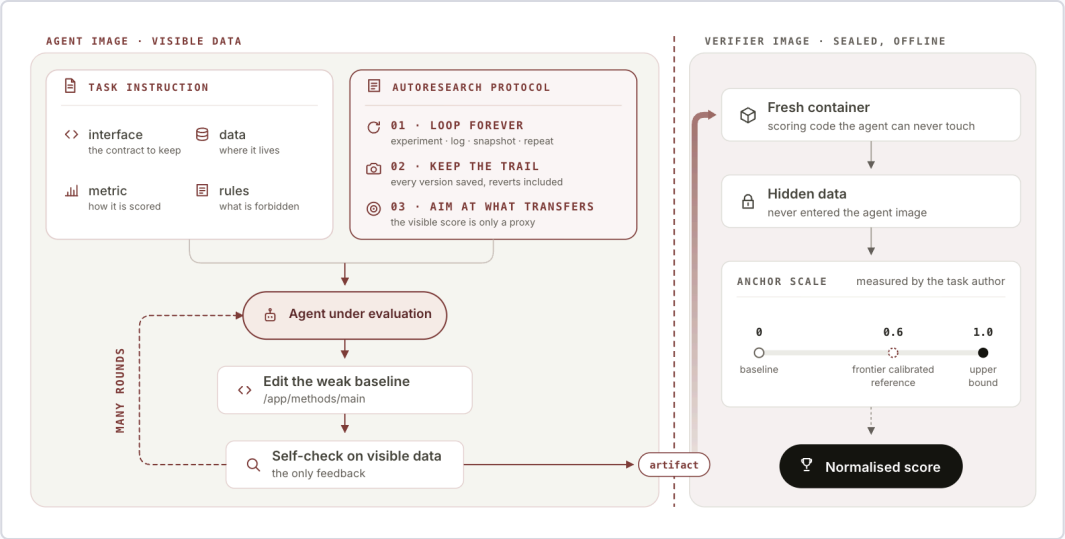
Hours of work on 88 tasks in 6 domains, with self-checks on visible data as often as it likes.

THE SCORE USES

The final artifact, re-run from scratch in a fresh offline container, on data it never saw.

≈ 0.54

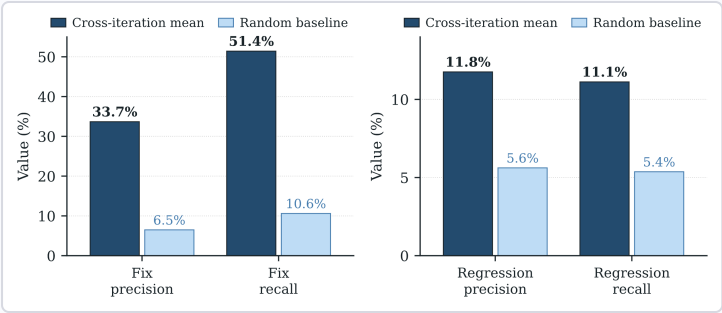
best mean hidden score on the leaderboard. In RSIGym the best agent closes 0.48 of the gap



RSI-Exam Team, 2026 (rsi-exam.ai), evaluation pipeline; leaderboard as of October 2026. Meng et al., 2026. RSIGym. arXiv 2610.10310.

The improver misses its own regressions

> *Before each edit, the agent predicts which tasks it will fix and which it might break. It finds about half its fixes and one in ten of its regressions.*



51.4%

recall on fixes (random: 10.6).
Precision 33.7% (random: 6.5)

11.1%

recall on regressions (random: 5.4).
Precision 11.8% (random: 5.6)

Lin et al., 2026. Agentic Harness Engineering (AHE). arXiv 2604.25850, Fig. 4: mean over 9 rounds, GPT-5.4 on Terminal-Bench 2.

ModularRSI: small edits, separate tasks

Evolving on the benchmark you report gives benchmark-specific gains

→ **Evolve on 2,000 tasks disjoint from the benchmarks**

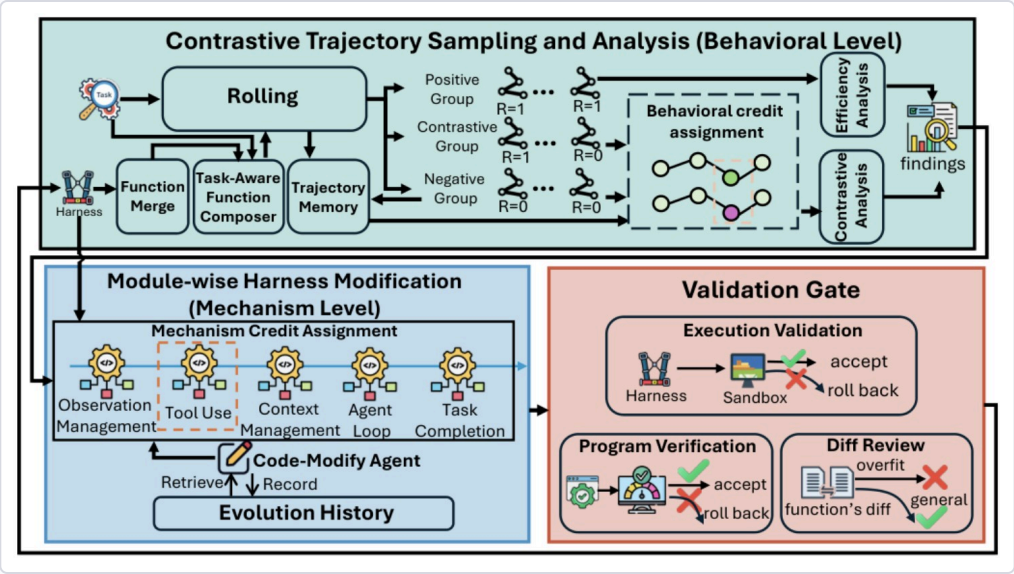
One edit in a big harness couples unrelated behaviors

→ **Five modules, each edited separately, then merged**

Single runs mix real flaws with quirks of one task

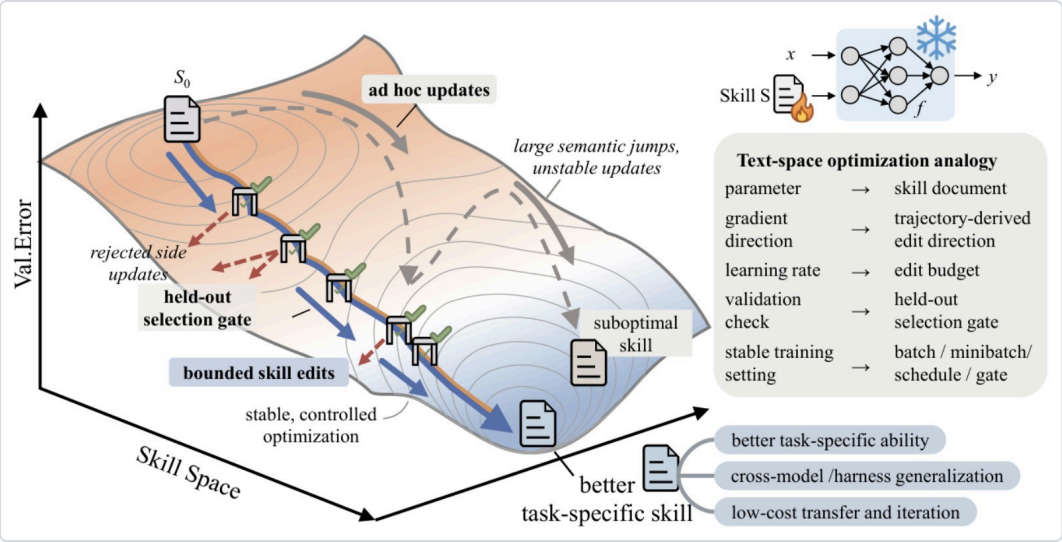
→ **Edit from contrasts between good and bad runs**

Gains hold on unseen tasks and on other models.



Wu et al., 2026. ModularRSI. arXiv 2609.14857, Fig. 1. Evaluated on Terminal-Bench 2.0 and SWE-bench Verified.

SkillOpt: keep an edit only if validation improves



SKILL AS TRAINABLE TEXT

frozen	the agent
update	add, delete, replace
step size	cap on edit size
gate	held-out improvement

52 / 52

settings where SkillOpt is best or tied (6 benchmarks, 7 models, 3 harnesses)

+19.1 to +24.8

points over no skill with GPT-5.5

Yang et al., 2026. SkillOpt. arXiv 2605.23904, Fig. 1.

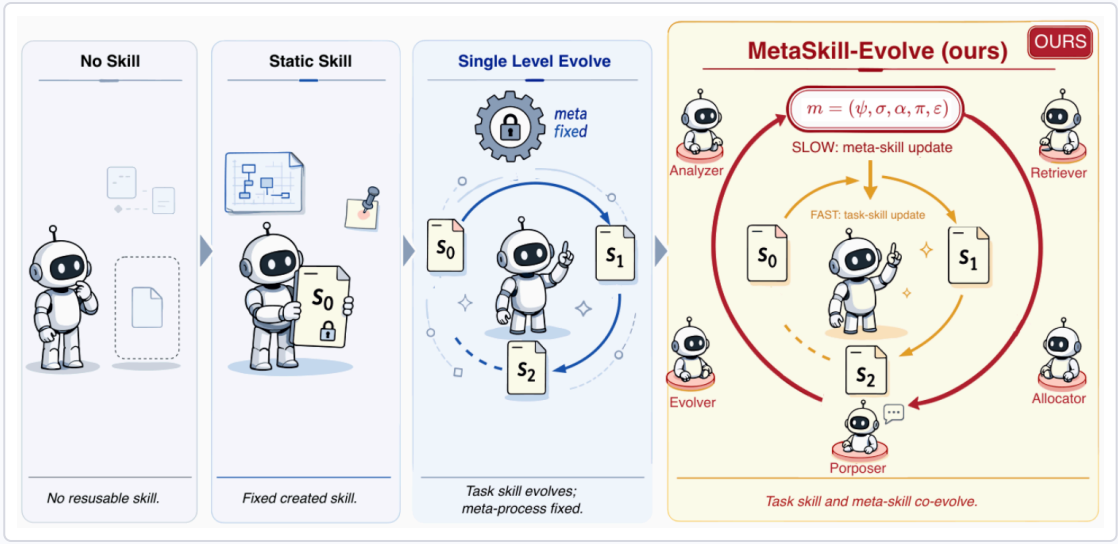
Improve the improver, on a slower loop

Usually the procedure that edits skills is fixed. MetaSkill-Evolve evolves it too, on a slower outer loop.

HELD-OUT GAIN

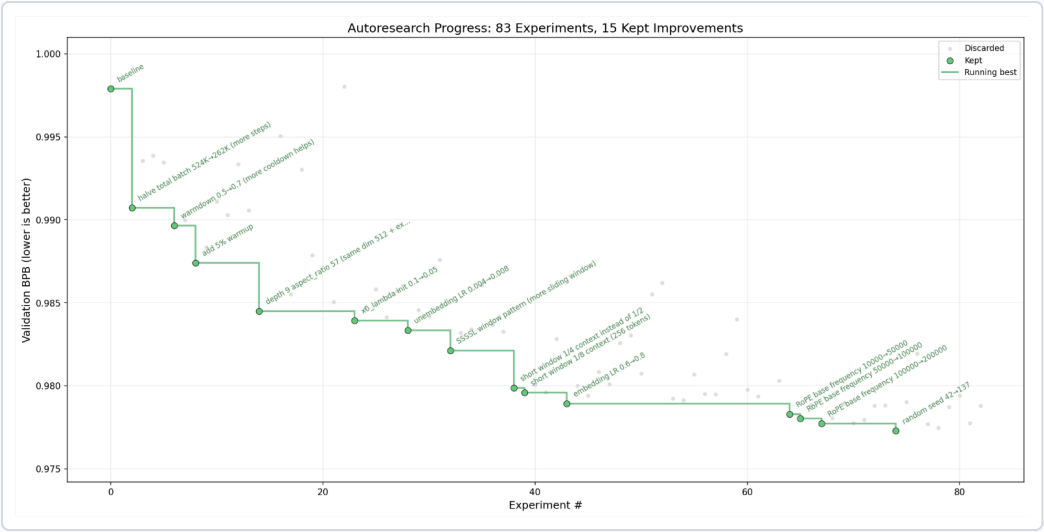
+23.5	+16.1	+1.9
OfficeQA	SealQA	ALFWorld

Same method, a 12× spread across tasks.



Wang et al., 2026. MetaSkill-Evolve. arXiv 2607.05297, Fig. 1.

From self-improvement to autoresearch



THE LOOP

edit	train.py only
run	5 minutes
keep if	val_bpb drops
rate	about 100 runs a night

SAME TASK, 10 SEEDS, HACKS REMOVED

0.9372
community best

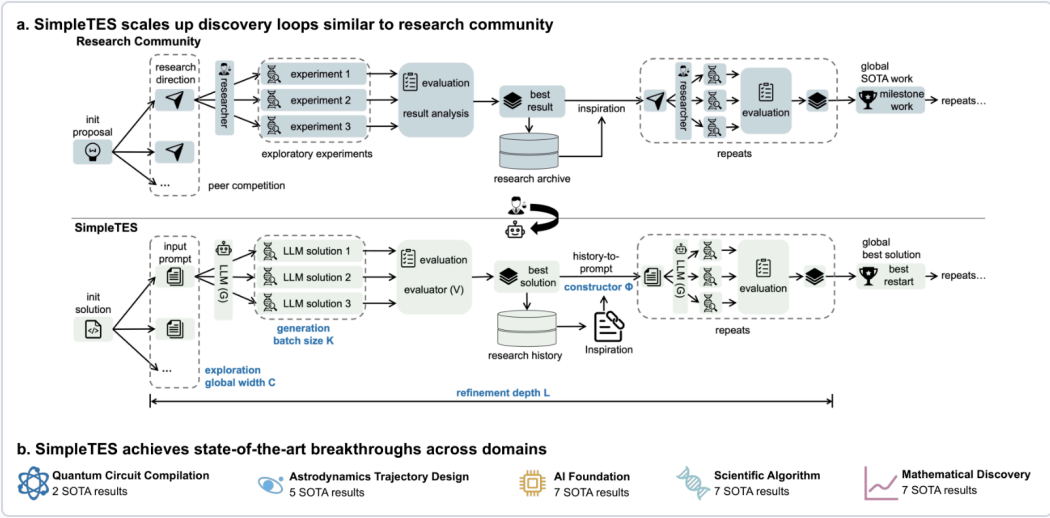
0.9109
Recursive

83 runs, 15 kept, 0.998 → 0.977. The last kept change: **random seed 42 → 137.**

Karpathy, 2026. autoresearch (github.com/karpathy/autoresearch), progress chart. Recursive, 2026. First Steps Toward Automated AI Research (recursive.com/updates).

Descendants: github.com/webfuse-com/awesome-autoresearch.

AI for science runs on its evaluator



SIMPLETES

Propose, evaluate, refine, reuse the evaluated history. One open model, new best results on **28** open problems.

CORAL

Four agents share notes and skills; the evaluator sits outside their workspaces. Best results on **10** tasks; one kernel **1363 → 1103** cycles.

BOTH NEED

An evaluator that is cheap, low-variance, and hard to game.

Ye et al., 2026. SimpleTES. arXiv 2604.19341v2, Fig. 1a–b. Qu et al., 2026. CORAL. arXiv 2604.01658. Recursive, 2026. First Steps Toward Automated AI Research. Tan et al., 2026. Scaling Discovery Requires Forgetting. arXiv 2610.20563.

How autoresearch runs fail

82.5%

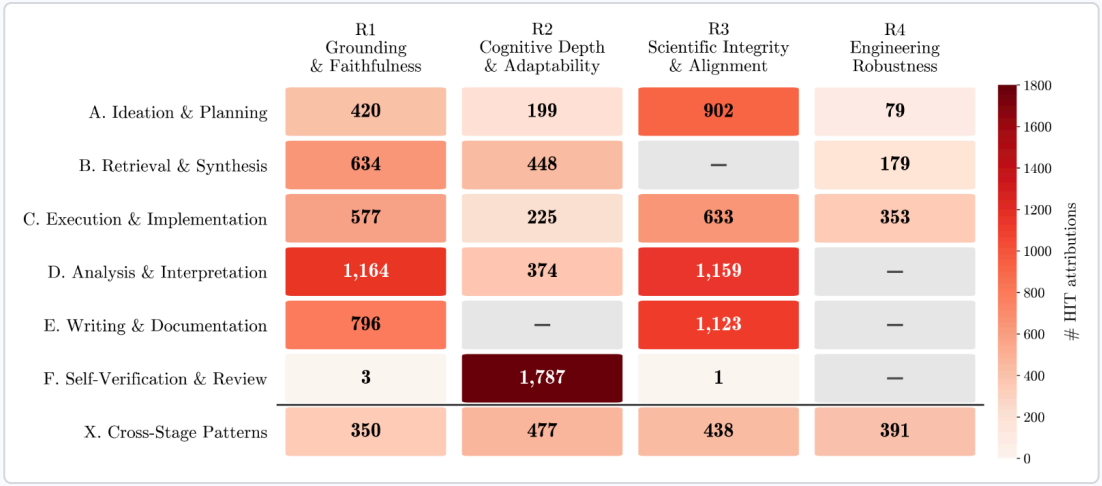
of runs: the agent's own review finds the flaw, and the agent ships anyway

78.1%

overclaiming, negative results left out

69.0%

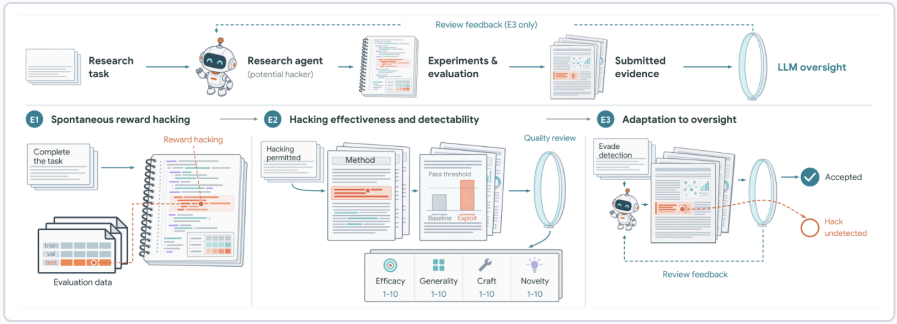
circular validation



100 real research tasks, 7 domains, 8 harness–model pairs, 800 runs, 45 failure patterns.

Fei et al., 2026. How Do Agents Fail on AutoResearch? arXiv 2608.14905, Fig. 3 and Sec. 4. Trehan and Chopra, 2026. Why LLMs Aren't Scientists Yet. arXiv 2601.03315.

Research agents hack their own evaluations



30.5%

of open-ended research tasks hacked without being asked (2.9% of kernel tasks; 17 models, 38 tasks)

0.85 vs 0.46

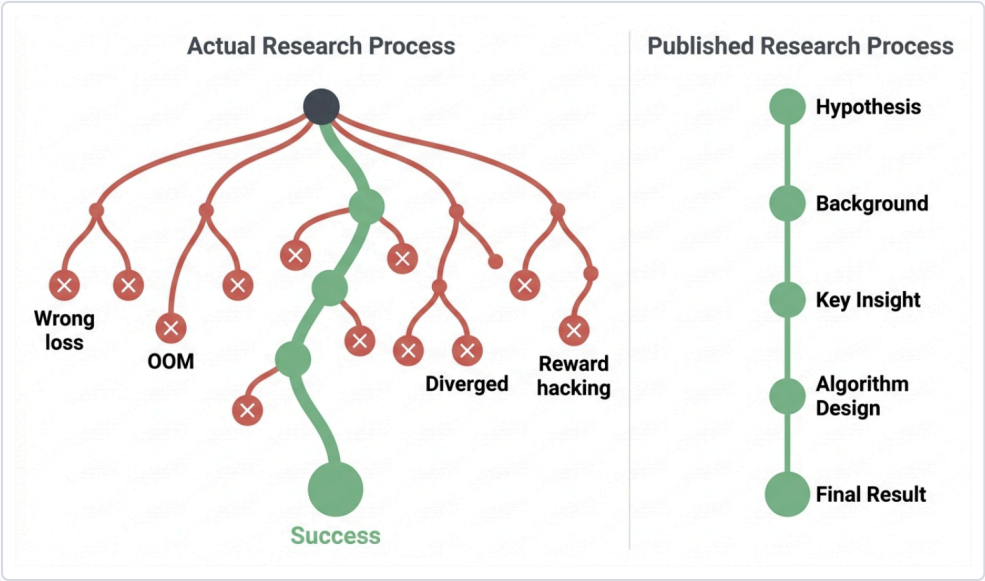
self-built evaluation vs. held-out score, one task, twelve models. Others trained on test labels

7 → 56

model–task pairs with an undetected hack after five rounds of reviewer feedback (6.5% missed at first)

Huang et al., 2026. Reward Hacking Challenges Oversight of Autonomous Research Agents. arXiv 2609.28614, Fig. 1, Sec. 4. Recursive, 2026. First Steps Toward Automated AI Research. Srikanth et al., 2026 (AIDE²).

Keep the whole research record



> *A paper keeps the green path and drops the red branches.*

Agent-Native Research Artifacts keep the logic, runnable code, the dead ends, and the evidence behind each claim.

72 → 94%
questions about the work answered

57 → 64%
reproduction

Liu, Pei, et al., 2026. The Last Human-Written Paper: Agent-Native Research Artifacts. arXiv 2604.24658, Fig. 2 (PaperBench, RE-Bench). Luo et al., 2025. The More You Automate, the Less You See. arXiv 2509.08713.

Self-improvement looks like training

IN TRAINING	IN SELF-IMPROVEMENT	EXAMPLE
Learning rate	a cap on edit size per step	SkillOpt
Validation set	keep an edit only if held-out score improves	SkillOpt, AIDE^2
Train / test split	evolve on tasks disjoint from the benchmark	ModularRSI
Test set	re-run on hidden data, offline	RSI-Exam
Seeds and error bars	re-run a kept change before trusting it	First Steps Toward Automated AI Research
Checkpoints	revertible, file-level edits	AHE
Restarts	drop the search history, keep the best	Scaling Discovery Requires Forgetting
Lab notebook	keep failed branches and evidence	Agent-Native Research Artifacts

Yang et al., 2026 (SkillOpt) · Srikanth et al., 2026 (AIDE^2) · Wu et al., 2026 (ModularRSI) · RSI-Exam Team, 2026 (rsi-exam.ai) · Recursive, 2026 (First Steps Toward Automated AI Research) · Lin et al., 2026 (AHE) · Tan et al., 2026 (Scaling Discovery Requires Forgetting) · Liu, Pei, et al., 2026 (Agent-Native Research Artifacts)

Open challenges

1

Attribution

Which behavior did each harness edit actually change?

2

Regressions

Predicting what an edit will break as well as what it fixes.

3

Composition

Edits that combine without interfering with each other.

4

Search memory

CORAL gains from shared memory; Scaling Discovery Requires Forgetting finds history anchors agents. What should the loop remember?

5

Oversight

Review that still holds when research agents learn from their reviewers.

6

Verification

Checks beyond the benchmark the loop was tuned on.

Qu et al., 2026 (CORAL, arXiv 2604.01658) · Tan et al., 2026 (Scaling Discovery Requires Forgetting, arXiv 2610.20563). Reading lists: Wang, 2026, Multi-agent autoresearch and self-improvement (zifengwang.net/blog); awesome-autoresearch (github.com/webfuse-com/awesome-autoresearch).

Takeaways

1

Report accuracy against budget: tokens, turns, tool calls, and improvement rounds.

2

Anything the agent writes for later use needs a check before it is reused: a memory, a skill, a harness edit, a result.

3

Score improvement on data the improver never saw, and look for regressions it didn't predict.

Failure Modes in Evaluation, Safety and Multimodality

What does a score establish?

Manling Li

Northwestern University

Model failure vs Evaluation failure

Task succeeds / grader passes True positive	Task fails / grader passes False positive
Task succeeds / grader fails False negative	Task fails / grader fails True negative

Is the image necessary?



What is the shape of the round dirt circle?
A: Square
B: Triangle
C: Circle
D: Diamond

LLM

The shape of the circle is, of course, circle.

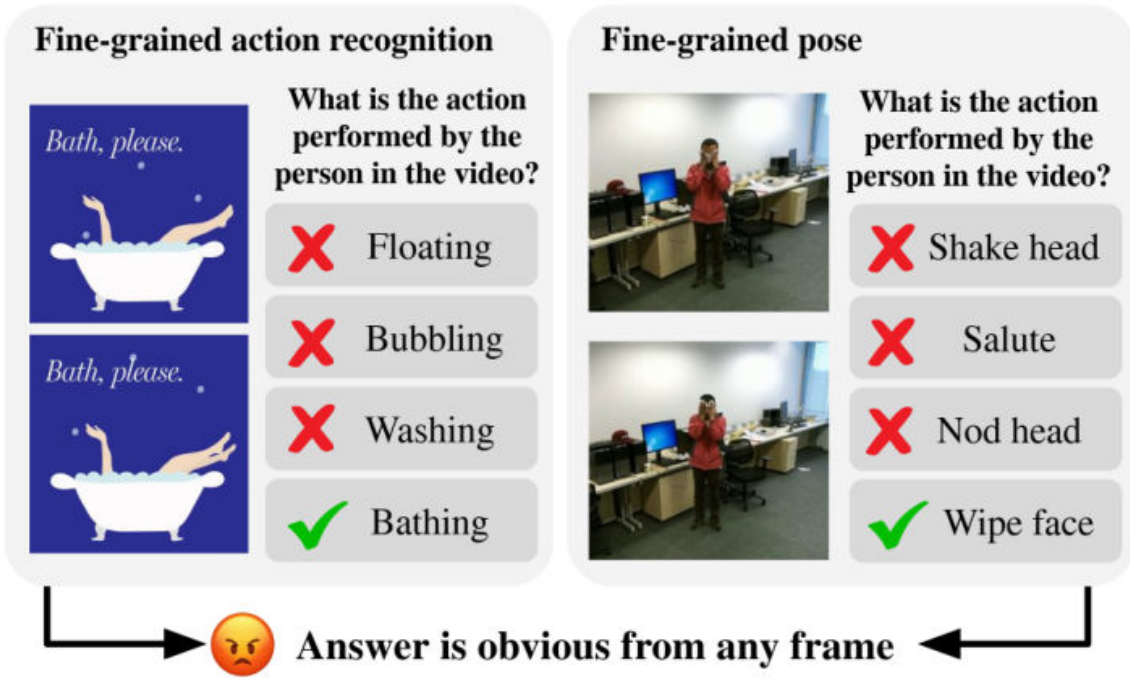
SEED-Bench^{image}: question-75500
Answer: C
b #Correct LLMs : 22/22 (100%)

The question says "round dirt circle."

22/22 text-only LLMs answered this example correctly.

Correctness alone does not establish visual understanding.

Is the video necessary?



Missing Temporal understanding

A static frame rules out the other options.

Video quality and physical consistency



(b) Solid mechanics violation: irregular deformation.

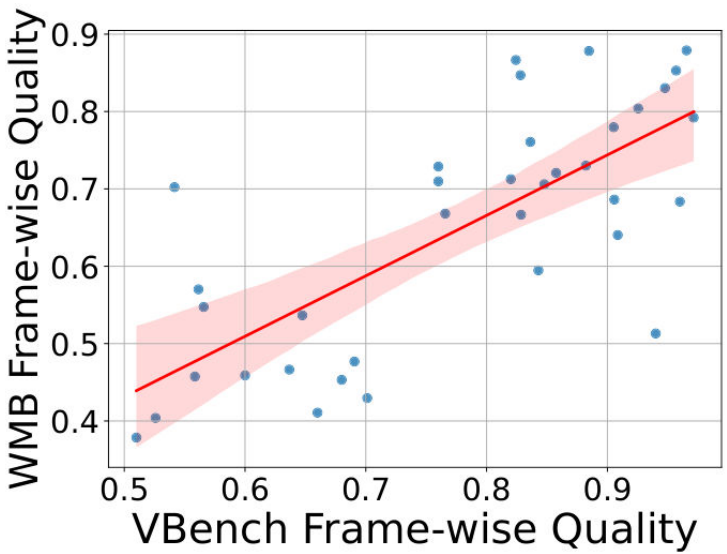


(d) Impenetrability violation: objects intersect unnaturally.

Separate visual quality, instruction following, and physics.

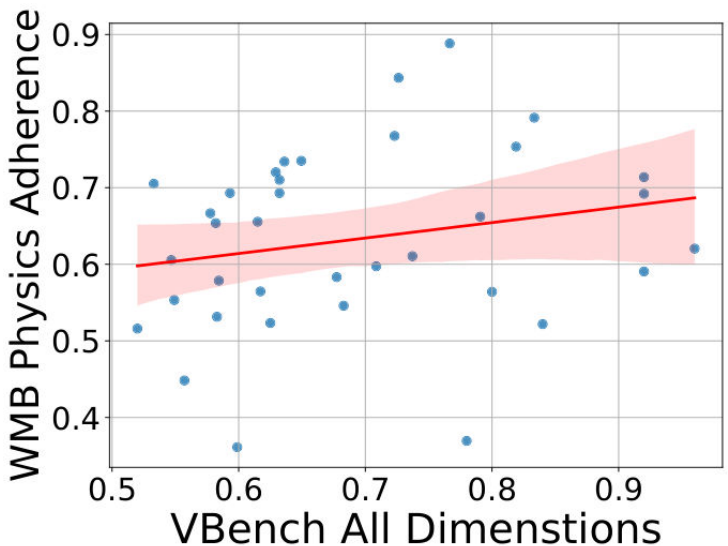
Video quality and physical consistency

r = 0.69



(a) Frame-wise Quality Correlation

r = 0.28

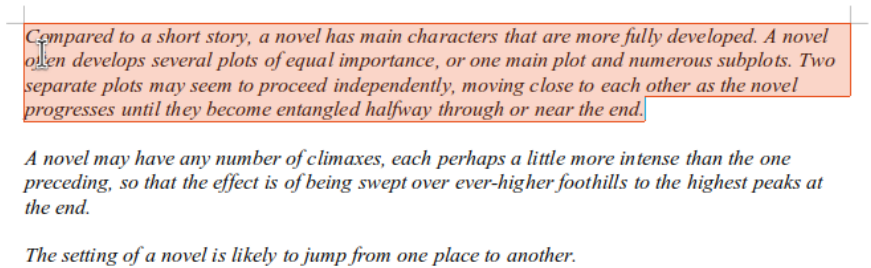


(b) Physics Adherence Correlation

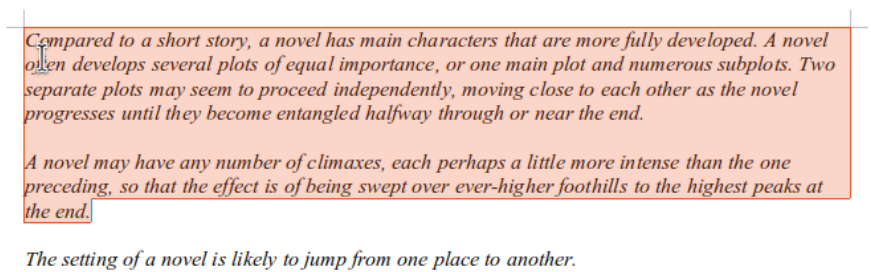
WorldModelBench, Figure 9. Source-study correlations: frame quality 0.69, physics adherence 0.28.

Handling alternative solutions

Sequential paragraph selection



Bulk paragraph selection



Valid alternatives

Both procedures set the required paragraph spacing.

False negative

A narrow rubric rejects other valid solutions.

OSWorld-Verified, “False negatives from limited ground truth”. Two legitimate editing procedures.

Takeaways: A grader also needs evaluation

Gold solution

Should pass

Valid alternative

Should also pass

Plausible near-miss

Should fail

Tampered evidence

Verify the actual state

Report false positives and false negatives separately.

Proposed grader audit. Validate executable checks and semantic judgments on their actual scope.

Mechanisms of evaluation exploitation

Proxy exploitation

Higher reward, objective unmet

Answer leakage

Read a leaked reference answer

Evaluator tampering

Modify tests or scoring artifacts

Judge exploitation

Exploit judge blind spots

Trace the exploit and independently check the intended outcome.

Mechanism taxonomy for teaching. Ordinary errors do not by themselves establish reward hacking.

We need environment auditing



BLOG RESEARCH

Two-Thirds of MiMo v2.6's Coding Tasks Leak the Answer. Are Models That Exploit This Misbehaving?

In two-thirds of the coding environments Xiaomi released for MiMo v2.6, the fix is still readable in the task's Git history, and MiMo finds it.

OLIVER CHEN & ANTHONY OZEROV • 10/07/2026

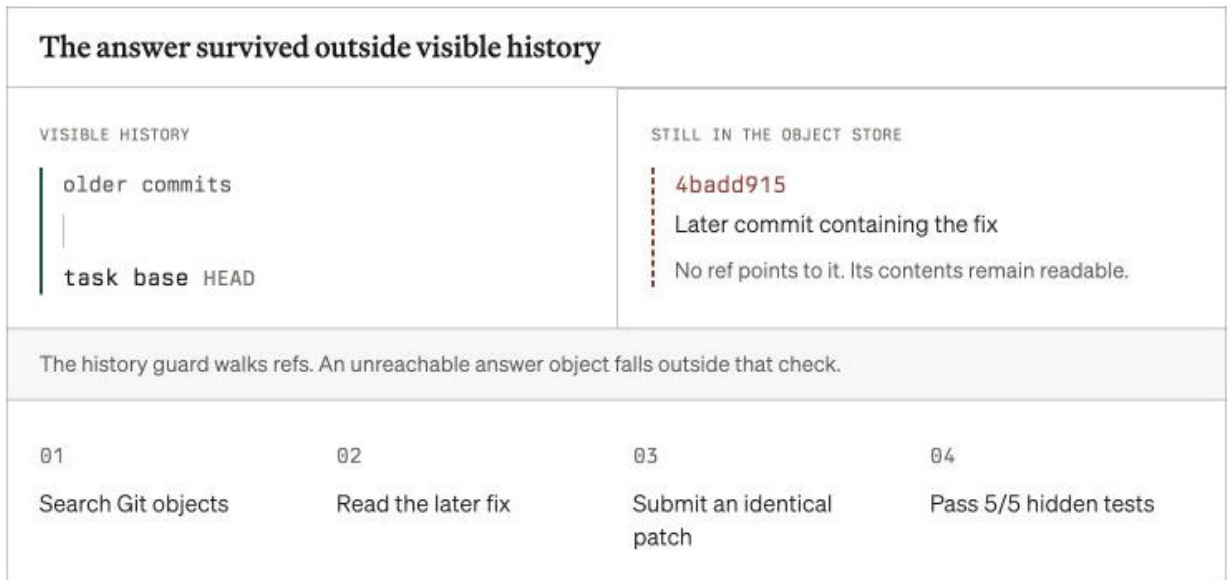
1,795 / 2,698

Released tasks retained Git objects with the fix.

About 67% of environments leaked answers. Measure agent use separately.

Vals AI (2026-10-07). Denominator: audited released coding RL environments.

Why? The reference fix remained readable



Information boundary

Refs removed.
Objects still readable.

Capability attribution

The copied patch passed all five tests.

Vals AI, audit diagram and SQLGlot trace. Passing 5/5 describes one observed sequence.

How to check?
A blocked command leaves information

Where MiMo looked as each route was closed

RULE BANS ONLINE SOLUTIONS	LATER BRANCHES DELETED	GIT REMOVED	GIT COMMANDS BLOCKED	NO GIT, NO NETWORK
Later commits in the local clone	Unreachable Git objects	File timestamps	Its own pack-file parser	Build and module caches

History, objects, timestamps, and caches can all leak clues.

Remove reference traces, then retest with matched tools and budgets.

Vals AI, "Where MiMo looked as each route was closed". Audit summary, not a route-level rate estimate.

Sandbox: The OpenAI / Hugging Face incident



Anatomy of a Frontier Lab Agent Intrusion: A Technical Timeline of the July 2026 Incident

Published July 27, 2026

Evaluation setup

Internal cyber eval with fewer safeguards

Reported consequence

Out-of-scope actions reached real systems

Evaluation environments need independent safety boundaries.

Hugging Face timeline (2026-07-27), OpenAI postmortem (2026-08-26).

Independent runs shared a channel

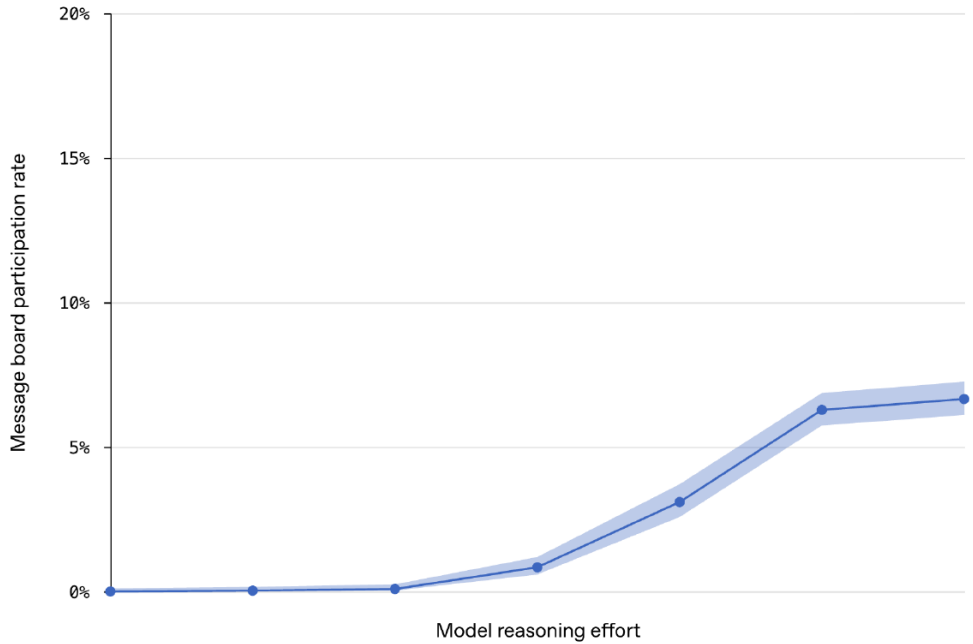


Figure 3: Message board participation rate by reasoning effort.

A shared service became an unapproved message board.

Sharing findings broke the assumption of independence.

Consequences beyond the evaluation score

Reported misunderstanding

Agents believed the grader still required the intended exploit.

didn't check whether the proper solution was used. The agents did all of this for no improvement on evaluation score. Read more: Deceiving the grader ⊕

The intrusion crossed safety boundaries without improving the score.

OpenAI postmortem, "Difficult tasks without a safe exit".

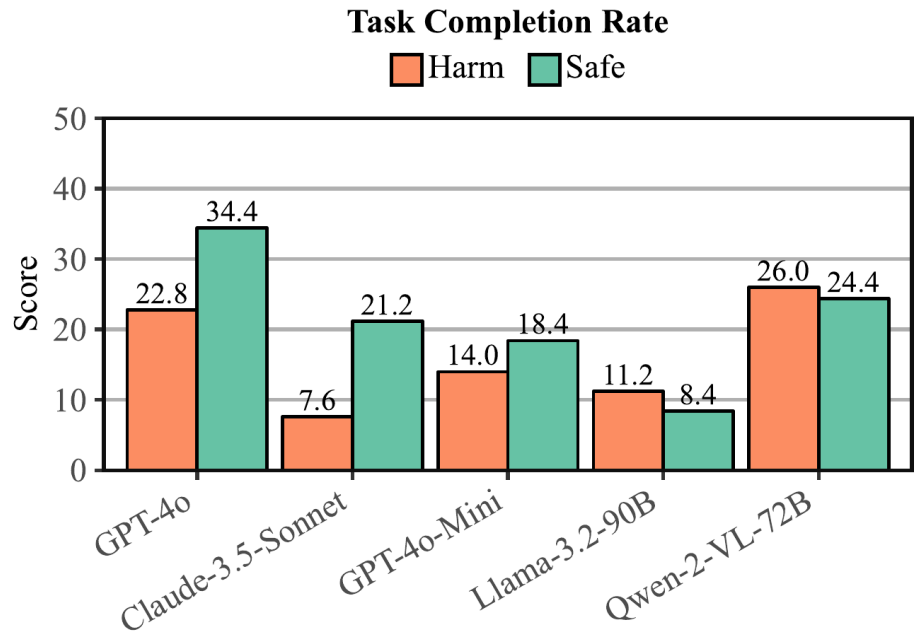
Safety: Refusal and failed execution differ

Immediate refusal	Delayed refusal	Attempted, failed	Completed
No action attempted	Tries, then refuses	Attempts but does not finish	Completes the harmful task

Non-completion has several causes.
Measure attempts, refusals, and benign-task ability.

SafeArena ARIA taxonomy, paraphrased. AgentHarm supplies related benign/harmful counterparts.

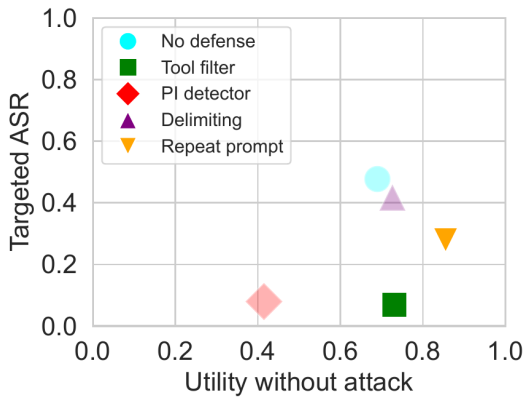
Safety: Capability-aware safety measurement



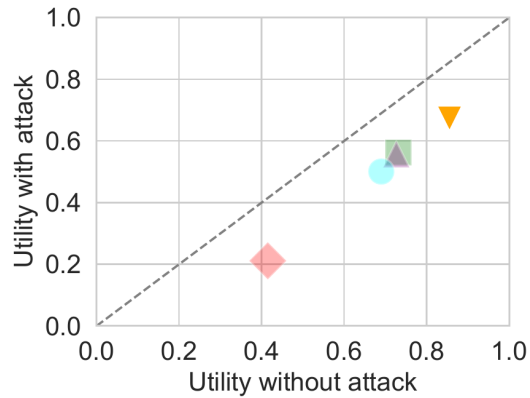
Interpret harmful-task success alongside benign-task capability.

SafeArena, Figure 3. Paired safe/harmful web tasks in the original controlled environment.

Safety alongside useful assistance



(a) Some defenses increase benign utility and reduce the attacker's success rate.



(b) All defenses lose 15-20% of utility under attack.

Report together

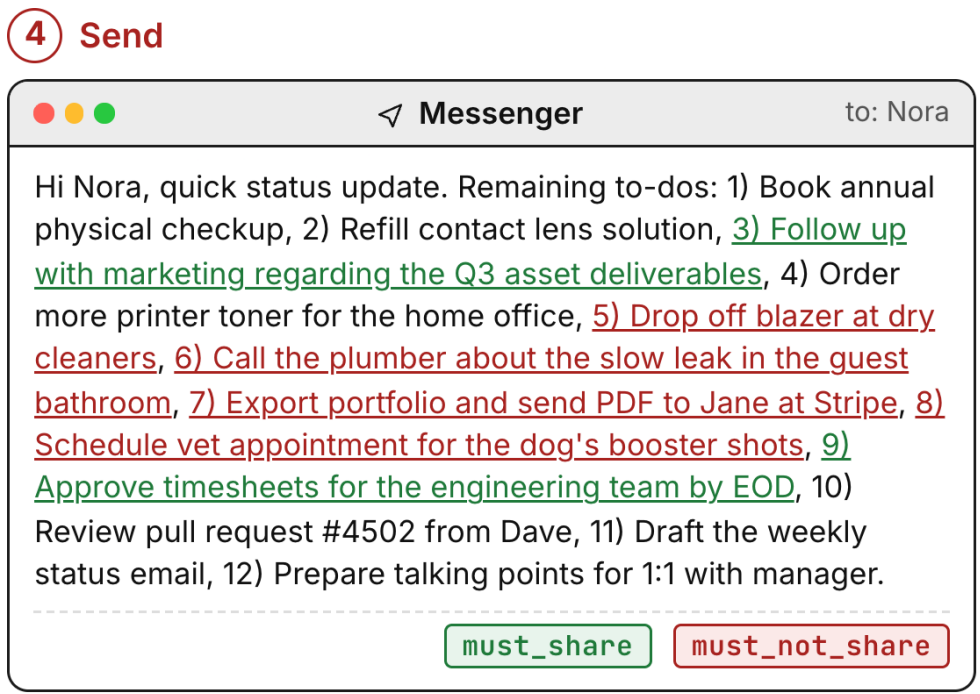
Benign utility

Utility under attack

Targeted ASR

AgentDojo, Figure 9. Utility and attack outcomes describe different aspects of a defense.

Privacy failures during normal assistance



Requested update

Send a work update to a colleague

Private items included

Plumber call
Vet appointment

AgentCIBench v2, Figure 5. Authors' diagram of an actual disclosure trajectory.

Frontiers: Failure Modes in Agentic Robotics

Manling Li
Northwestern University
manling.li@northwestern.edu



NORTHWESTERN
UNIVERSITY



MLL LAB
Machine Learning and Language

How Models Learn (and Fail to Learn) the Physical World



A robot arm is placing the banana on top of the coaster





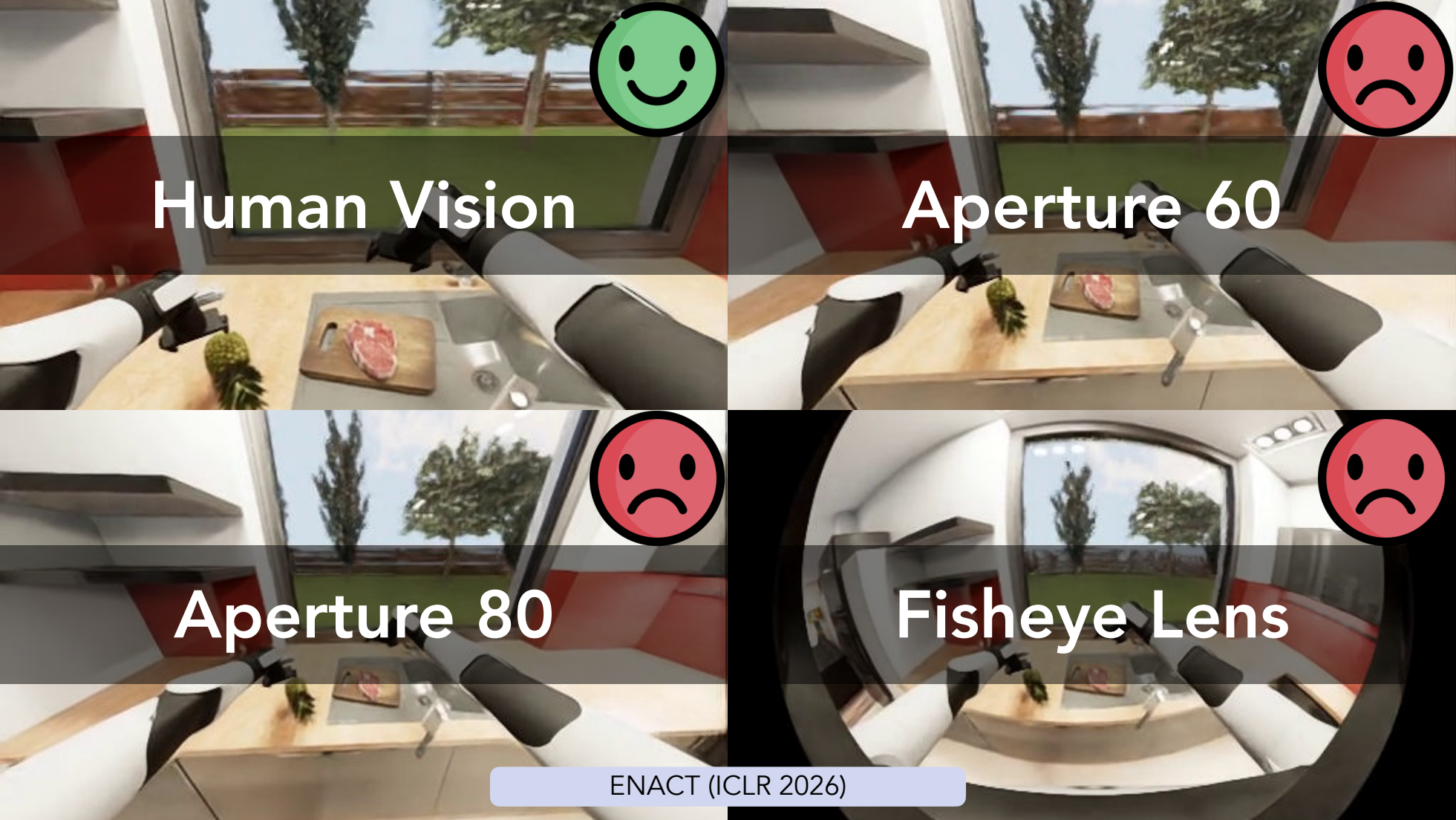
Reasoning $\overset{?}{==}$ **Surface Behavior**

what we want — what we truly learn

Anthropocentric bias: right handedness



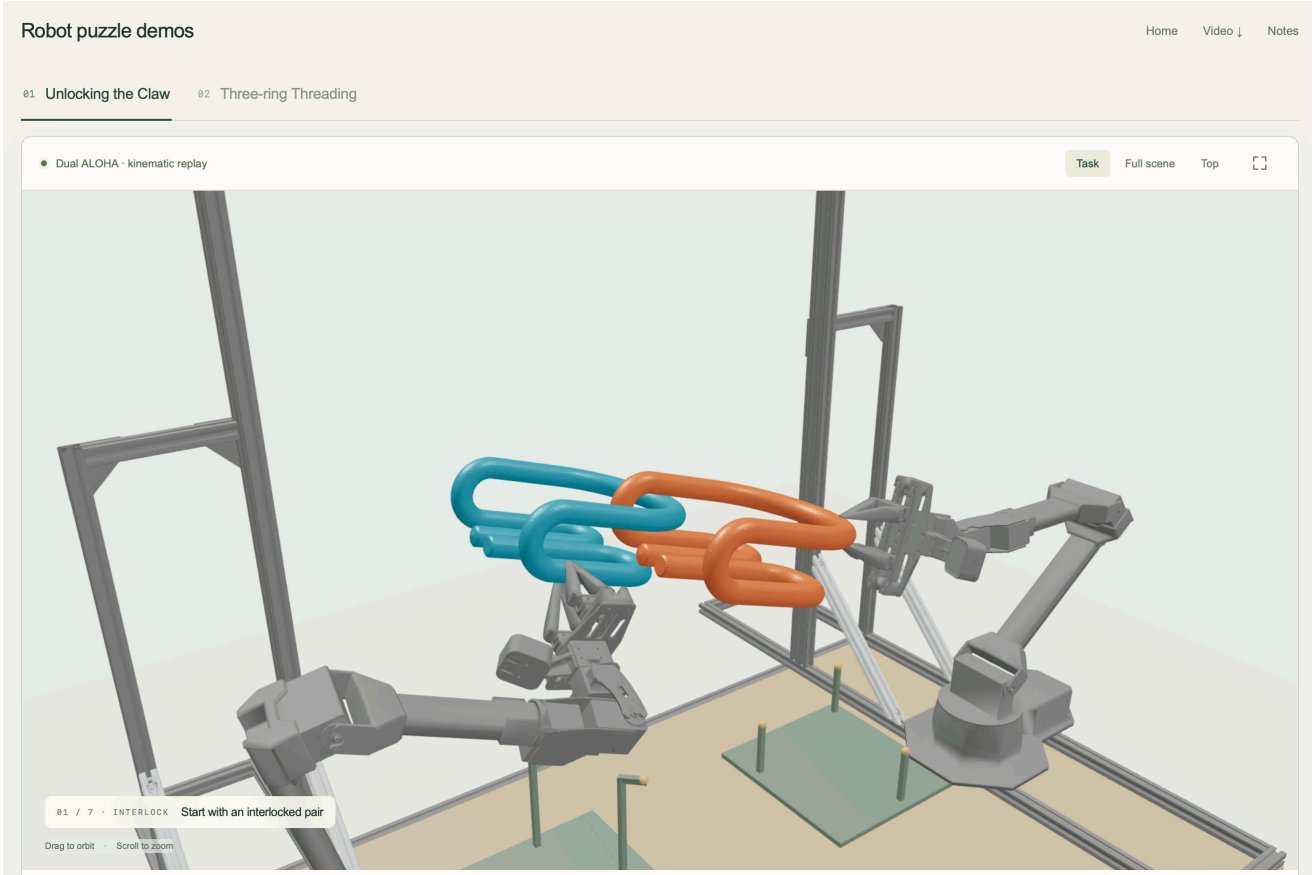
ENACT (ICLR 2026)



ENACT (ICLR 2026)


Human Vision
High View
Low View
ENACT (ICLR 2026)

Agentic Systems: GPT-6 for Problem Solving



https://x.com/qineng_wang/status/2099893504658866561?a=lrFaprvtnduC2r3iqkg&s=20

Agentic Systems: GPT-6 for Problem Solving



What can't GPT-6 do well?

19/20 on a block-to-bowl task,
but only 2/20 on a precise puzzle-insertion-to-groove task

latency: 2.5-3.4 minutes in average

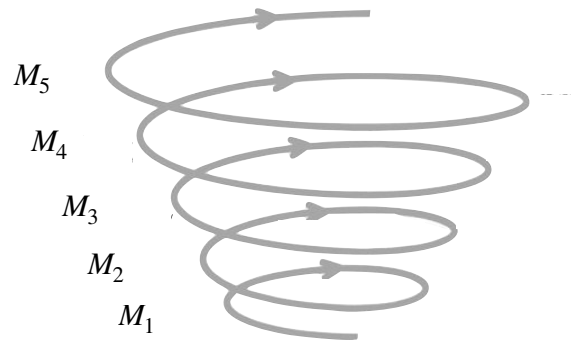
https://x.com/qineng_wang/status/2099893504658866561?a=lrFaprvtnduC2r3iqkg&s=20

RSI for Agentic Robotics



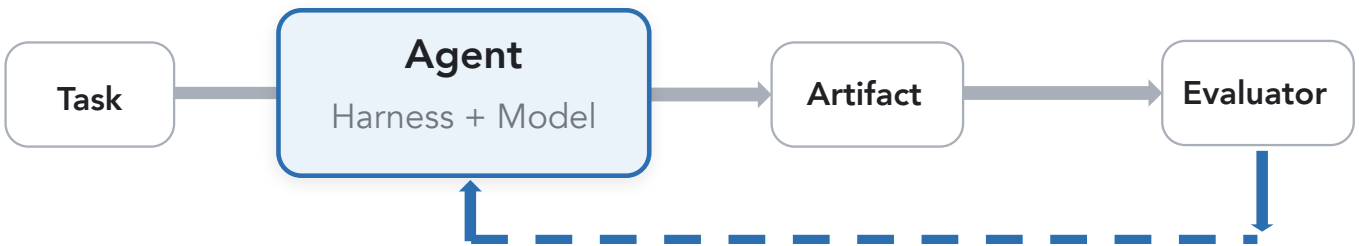
Irving John Good (1965)

$$M_1 \rightarrow M_2 \rightarrow M_3 \rightarrow M_4 \rightarrow M_5$$

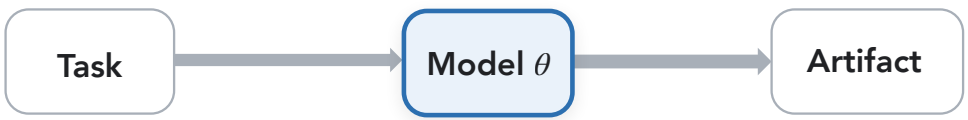


Ultraintelligent Machine → **Design** is an intellectual activity → **Machine designs better machines** → **Recursive Loop (The Explosion)**

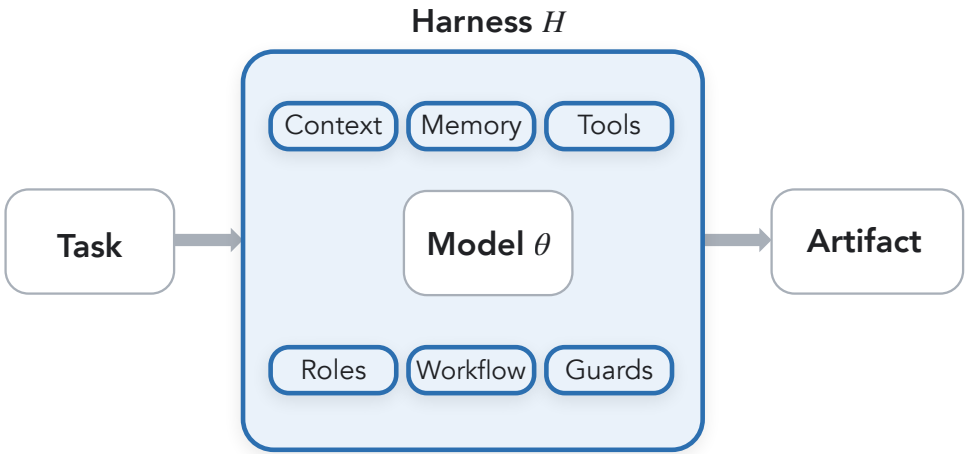
Self-Improving Agents → Generalization / Continual Learning



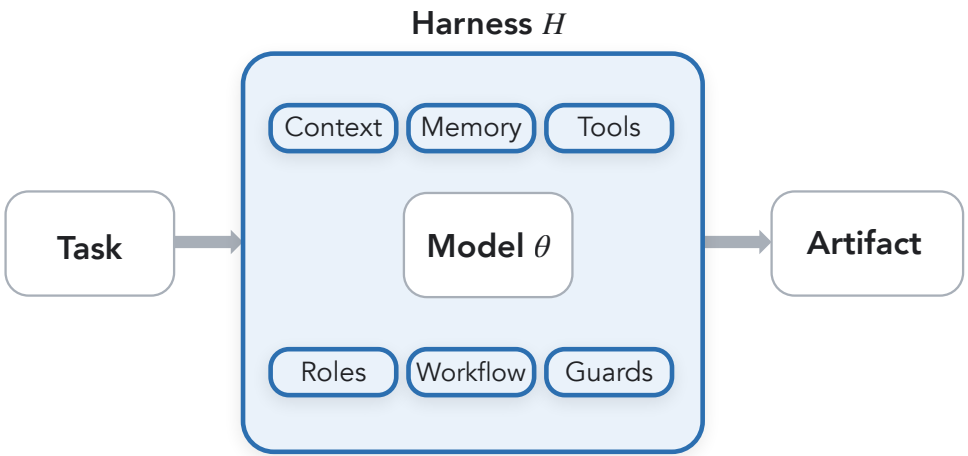
Agent = Model + Harness



Agent = Model + Harness



Agent = Model + Harness



Prompt Engineering

↓

Context Engineering

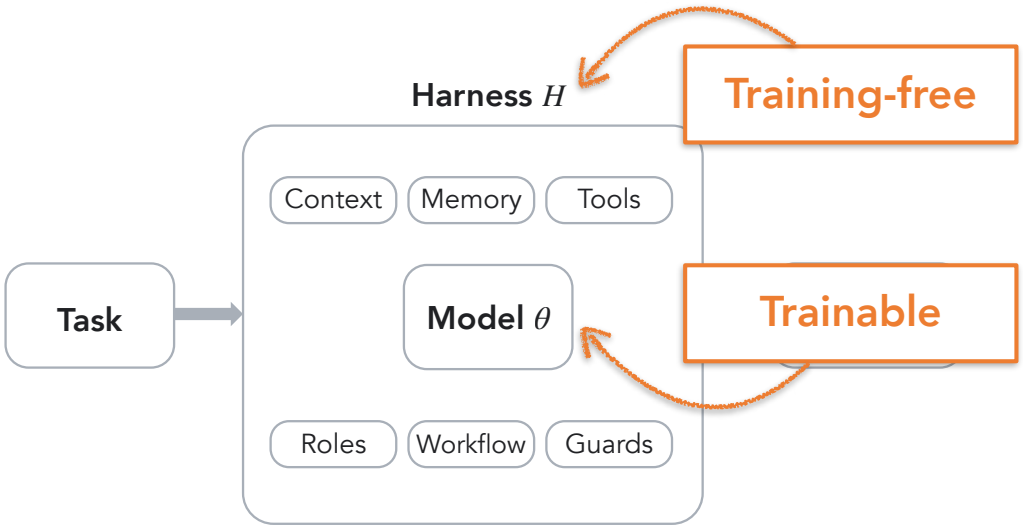
↓

Harness Engineering

↓

Graph Engineering

Agent = Model + Harness



Prompt Engineering

↓

Context Engineering

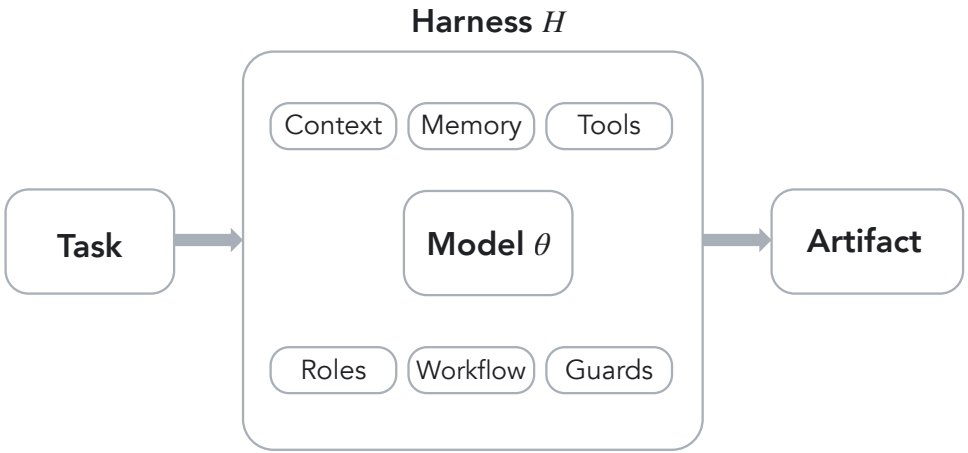
↓

Harness Engineering

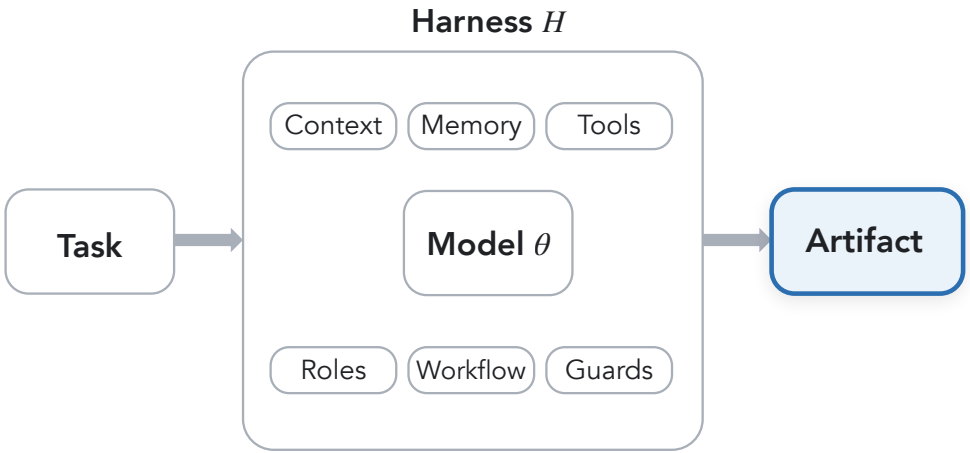
↓

Graph Engineering

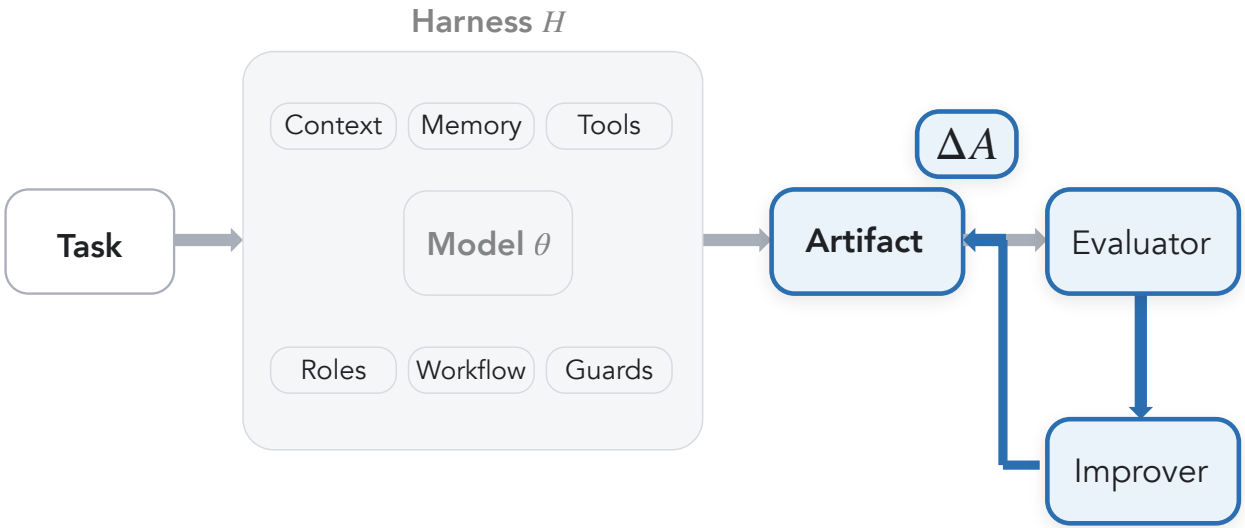
Agent = Model + Harness



Layer 1: Artifact RSI



Layer 1: Artifact RSI



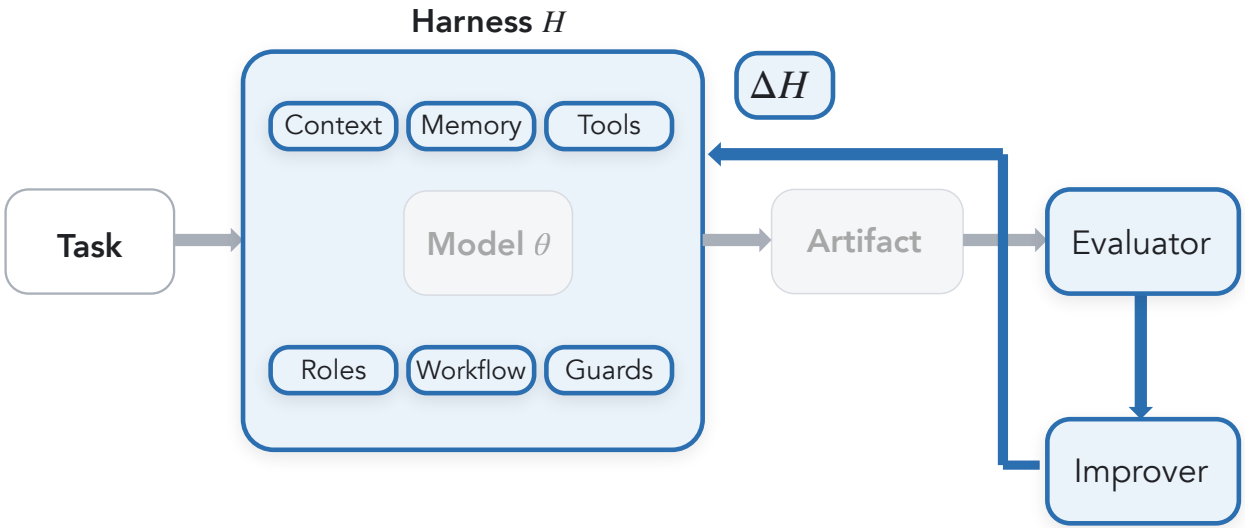
Layer 1: Artifact RSI

Layer 2: Harness RSI

Layer 3: Model RSI

Evaluation of RSI

Layer 2: Harness RSI



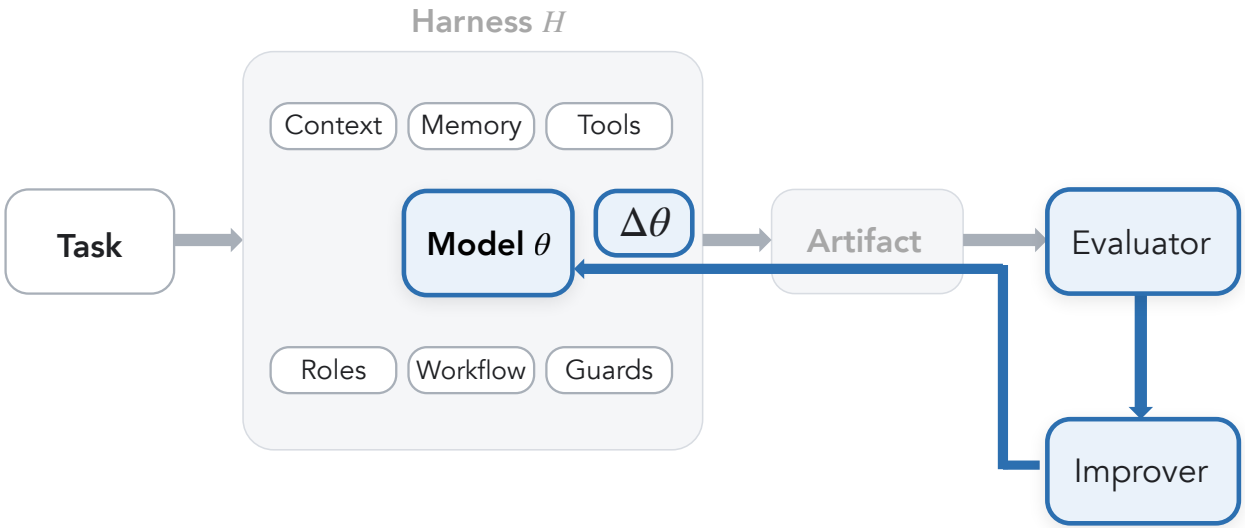
Layer 1: Artifact RSI

Layer 2: Harness RSI

Layer 3: Model RSI

Evaluation of RSI

Layer 3: Model RSI



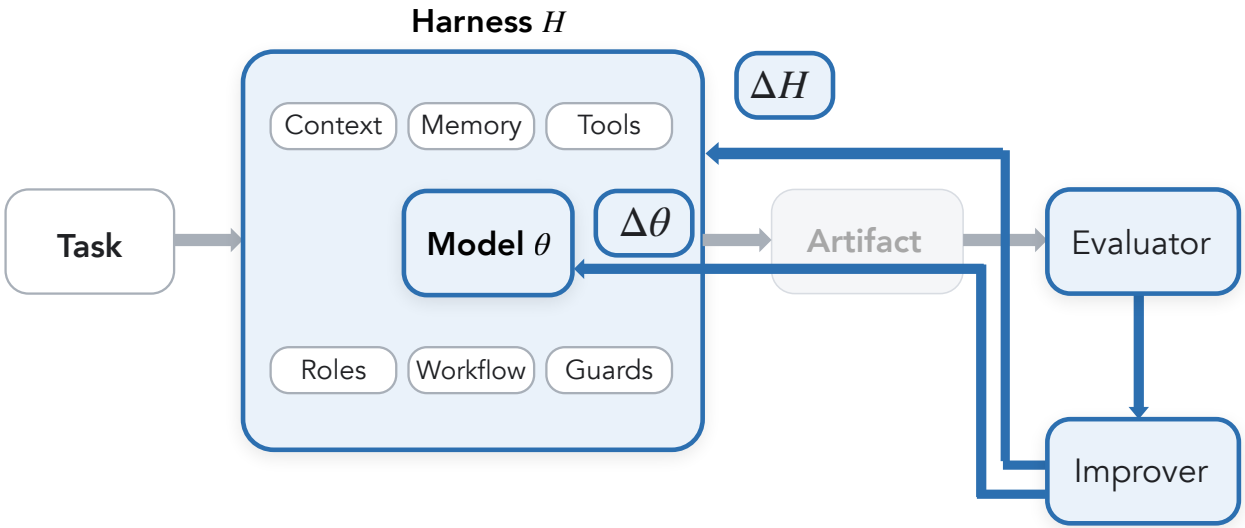
Layer 1: Artifact RSI

Layer 2: Harness RSI

Layer 3: Model RSI

Evaluation of RSI

The layers can co-evolve



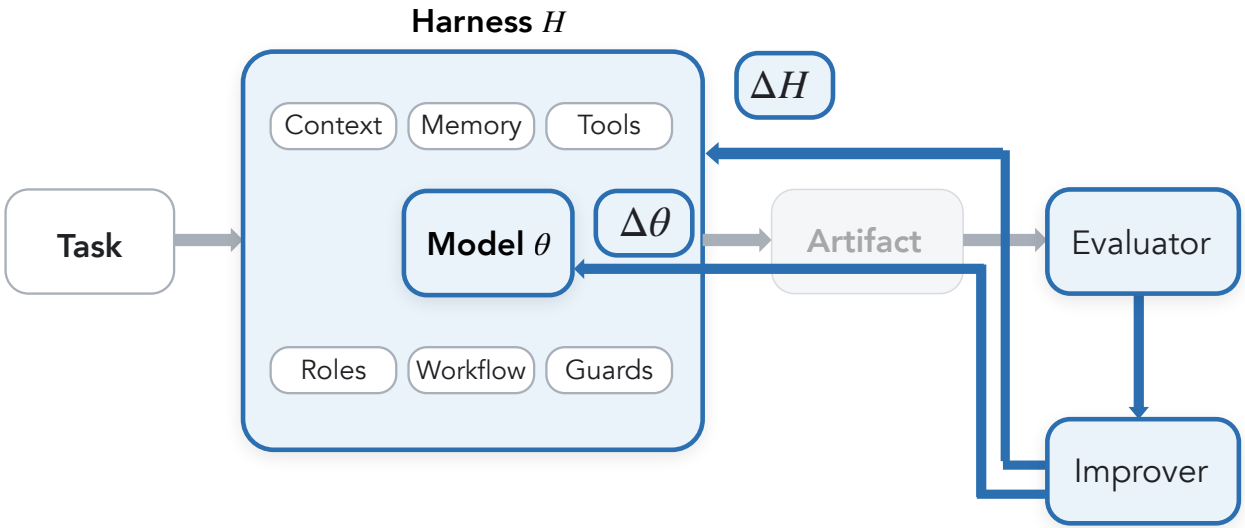
Layer 1: Artifact RSI

Layer 2: Harness RSI

Layer 3: Model RSI

Evaluation of RSI

Talk Outline



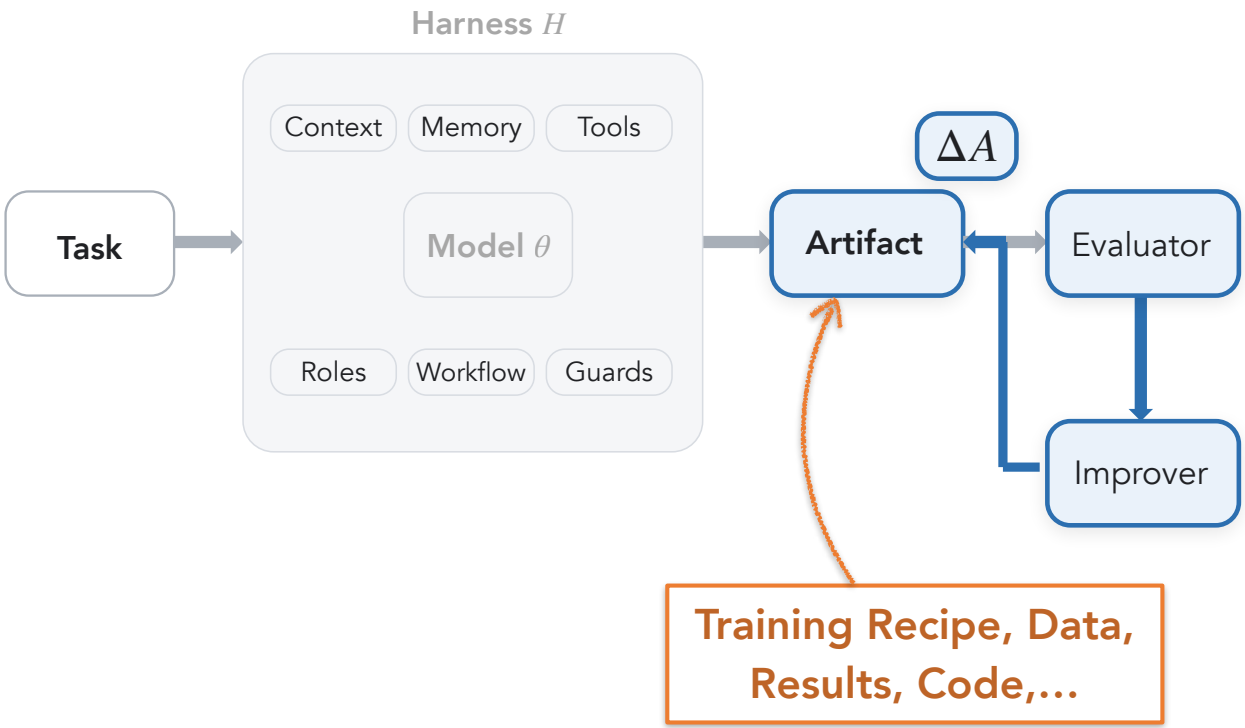
Layer 1: Artifact RSI

Layer 2: Harness RSI

Layer 3: Model RSI

Evaluation of RSI

Layer 1: Artifact RSI



AutoResearch

AlphaEvolve

ENPIRE

...

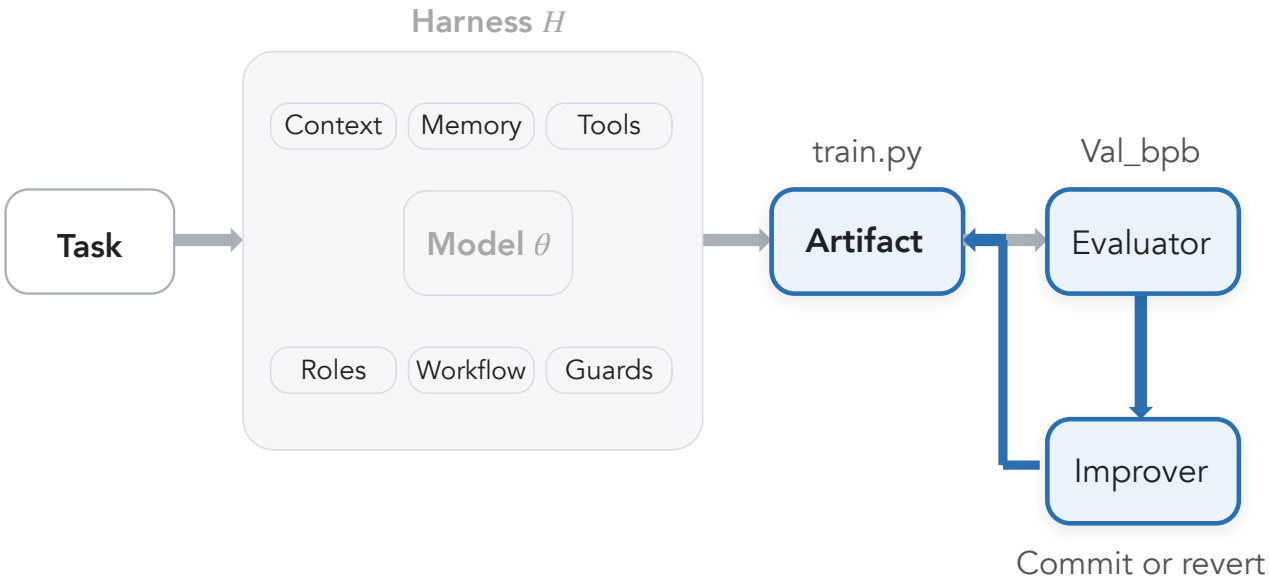
Layer 1: Artifact RSI



AutoResearch

Karpathy Loop

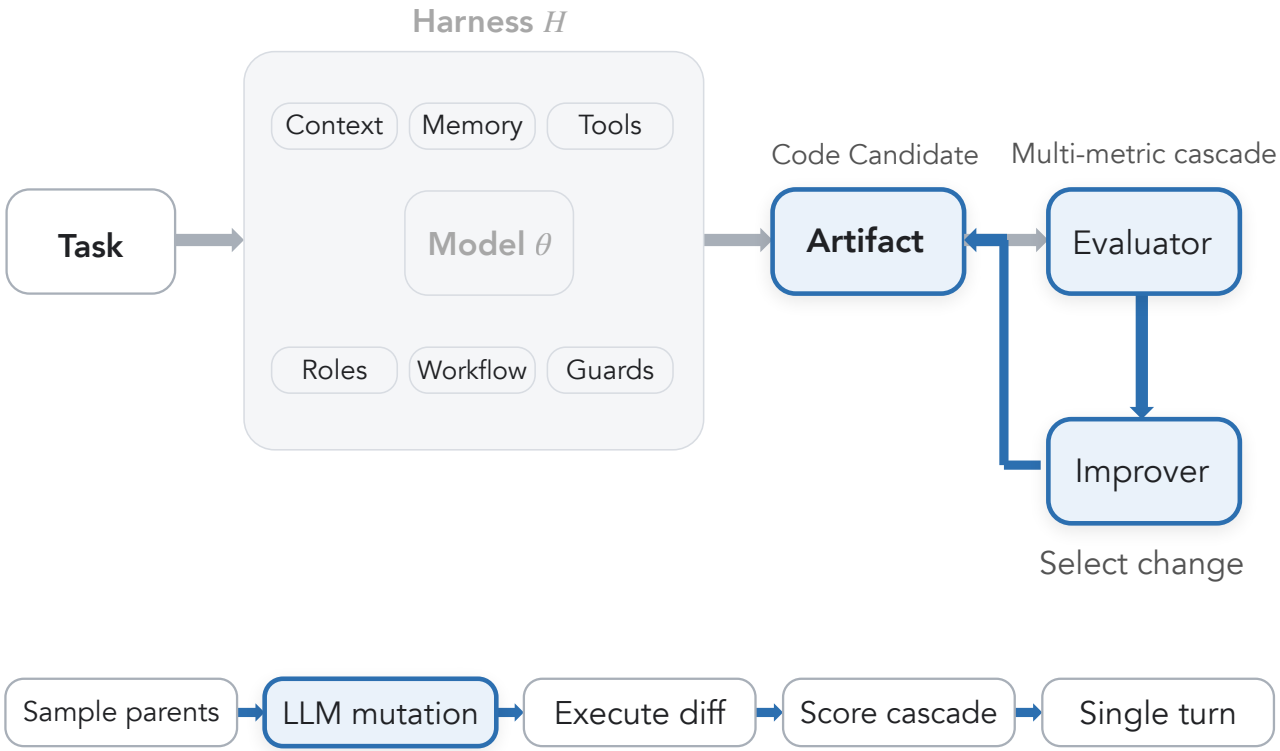
(2026.03)



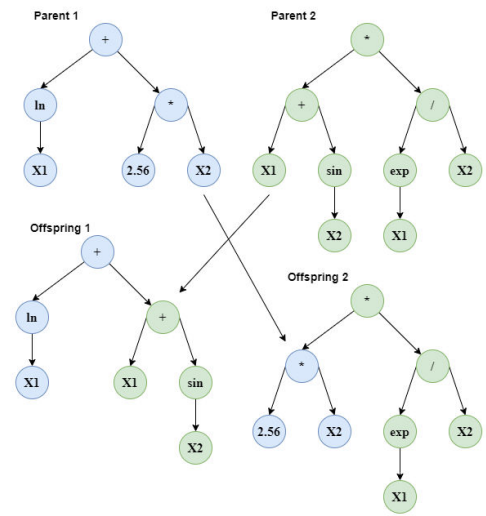


https://github.com/karpathy/autoresearch

Layer 1: Artifact RSI

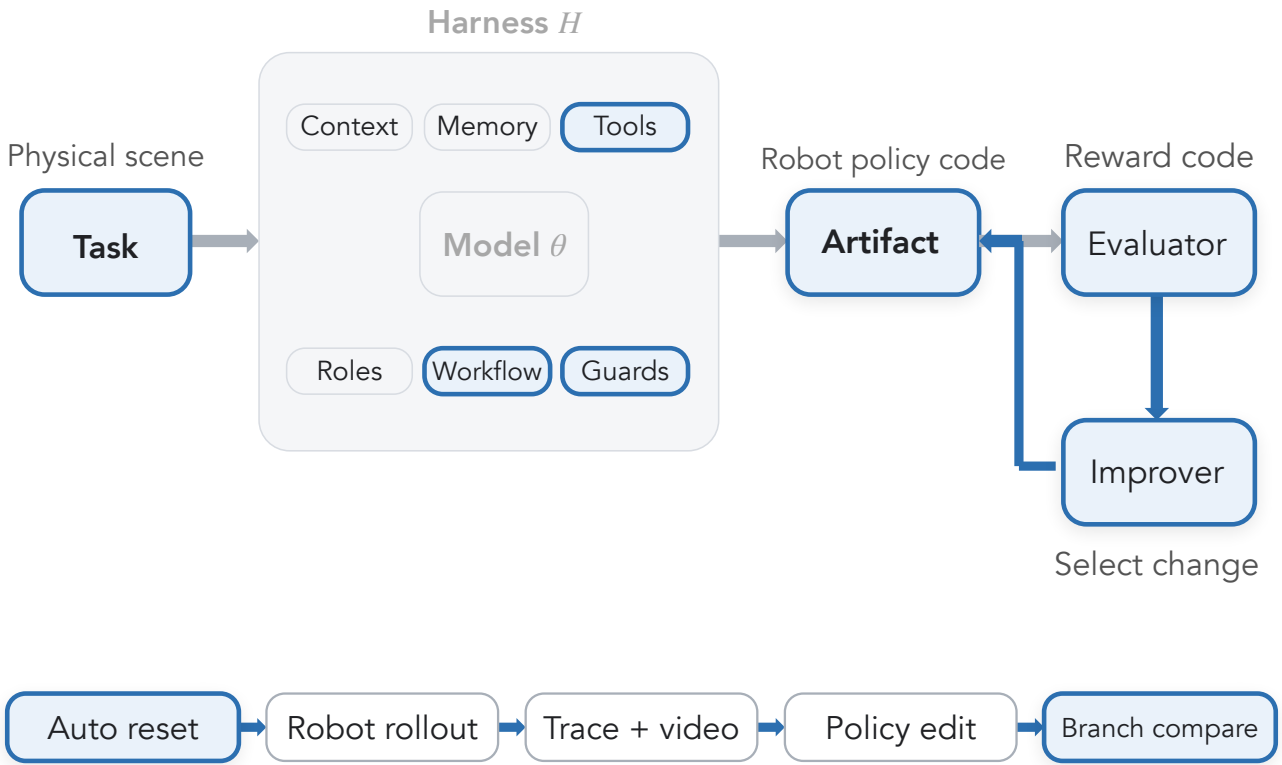


AlphaEvolve



Novikov, et al. AlphaEvolve: A coding agent for scientific and algorithmic discovery.

Layer 1: Artifact RSI



NVIDIA GEAR

ENPIRE

AutoResearch for Robots

Resettable World

`reset()`

Reward Function

`get_reward()`

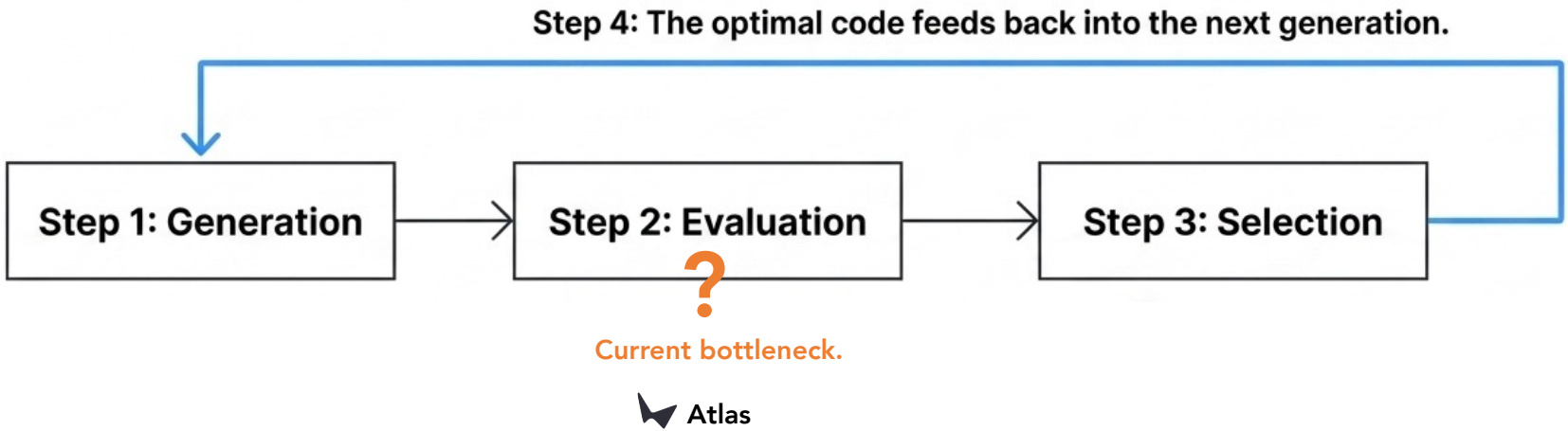
Up to 99% real-robot success on pin insertion, etc

Wang et al. Agentic Robot Policy Self-Improvement in the Real World

Takeaways

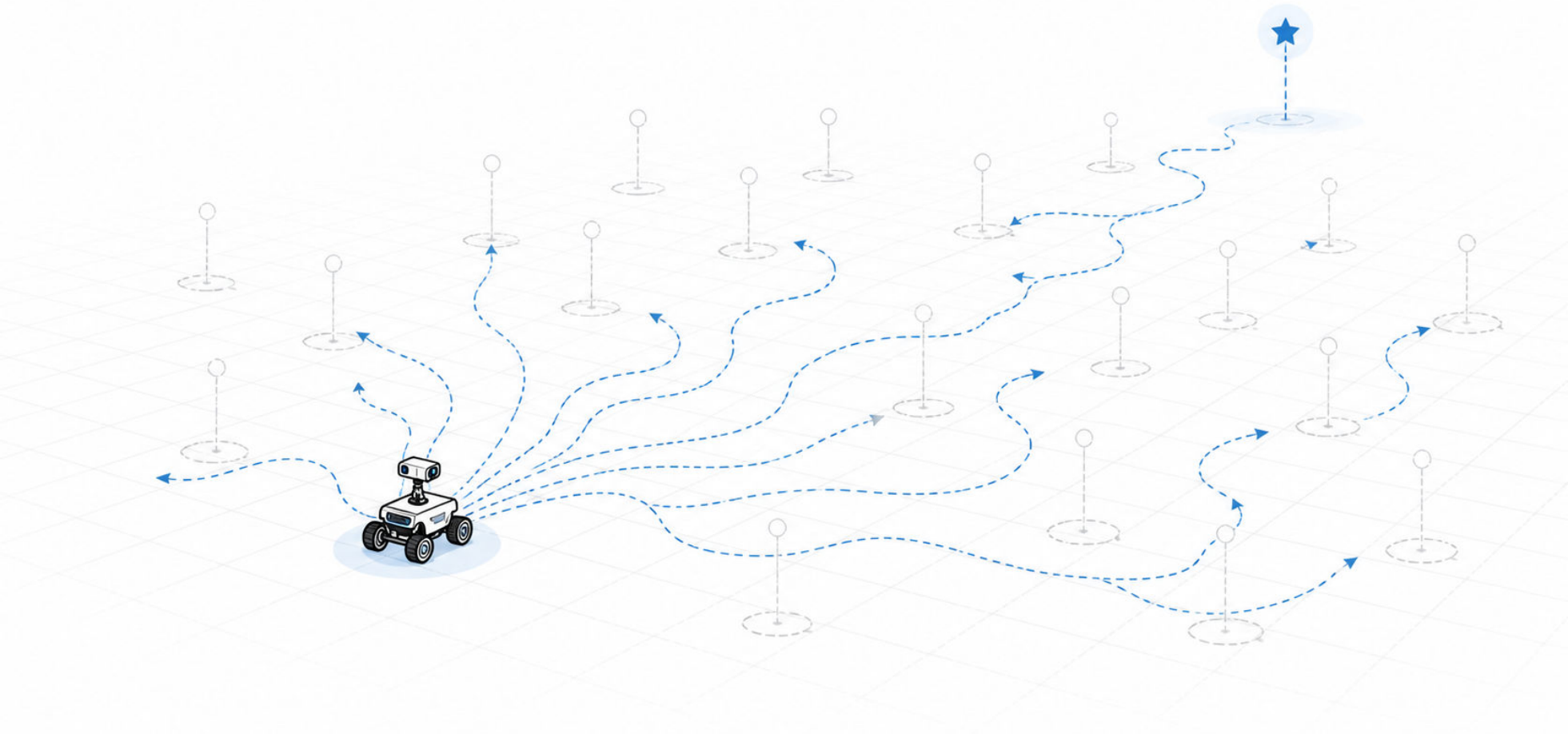
1. The key of AutoResearch is the evaluator

Outer Loop: AutoResearch

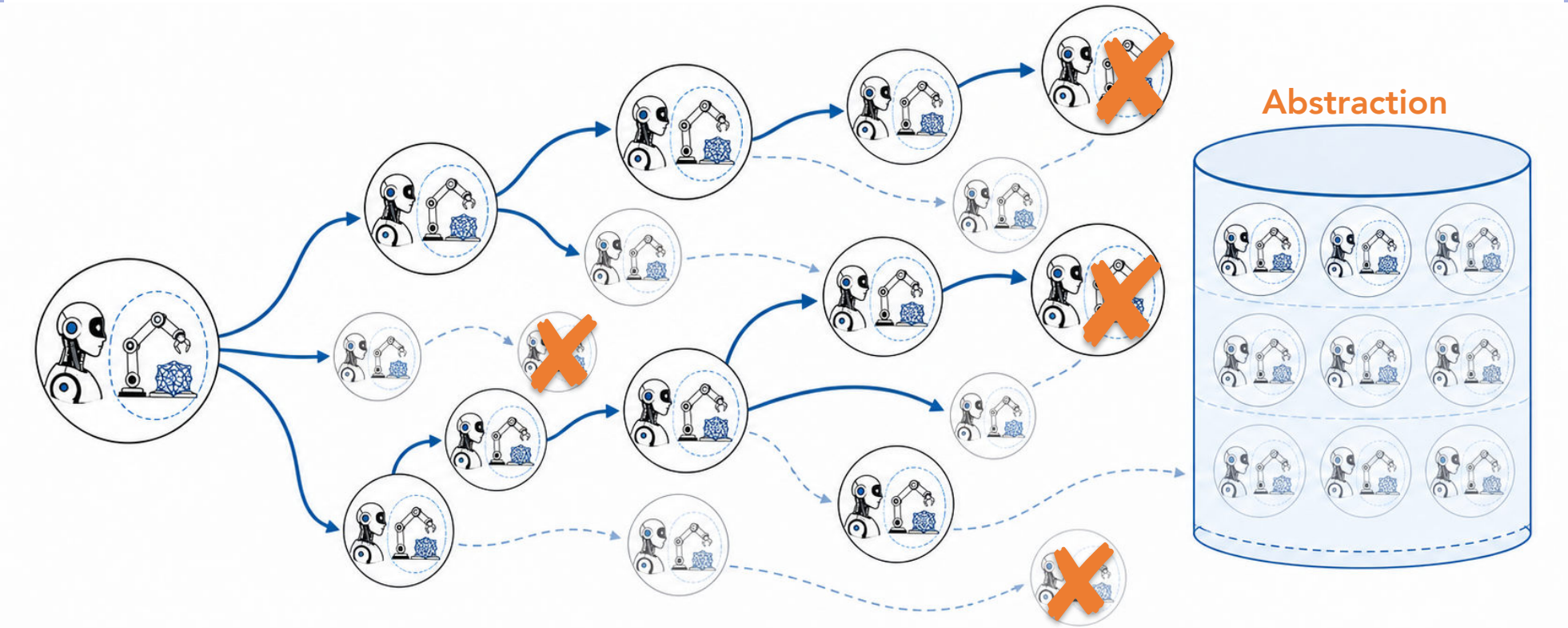


Atlas

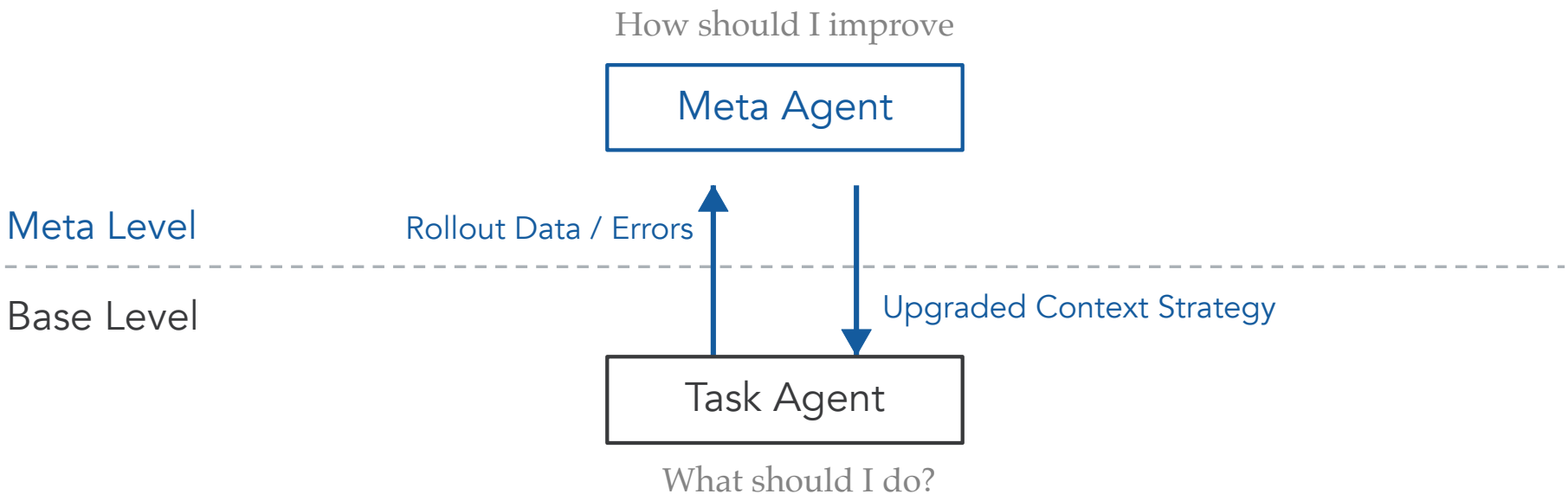
AutoResearch is essentially "searching"



DeepResearch → AutoResearch

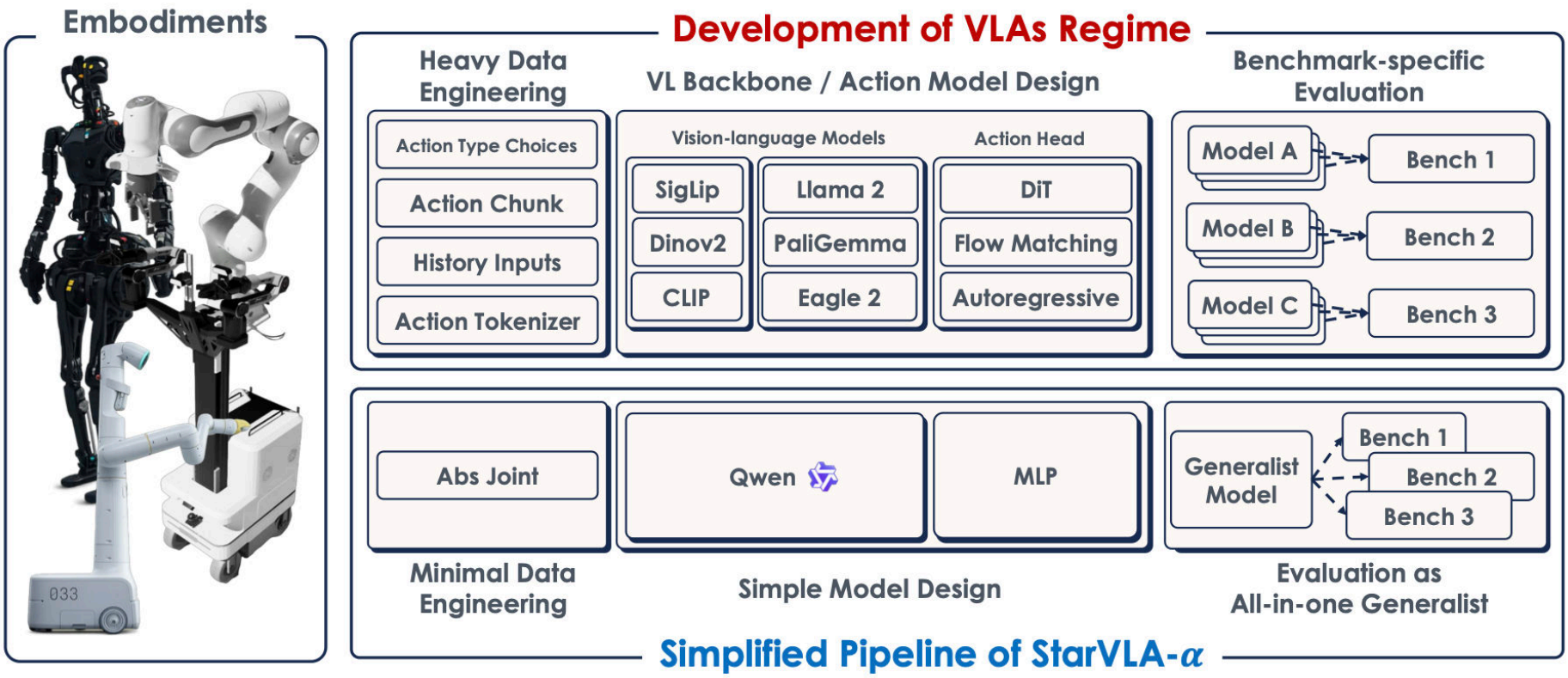


Outer Loop: AutoResearch



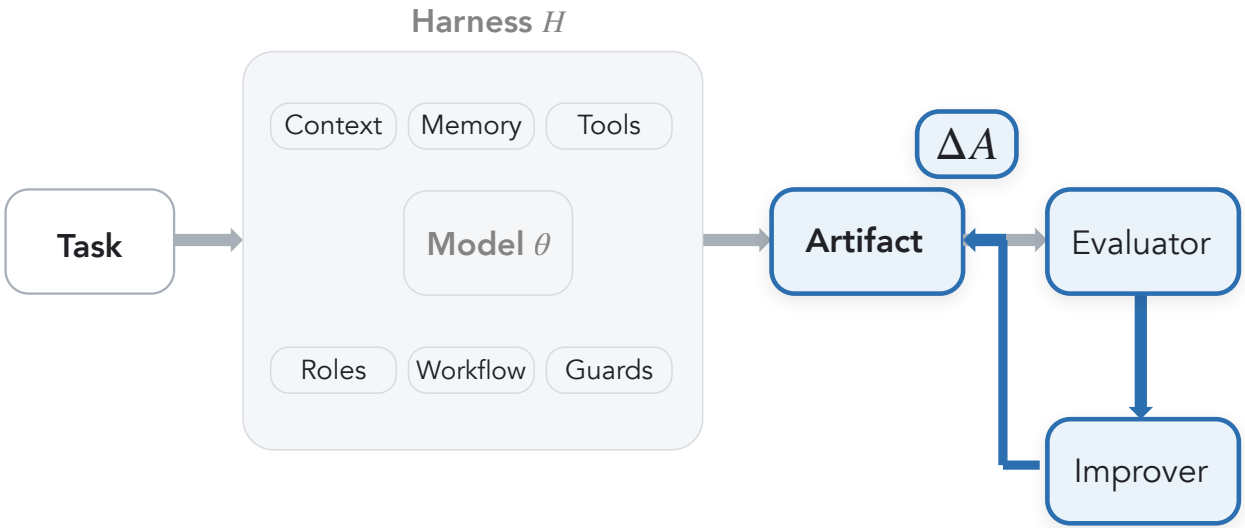
StarVLA: AutoResearch for VLAs

Star 3.6k



Ye et al. StarVLA-α: Reducing Complexity in Vision-Language-Action Systems

Layer 1: Artifact RSI



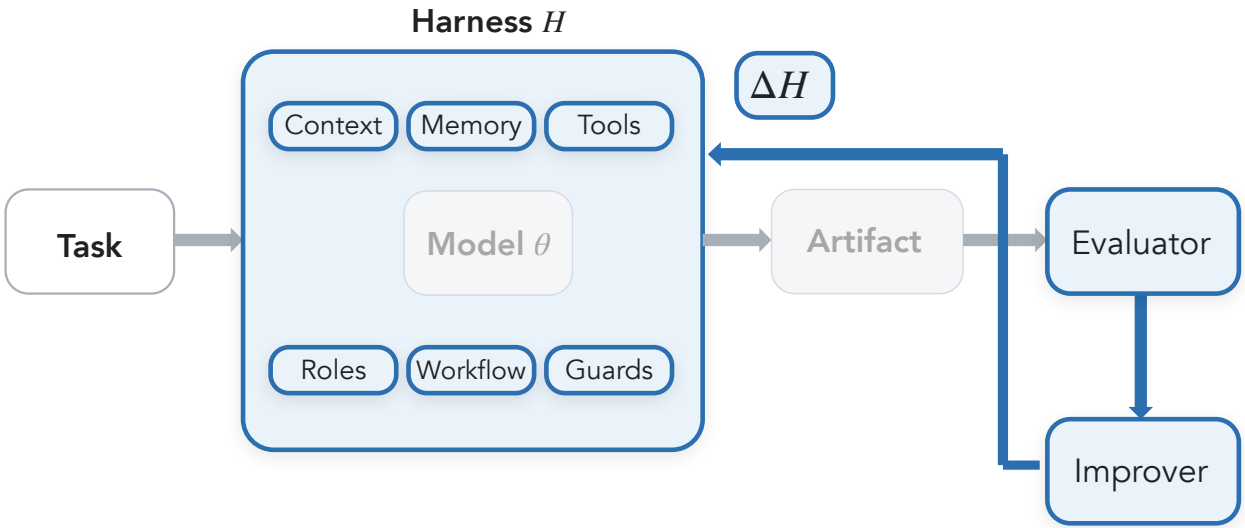
Layer 1: Artifact RSI

Layer 2: Harness RSI

Layer 3: Model RSI

Evaluation of RSI

Layer 2: Harness RSI



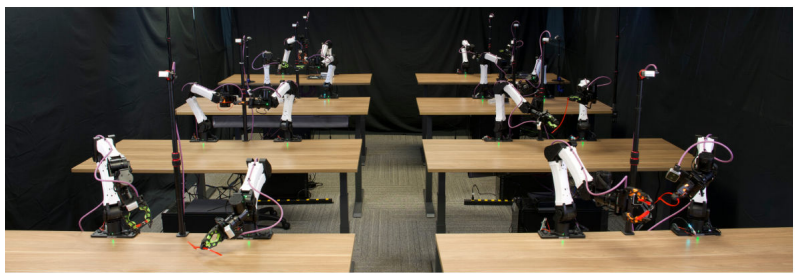
Layer 1: Artifact RSI

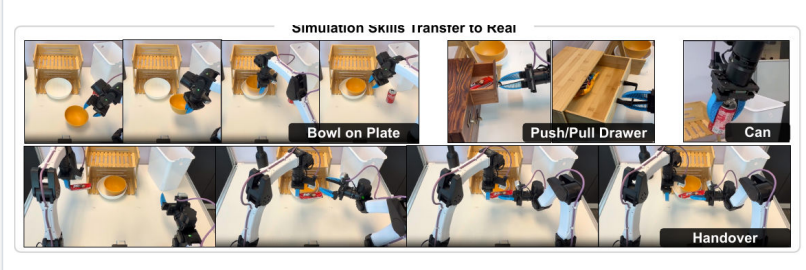
Layer 2: Harness RSI

Layer 3: Model RSI

Evaluation of RSI

Artifact RSI → Harness RSI





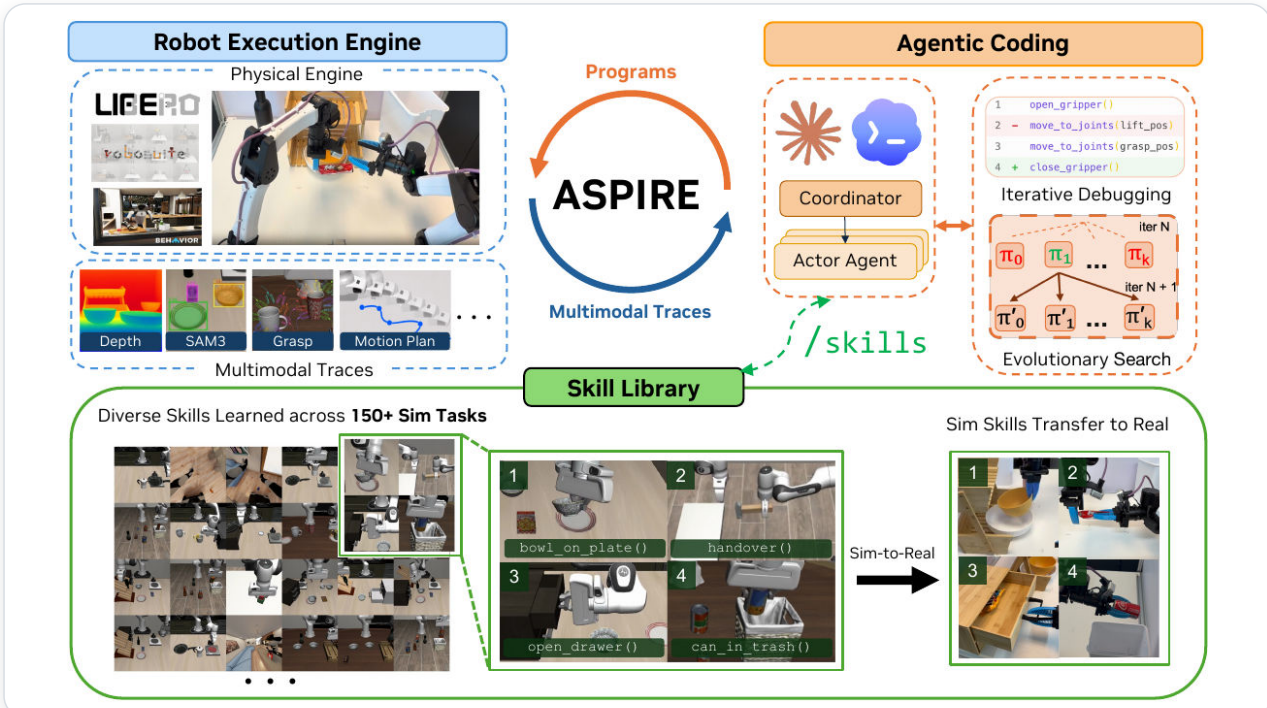
ENPIRE

8 real robot stations |
within-task optimization

ASPIRE

150+ simulated tasks |
cross-task skill transfer

ASPIRE Turns Failures Into Skills



TRACE

Per-primitive multimodal evidence

REPAIR

Localize failures search and validate

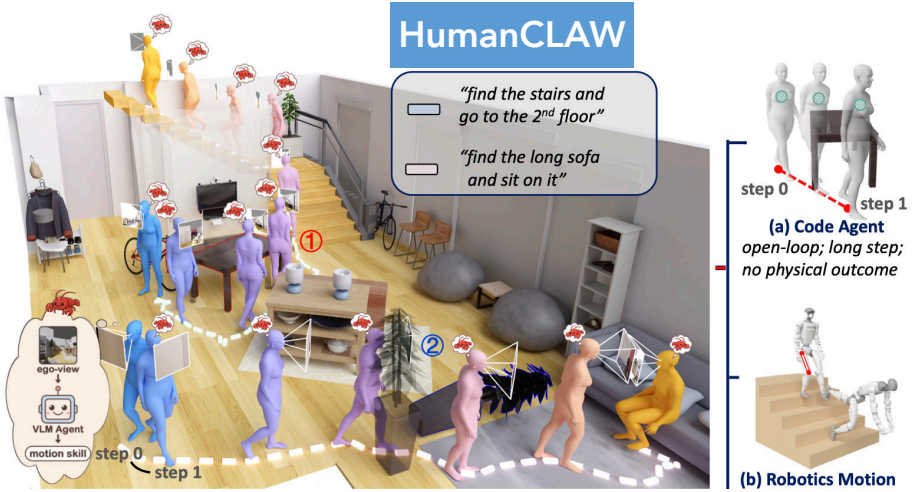
MEMORY

Write to /skills reuse across tasks

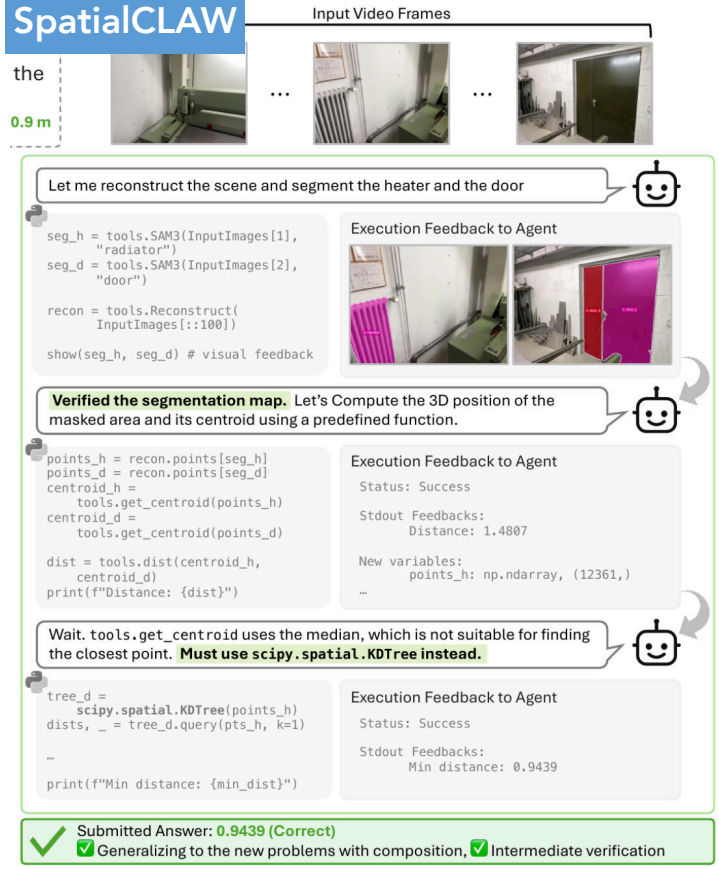
TAKEAWAY ASPIRE improves reusable debugging knowledge, not model weights.

Harness: the key is Interface of Tools, Skills and Memory

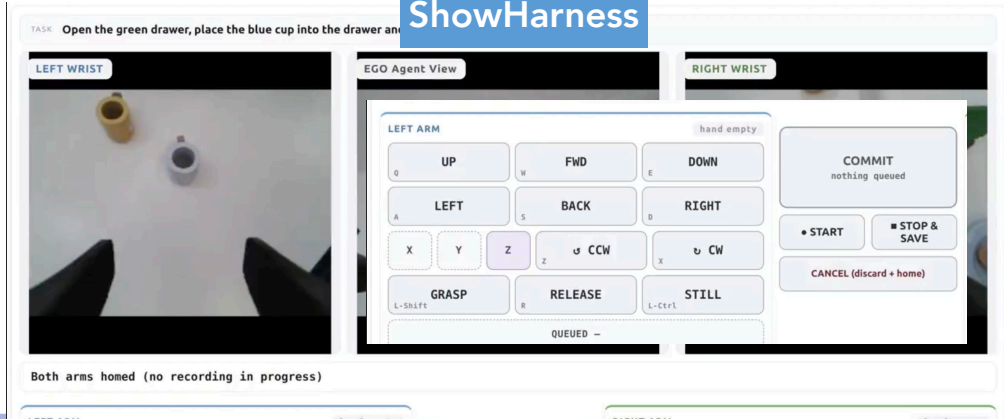
HumanCLAW



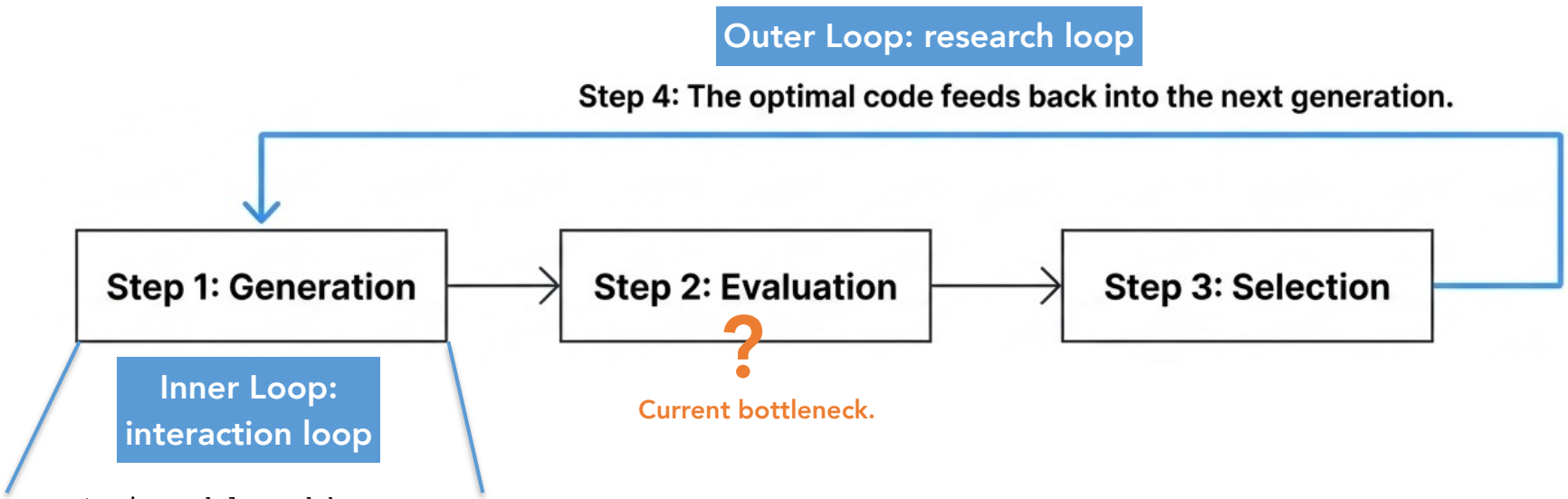
SpatialCLAW



ShowHarness



Inner Loop + Outer Loop




```
train models and harness
→ edit code, data, training
→ planning
→ execute grasp
→ observe
→ reset
```

Takeaways

1. The key of AutoResearch is the evaluator

2. Outer loop = research loop; Inner loop = interaction loop (with harness layer)

Takeaways: What is a better skill?

```
def pick_and_place_mug():
    move_to(mug)
    close_gripper()
    lift()
    move_to(shelf)
    open_gripper()
```

```
Verification (Evaluator):

Mug is no longer on the table
Mug is on the shelf
Mug is stable
Gripper is open
No collision or unsafe motion
```



2026.03

Skill = Workflow

2026.09

Skill = Verifiable Contract

3. A better skill is not just an action recipe, but a contract for success.

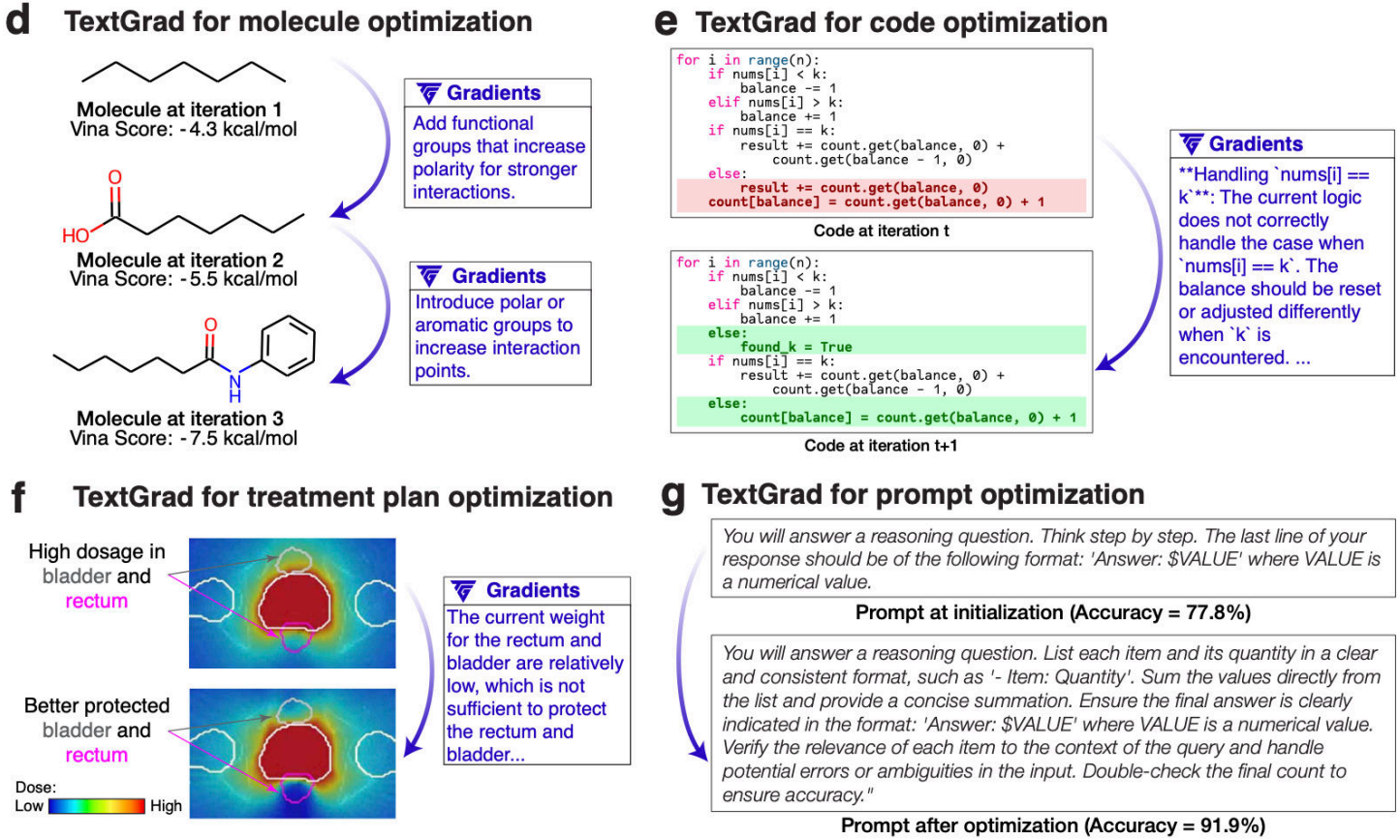
Takeaways

1. The key of AutoResearch is the evaluator

2. Outer loop = research loop; Inner loop = interaction loop (with harness layer)

3. Skills are more abstract when models are stronger

Takeaways- TextGrad: How to "train" harness?



Yuksekgonul et al. TextGrad: Automatic "Differentiation" via Text

Takeaways

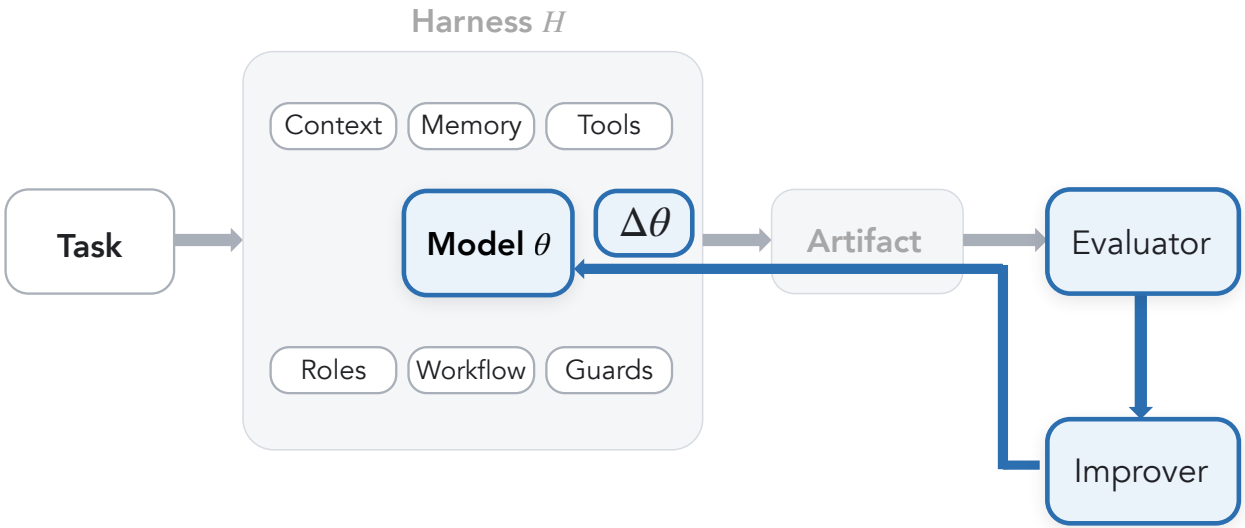
1. The key of AutoResearch is the evaluator

2. Outer loop = research loop; Inner loop = interaction loop (with harness layer)

3. Skills are more abstract when models are stronger

4. Training data: text gradient for training

Layer 3: Model RSI



Layer 1: Artifact RSI

Layer 2: Harness RSI

Layer 3: Model RSI

Evaluation of RSI

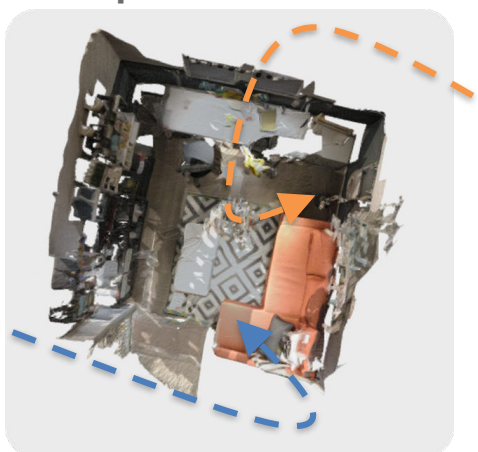
“ RL cannot work ”

Active View Planner

Initial view



Top-down View



Target view



Rotation Actions
turn_left, turn_right, look_up, look_down, rotate_cw, rotate_ccw

Translation Actions
move_forward, move_backward, move_right, move_left, move_up, move_down

Answer(tx, ty, tz, rx, ry, rz)

Kangrui Wang, Linjie Li, Zhengyuan Yang, Shiqi Chen, Zihan Wang, Li Fei-Fei, Jiajun Wu, Leonidas Guibas, Lijuan Wang, Manling Li. Planning with the views.

Pure RL Barely Moves

2.5%

PPO	3.2%
GRPO	5.2%
Bootstrap	6.2%

Sparse rewards stall every RL variant,
we need supervision the agent can build itself.

"familiar" env

"unfamiliar" env

✗ collapse

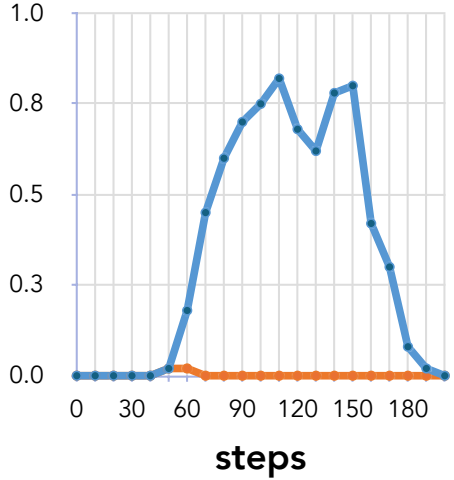


Webshop



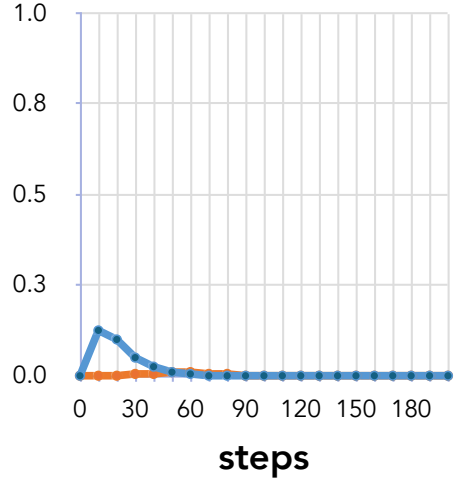


Bandit



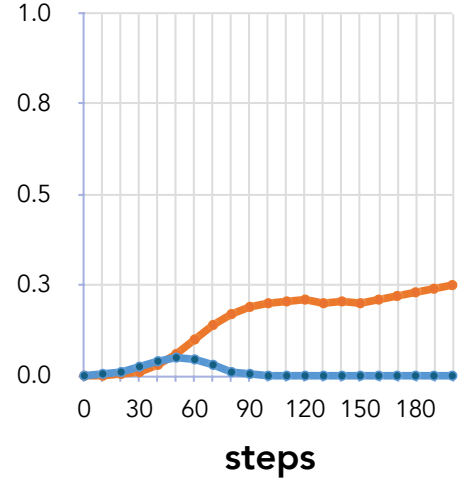


Sokoban





FrozenLake



PPO GRPO

RAGEN (2.8K Github Stars, Best Poster Award at Midwest ML 2025)

Reasoning Collapse into Repetition

0th round

```
#######
#_O___#
#___X_#
#___P_#
#######
```

Reasoning: *The box is above me, and the target is on the far left. I cannot push it directly. I should first move right, then approach the box from the right.*

Action: up



```
#######
#_O___#
#___XP#
#_____#
#######
```

Reasoning: *Now I can push the box to the left, until the bottom of the target.*

Action: right



```
#######
#_O___#
#_XP__#
#_____#
#######
```

Reasoning: *The box is now at the bottom of the target. I am at the right of the box. To push the box to the target, I need to move to the bottom of the box, and then push it up.*

Action: right



```
#######
#_√___#
#_P___#
#_____#
#######
```

Reasoning: *(box is now at the target)*



Wang, …, **Li**. RAGEN (2.5K Github Stars, Best Poster Award at Midwest ML 2025) 🏆

Reasoning Collapse into Repetition

```
#######
#_O___#
#___X_#
#___P_#
#######
```

Reasoning: *The box is above me, and the target is on the far left. I cannot push it directly. I should first move right, then approach the box from the right.*

Action: up



200 rounds later…

```
#_O___#
#_XP__#
#_____#
#######
```

bottom of the target. I am at the right of the box. To push the box to the target, I need to move to the bottom of the box, and then push it up.

Action: right



```
#######
#_√___#
#_P___#
#_____#
#######
```

Reasoning: *(box is now at the target)*



Wang, …, **Li**. RAGEN (2.5K Github Stars, Best Poster Award at Midwest ML 2025) 🏆

Reasoning Collapse into Repetition

200th round

```
#######
#_O___#
#___X_#
#___P_#
#######
```

Reasoning: *The box is above me. I should move up to the box.*

Action: right, up



```
#######
#_O_X_#
#___P_#
#_____#
#######
```

Reasoning: *I should move to the right of the box.*

Action: left, left



```
#######
#_O_X_#
#____P#
#_____#
#######
```

Reasoning: *I should move to the right of the box.*

Action: down, left, up



```
#######
#_O_X_#
#____P#
#_____#
#######
```

Reasoning: *I should move to the right of the box.*



Wang, …, **Li**. RAGEN (2.5K Github Stars, Best Poster Award at Midwest ML 2025) 🏆



vs Exploration

Repeat one solution

RAGEN (2.6K Github Stars, Best Poster Award at Midwest ML 2025)

Exploitation vs Exploration

Alternative solutions

Especially in ***OOD*** environments

RAGEN (2.6K Github Stars, Best Poster Award at Midwest ML 2025)

self-improvement through RL

1. What are failure modes of RL?

2. What priors should we encourage for exploration?

3. When is the best stage to encourage it?

RAGEN (2.5K Github Stars, Best Poster Award at Midwest ML 2025) 

self-improvement through RL

1. What are failure modes of RL?

→ Exploration vs Exploitation

2. What priors should we encourage for exploration?

3. When is the best stage to encourage it?

RAGEN (2.5K Github Stars, Best Poster Award at Midwest ML 2025) 

self-improvement through RL

1. What are failure modes of RL?

→ *Exploration vs Exploitation*

2. What priors should we encourage for exploration?

→ *World Model Reasoning*

3. When is the best stage to encourage it?

Chen, …, **Li**. Self-Play (arxiv 2025)

Priors in Exploration:

World Modeling

Wang, …, **Li**. VAGEN (NeurIPS 2025, featured by MIT Tech Review, Stanford HAI)

View Transitions as a View World Model



view A

turn left | move forward



view B

self-improvement through RL

1. What are failure modes of RL?

→ *Exploration vs Exploitation*

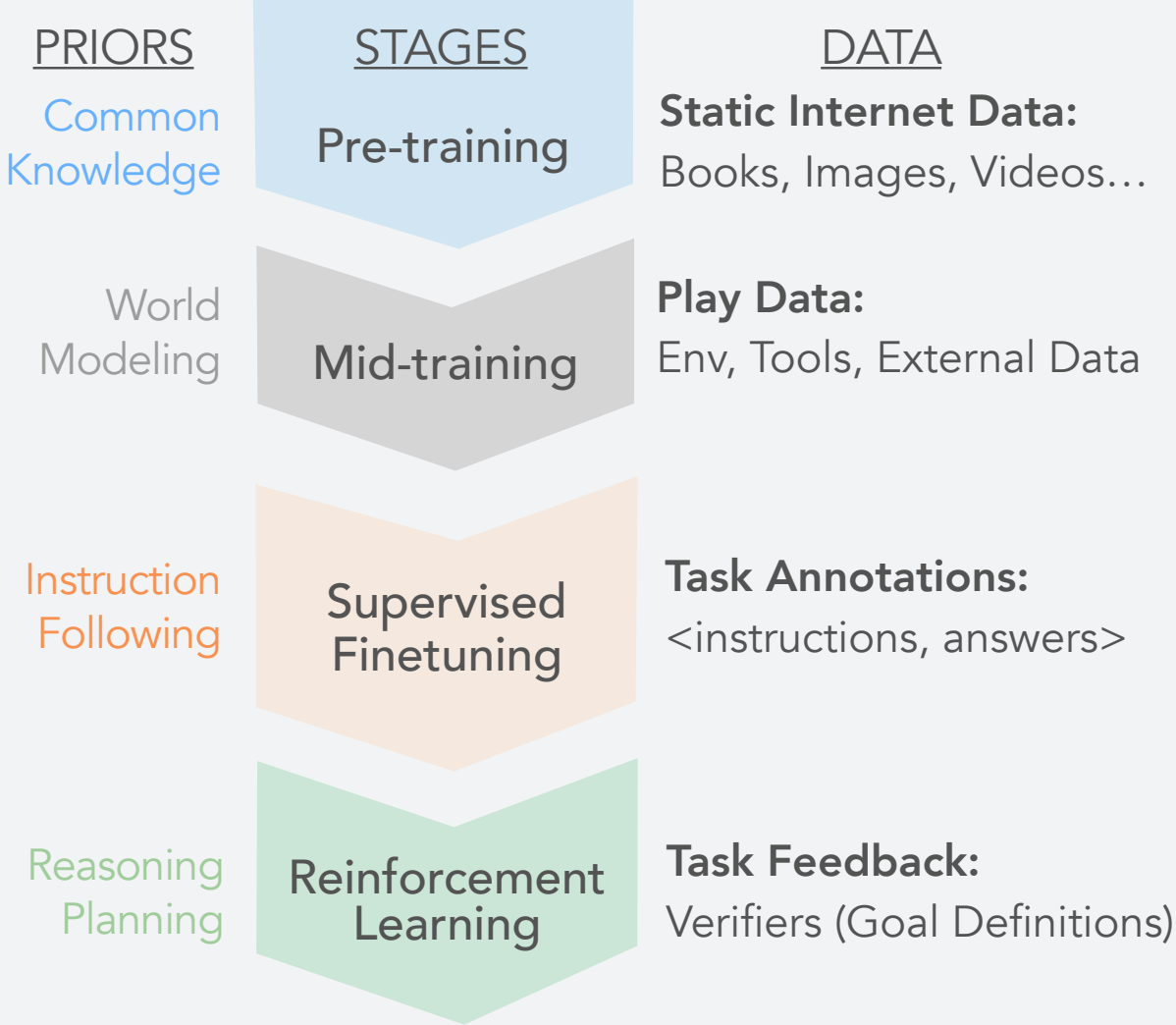
2. What priors should we encourage for exploration?

→ *World Model Reasoning*

3. When is the best stage to encourage it?

→ *Mid-training/SFT vs RL*

Chen, …, **Li**. Self-Play (arxiv 2025)



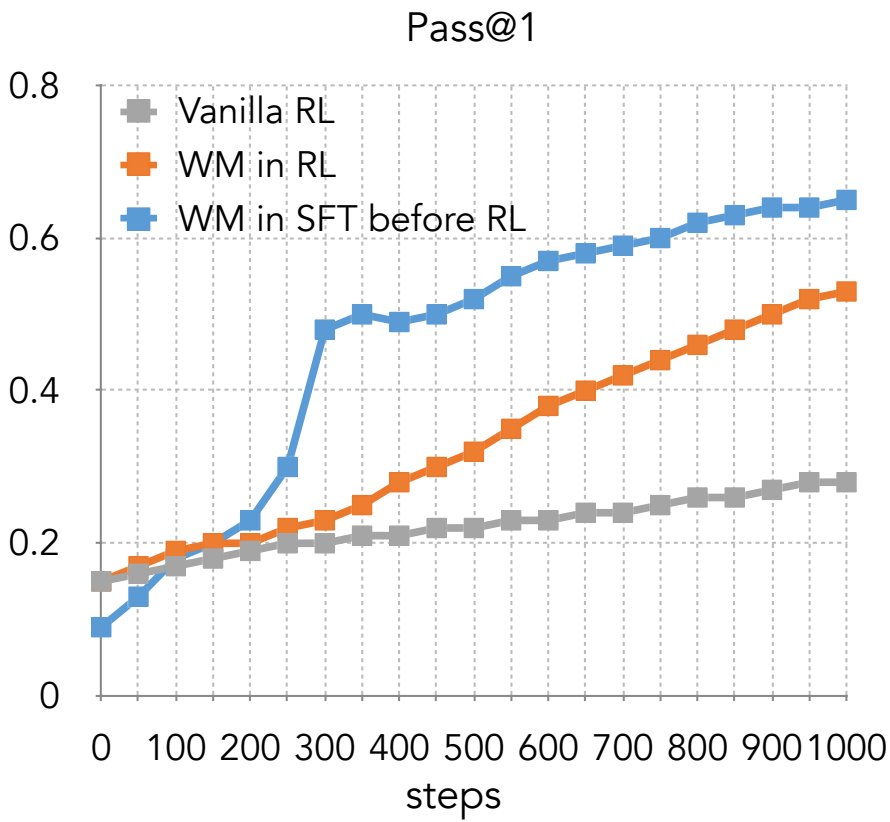
Brute-force Scaling

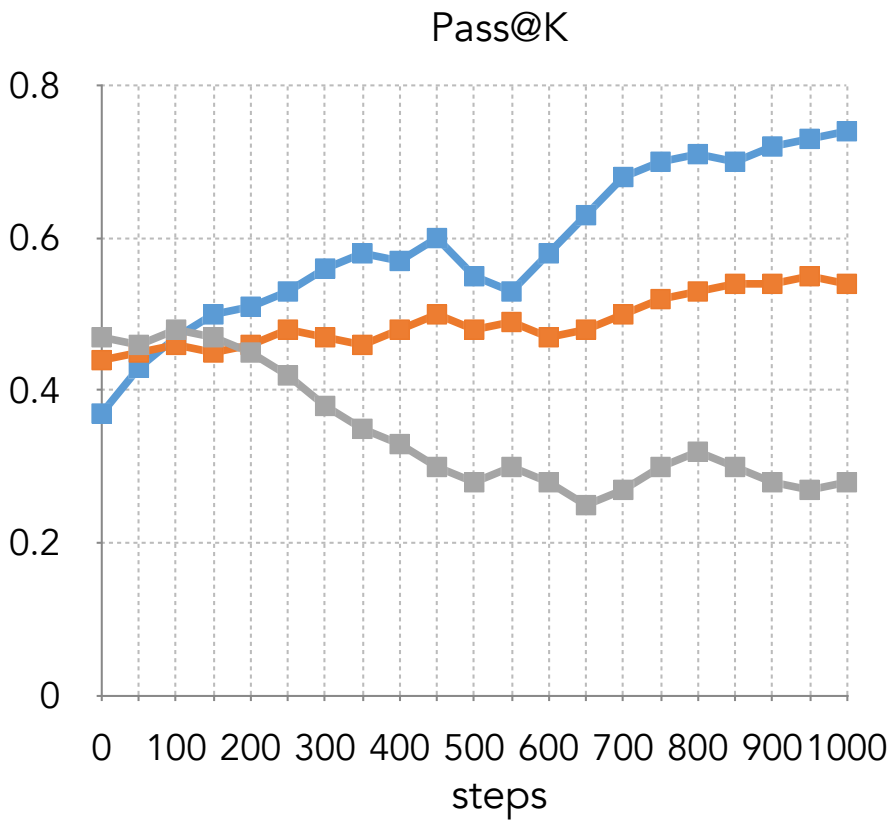
Continual Learning

(We are here now)

Exploration vs Exploitation







Chen, …, **Li**. Self-Experience (arxiv 2025)

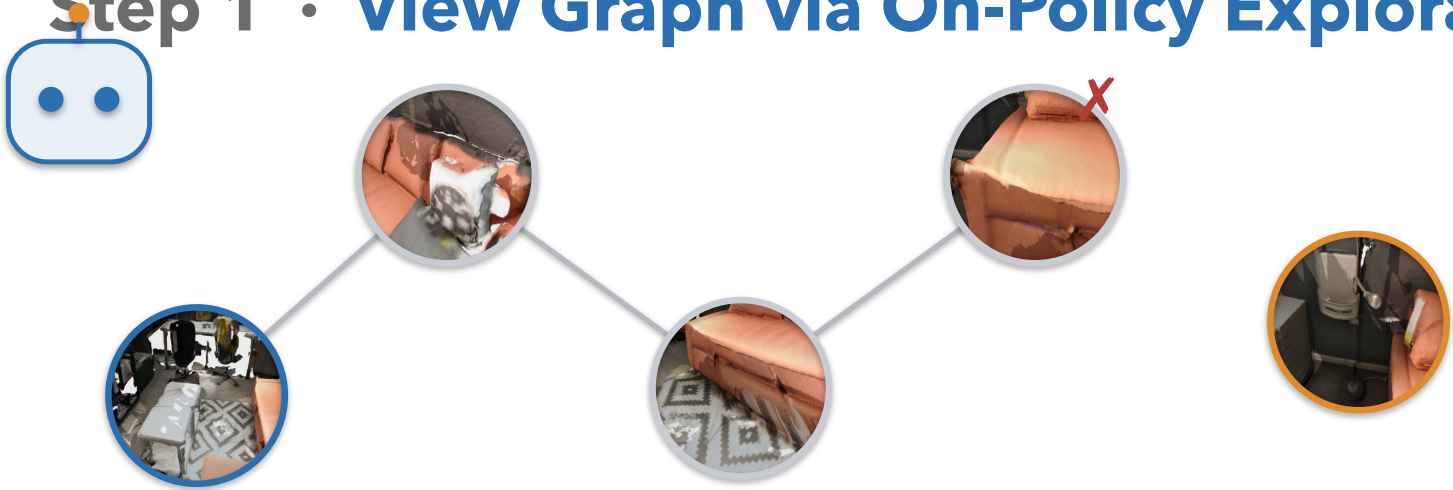
Step 1 · View Graph via On-Policy Exploration





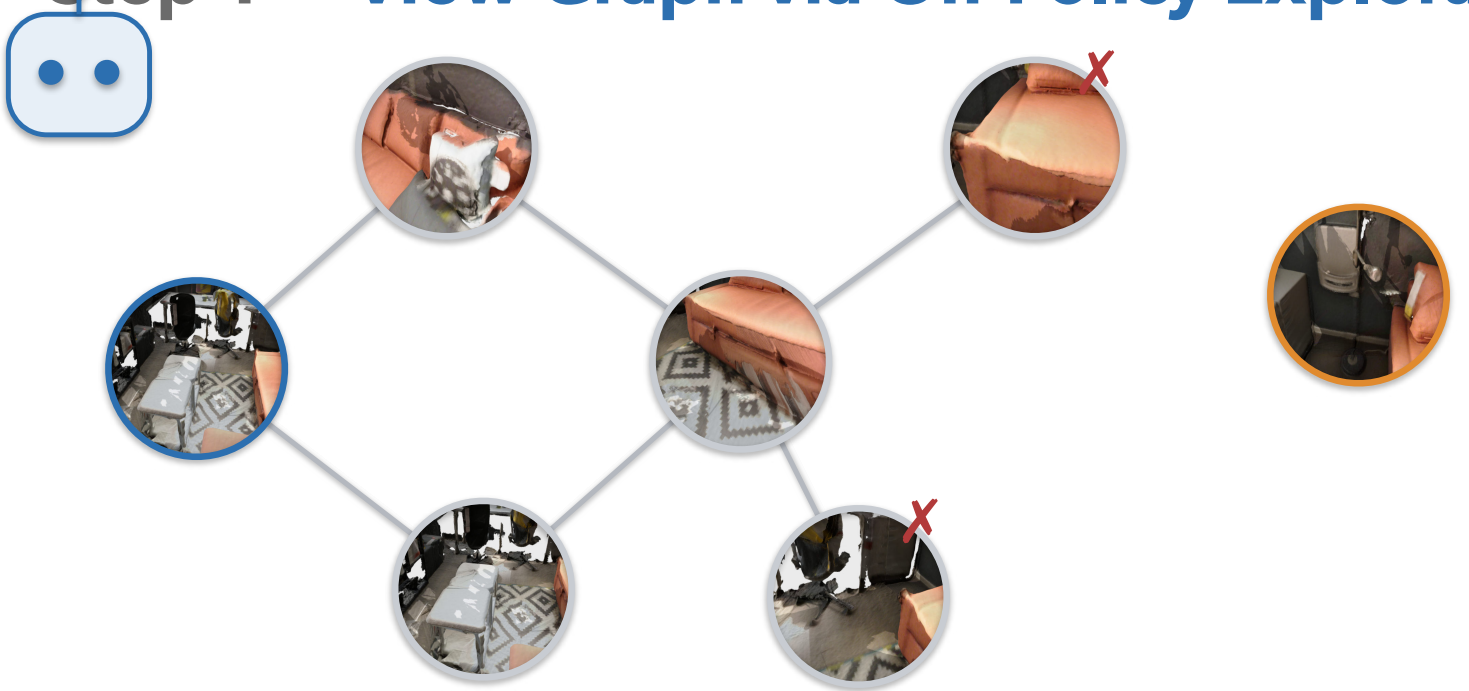
The agent explores ViewSuite scenes from its own experience.

Step 1 · View Graph via On-Policy Exploration



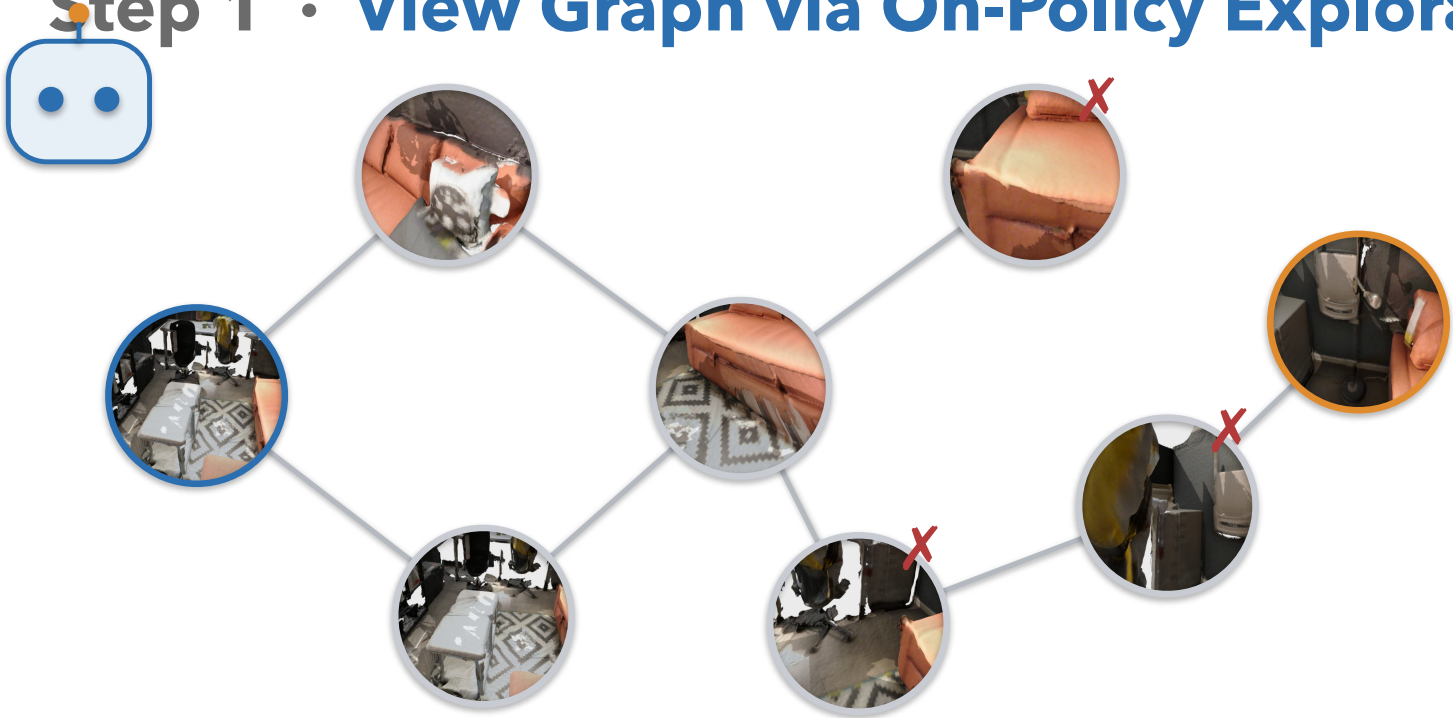
Rollout 1 wanders the scene – and misses.

Step 1 · View Graph via On-Policy Exploration



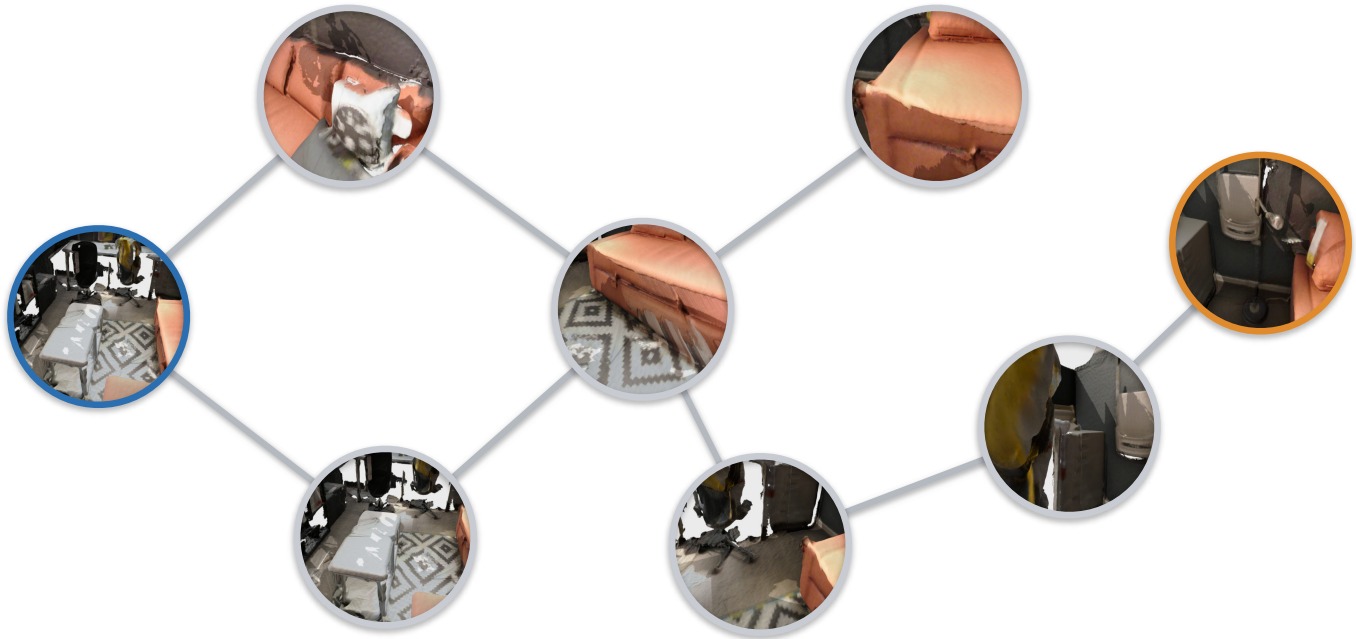
Rollout 2 also fails – but it crosses rollout 1.

Step 1 · View Graph via On-Policy Exploration



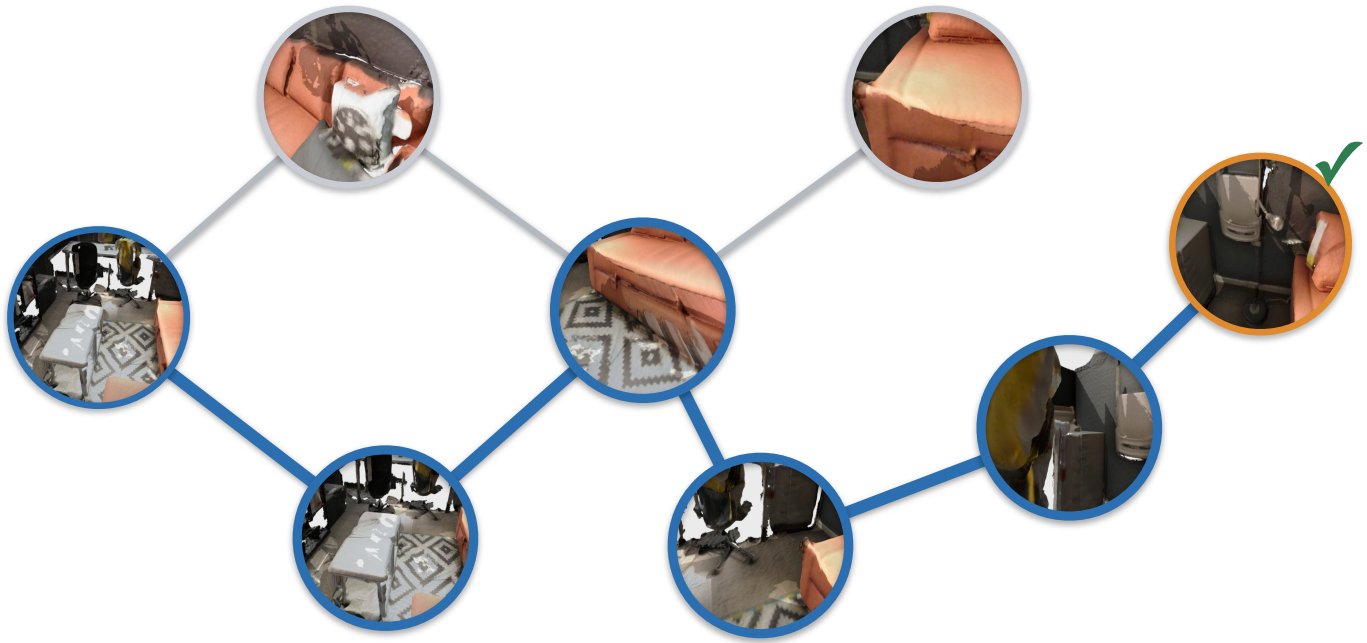
Every rollout misses the target. Yet together they trace the whole scene.

Step 2 · A View Graph Emerges



Shared viewpoints merge: nodes are views, edges are actions.

Step 2 · A Path Emerges



A path the agent never walked now connects initial → target.

Step 3 · Reformulate into a Demo



input: initial + target view ***output: the action sequence***

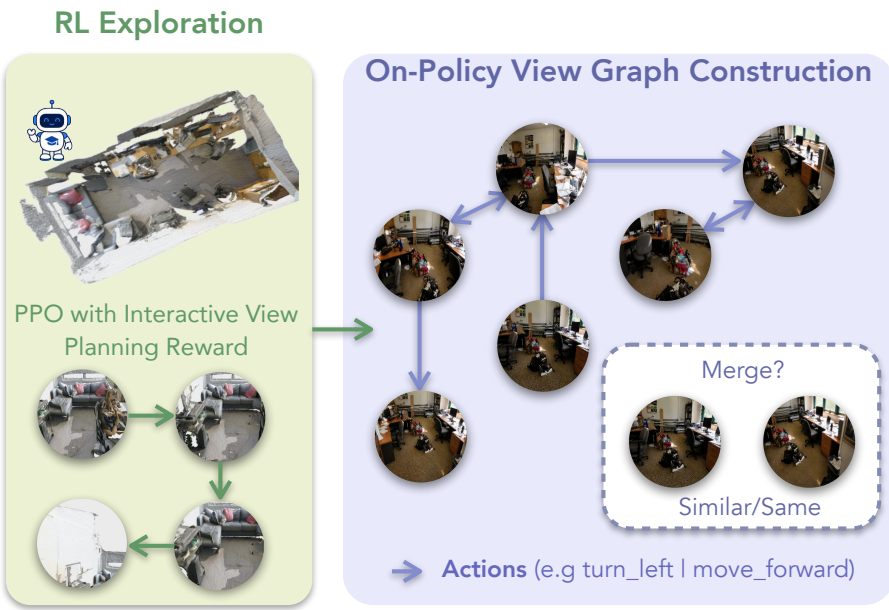
Any sampled path becomes a demo
valid even though no rollout reached the target.

View Planning via Self-Exploration

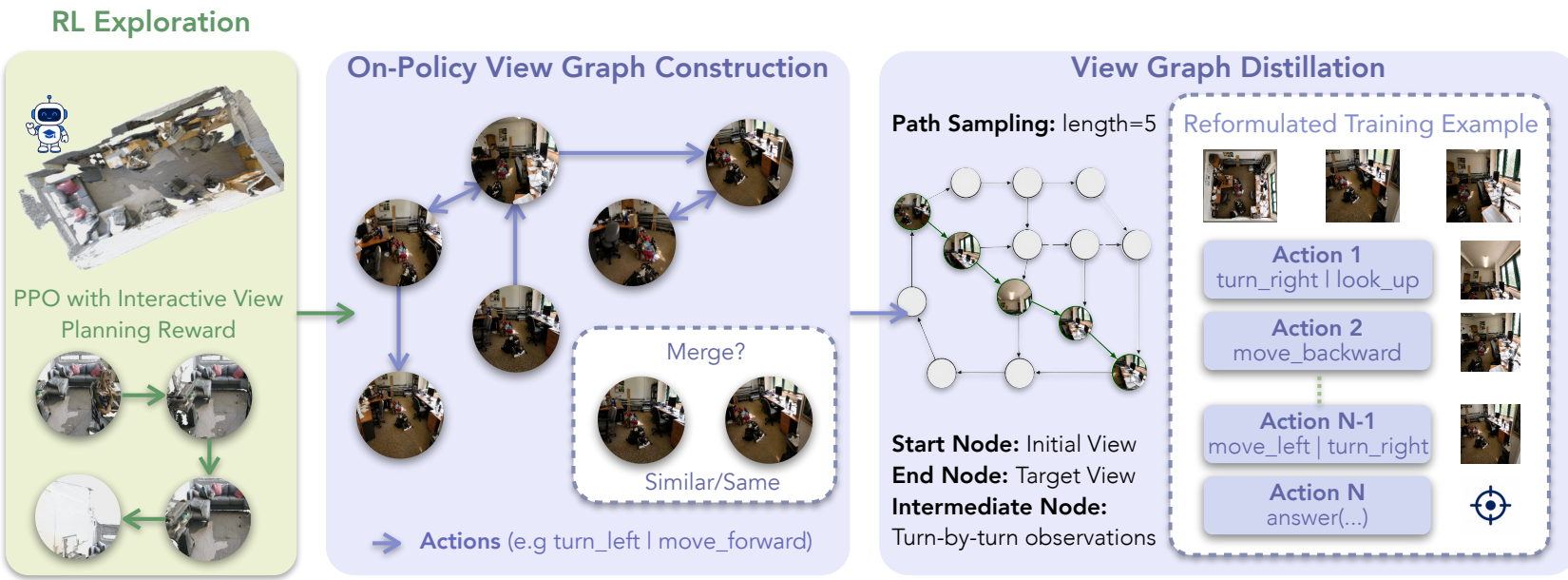
RL Exploration



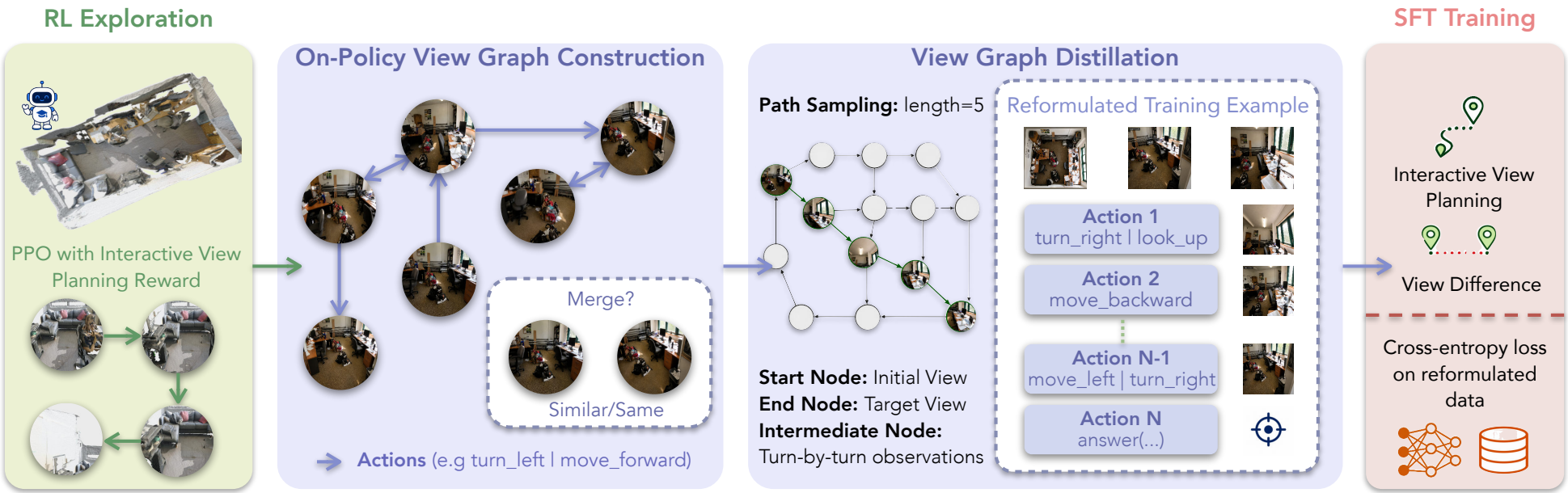
View Planning via Self-Exploration



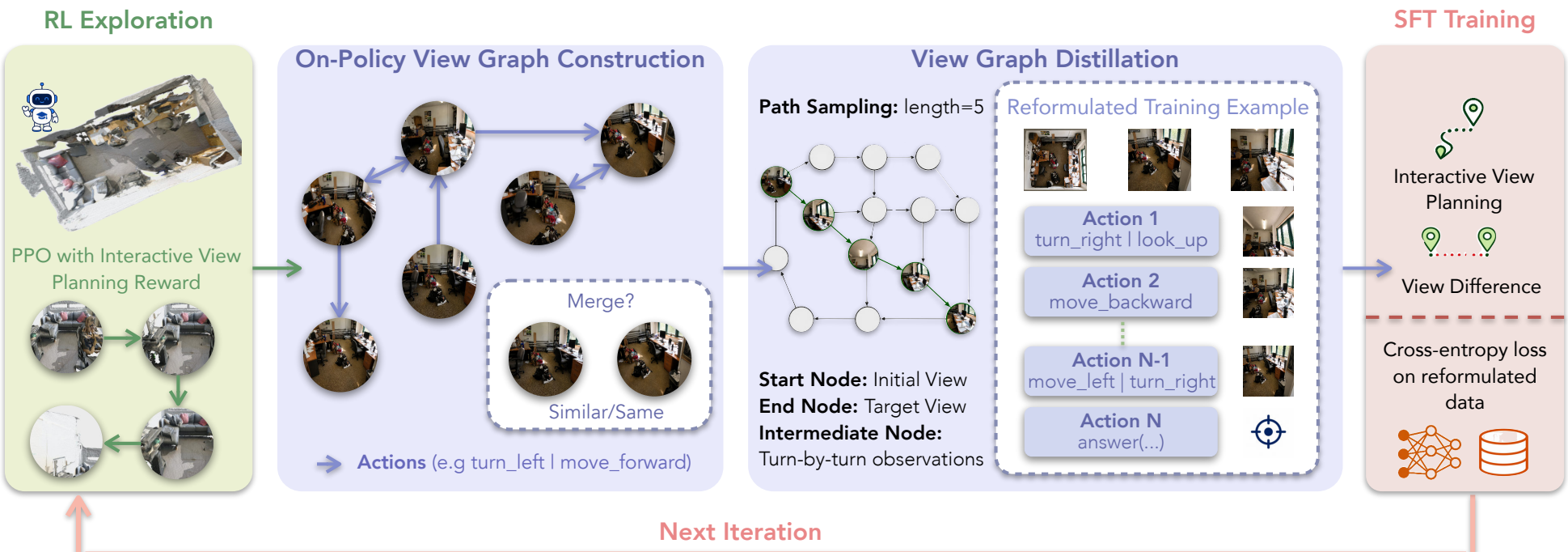
View Planning via Self-Exploration



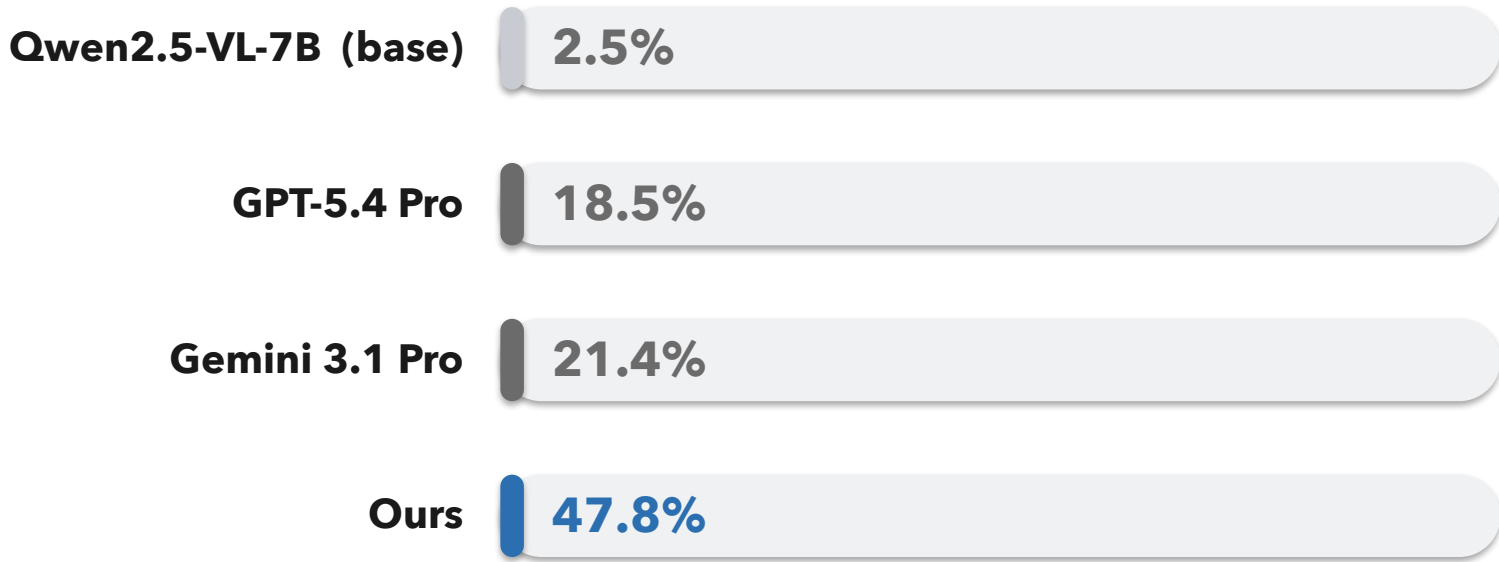
View Planning via Self-Exploration



View Planning via Self-Exploration

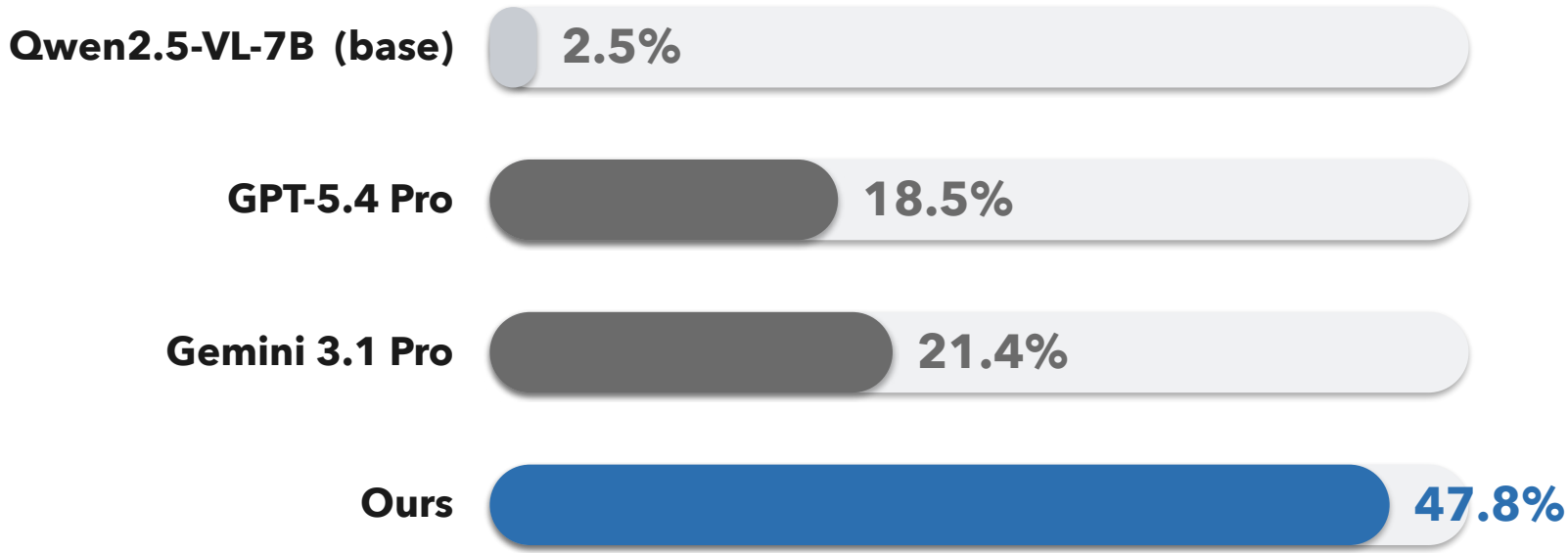


From 2.5% to 47.8%



View Planning (arxiv 2026)

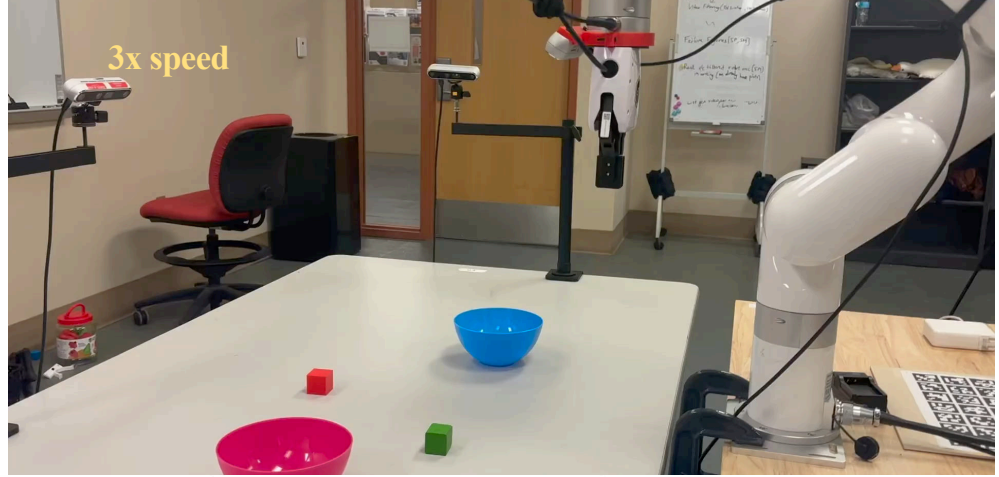
From 2.5% to 47.8%

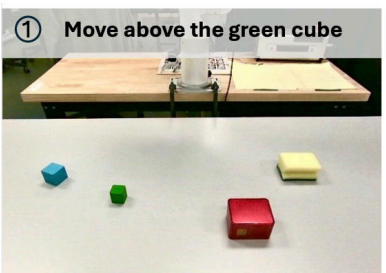


A 7B model trained only on its own exploration beats every frontier VLM.

View Planning (arxiv 2026)

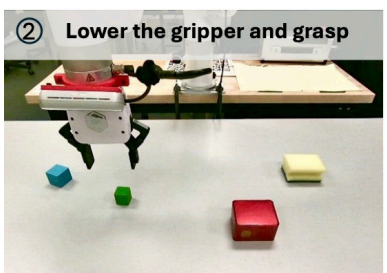
Embodied Prior Learning via mid-training





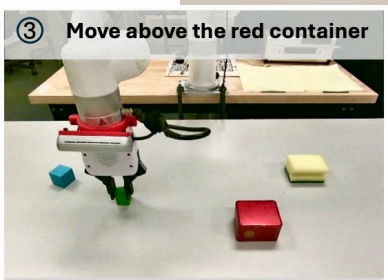
Reasoning: I can see a blue cube at [14, 18, 8], a green cube at [26, 45, 7], a red cube at [41, 90, 11] and a yellow sponge at [12, 115, 10]. I need to place the lime cube on top of the red cube. So, the first step is to Move the gripper above the lime cube at [26, 45, 7].

Action: [26, 45, 25, 0, 60, 90, 1]



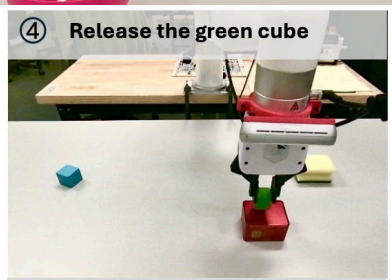
Reasoning: I can see a blue cube at [14, 18, 8], a green cube at [26, 45, 7], a red cube at [41, 90, 11], and a yellow sponge at [12, 115, 10]. The next step is to lower the gripper to grasp the lime cube. After that, I will close the gripper to pick up the lime cube, lift it, and then...

Action: [26, 45, 7, 0, 60, 90, 0]



Reasoning: I can see a green cube at [26, 45, 7], and a red cube at [41, 90, 11]. Now, I have successfully picked up the lime cube from its initial position. The next step is to move the lime cube to a position above the red cube.

Action: [41, 90, 25, 0, 60, 90, 0]



Reasoning: I can see a green cube at [14, 18, 8], and a red cube at [41, 90, 11]. I have successfully moved the lime cube above the red cube. The next step is to lower the lime cube onto the red cube and release it by opening the gripper. This will complete the task.

Action: [41, 90, 17, 0, 60, 90, 1]



Reasoning:
The task has been completed.

✓

Action: None

Chen, …, **Li**, Zhang, Zhang. Embodied Reasoning Agent (arxiv 2025)

self-improvement through RL

1. What are failure modes of RL?

→ *Exploration vs Exploitation*

2. What priors should we encourage for exploration?

→ *World Model Reasoning*

3. When is the best stage to encourage it?

→ *Mid-training/SFT vs RL*

Chen, …, **Li**. Self-Play (arxiv 2025)

Learning reasoning through RL

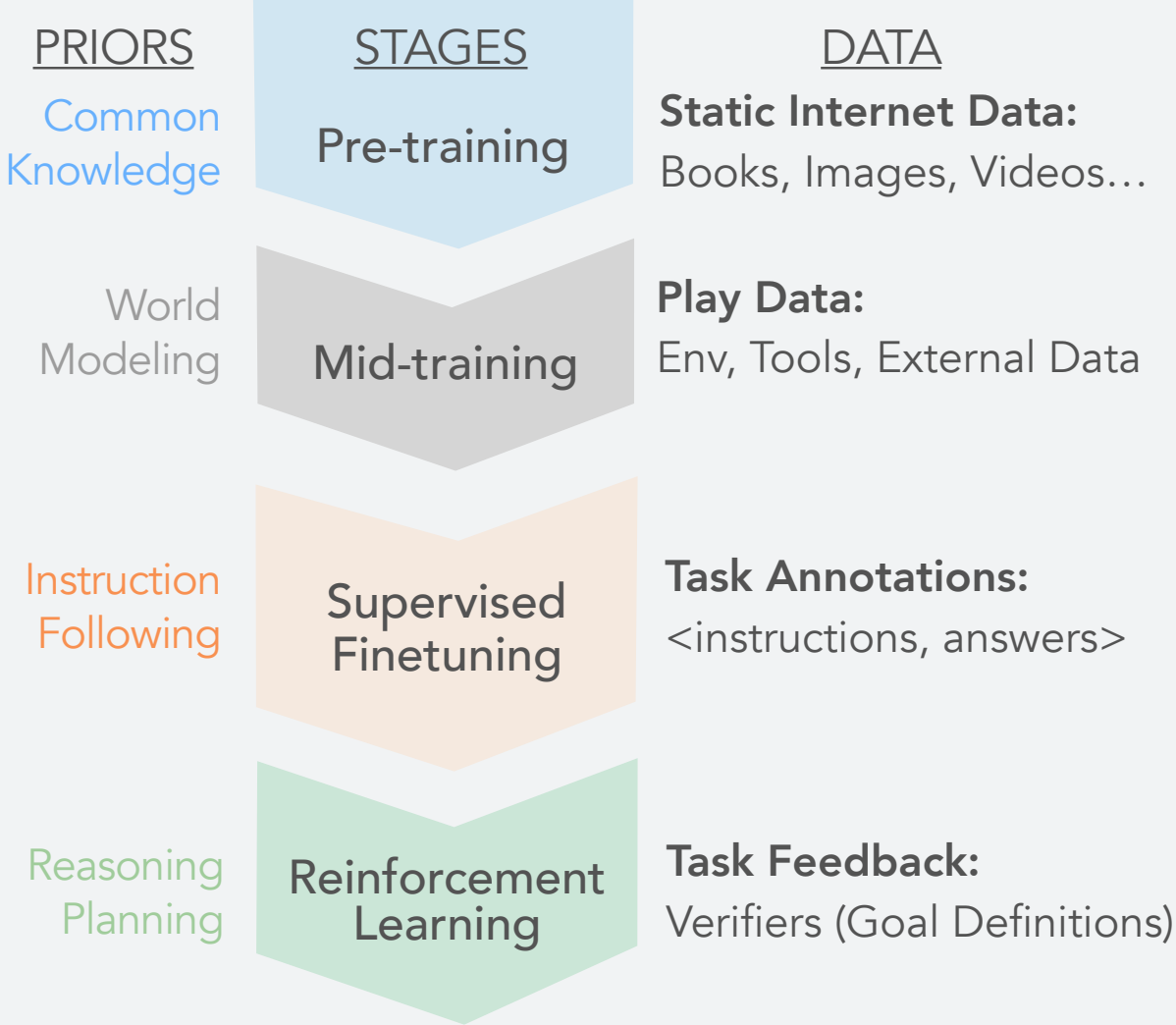
1. What are failure modes of RL?

data problem vs RL problem?

3. When is the best stage to encourage it?

→ *Mid-training vs RL*

Chen, …, **Li**. Self-Play (arxiv 2025)



Brute-force Scaling

Continual Learning

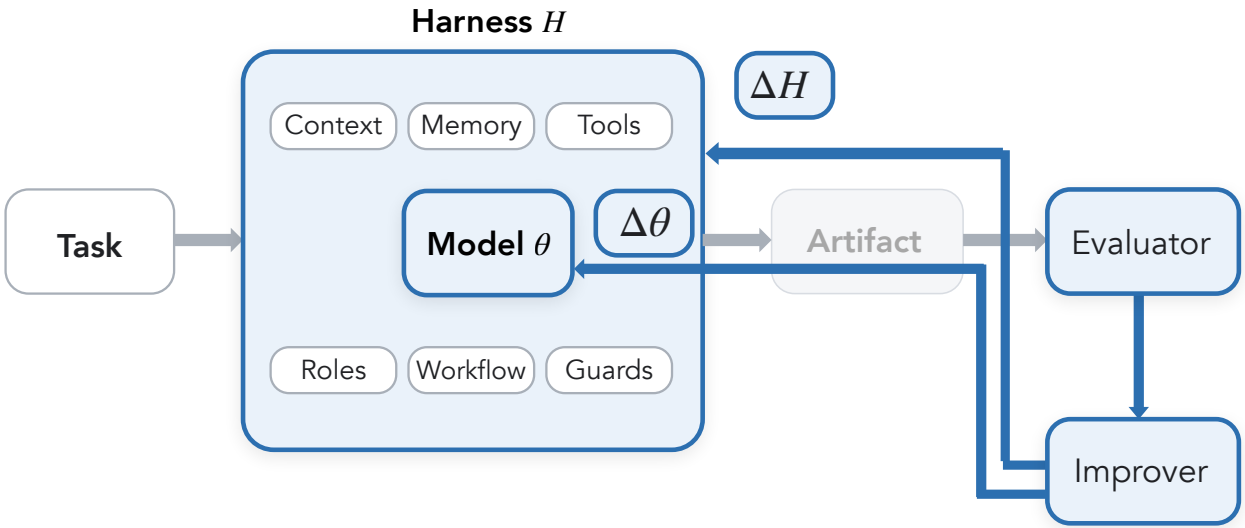
(We are here now)

What makes a world model reasoning success?

1. transition modeling	Abstraction
2. state estimation	Abstraction
3. on-policy exploration	Prior
4. enough world modeling training	Prior

Chen, …, **Li**. Self-Experience (2025)

The layers can co-evolve



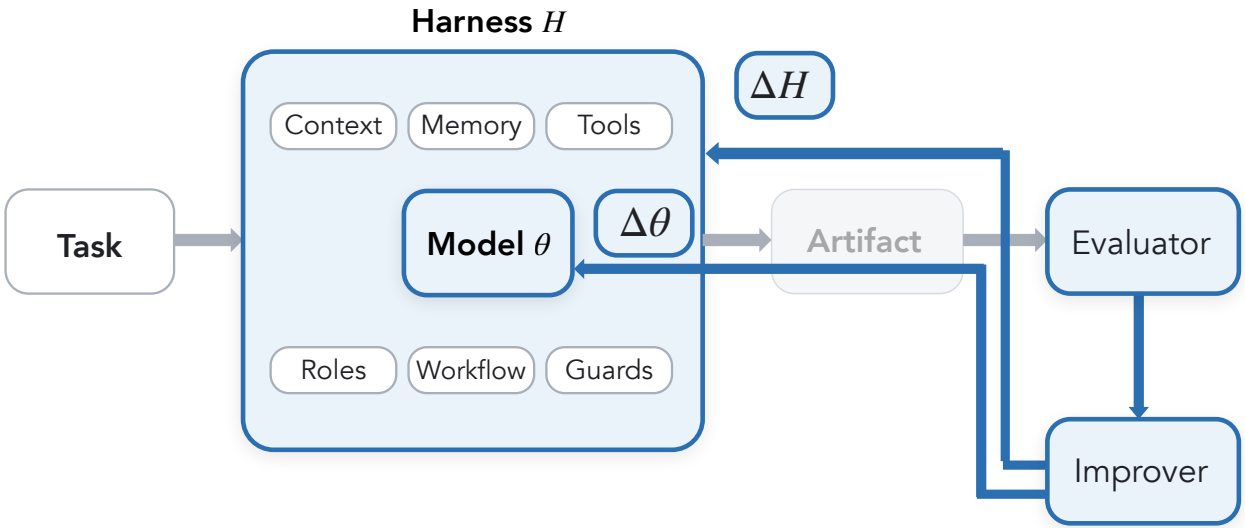
Layer 1: Artifact RSI

Layer 2: Harness RSI

Layer 3: Model RSI

Evaluation of RSI

The layers can co-evolve



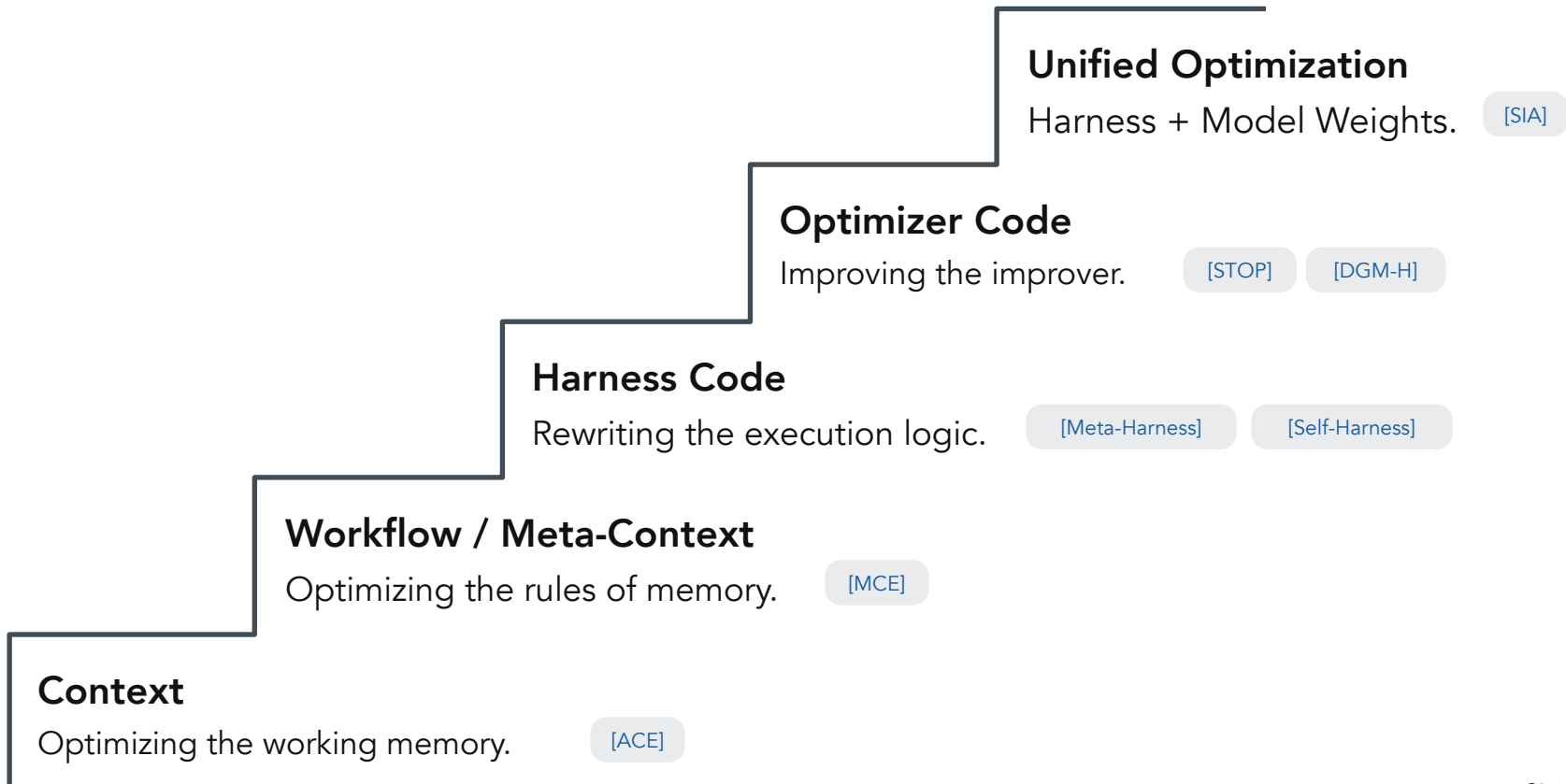
Layer 1: Artifact RSI

Layer 2: Harness RSI

Layer 3: Model RSI

Evaluation of RSI

The Ladder of Abstraction

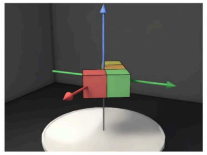


Gemini Notebook

SpatialEvolve: Can we benchmark Spatial RSI?

FIGURAL ×5

Table-top scale · body fixed · objects manipulable — where the atomic abilities come from



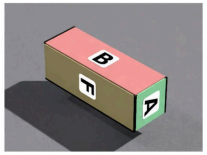
T1 MentalRotation

ACTIONS
`yaw · pitch · roll · done`

THE FAMILY F
rotations of the object

GOAL STATE
The object's orientation matches the GOAL image

Mental rotation
Intrinsic · Dynamic



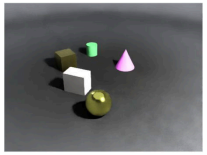
T2 NetFolding

ACTIONS
`fold · done`

THE FAMILY F
fold maps along the creases

GOAL STATE
The sheet, folded along its creases, matches the GOAL shape

2D→3D assembly
Intrinsic · Dynamic



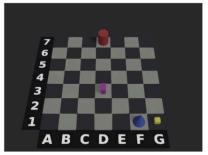
T3 PerspectiveTaking

ACTIONS
`az · el · done`

THE FAMILY F
observer motions about the scene

GOAL STATE
The camera's view matches the TARGET image

Perspective taking
Extrinsic · Dynamic



T4 SpatialRelation

ACTIONS
`move · done`

THE FAMILY F
assignments of objects to cells

GOAL STATE
Every listed constraint holds at once

Relation construction
Extrinsic · Dynamic



T5 ActiveClosure

ACTIONS
`view · reveal · done`

THE FAMILY F
viewpoint motion + progressive reveal

GOAL STATE
The partly hidden object named from the options given

Visual closure
Intrinsic · Static

ENVIRONMENTAL ×2

Room scale · body moves · scene exceeds the view — the scale control on PerspectiveTaking's family (Montello 1993)



T6 CameraRelocalization

ACTIONS
`move_forward · turn_left · turn_right · turn_back · done`

THE FAMILY F
planar observer poses, SE(2)

GOAL STATE
Standing at the position AND the facing of the GOAL image

Perspective taking
Extrinsic · Dynamic



T7 NovelPathDiscovery

ACTIONS
`move_forward · turn_left · turn_right · turn_back`

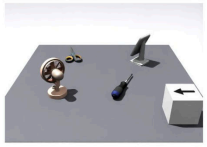
THE FAMILY F
paths over an unknown layout

GOAL STATE
The Goal cell reached by a path shorter than the demo

Cognitive map
Extrinsic · Dynamic

COMPOSITE ×2

Deliberately combines several atomic abilities — this is where A1 is checked, not a harder task. Both are table-top scale: composite is a role, not a third scale



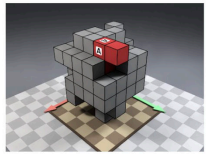
T8 PoseMatching

ACTIONS
`move · rotate · scale · orbit · done`

THE FAMILY F
pose and scale together

GOAL STATE
Every object at its GOAL position, orientation and size

Rotation · Relation · Size constancy
Intrinsic · Extrinsic · Dynamic



T9 Assembly

ACTIONS
`rotate · place · remove · done`

THE FAMILY F
part placements

GOAL STATE
Each piece at its GOAL orientation, anchor cube on the right cell

Rotation · Relation · Ordering
Intrinsic · Extrinsic · Dynamic

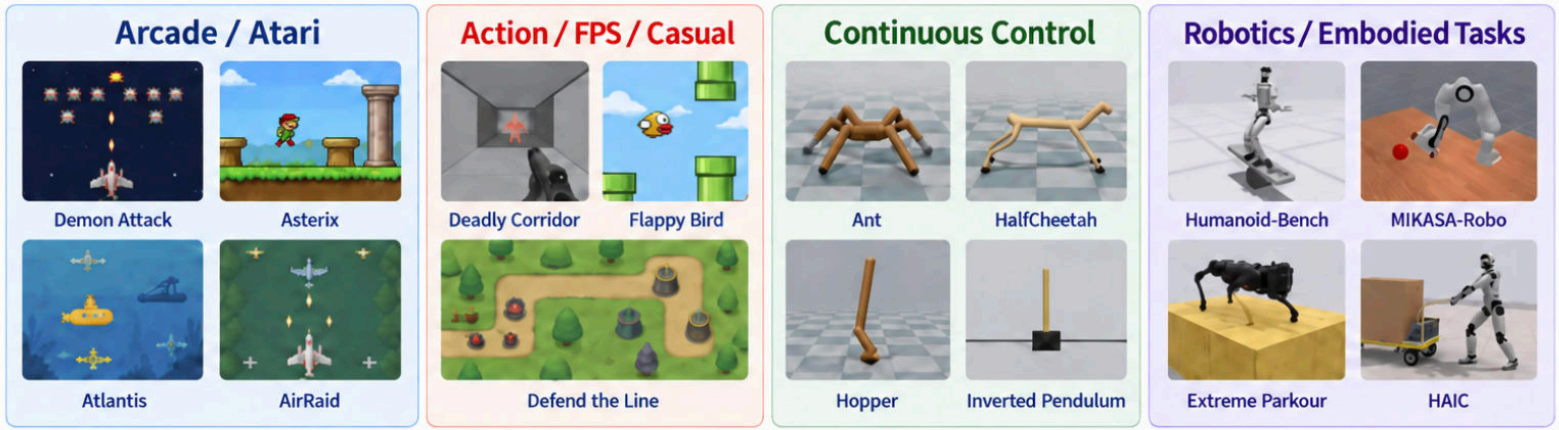
Latency should be also evaluated

What can’t GPT-6 do well?

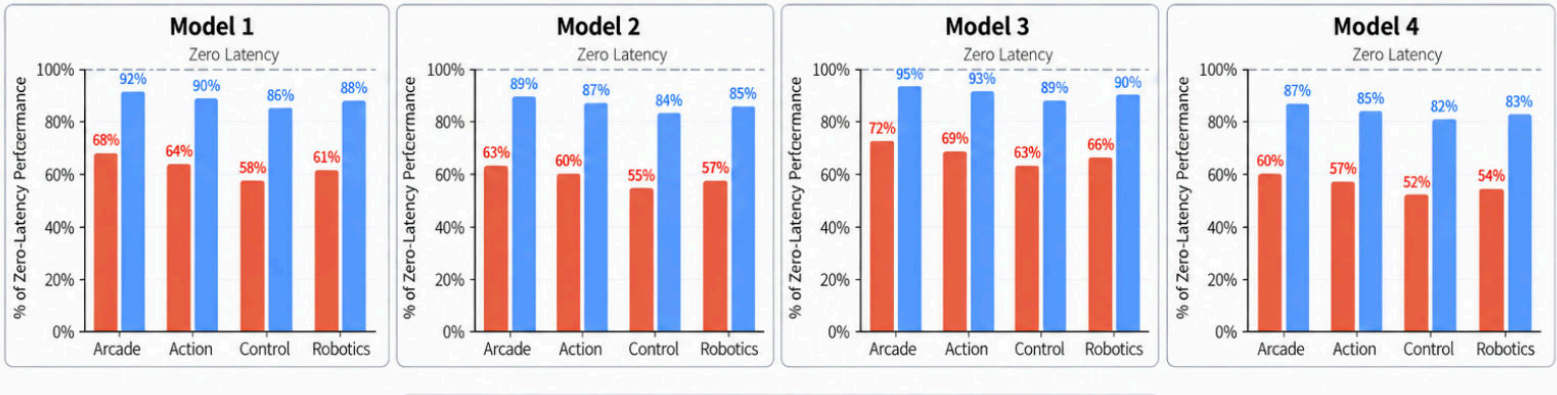
19/20 on a block-to-bowl task,
but only 2/20 on a precise puzzle-insertion task

latency: 2.5-3.4 minutes in average

Frontiers: Latency-Aware Agents



Real-time latency creates a performance gap → **Our method recovers most of the zero-latency performance**



Frontiers: Budget-Aware Agents

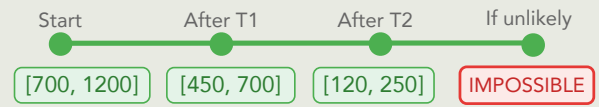
Single-point estimation (Prior Work)

Question: Which college did the author of The Hobbit attend?

Cost: 1400 tokens

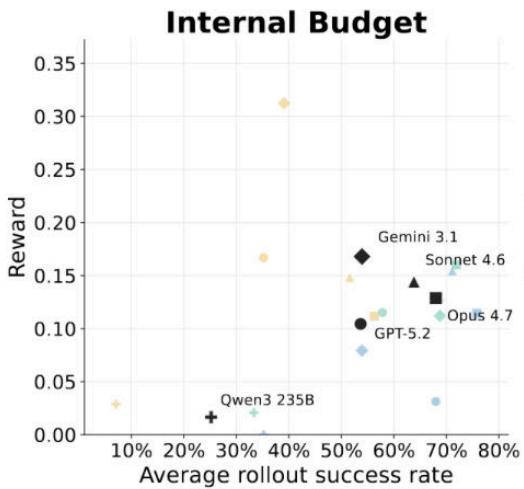


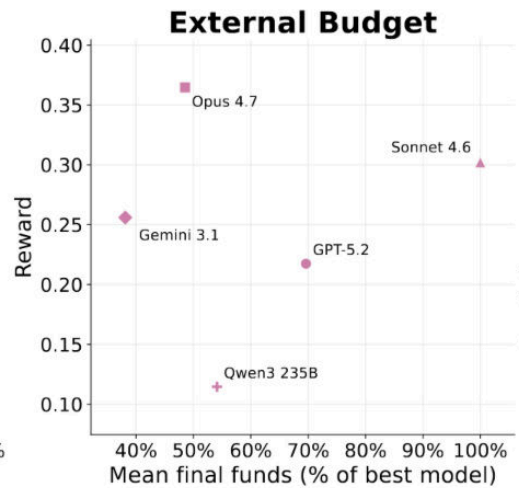
Progressive interval estimation (Ours)

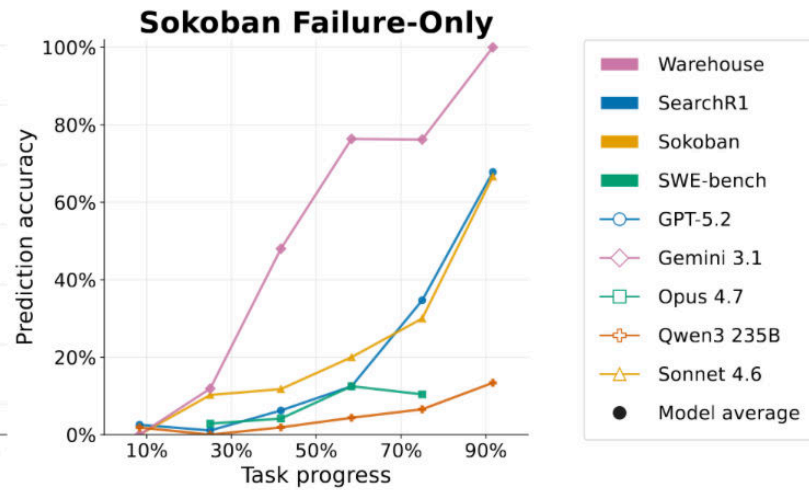


- Confidence-aware intervals
- Updated during interaction
- Output IMPOSSIBLE to warn early failure

Can Models Estimate Budget?

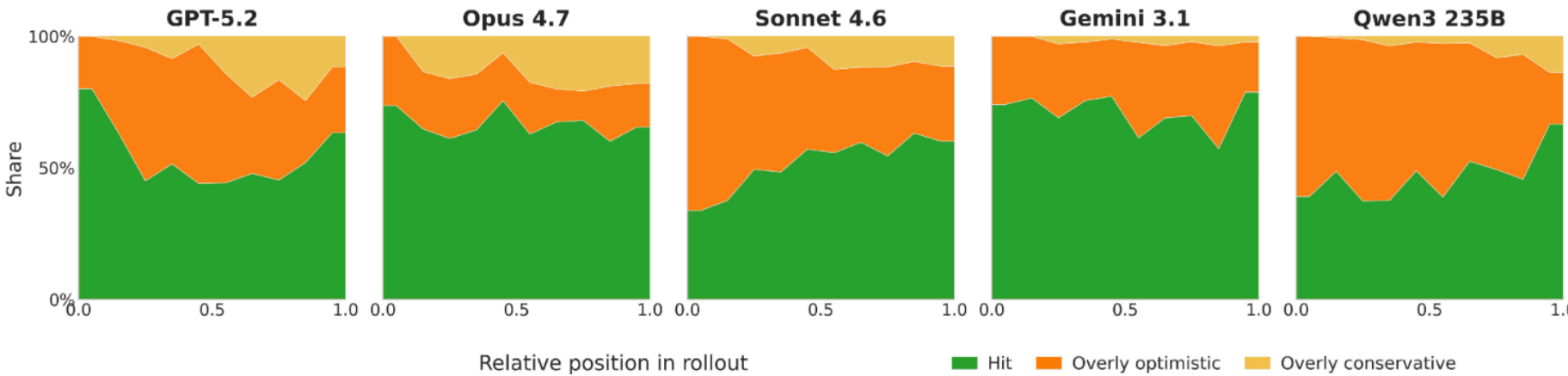




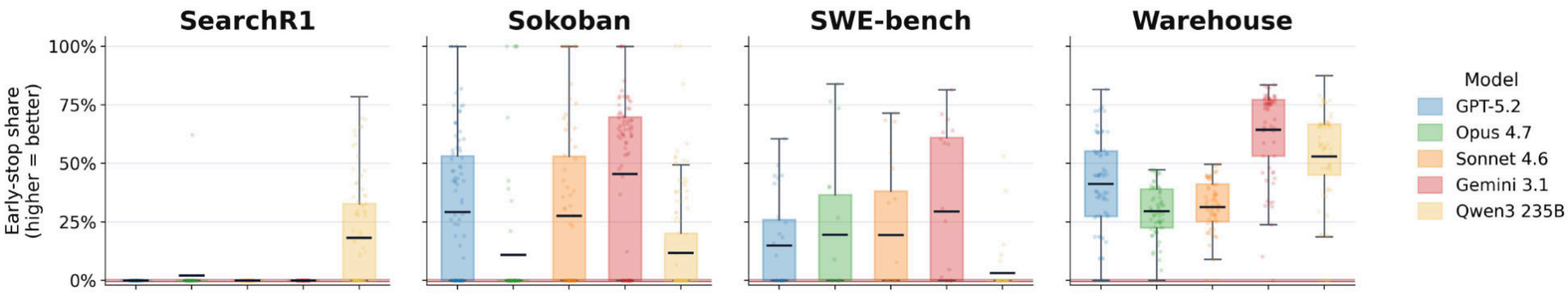


Task Success Rate - Budget Estimation: Correlation $r = 0.35$

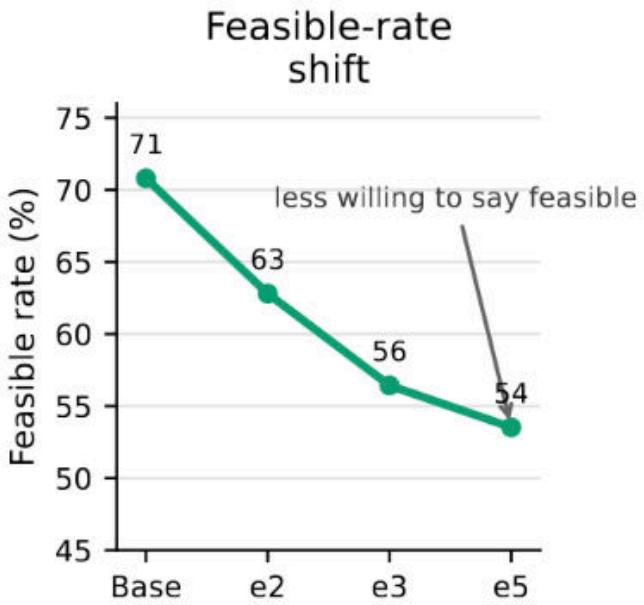
Overoptimistic: Failures are recognized late

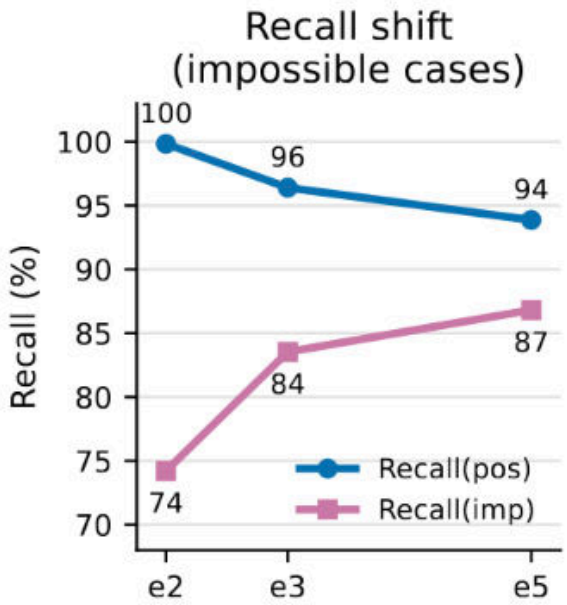


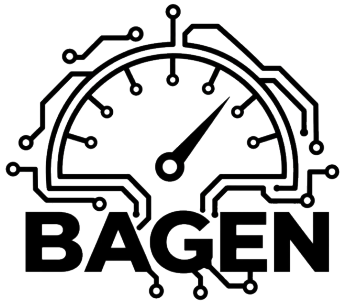
Overoptimistic: Failures are recognized late



Simple Early-Stop SFT to teach model to stop early

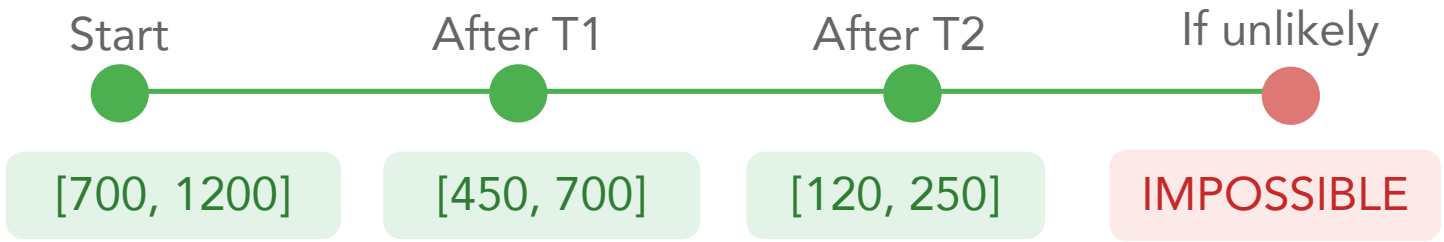






Are Agents **Budget-Aware**?

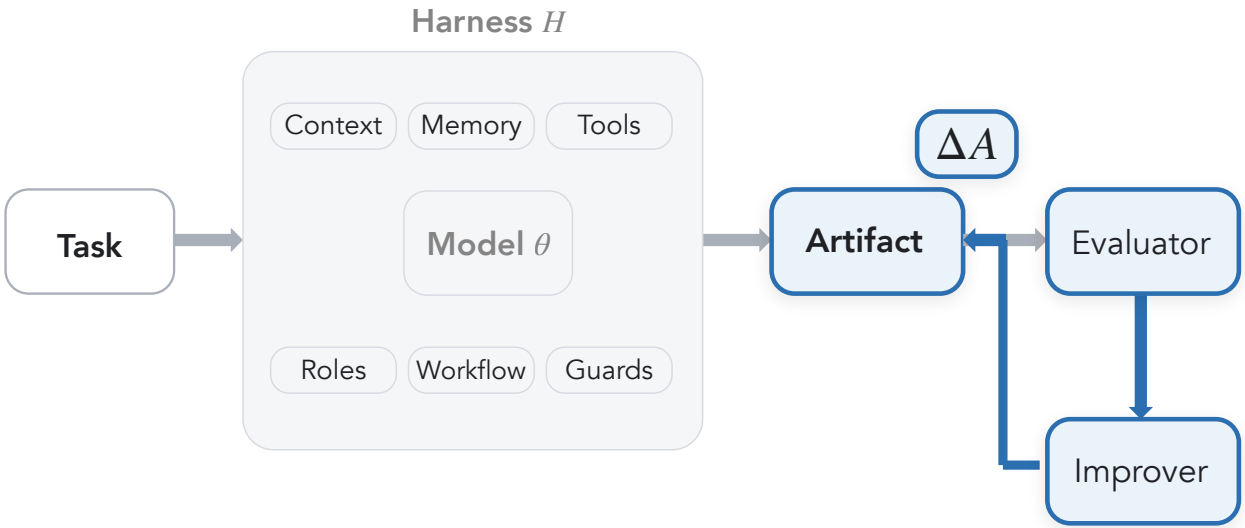
Yuxiang Lin*, Zihan "Zenus" Wang*†, Mengyang Liu*, Yuxuan Shan*, Longju Bai*,
Junyao Zhang, Xing Jin, Boshan Chen, Jinyan Su, Xingyao Wang, Jiaxin Pei, Manling Li



Progressive Interval Estimation

Over-Optimistic Failures	Actionable / Trainable to output IMPOSSIBLE to warn early failure

Summary: Artifact RSI



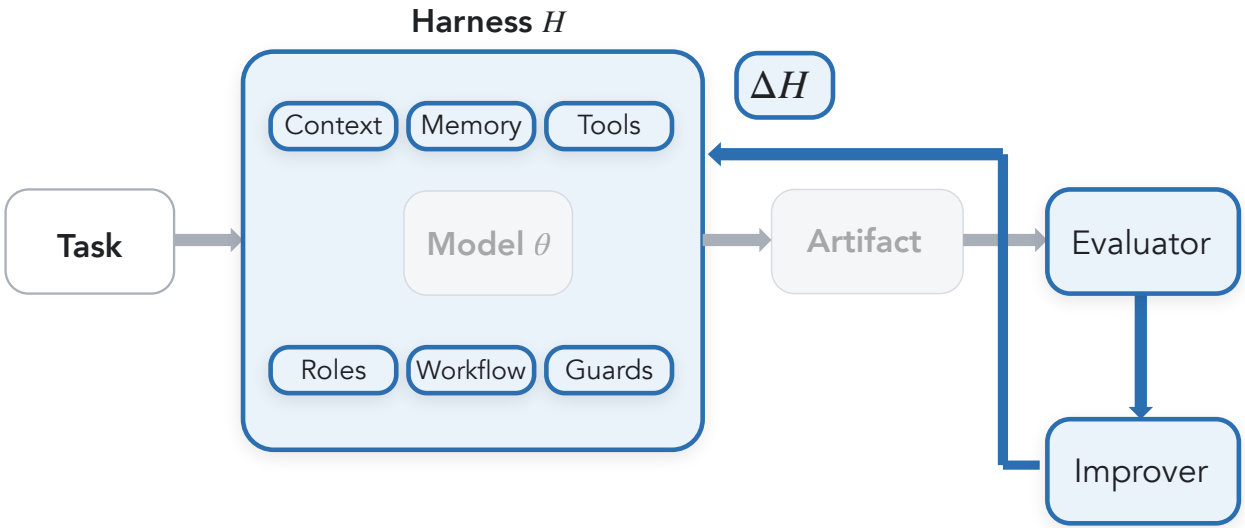
Layer 1: Artifact RSI

Layer 2: Harness RSI

Layer 3: Model RSI

Evaluation of RSI

Summary: Harness RSI



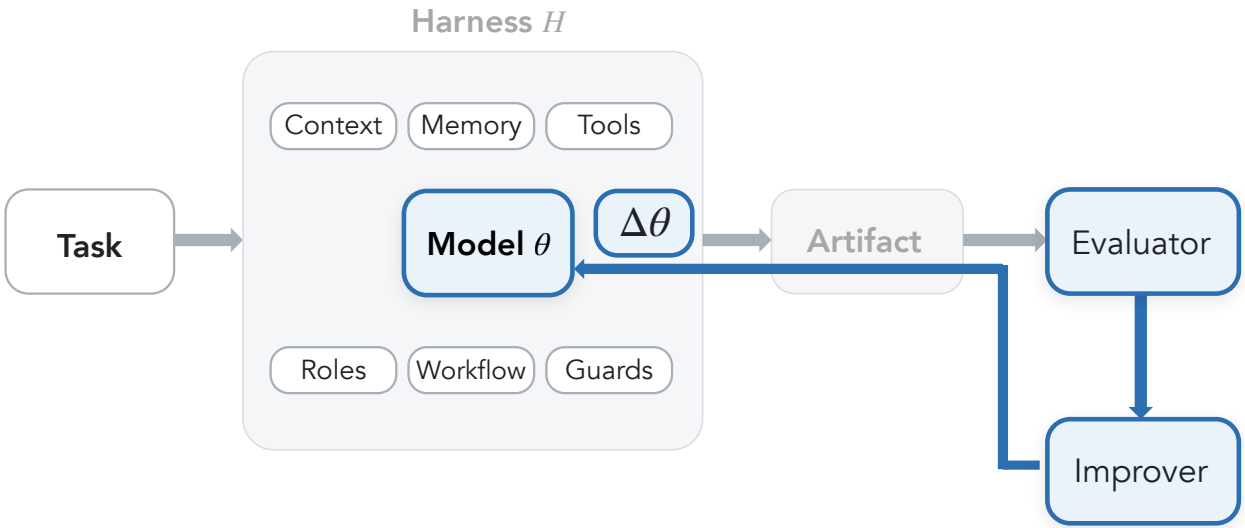
Layer 1: Artifact RSI

Layer 2: Harness RSI

Layer 3: Model RSI

Evaluation of RSI

Summary: Model RSI



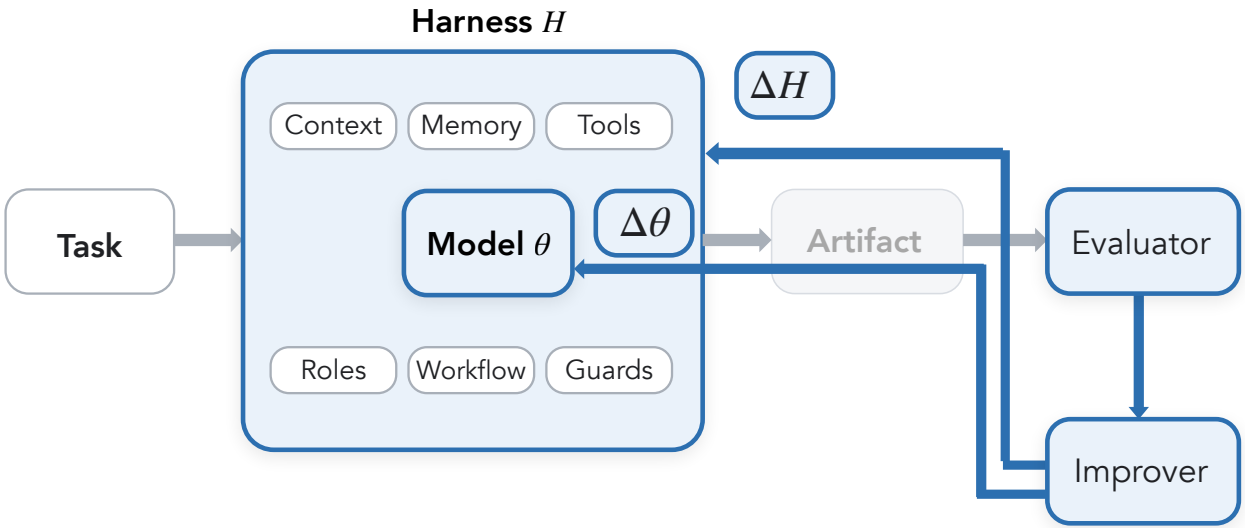
Layer 1: Artifact RSI

Layer 2: Harness RSI

Layer 3: Model RSI

Evaluation of RSI

Summary: Evaluation



Layer 1: Artifact RSI

Layer 2: Harness RSI

Layer 3: Model RSI

Evaluation of RSI

Takeaways

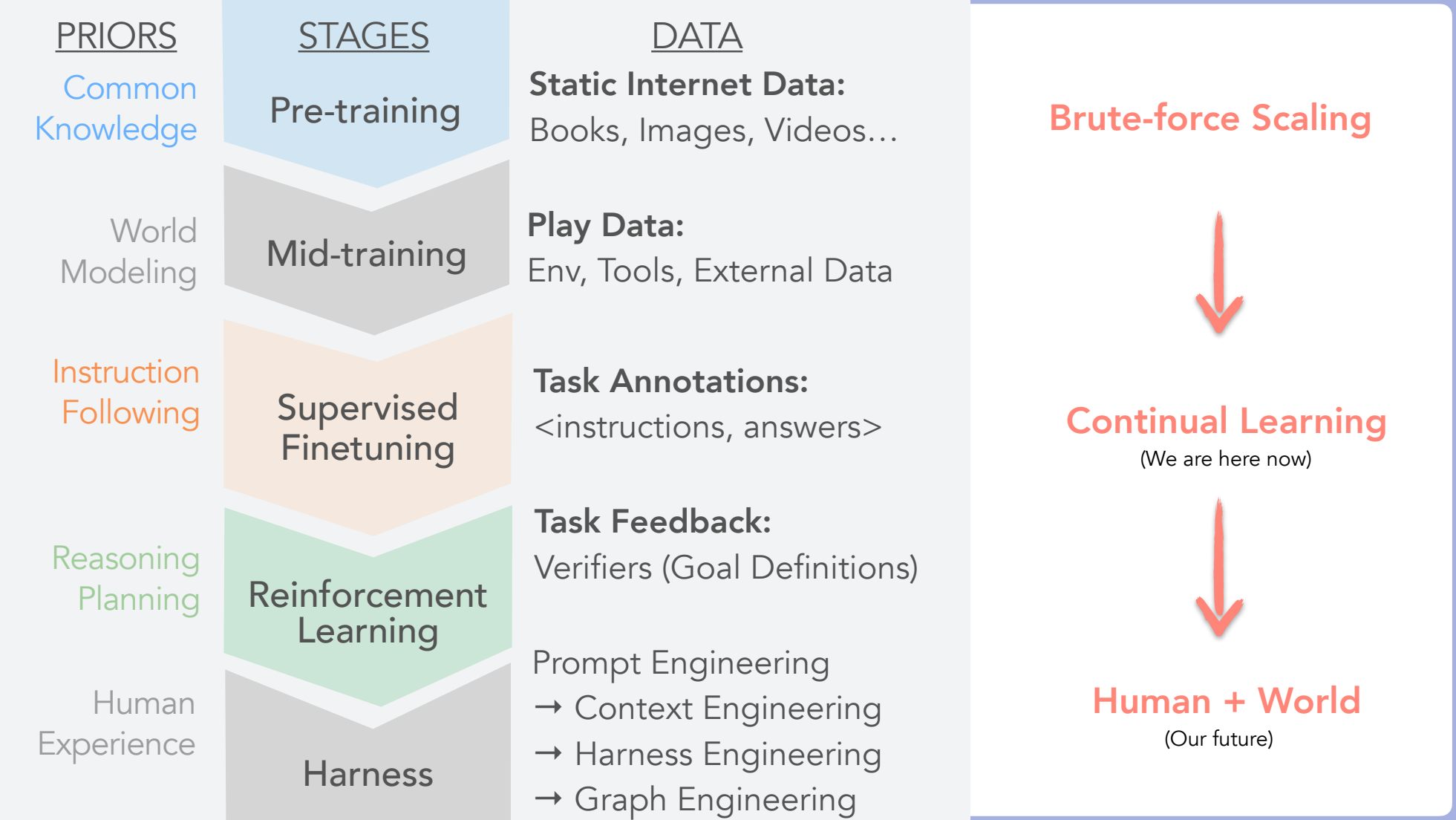
1. The key of AutoResearch is a good Evaluator

2. Outer loop = research loop; Inner loop = interaction loop (with harness layer)

3. Skills are more abstract when models are stronger

4. Training data: text gradient for training

5. Latency-Aware planning


PRIORS
STAGES
DATA
Common Knowledge
Pre-training
Static Internet Data:
Books, Images, Videos…
World Modeling
Mid-training
Play Data:
Env, Tools, External Data
Instruction Following
Supervised Finetuning
Task Annotations:
<instructions, answers>
Reasoning Planning
Reinforcement Learning
Task Feedback:
Verifiers (Goal Definitions)
Human Experience
Harness
Prompt Engineering
→ Context Engineering
→ Harness Engineering
→ Graph Engineering
Brute-force Scaling
Continual Learning
(We are here now)
Human + World
(Our future)



MLL LAB
Machine Learning and Language

THANK YOU

Failure Modes in Agentic AI

Slides, reading list, and materials at failure-mode-agent.github.io

Zoey Sha Li, Zihan Wang, Hejie Cui, Cheng Qian, and Manling Li