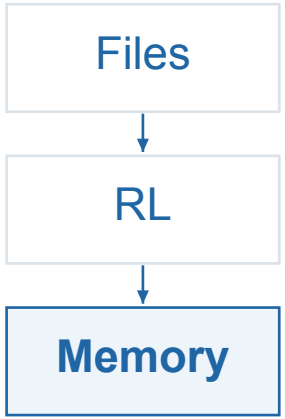
Coding Agent Memory Post-training



Unlocking the Memory Potential of Pre-trained File Operations for Long-Horizon Tasks via Reinforcement Learning

Familiar file operations → learned, useful memory

Lirui Luo · Kelong Mao · Heming Xia · Rongqing Li · Xinwei Yang · Luyu Chen · Kieran Wong · Yudong Guo · Xinrui Wang · Jiayin Zhu · Simiu Gu · Sulong Xu · Cong Fang

JD.com

Project · code · slides

The story in four questions

JD.com

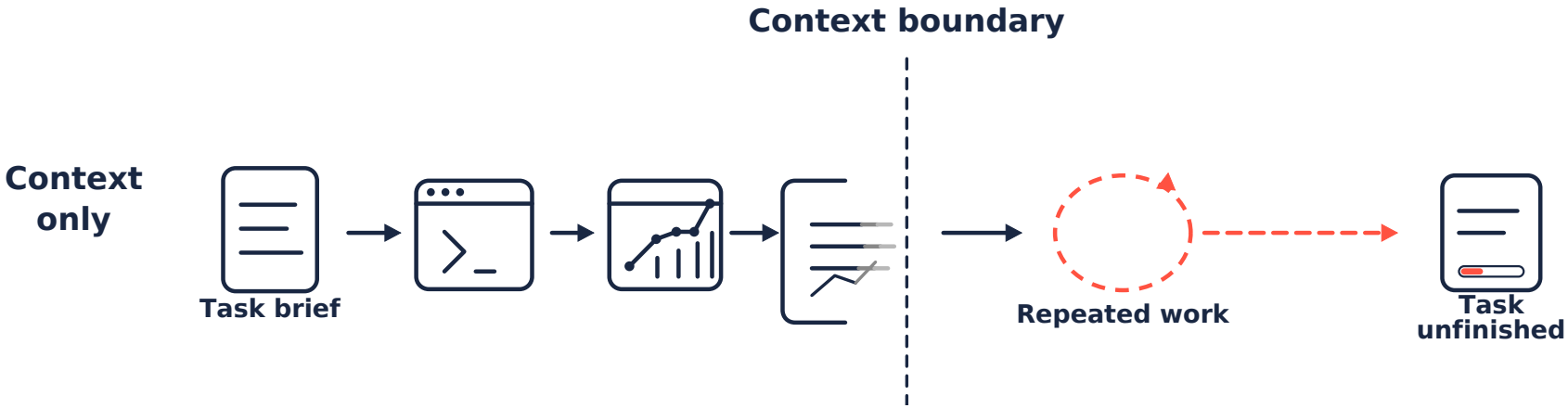
01	**Background**	What memory problem remains? Need → underused tools → prior mismatch → question
02	**Environments**	What must the agent remember? Four domains, one persistent workspace
03	**Method**	How does memory become a learned skill? Task reward + fully asynchronous PPO
04	**Results & analysis**	Does it help, why, and does it transfer? Q1 Effectiveness → Q2 Mechanism → Q3 Transfer

Why memory matters → how it is learned → what it enables.

Long tasks need memory beyond the active context

JD.com

Earlier findings still matter after they leave the prompt.

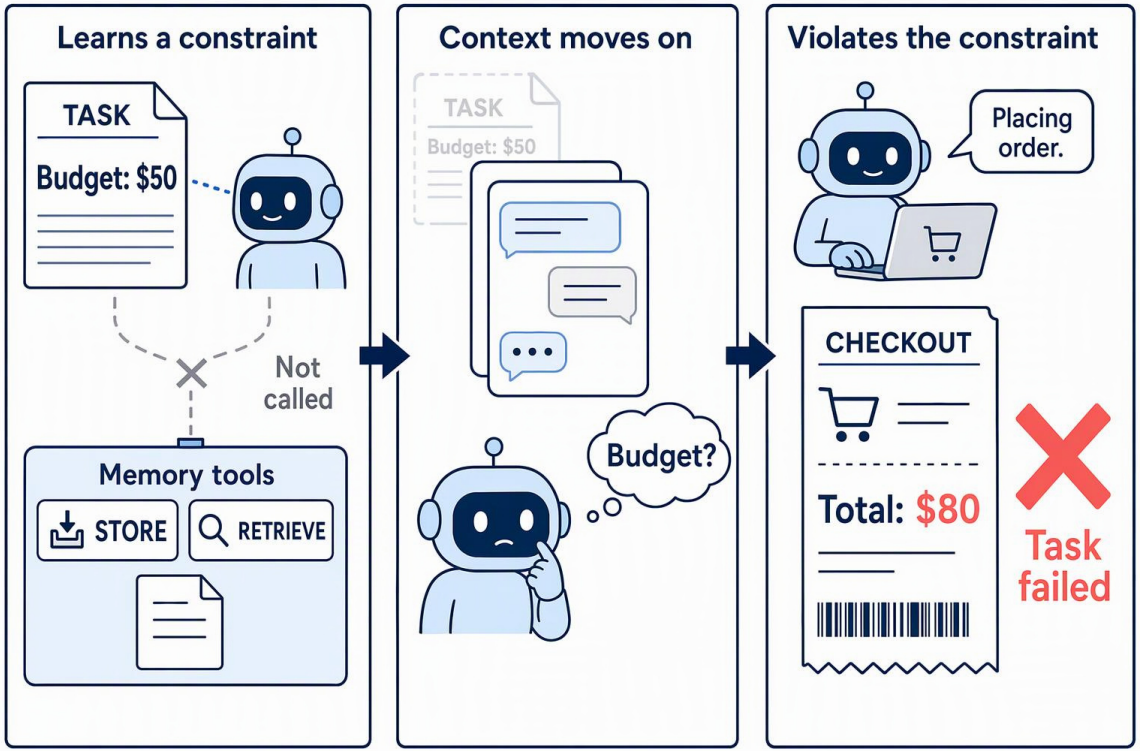


Without memory: repeat work, forget constraints, lose progress.

Figure 1 · context-only row

Having memory tools does not ensure their use

JD.com



OUR HYPOTHESIS

Post-training memory actions can be poorly aligned with the model's pre-trained behavioral priors.

How can post-training teach effective memory use?

Three contributions to learning reusable memory

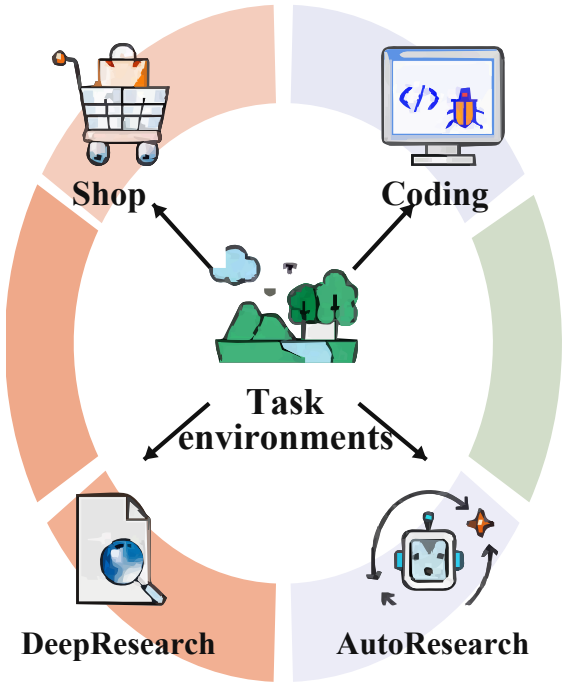
JD.com

01	CAMG	**A long-horizon environment suite**	Four domains with native tools, shell access and persistent files.
02	CAMG-RL	**Learning memory from task reward**	One policy learns task actions and ordinary file-memory operations.
03	Empirical study	**Useful memory that can be reused**	Stronger task execution, content-sensitive behavior and transfer.

Environment suite → learning method → reusable behavior.

CAMG: four kinds of long-horizon work

JD.com



WHAT MUST THE AGENT REMEMBER?

Shop

Preferences and purchase history.

Coding

Diagnosis, patches and test progress.

DeepResearch

Sources and intermediate findings.

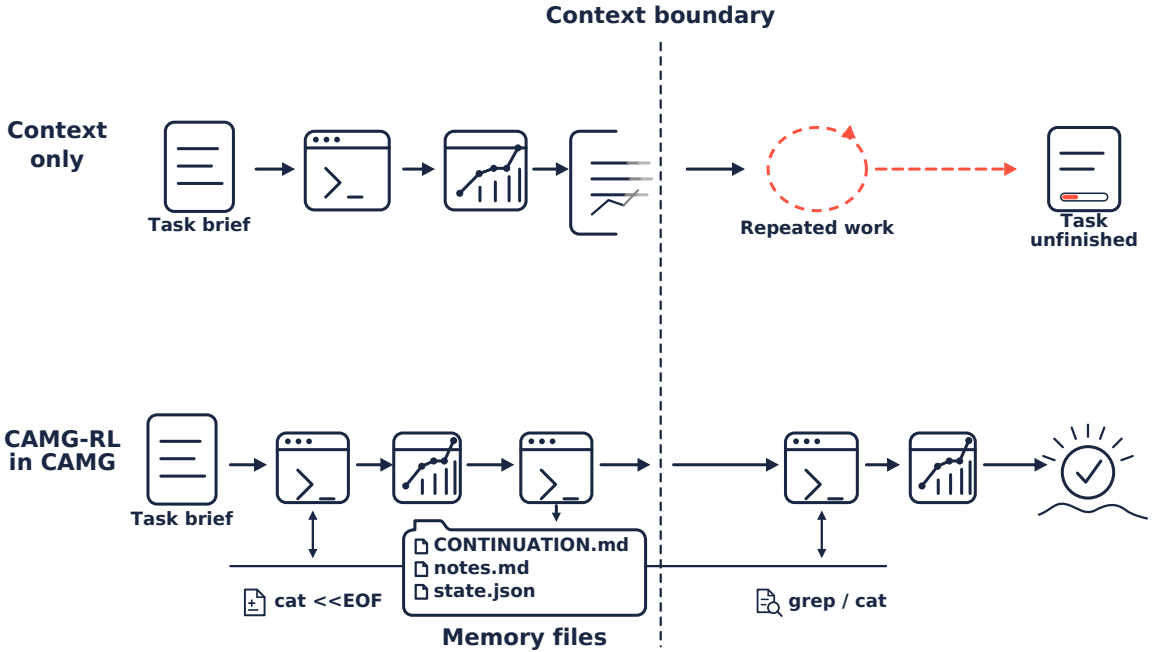
AutoResearch

Hypotheses, trials and results.

Later decisions depend on information gathered earlier.

Use files to carry working state beyond the context

JD.com



Write

Plans, evidence, intermediate results

Keep

Files survive context replacement

Read

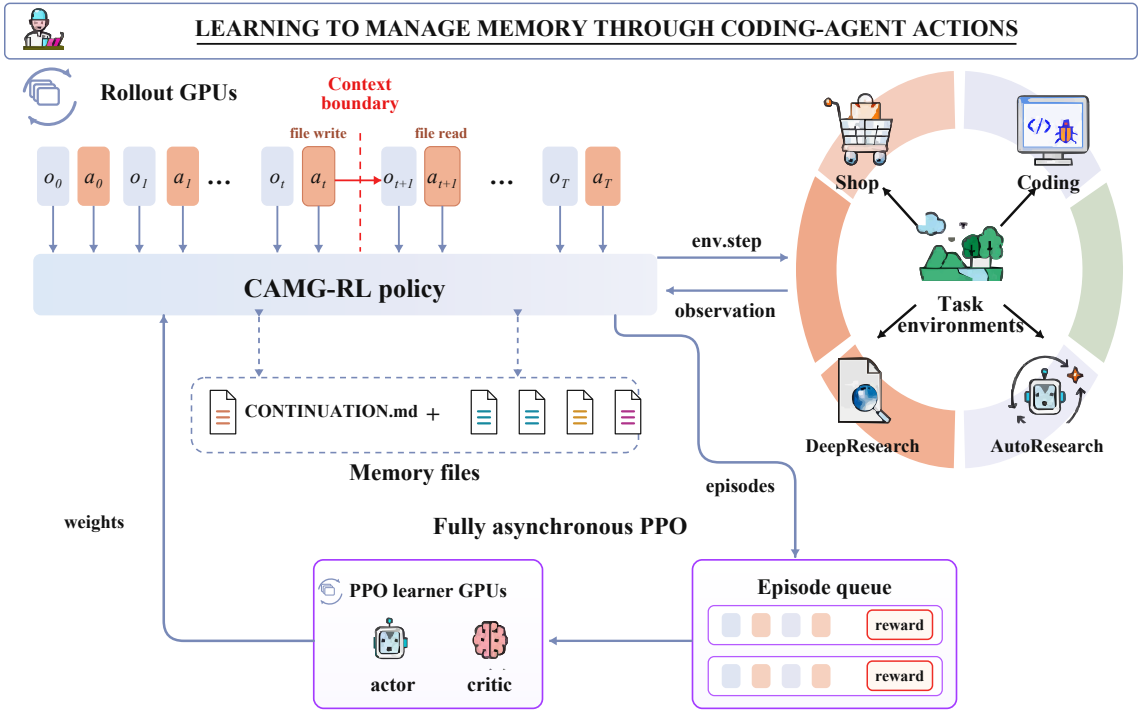
Recover state and continue the task

A familiar interface gives RL somewhere useful to start.

Figure 1 · conceptual illustration

Task reward trains both acting and remembering

JD.com



Shared task policy

Task and file actions in one trajectory.

Task reward

Learn what to retain and reuse.

Fully asynchronous PPO

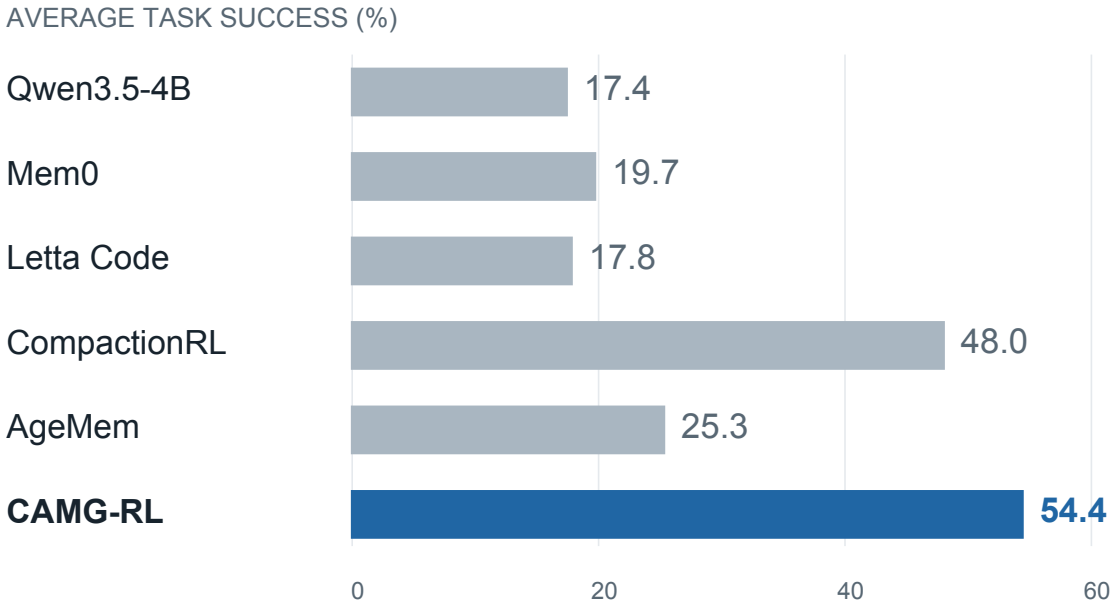
Rollout and learning proceed concurrently.

No extra reward for writing a memory file.

CAMG-RL reaches 54.4% average task success

JD.com

Compare six approaches with Qwen3.5-4B on the same held-out tasks.



+6.4 pp

vs. CompactionRL on held-out tasks

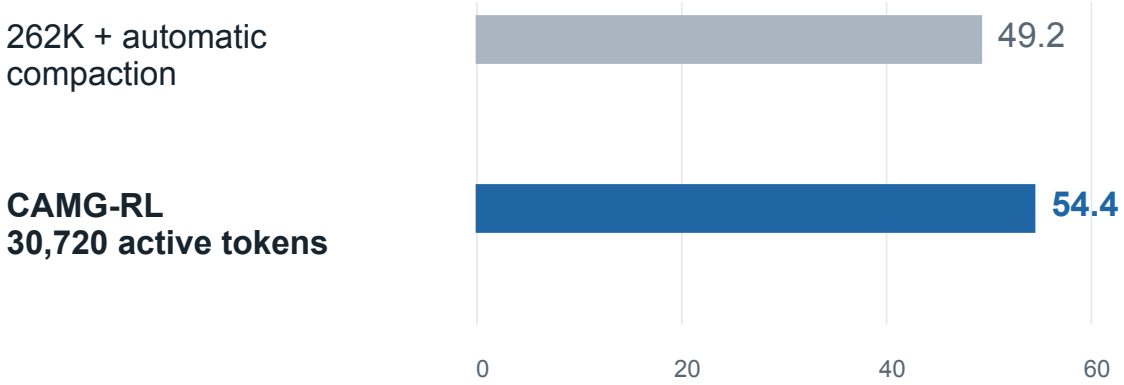
Largest gains: Coding +7.1 pp · AutoResearch +14.1 pp

Learned file memory beats a 262K-context reference

JD.com

Same base, PPO, task rewards and 200 updates; context and compaction differ.

AVERAGE SUCCESS (%)



+5.2 pp

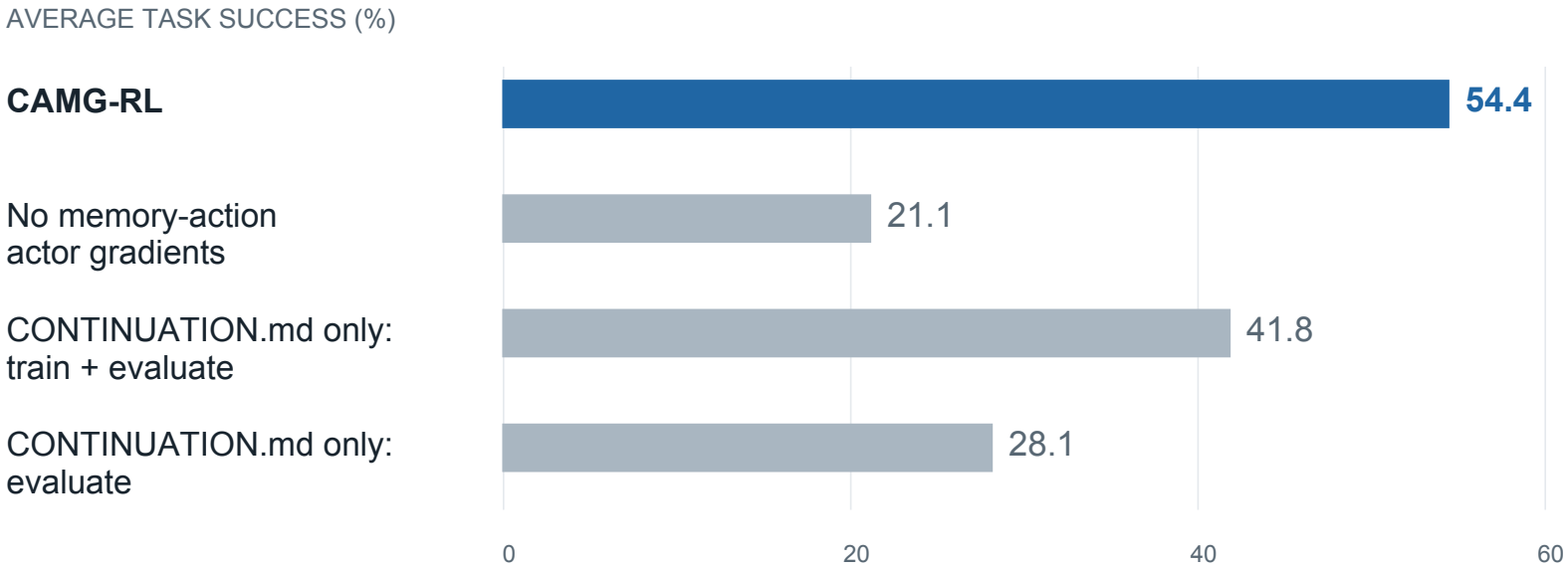
Average task success with 30,720 active tokens

A larger window alone is not the only route to stronger execution.

Memory actions need the learning signal

JD.com

Remove memory-action actor gradients (keep rewards/critic), or use only CONTINUATION.md.



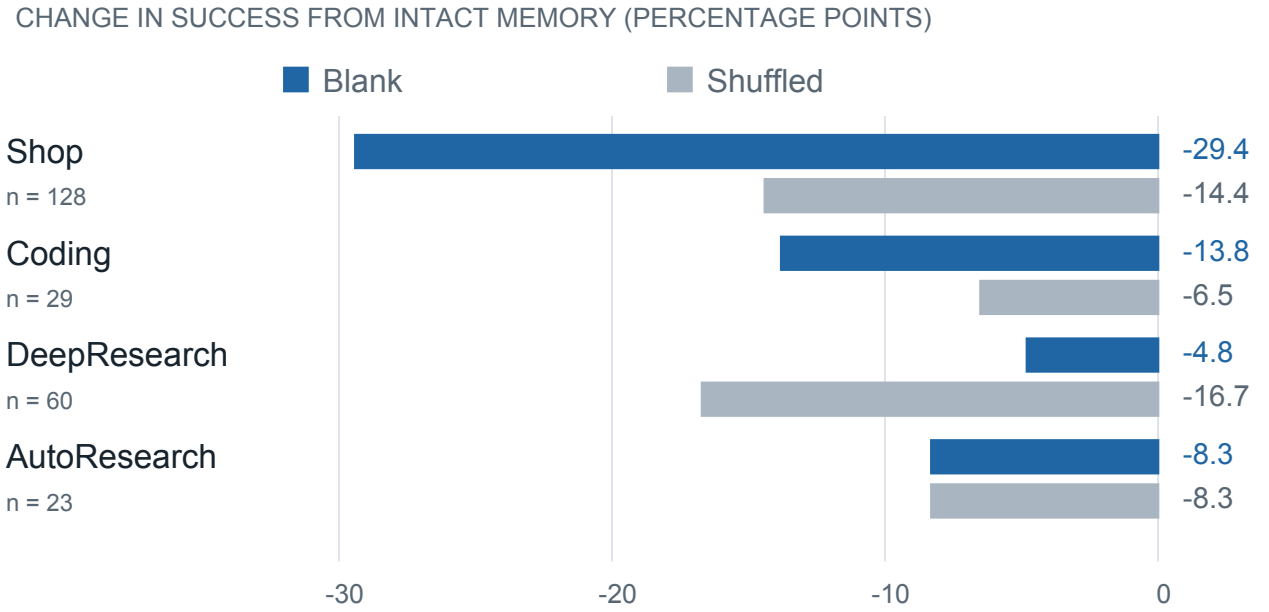
Task reward can train what to save—and how to use it.

Figure 4

The saved content changes what happens next

JD.com

Resume from the same post-reset state with intact, blank or shuffled memory.



−14.1 pp

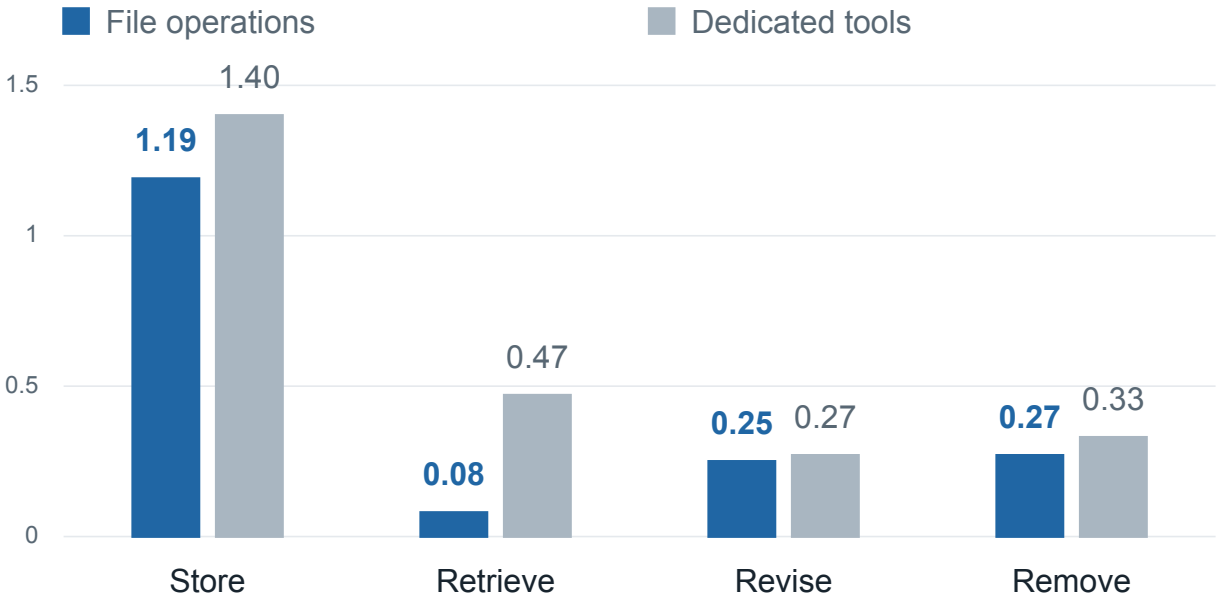
Equal-environment mean when memory is blanked

Erase or shuffle the memory: success falls in every domain.

The file interface is familiar before RL

JD.com

Score file and dedicated-tool memory actions with frozen Qwen3.5-4B.



NEGATIVE LOG-LIKELIHOOD
NATS / ACTION TOKEN
LOWER IS BETTER

0.08

Retrieval NLL with files vs. 0.47 with dedicated tools

Build on an interface the model already knows.

Figure 3a

Case study: carry a Scrapy fix across contexts

JD.com

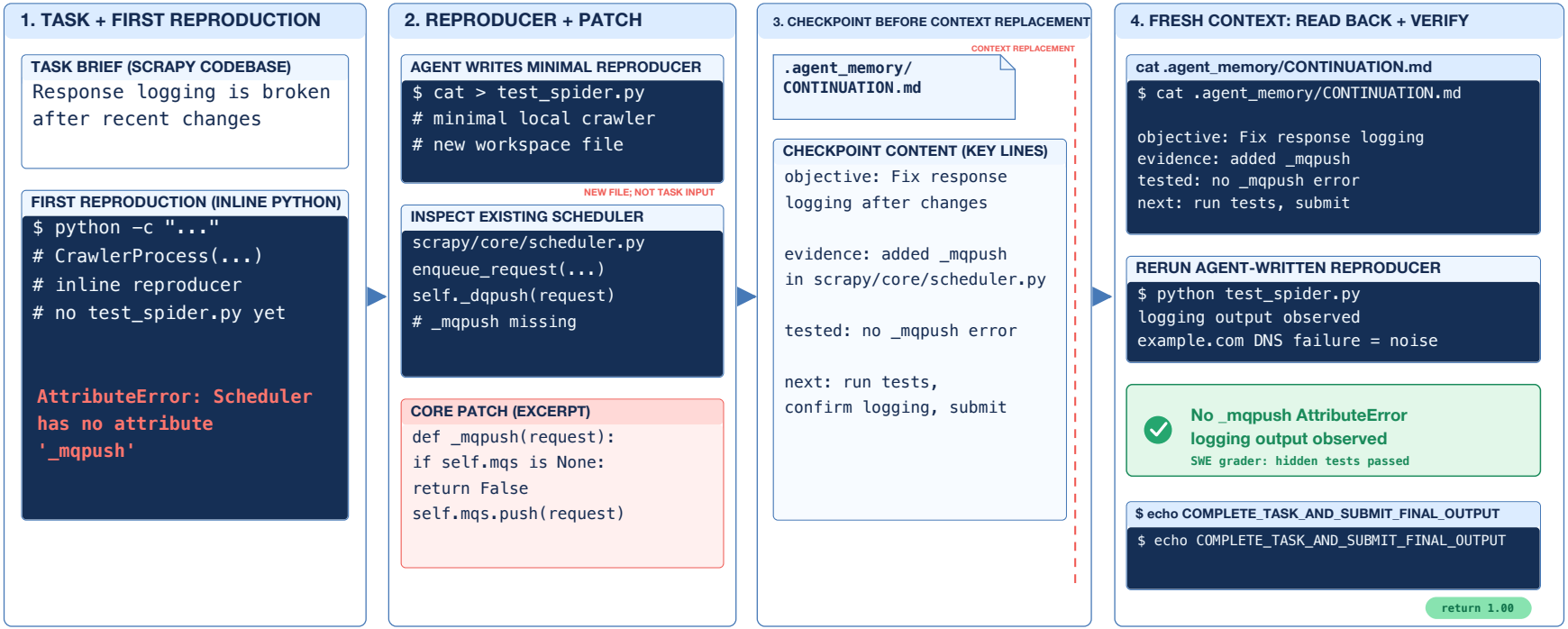
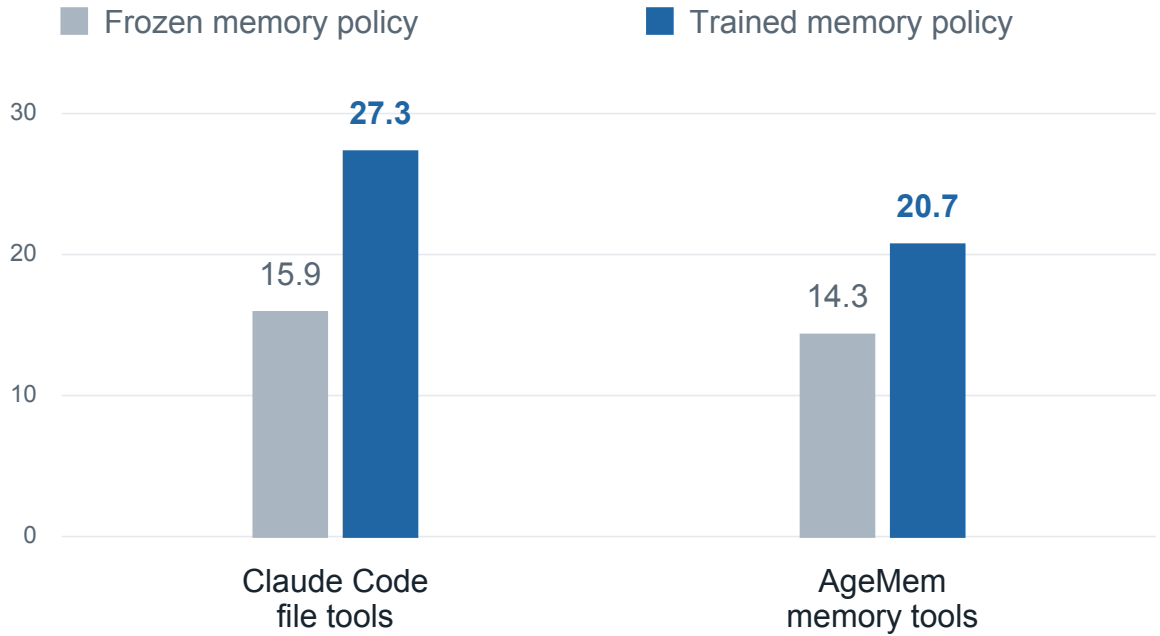


Figure 5

The learned memory policy transfers across interfaces

JD.com

Keep the task executor fixed; replace only the memory policy, with no further training.



AVERAGE SUCCESS (%)

+11.4 pp

Claude Code file tools

+6.4 pp

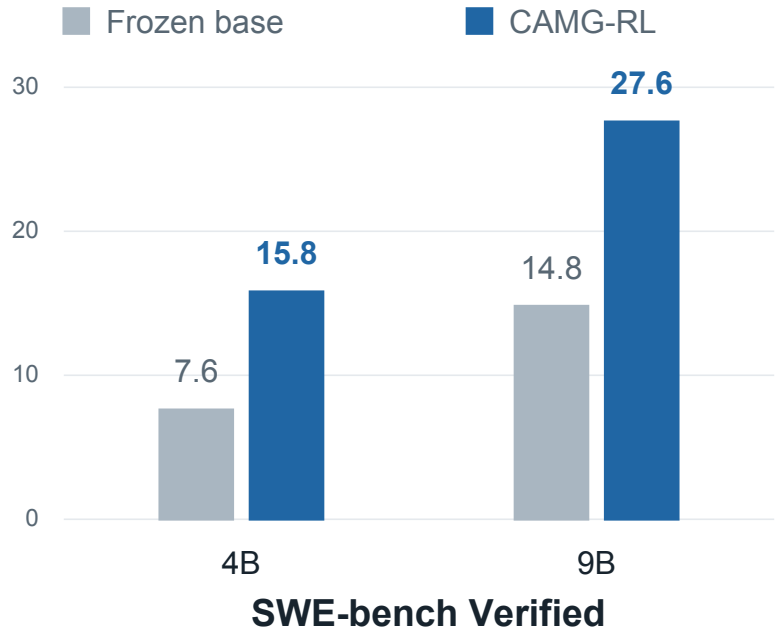
AgeMem memory tools

A trained memory policy improves success through both interfaces.

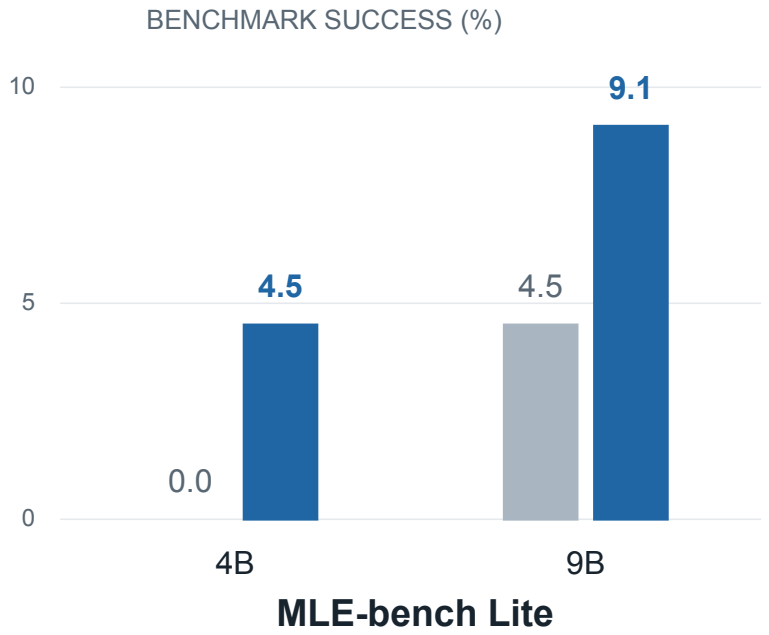
Gains carry over to external benchmarks

JD.com

Compare frozen base and trained policies at the same 4B and 9B model scales.



SWE-bench Verified



MLE-bench Lite

Improve the trained policy's own base on both benchmarks.

Takeaways

JD.com

01 **Memory interfaces should build on pre-trained skills.**

File operations give memory learning a familiar starting point.

02 **Learned memory behaviors generalize to new tasks.**

A shared policy uses memory across environments and external tasks.

liruiluo.github.io/agentmemorygym