ReAct → AGMem

ICLR 2023

ICML 2026 Workshop

Presented by Arvind Jayaraman, Efstratios Reppas, Aryan Roy

Georgia Tech

Outline

Part 1 · ReAct (seminal paper)

- Problem, related work and the ReAct formulation
- Approach: Wikipedia API, four benchmarks, CoT-SC hybrids
- Results: reasoning, decision-making and scaling
- Our take, limitations and discussion
- What persisted and what changed since ReAct

Part 2 · Naive Visual Memory Is Not Enough (frontier paper)

- Problem and four failure modes of GUI agents
- Experiments: why full-image memory falls short
- AGMem: action-grounded visual memory and results
- Omissions, limitations and discussion

ReAct : Synergizing Reasoning And Acting In Language Models

Shunyu Yao, Jeffrey Zhao , Dian Yu , Nan Du , Izhak Shafran , Karthik Narasimhan , Yuan Cao

ICLR 2023

Georgia Tech

Problem Statement

Reasoning and acting are studied apart

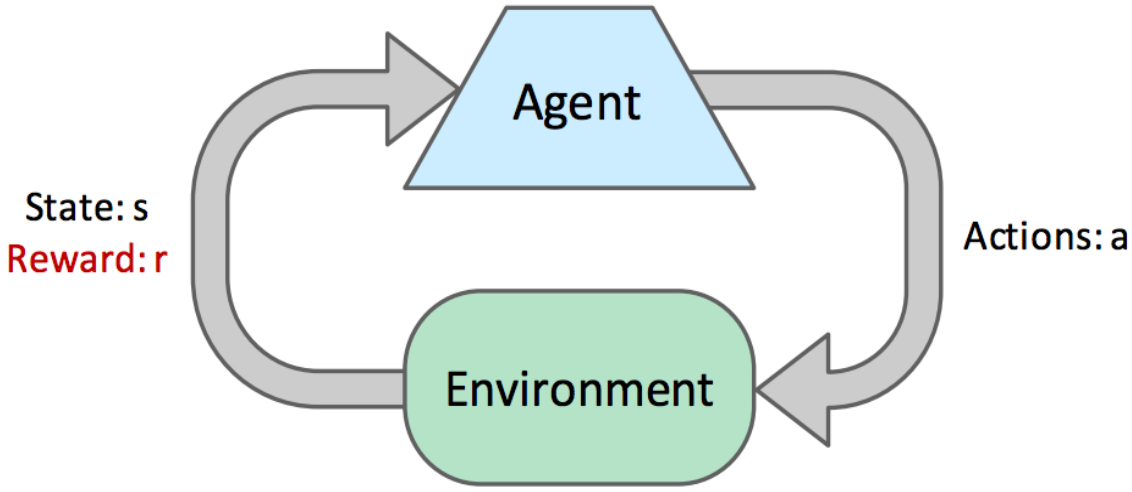
Bridge the gap: chain-of-thought reasons without acting, and action-only agents act without reasoning.

Real environments are noisy

General-purpose agents face HTML pages laden with ads, buggy UI elements and more.

Without reasoning, an agent can't adapt

An agent that cannot reason is essentially an offline planner: it can't revise its plan as observations arrive.



CORE QUESTION

How can we interleave internal reasoning with external interaction in a closed feedback loop?

Georgia Tech

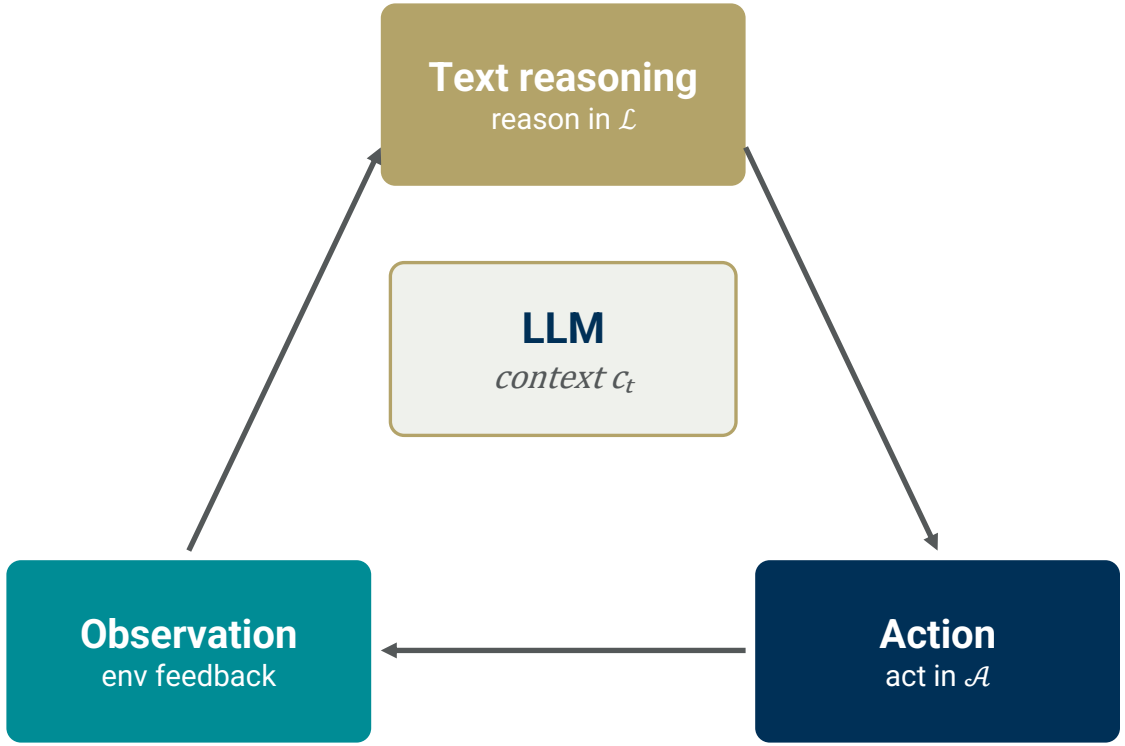
Related Work

Category	Method / Paper	What it Does	Drawbacks
Pure Reasoning	**chain-of-thought (CoT)** *(Wei et al., 2022)*	Generates step-by-step internal reasoning traces before outputting a final response.	No External Grounding
Pure Reasoning	**Least-to-Most / Self-Consistency** *(Zhou et al.; Wang et al., 2022)*	Decomposes complex problems into sub-questions or samples multiple reasoning paths to vote on the final answer.	Static & Offline
Pure Acting	**WebGPT / Act-Only Agents** *(Nakano et al., 2021)*	Maps environment observations directly to browser/API actions via imitation or RL.	Myopic Execution
Pure Acting	**BUTLER** *(Shridhar et al., 2020)*	Imitation-learning agent for ALFWorld, trained on 10^5 expert trajectories per task type.	Data-Hungry
Embodied Precursors	**SayCan & Inner Monologue** *(Ahn et al.; Huang et al., 2022)*	Grounds high-level LLM subgoals using robotic affordance functions and basic environment feedback.	Domain-Restricted

Georgia Tech

The ReAct Formulation

- Standard agent: at step t it sees observation $o_t \in \mathcal{O}$, takes action $a_t \in \mathcal{A}$ via policy $\pi(a_t \mid c_t)$
- Context $c_t = (o_1, a_1, \ldots, o_{t-1}, a_{t-1}, o_t)$: mapping $c_t \to a_t$ is hard when it needs implicit, multi-step reasoning
- **ReAct's key move:** augment the action space with language, $\hat{A} = \mathcal{A} \cup \mathcal{L}$
- A text reasoning step $\hat{a}_t \in \mathcal{L}$ does not touch the environment and yields no observation; it only updates context: $c_{t+1} = (c_t, \hat{a}_t)$
- $\mathcal{L}$ is unbounded, so the policy needs a strong language prior: ReAct uses a frozen LLM (PaLM-540B) prompted with a few example trajectories



Text reasoning costs no environment step, but reshapes every action that follows

Georgia Tech

ReAct in Action: HotpotQA

Q: *Aside from the Apple Remote, what other device can control the program the Apple Remote was originally designed to interact with?*

Text reasoning	*I need to search Apple Remote and find the program it was designed for.*
Act	Search[Apple Remote]
Obs	…originally designed to control the Front Row media center program…
Text reasoning	*Next, search Front Row and find what other device controls it.*
Act → Obs	Search[Front Row] → Could not find. Similar: 'Front Row (software)', …
Text reasoning	*Front Row is not found. I should search Front Row (software).*
Act → Obs	Search[Front Row (software)] → …controlled by an Apple Remote or keyboard function keys
Act	Finish[keyboard function keys] ✓

Trajectory condensed from Yao et al. (2023), Fig. 1

What the text reasoning did

- Decomposed the question into two lookups
- Recovered from a failed search by reformulating
- Pulled the answer out of a noisy observation

In the paper's Fig. 1, Standard, CoT-only and Act-only prompting all answer this question wrong.

Georgia Tech

Approach: Building ReAct

1

Few-shot prompting, no training

A frozen PaLM-540B is prompted with a handful of human-written trajectories (3–6 for QA, 1–2 for ALFWorld). Text reasoning, actions and observations are all just text.

2

Dense vs. sparse text reasoning

Reasoning-heavy tasks (HotpotQA, FEVER) alternate text reasoning → action → observation every step. Long-horizon tasks (ALFWorld, WebShop) let the model decide when to think.

Georgia Tech

Approach: A Three-Action Wikipedia API

`search[entity]`

Returns the first 5 sentences of the entity's Wikipedia page, or the top 5 similar titles if no page matches

`lookup[string]`

Returns the next sentence on the current page that contains the string, like Ctrl+F in a browser

`finish[answer]`

Submits the answer and ends the episode

Deliberately weaker than a real retriever. The model cannot pull the right passage in one call, so it has to search, read and refine queries in text reasoning the way a person browsing Wikipedia would.

Georgia Tech

Approach: Four Benchmarks

Benchmark	Task	What the agent does	Metric	Text reasoning	Demos
HotpotQA Knowledge-intensive	Multi-hop question answering	Answers questions that need facts from two or more Wikipedia pages, given only the question	Exact match	Dense	6
FEVER Knowledge-intensive	Fact verification	Labels a claim Supports, Refutes or Not Enough Info using Wikipedia evidence	Accuracy	Dense	3
ALFWorld Decision-making	Text-based household game	Completes multi-step chores such as finding, cleaning and placing objects across 6 task types	Success rate	Sparse	2
WebShop Decision-making	Simulated online store	Searches and browses 1.18M real products to buy one that matches a text instruction	Score, success rate	Sparse	1

Dense: a text reasoning step before every action. Sparse: the model decides when to think. Demos: hand-written in-context trajectories (ALFWorld: per prompt, 6 prompts per task type). Eval: 134 unseen ALFWorld games, 500 WebShop test instructions. Source: Yao et al. (2023), Sections 3–4.

Georgia Tech

Approach: Prompting and Finetuning Details

1 Prompting: no training, only worked examples in the prompt

- **Examples:** 6 hand-written solved cases for HotpotQA, 3 for FEVER; adding more did not help
- **ALFWorld:** each prompt shows 2 example runs for the task type
- **Decoding:** greedy, capped at 7 / 5 steps (HotpotQA / FEVER); only 0.84% / 1.33% of correct runs hit it

2 Finetuning: smaller models learn from ReAct's own successes

- **Data:** 3,000 correct HotpotQA runs from the prompted 540B model train PaLM-8B and 62B
- **Training:** ReAct and Act improve up to 4,000 steps; Standard and CoT degrade after 1,000–2,000

Improving ReAct: Combining It with CoT-SC

What is CoT-SC?

Chain-of-thought with self-consistency (Wang et al., 2022)

- Sample **21 chains** at temperature 0.7: **21× the cost** of one CoT pass
- Answer with the **majority vote**
- Uses only the model's **internal knowledge**: no actions, no retrieval

ReAct → CoT-SC **Best on HotpotQA: 35.1 EM**

Run ReAct → No answer within 7 steps (HotpotQA) or 5 (FEVER) → **Fall back to CoT-SC**

Starts from scratch; ReAct's attempt is discarded

CoT-SC → ReAct **Best on FEVER: 64.6% accuracy**

Run CoT-SC → Majority answer gets fewer than 11 of 21 votes → **Switch to ReAct**

Fresh ReAct run on the question

Each hybrid backs off when its own signal is weak: ReAct can loop on failed searches, while CoT-SC can hallucinate.

Georgia Tech

Improving ReAct: Why Hand Off to CoT-SC?

Text reasoning plans searches

ReAct's text reasoning decides what to look up next. When searches return nothing useful, it **repeats the same text reasoning and action** instead of answering from memory.

One trajectory, no vote

ReAct decodes greedily, producing a **single answer path with no confidence signal**. CoT-SC's 21 samples and majority vote supply one.

More steps don't help

Raising ReAct's step budget **did not improve results**, so stopping and switching costs less than letting it loop.

Both switches are hand-set rules, not learned. ReAct → CoT-SC discards ReAct's attempt and answers from scratch; CoT-SC → ReAct runs ReAct only when the vote is split.

Georgia Tech

Budgets: ReAct Trades Training Cost for Inference Cost

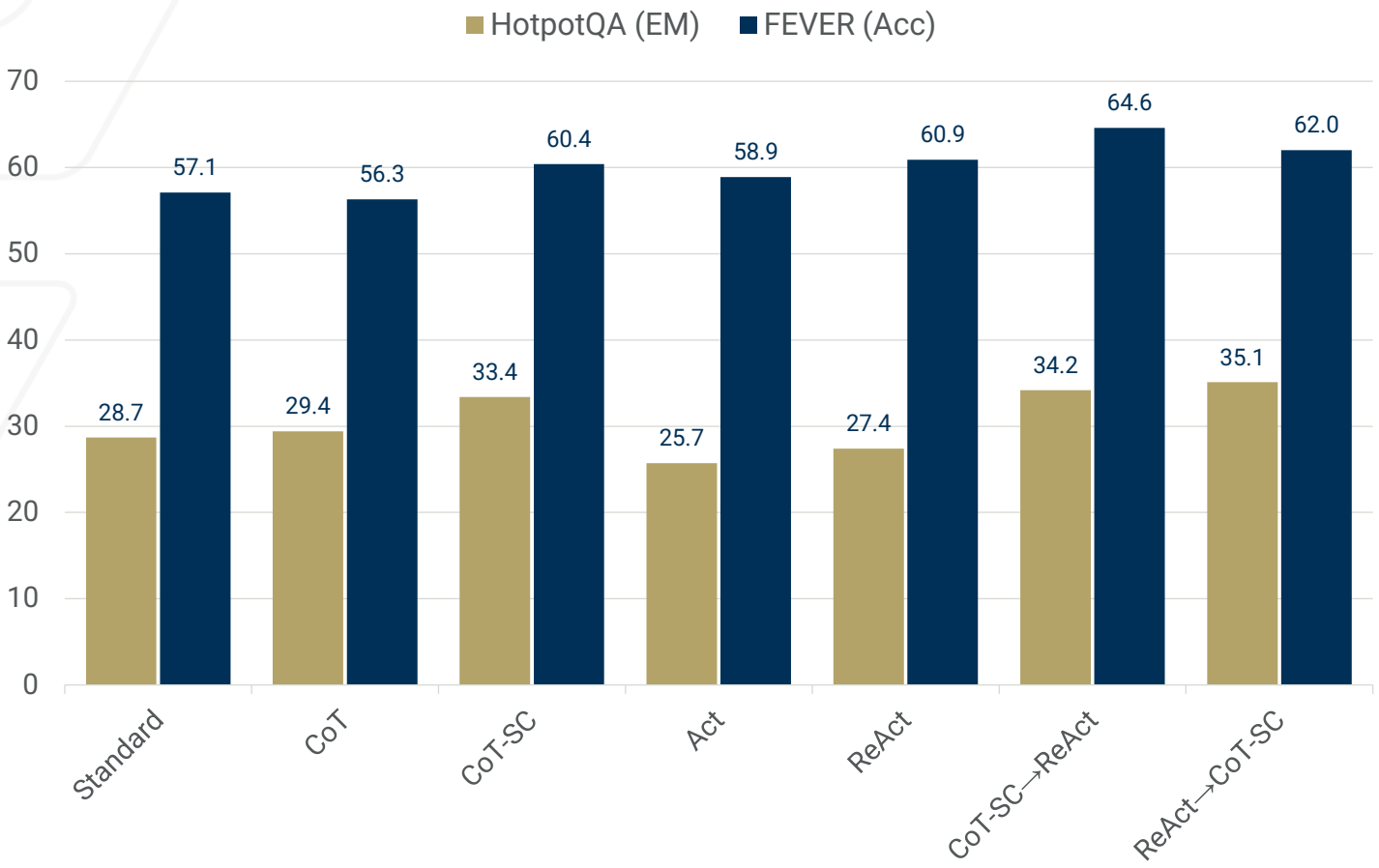
Method	Training cost	Inference cost per question
ReAct, prompted PaLM-540B, frozen	None: 1–6 hand-written demos	1 greedy pass of a 540B model
CoT-SC PaLM-540B, frozen	None: ReAct's demos, text reasoning only	21 sampled passes of a 540B model
ReAct, finetuned PaLM-8B and 62B	3,000 trajectories, 4,000 steps	1 pass of a smaller model
BUTLER ALFWorld baseline	10^5 expert trajectories per task type	Beam search, small trained model
IL+RL WebShop baseline	1,012 human trajectories, then RL	1 pass, small trained model

Training-time vs. test-time adaptation. BUTLER and IL+RL pay up front for a fixed policy; ReAct adapts at test time through demos and its own text reasoning. Adaptive Computation Time (Graves, 2016) learns how much test-time compute to spend; ReAct sets that budget by hand.

Prompted methods skip training but run a 540B model at every step; baselines train small models on large datasets. Source: Yao et al. (2023), Sections 3–4, Appendix B.

Georgia Tech

Results: Knowledge-Intensive Reasoning



PaLM-540B, few-shot prompting. Supervised SoTA: 67.5 HotpotQA EM / 89.5 FEVER Acc.

Reasoning helps acting

ReAct beats Act-only on both tasks (27.4 vs 25.7; 60.9 vs 58.9).

Grounding vs. flexibility

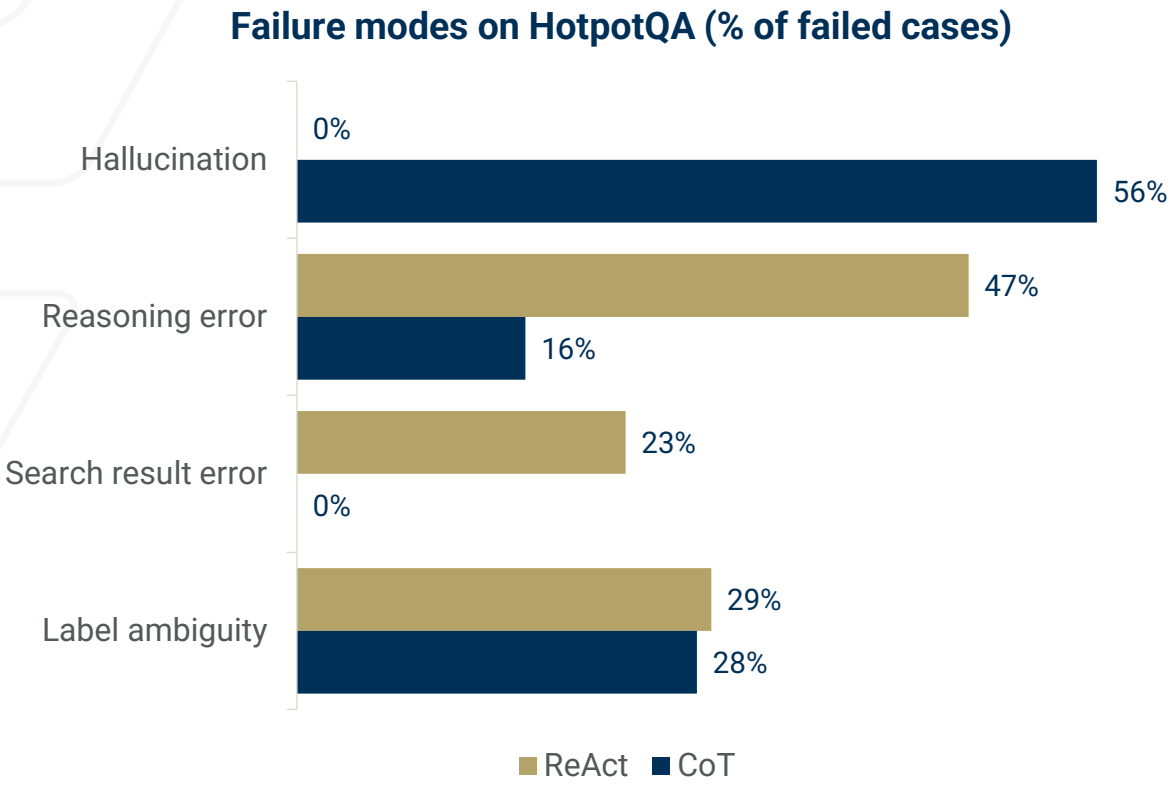
ReAct trails CoT on HotpotQA (27.4 vs 29.4) but wins on FEVER (60.9 vs 56.3), where facts must be checked.

Best of both worlds

Hybrids top both tasks: 35.1 EM on HotpotQA, 64.6% on FEVER.

Georgia Tech

Why It Works: ReAct and CoT Fail Differently



Manual analysis of 50 random successes and 50 failures per method. Label ambiguity is equal (29% vs. 28%). The failures are complementary, which is what the CoT-SC hybrids exploit.

1. **CoT hallucinates; ReAct doesn't**
   56% of CoT failures are made-up facts. ReAct: 0%, since its facts come from Wikipedia.

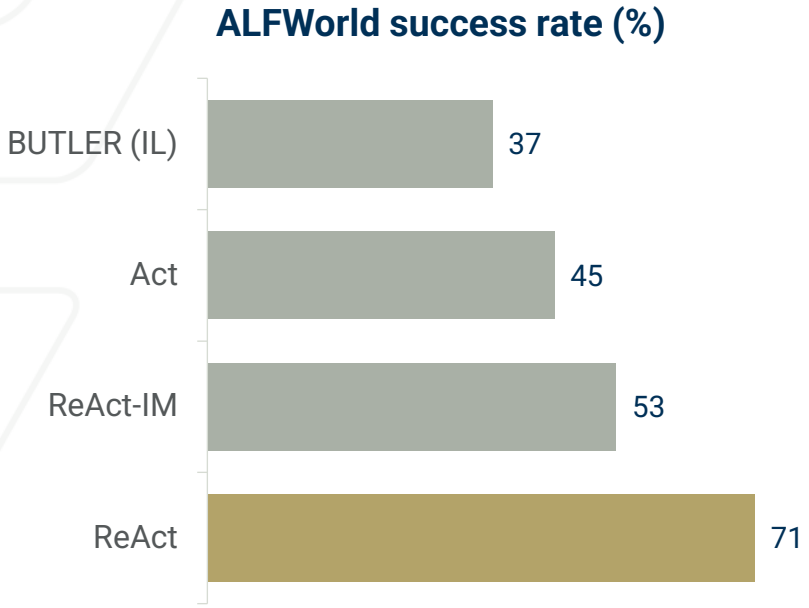
2. **ReAct gets stuck in loops**
   47% of failures are reasoning errors, often repeated text reasoning and actions (CoT: 16%).

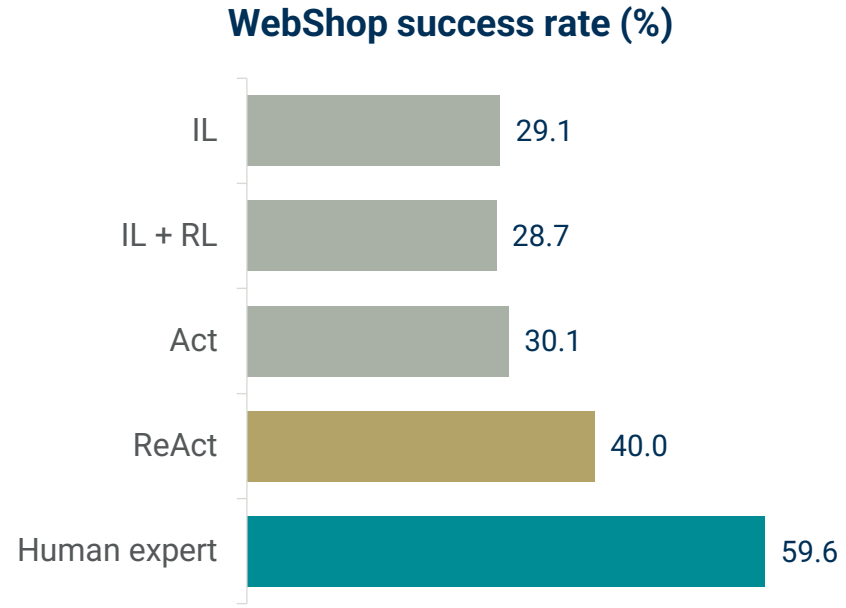
3. **Bad searches derail ReAct**
   23% of ReAct failures follow an uninformative search it can't recover from. CoT never searches.

4. **ReAct's successes are more trustworthy**
   6% of ReAct's correct answers rest on hallucinated reasoning vs. 14% for CoT.

Georgia Tech

Results: Interactive Decision-Making





+34

pts over imitation-learning BUTLER on ALFWorld

+10

pts WebShop success rate over the best baseline

ALFWorld: best of 6 prompts (BUTLER best of 8). WebShop SR = % of episodes buying an item meeting all requirements.

Only 1–2 in-context examples vs. IL/RL methods trained on thousands of expert trajectories. ReAct-IM (Inner-Monologue-style text reasoning that only restates feedback) does worse: high-level goal decomposition and progress tracking are what help.

Georgia Tech

Scaling: Finetuned ReAct Beats Larger Prompted Models

Prompting vs. finetuning

- PaLM-8B, 62B and 540B on HotpotQA
- Finetuned on **3,000 correct ReAct trajectories** generated by prompted PaLM-540B
- Prompted at 8B and 62B, ReAct is the **weakest of four methods**: small models struggle to learn reasoning and acting from a few examples

8B vs. 62B HotpotQA

Finetuned PaLM-8B ReAct → outperforms → **Every prompted PaLM-62B method**

Standard, CoT and Act prompting at 62B

62B vs. 540B HotpotQA

Finetuned PaLM-62B ReAct → outperforms → **Every prompted PaLM-540B method**

Standard, CoT and Act prompting at 540B

Finetuning ReAct or Act beats finetuning Standard or CoT: the first two teach the model to look facts up, the others to memorize them, including hallucinated ones.

Georgia Tech

Our Take: The Evidence Shows Within-Task Generalization

1

Unseen games, seen task types

ALFWorld's 134 games are new, but each prompt comes from the same task type. That tests new instances, not new tasks.

2

Headline numbers involve selection

ALFWorld reports the best of 6 prompts. The worst ReAct prompt (48%) still beats the best Act prompt (45%), so the gain holds.

3

The weak API favors reasoning

Search returns only 5 sentences, so answers need several refined queries. A strong retriever could shrink ReAct's edge over Act.

4

Baselines and hybrids are hand-built

Act is ReAct with text reasoning deleted, not a tuned agent. The hand-off thresholds get no sensitivity test.

Georgia Tech

Where ReAct's Claims Overreach

CLAIM

Reasoning + acting helps across tasks

COUNTER-EVIDENCE

Alone, ReAct loses on HotpotQA to both CoT (29.4) and plain Standard prompting (28.7). It only comes out ahead when paired with CoT-SC.

CLAIM

A few demos beat trained agents

COUNTER-EVIDENCE

The edge depends on a 540B model. Prompted ReAct is the weakest method at 8B and 62B, where finetuning wins.

CLAIM

Text reasoning makes agents interpretable and steerable

COUNTER-EVIDENCE

Steering is shown with one edited ALFWorld run. And 6% of ReAct's correct answers rest on hallucinated reasoning, so a trace can read well and still be wrong.

The gains are real but conditional: on the task, on model scale, and on a single anecdote.

Georgia Tech

Limitations & Societal Implications

Limitations

- **Context-bound prompting:** complex tasks need more demonstrations than fit in the prompt
- **Reasoning loops:** the agent repeats earlier text reasoning and actions and can't escape
- **Only as good as its tools:** uninformative search results derail whole trajectories
- **Still far from supervised SoTA:** 30+ points behind on HotpotQA and FEVER

Societal Implications

- **Interpretable & steerable:** humans can read text reasoning and edit it mid-task to correct the agent
- **Plausible ≠ faithful:** fluent reasoning traces can earn trust they haven't earned
- **Real actions, real stakes:** the paper stays in sandboxes (Wikipedia API, simulated shop); live web or tool access raises safety risks
- **Blueprint for modern agents:** the text reasoning–act–observe loop now underlies web, coding and tool-using assistants

Georgia Tech

Discussion Questions

1. ReAct's text reasoning is written for humans to read. Is it a faithful explanation of the model's behavior, or plausible narration?

2. ReAct alone lost to CoT on HotpotQA. When should an agent trust its own knowledge versus going to look things up?

3. Text reasoning editing lets humans steer an agent mid-task. Who should be allowed to intervene, and how would we audit it?

4. Finetuned 8B ReAct beat prompted 62B models. Should agent behavior be prompted in, or trained in?

From Hand-Written Demos to Retrieved Experience

Demos are memory

ReAct already puts prior experience in the loop: **1–6 trajectories, chosen once by hand** and shown on every query.

Task type is the key

ALFWorld prompts are matched to the task type. That is **retrieval done by a person** instead of an index.

Fallbacks are recovery

CoT-SC hand-offs and human text reasoning edits rescue a stuck agent. AGMem later **stores such fixes as recovery memory**.

The paper names the limit: complex tasks need more demos than fit in context. Memory banks retrieve experience instead. What should be stored, and in what form?

Georgia Tech

What Persisted and What Changed Since ReAct

Persisted

Reason → act → observe
Still the core loop

Frozen model
Experience enters through the prompt

Getting stuck
Still the main weak spot

Changed

	ReAct (2023)		GUI agents (2026)
Observations	Text snippets	→	**Screenshots**
Actions	search / lookup	→	**Clicks, shortcuts**
Experience	Hand-written demos	→	**Retrieved memory**
Failures	Hallucination, loops	→	**Perception, grounding**

THE OPEN QUESTION

Which failures does visual memory fix and which does it create?

Georgia Tech

Naive Visual Memory is Not Enough: A Failure-Mode Study of GUI Agents

Seoyoung Choi, Minseok Ko, Hyunseek Lee, Kunwoong Kim, Woomin Song, Chanseok Jeon, Jinwoo Shin
ICML 2026 Workshop on Failure Modes in Agentic AI

Presented by Arvind Jayaraman, Efstratios Reppas, Aryan Roy

Georgia Tech

Problem Statement

GUI agents still fail in non-trivial ways

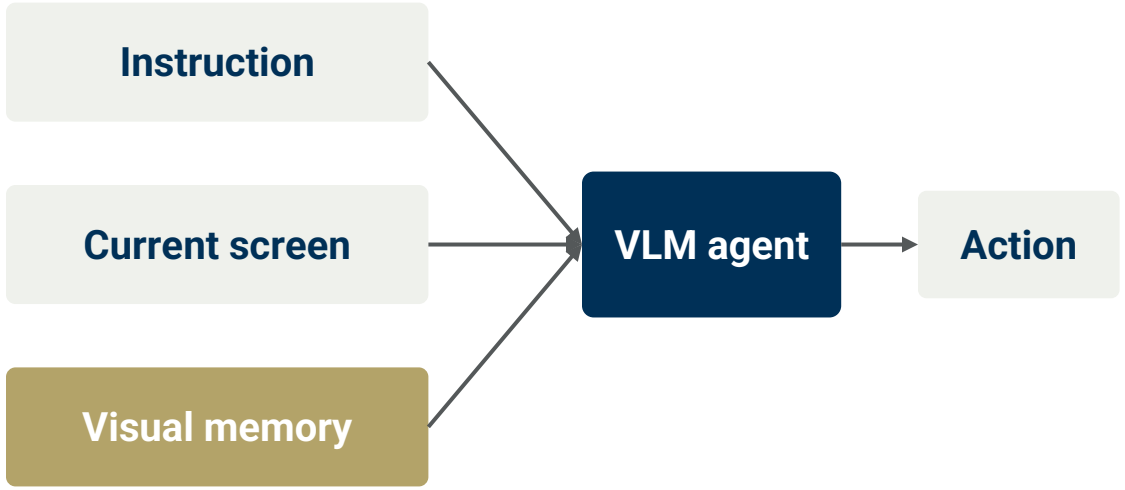
Even frontier VLMs break on real desktop and web tasks.

Memory is the go-to fix

Agents retrieve past trajectories; recent work stores full screenshots.

Its effect is poorly understood

Which failures does visual memory fix, and which does it worsen?



Memory: screenshots from past successful trajectories

CORE QUESTION

What are the core failure modes of GUI agents, and how does visual memory change them?

Georgia Tech

Contributions

1

Failure taxonomy

Four GUI failure modes, plus an LLM-as-Judge protocol to label them on online and offline benchmarks.

2

Visual memory analysis

First study of how memory shifts failures: full screenshots cut state-level errors but amplify action-level ones.

3

AGMem

Action-grounded visual memory: store cropped, action-relevant regions, built automatically from trajectories.

+8.9

pts OSWorld task success with AGMem

18.3% vanilla → 20.4% full-image memory → 27.2% AGMem (GPT-5.4-mini)

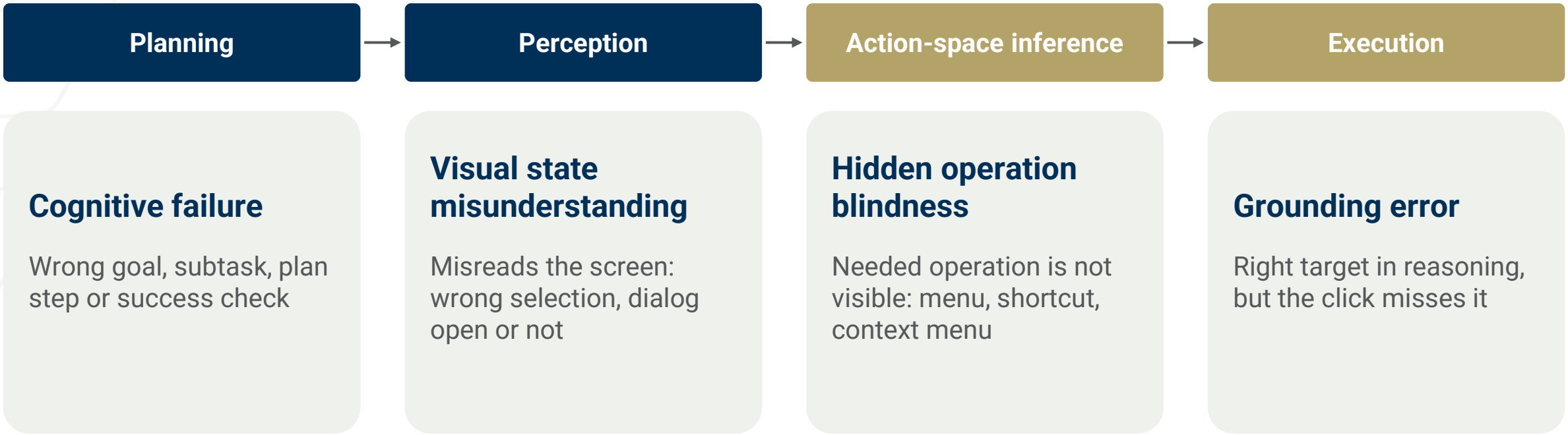
Georgia Tech

Related Work

Category	Methods	What they do	Gap
Text agentic memory	*Synapse, AWM, MemP, A-Mem, ExpeL, ReasoningBank*	Reuse past trajectories, workflows or lessons as text	Miss visual cues: icons, layouts, selections
Visual memory for GUI agents	*Agent S, CoMEM, WebRAgent, Wu et al. (2025a)*	Retrieve full screenshots, trajectories or embeddings	Whole screens; no focus on the acted-on region
GUI agent models	*CogAgent, OS-Atlas, Aguvis, OpenCUA*	Train stronger models on larger GUI datasets	No reuse of experience at test time
Visual focusing	*Set-of-Mark, V*, CropVLM, GUIPruner, FocusUI*	Crop, mark or prune the current observation	Applied to the current screen, not to memory

Georgia Tech

Four Failure Modes Along the Agent Pipeline



Multi-label: one failed task can carry several modes.

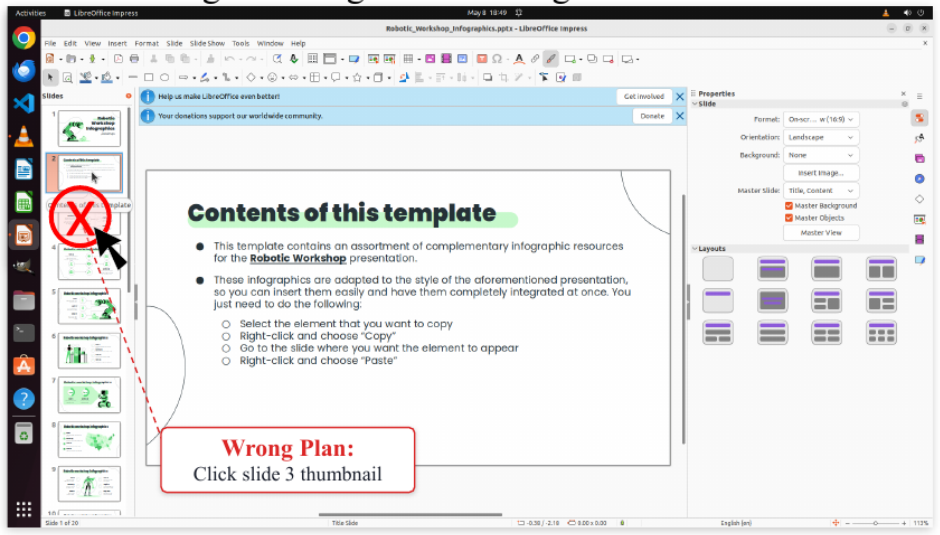
Georgia Tech

Examples: State-Level Failures

Planning **Cognitive failure**

Already on slide 2, yet plans to open slide 3's settings.

Task: Set the given image as the background of the second slide.

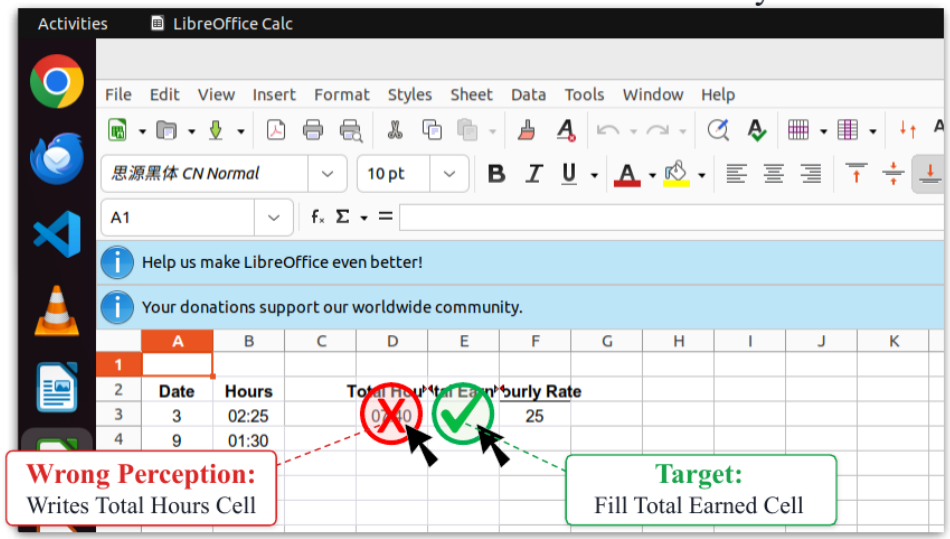


Slide 2 is currently selected, so I'll click slide 3's thumbnail to access the next page's background settings.

Perception **Visual state misunderstanding**

Misreads the columns; writes into Total Hours instead of Total Earned.

Task: Fill total earned = total hours × hourly rate.



*Total hour is in column C and Total Earned in column D. I'll fill D3 with C3*24*F3*

Georgia Tech

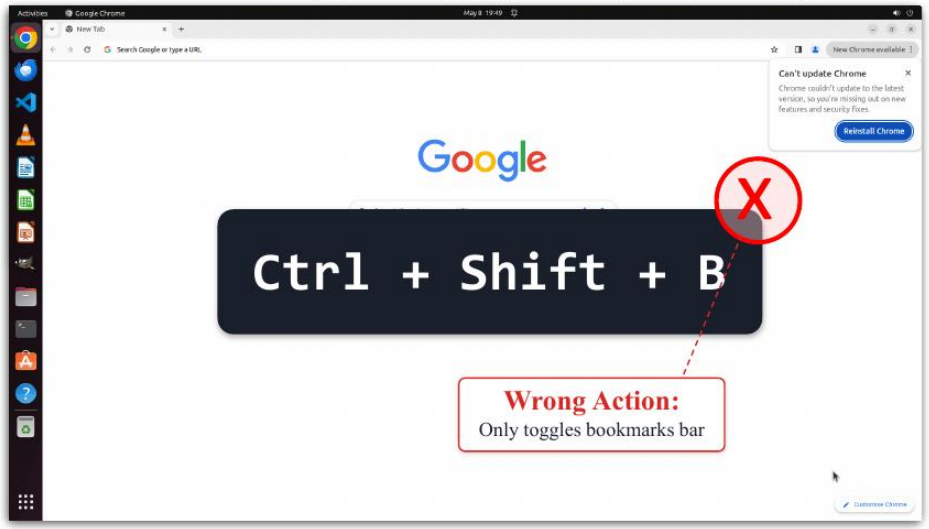
Examples: Action-Level Failures

Action space

Hidden operation blindness

'Add folder' isn't on screen; Ctrl+Shift+B only toggles the bar.

Task: Create a new folder on the bookmarks bar.



I don't see an 'Add folder' option on the screen. I'll use Ctrl+Shift+B to create the folder directly.

Execution

Grounding error

Correct target ('Close Tab'), but the click lands just off it.

Task: Remove the dock on the left side of the screen.



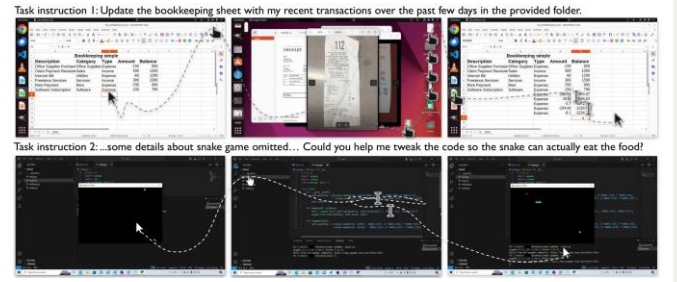
I'll click 'Close Tab' in the dock menu to remove the dock.

Georgia Tech

Three Benchmarks

ONLINE · DESKTOP

OSWorld



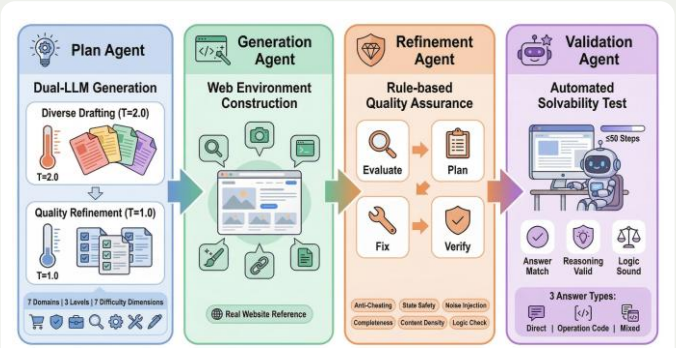
Real apps in Ubuntu VMs: office, browser, GIMP, VLC, VS Code

Scored by scripts that check the final state

Used here: 316 tasks, 1920×1080

ONLINE · WEB

WebForge



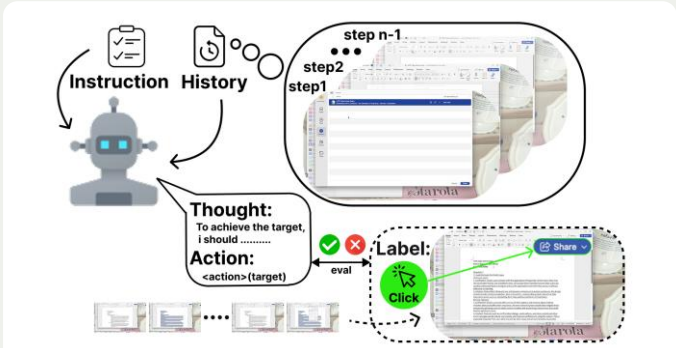
Auto-built websites: LLM agents plan, generate, refine, validate

934 tasks, 7 domains, 3 difficulty levels; answer checked

Used here: random 50 tasks

OFFLINE · DESKTOP

AgentNetBench



Predict the next action from instruction, history, screenshot

Scored against human ground-truth actions, per step

Used here: step accuracy

Images: OSWorld project page; WebForge repo (construction pipeline, no task screenshots published); OpenCUA repo.

Georgia Tech

Experimental Setup

Agent

Model: GPT-5.4-mini

Input: screenshots only, no accessibility tree or HTML

Each step sees: instruction, last 3 screenshot–action pairs, current screen, retrieved memory

Budget: up to 50 steps per episode

Failure labeling

Judge: Codex-based LLM-as-Judge

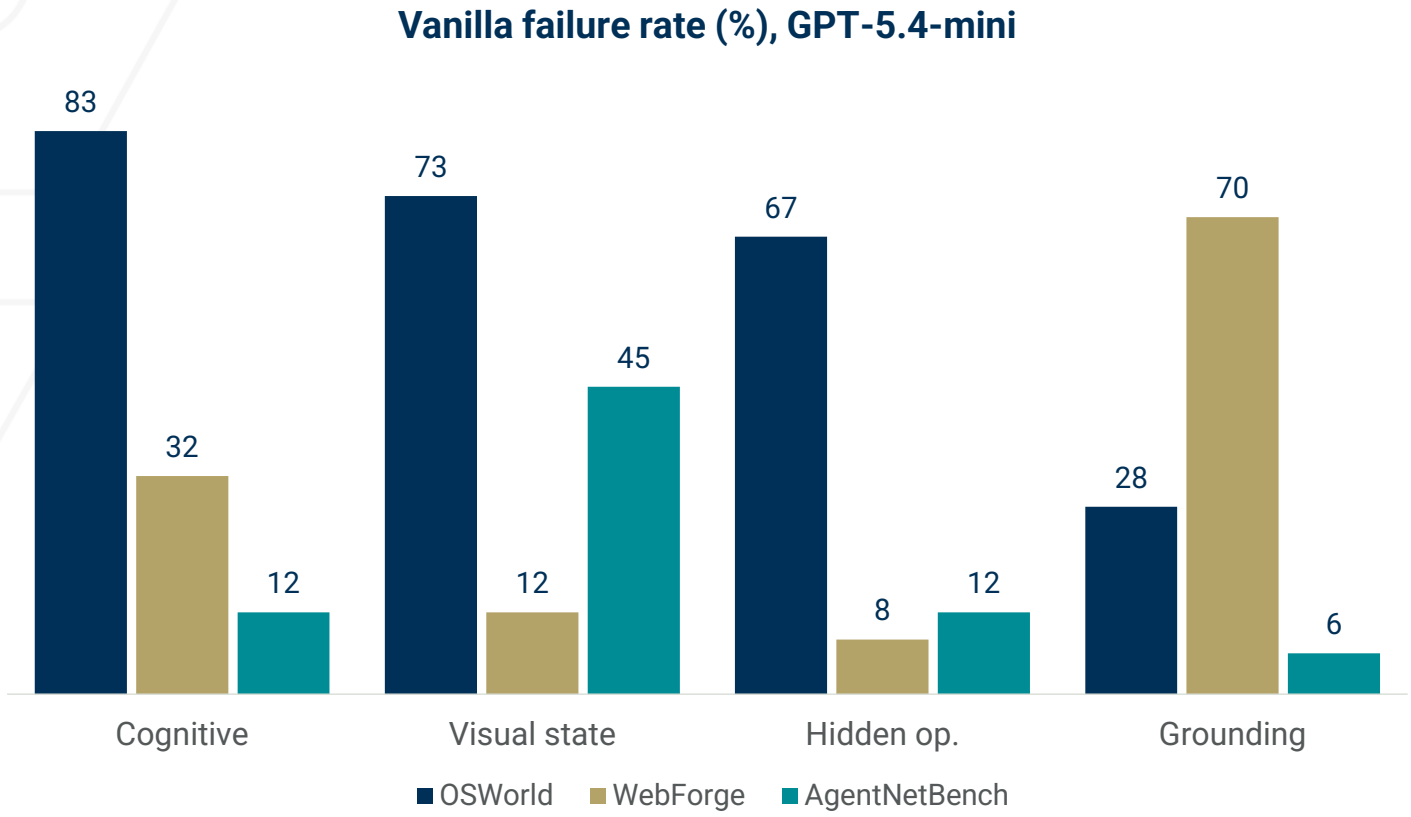
Evidence rule: a label only if it is visible in the trace

Multi-label: rates need not sum to 100%

Unit: task-level on OSWorld and WebForge, action-level on AgentNetBench

Georgia Tech

Each Benchmark Has Its Own Dominant Failure



Multi-label rates. Values not stated in the text are read off the paper's Fig. 2.

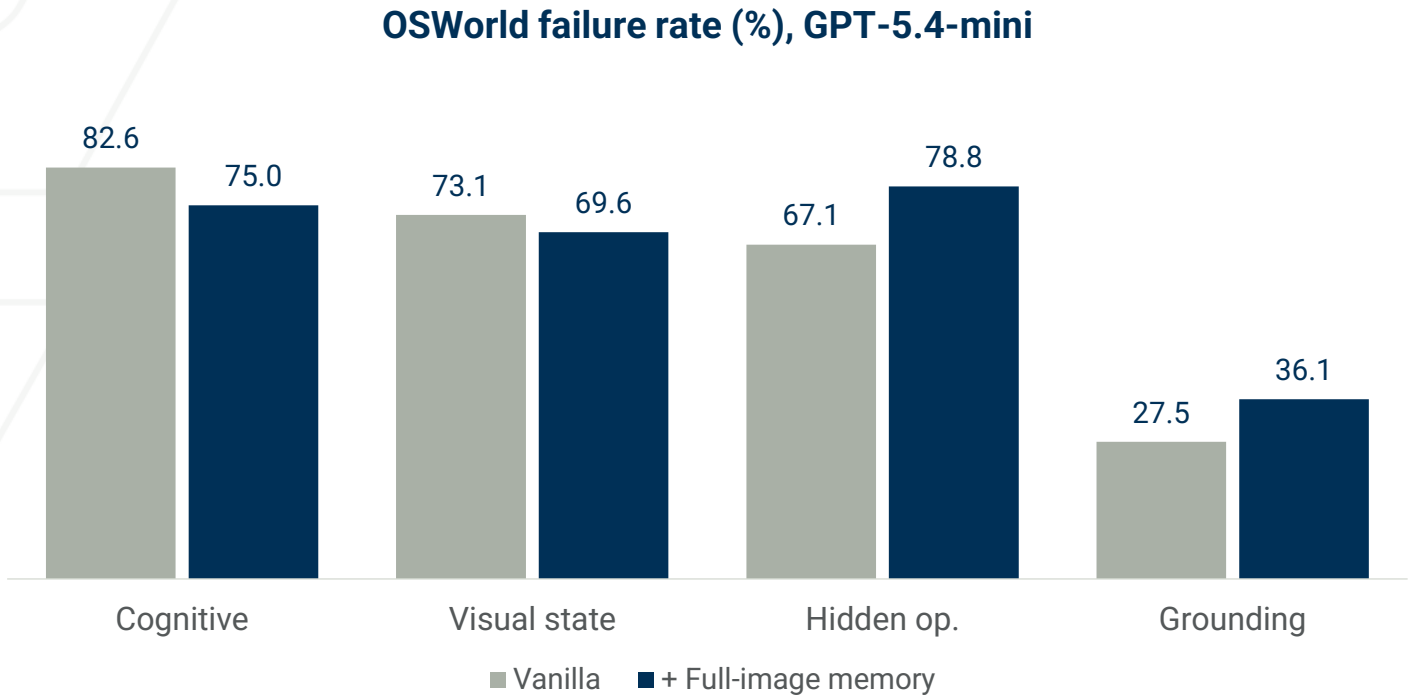
OSWorld
Cognitive 82.6%, hidden op. 67.1%
Long-horizon, menu-heavy desktop apps

WebForge
Grounding error 70.0%
Dense pages, many small click targets

AgentNetBench
Visual state misread 45.1%
Per-step scoring isolates state recognition

Georgia Tech

Full-Image Memory Has a Divergent Effect



−7.6

pts cognitive failure (82.6 → 75.0); visual state −3.5

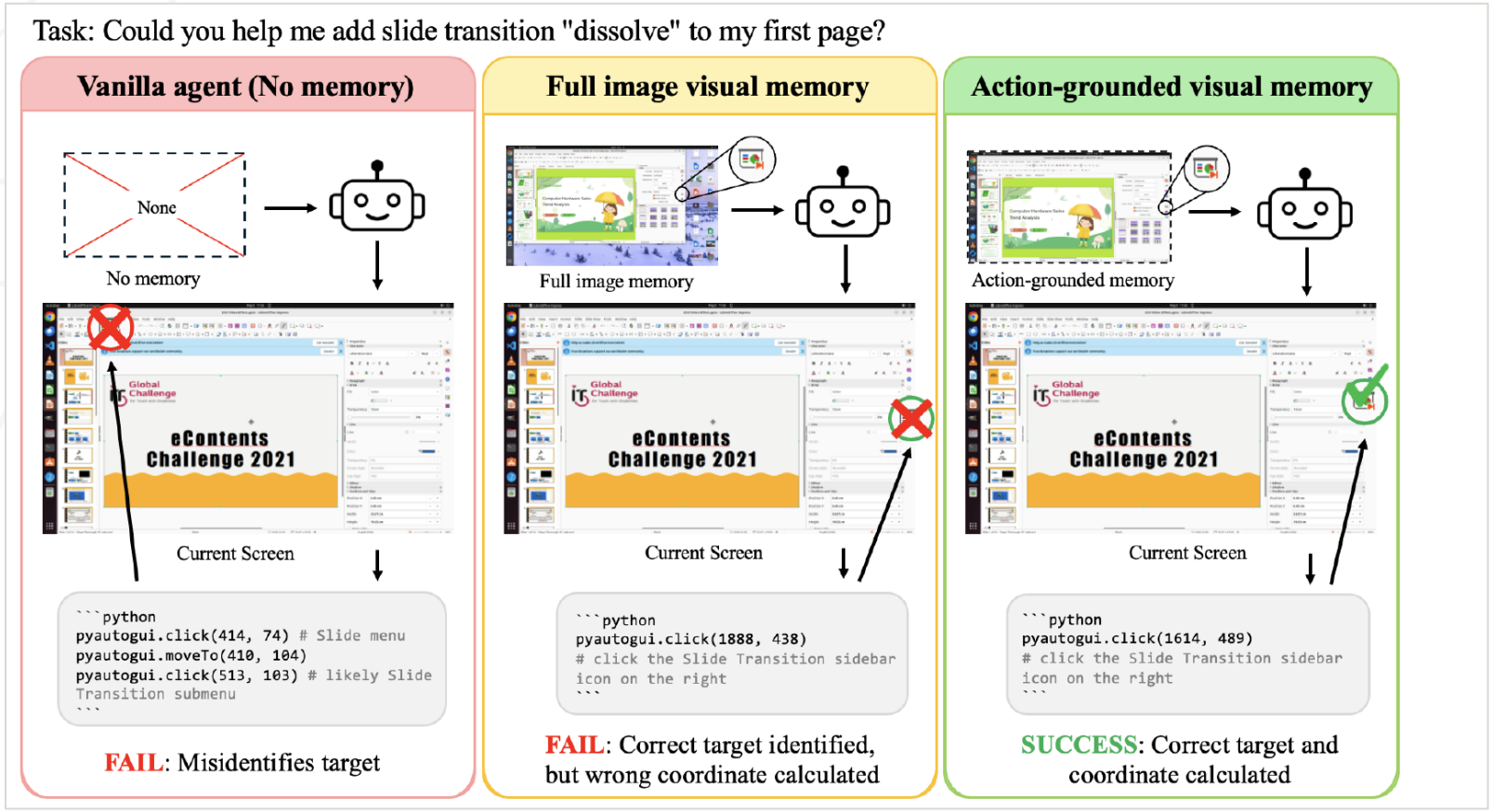
+11.7

pts hidden operation blindness (67.1 → 78.8); grounding +8.6

Which pixels are stored matters as much as whether memory is used. Memory should be smaller and more selective.

Georgia Tech

Key Idea: Ground the Memory in the Action



No memory
Guesses the wrong menu

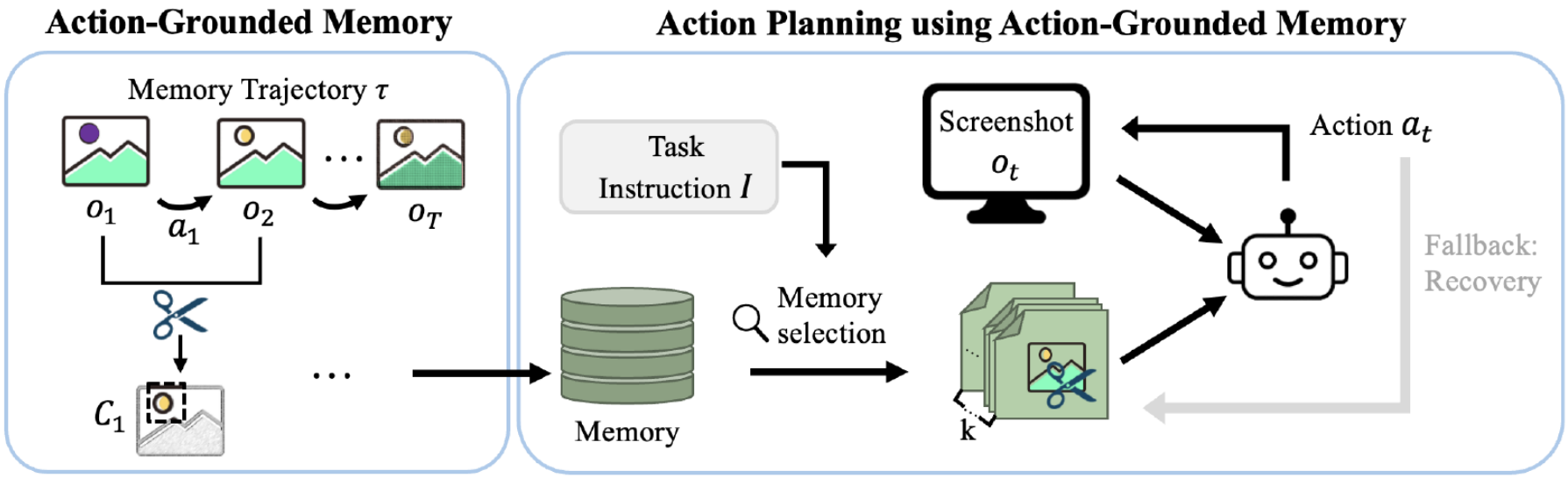
Full screenshot
Right target, wrong coordinates

Action-grounded crop
Right target, right click

Full-image screenshots can fail because it adds context that diverts from the task + inefficient

Georgia Tech

AGMem: Action-Grounded Visual Memory



1 Subtask plan

Split the instruction into subtasks; agent reports its current one each step

2 Action-relevant views

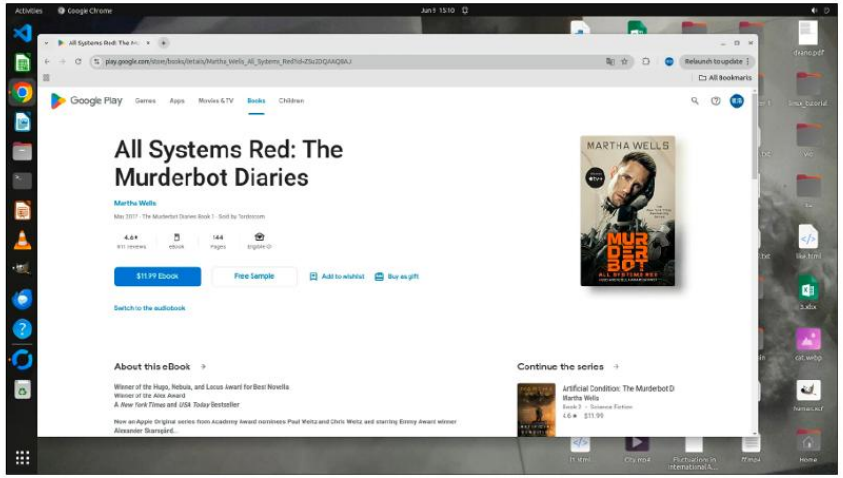
Each memory step stores a tight crop, not the full screen

3 Retrieval + recovery

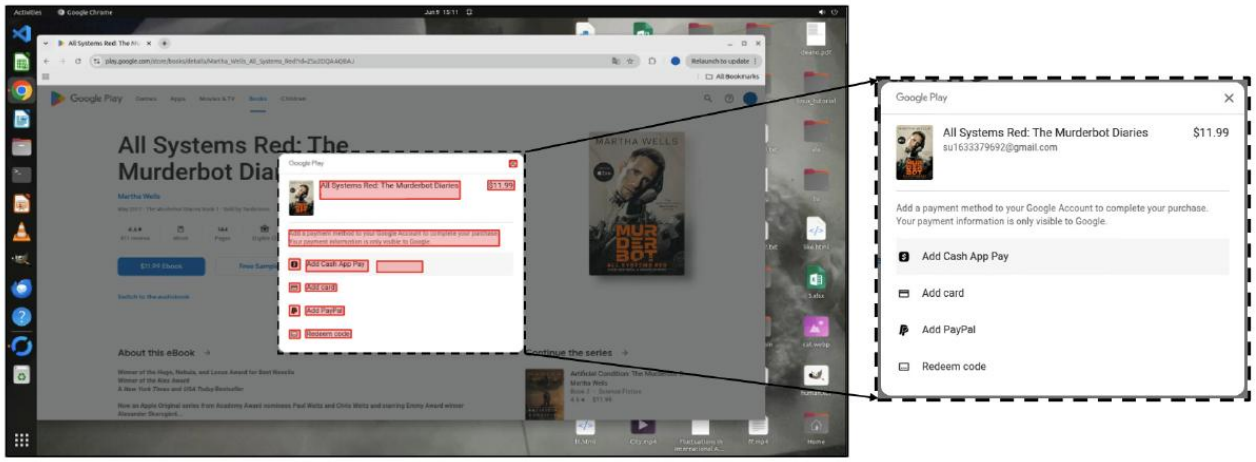
Two-stage retrieval; switch to recovery memory on errors

Georgia Tech

Building Action-Relevant Views



(a) Screenshot o_t at time t.



(b) o_{t+1} after a_t, highlighted by the action-relevant crop C_t.

1 Pixel difference

between screens before and after the action

→

2 Clean the mask

morphological opening and closing

→

3 Snap to UI

smallest window, panel or dialog containing the change

→

4 Fallbacks

no container: keep the box; no change: full screen

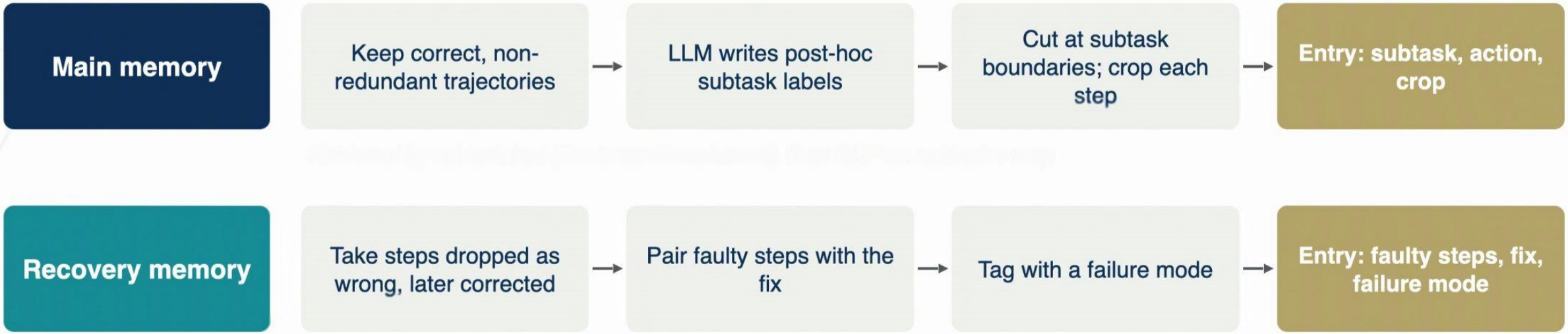
Memory step = (subtask, action, crop). Query at test time: current subtask + previous step's crop.

Georgia Tech

Building the Memory Bank from AgentNet

AgentNet **(OpenCUA, Wang et al. 2026)**

22.6K human demonstrations · Windows, macOS, Ubuntu · 200+ apps and sites · each step: screenshot + action



Identified by dataset metadata

Successes teach what to do; corrected mistakes teach how to recover.

Georgia Tech

Narrowed Retrieval and Recovery Memory

Two-stage retrieval

Trajectory level

Match task subtasks to stored subtask labels (**Sentence-Transformer**); pick *k* trajectories covering all subtasks

↓

Step level

CLIP text + image (0.5 / 0.5) vs. current subtask and previous crop

↓

Top-5 memory steps → prompt

Recovery-aware verification

Recovery-state detector

LLM checks recent screens, actions and the proposed action

↓

Error flagged

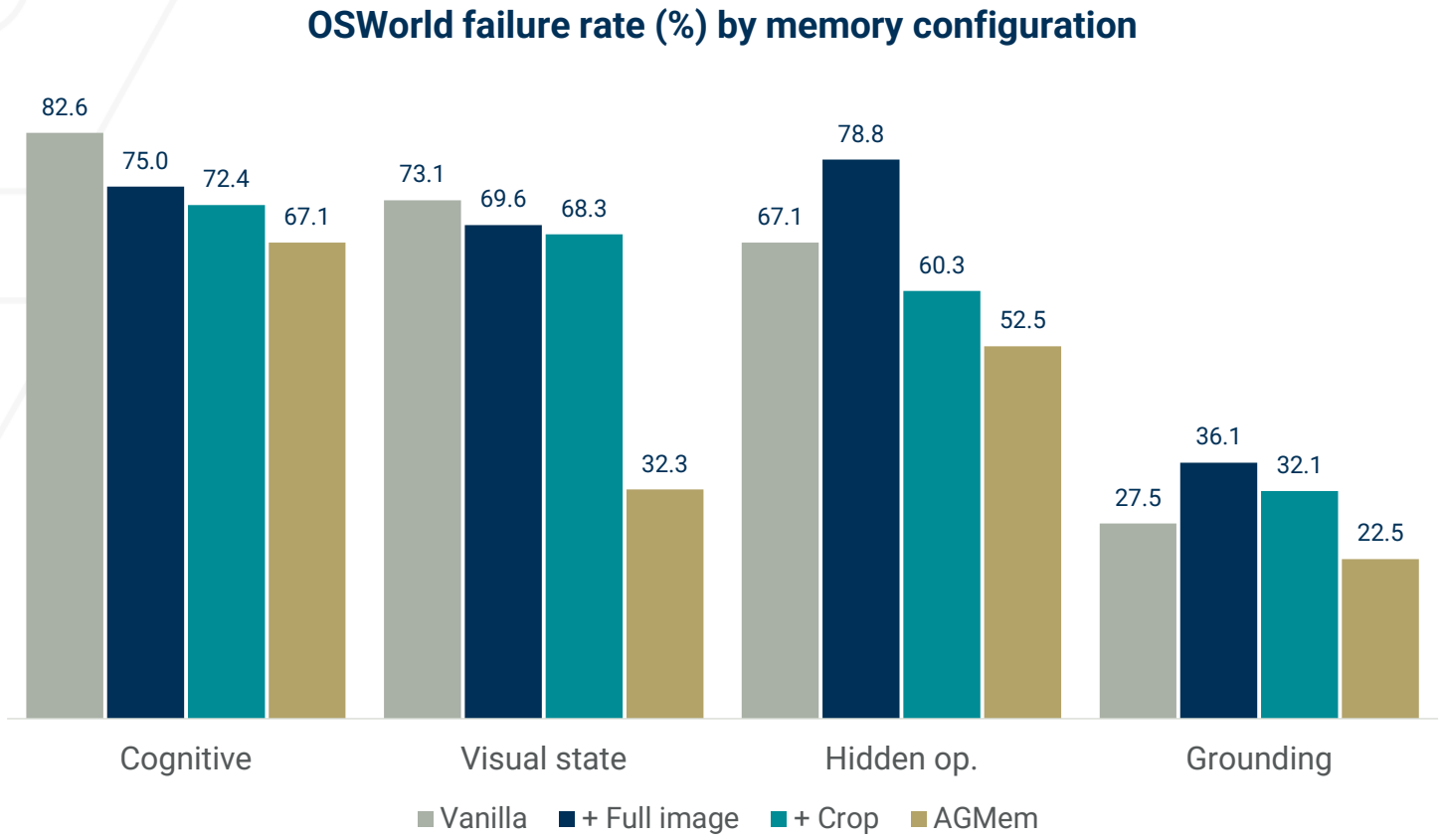
Block the action; retrieve from recovery memory: faulty sub-trajectory + fix, labeled by failure mode

↓

Corrective example → prompt

Georgia Tech

AGMem Reduces Every Failure Mode



−40.8

pts visual state misunderstanding (73.1 → 32.3)

Cropping alone isn't enough

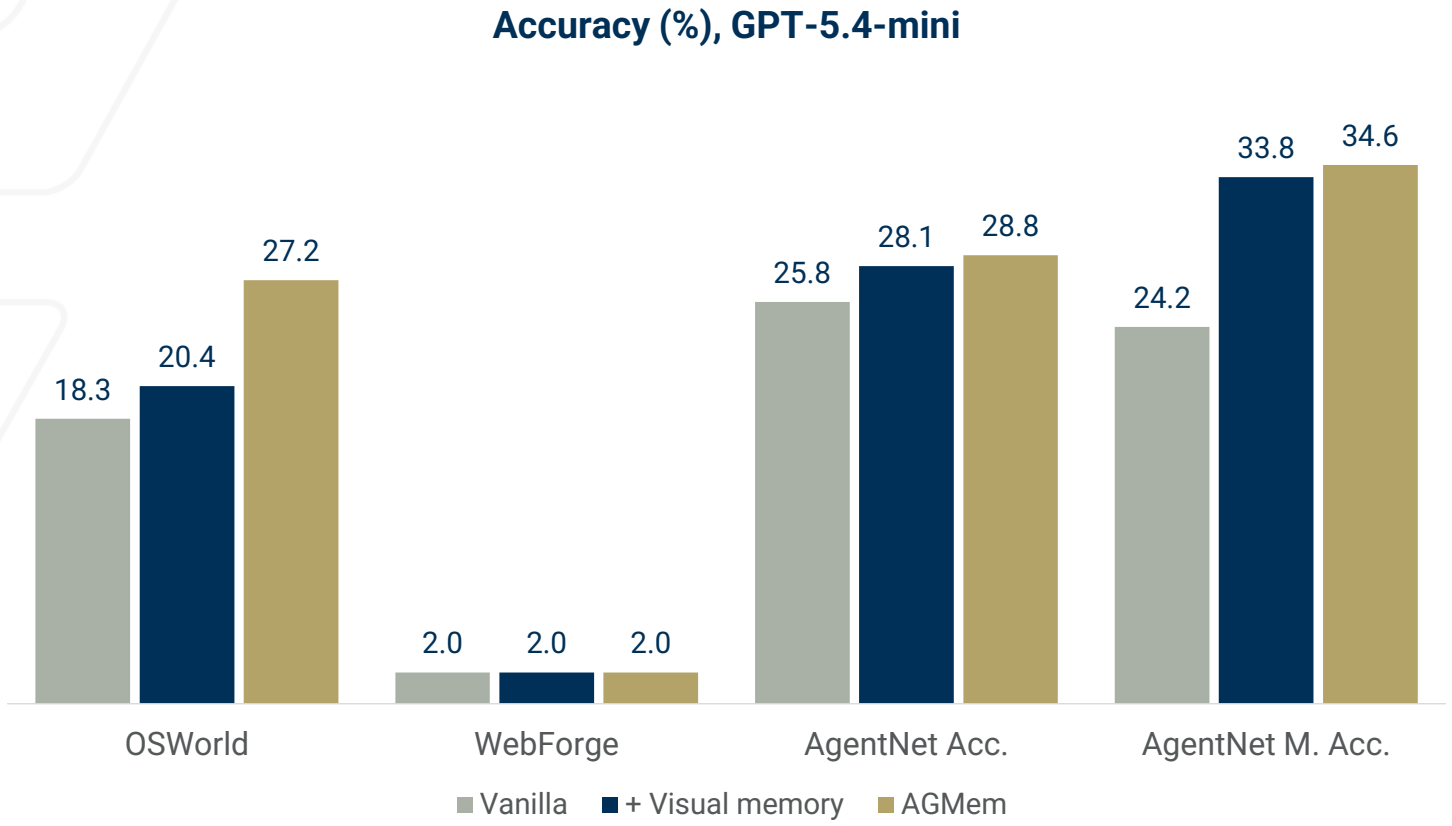
Crop-only still misreads state 68.3% of the time; subtask-aligned retrieval closes the gap.

Success rate climbs with each step

18.3 → 20.4 → 25.8 → 27.2%

Georgia Tech

Main Results Across Benchmarks



OSWorld/WebForge: task success. AgentNetBench: step accuracy and milestone accuracy.

+8.9

pts OSWorld success over vanilla (18.3 → 27.2)

+3.0

pts AgentNetBench step accuracy (25.8 → 28.8)

No gain on WebForge

2.0% for all three methods

Georgia Tech

Case Study: Recovering From a Faulty State

Task: *stream an HLS video in VLC (network-stream dialog + .m3u8 URL)*

Vanilla	Reaches the dialog →	Misclicks Play →	Assumes task is done →	**FAIL**
+ Action-grounded memory	Clearer UI regions →	Better example actions →	Wrong premise remains →	**FAIL**
AGMem (+ recovery)	Detector flags the error →	Recovery memory: Ctrl+N →	Dialog reopened, URL played →	**SUCCESS**

Grounded memory picks the right actions; recovery memory is needed once the agent is already off track.

Georgia Tech

Omissions

1 No efficiency results

Sec. 4.2

Average token use per sample is promised, but never reported

2 Fifth category dropped

App. C.1

The judge also labels Memory Over-following; the conclusion cites it, yet no rate is shown

3 Successes go unlabeled

App. C.1

Errors the agent recovers from, or that don't sink the task, are never counted

4 No component ablations

Table 2

Subtask plan, two-stage retrieval and recovery memory are only tested together

They generally need many more experiments (ablations, benchmarks, metrics etc)

Georgia Tech

Limitations & Implications

Limitations

One backbone: all analysis and gains use GPT-5.4-mini

Uneven gains: WebForge stays at 2.0% for every method

Judge-made labels: LLM-as-Judge taxonomy, no reported human agreement

Inconsistent numbers: OSWorld gain stated as 6.8, 8.9 and 9.1 pts

Implications

Diagnose before designing: failure breakdowns expose trade-offs accuracy hides

Less context can be more: crop and narrow what the agent sees

Remember how to recover: store fixes for mistakes, not only successes

Screen memory is sensitive: filter private content, keep human oversight

Georgia Tech

Discussion Questions

1. Full-image memory helps perception but hurts action. Isn't the problem the memory, or the model's limited visual attention?

2. AGMem crops where pixels change. What about actions whose effect is invisible or far from the click, like shortcuts?

3. Memory allocation is done heuristically and very high-level. Isn't that an algorithmic issue we should try to fix instead?

4. Should agents learn to recover from failures, rather than retrieve recovery examples?

Bitter Lesson...

Georgia Tech