FARPO: FOLDING-AWARE REWARD-DECOUPLED POLICY OPTIMIZATION FOR LONG-HORIZON MOBILE GUI AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Long-horizon mobile GUI agents must decide not only which button to tap but also what to remember: which history to compress, which screen facts to keep, and when. MemGUI-Agent casts these decisions as first-class actions through the Context-as-Action (ConAct) interface, but its supervised policy only imitates demonstrations and never compares alternative memory decisions sampled from itself. We study reinforcement-learning post-training of such policy-managed memory and find that group relative policy optimization (GRPO), applied naively, fails in two ways. First, aggregating heterogeneous verifier rewards (format, action type, parameters, folding range) before group normalization lets the component with the largest within-group spread dictate the update, regardless of the intended weights. Second, span-level history folds, the operation that actually compresses context, are only 22.7% of annotated folds, and the majority, step folds, are almost always already correct, so under natural sampling most prompt groups receive identical rewards and the policy collapses to step-only folding (deep-fold rate 17.7% $\rightarrow$ 0.2%). We introduce **F**olding-**A**ware **R**eward-Decoupled **P**olicy **O**ptimization (**FARPO**), which standardizes each verifier component within its group before weighting and controls span exposure through a span-to-step ratio ρ; both mechanisms are characterized analytically. The resulting MemGUI-8B-RL reaches 48.4% Pass@3 and 39.0% information retention on MemGUI-Bench and 19.7% success on the out-of-distribution MobileWorld, more than doubling the backbone model on every metric, with the gain concentrated on memory-intensive tasks; it is the best open-data 8B model on both benchmarks. A seven-ratio sweep shows folding exposure to be a controllable hyper-parameter with a bimodal footprint, and a stepwise ablation attributes the gain to each mechanism. An anonymized release of the code, data, trained model, and evaluation logs is publicly available at https://memgui-rl-anonymous.github.io/.

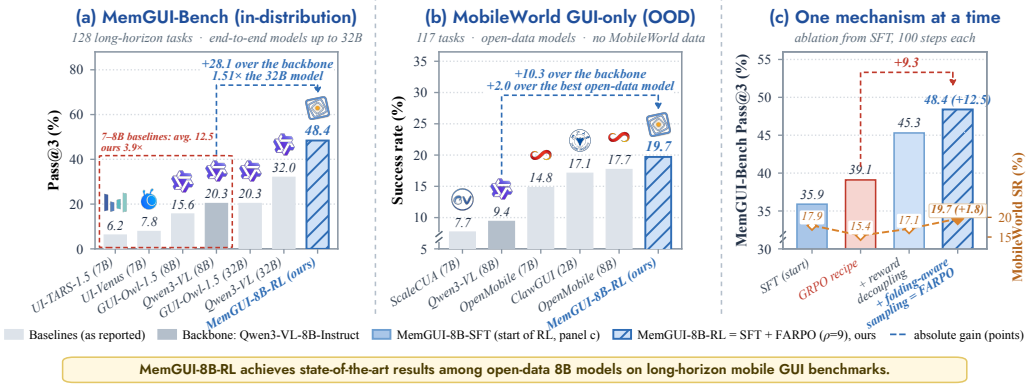


Figure 1: **Main results of MemGUI-8B-RL. (a)** MemGUI-Bench Pass@3 vs. end-to-end models up to 32B and the backbone. **(b)** Out-of-distribution MobileWorld GUI-only vs. open-data models (axis from 5%). **(c)** RL gain over MemGUI-8B-SFT, one mechanism at a time: the GRPO recipe, then reward decoupling, then folding-aware sampling (= FARPO), 100 steps each; bars Pass@3 (axis from 30%), line MobileWorld SR (Table 5).

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

1 INTRODUCTION

A mobile GUI agent that books a flight, compares three apartments, or transcribes reviews across applications has to carry information forward for dozens of steps (Kong et al., 2025; Liu et al., 2026b). Recent end-to-end agents read screens and emit actions well (Qin et al., 2025; Gao et al., 2026; Xu et al., 2026), and agentic frameworks add planning and memory modules around a backbone (Agashe et al., 2025; Wang et al., 2024), but long-horizon, cross-application benchmarks still expose failures of memory rather than perception: facts are lost, progress is mis-tracked, and the growing history dilutes what matters (Liu et al., 2026b). Learned memory management has been studied for documents and long-horizon agents (Yu et al., 2025a; Zhou et al., 2025; Sun et al., 2026); MemGUI-Agent brings it to mobile GUI control as *policy-managed memory*: through the Context-as-Action (ConAct) interface, the same policy that chooses the UI action also emits a *folding command* that compresses an interval of its own history into a summary, and memory operations that store persistent UI facts (Liu et al., 2026a). Supervised fine-tuning on MemGUI-3K yields MemGUI-8B-SFT, an 8B policy that outperforms end-to-end models up to 32B on MemGUI-Bench.

Supervised learning, however, only imitates the demonstrated memory decisions; it never asks whether an alternative fold or memory write sampled from the current policy would have been better. Group relative policy optimization (GRPO) (Shao et al., 2024) offers this comparison without a critic, and rule-based verifiers with several reward components make it cheap on annotated GUI states (Lu et al., 2026a; 2025);[1] two observations show that the direct recipe is inadequate (Figure 2).

> RESEARCH QUESTION Can annotation-based RL improve a policy that manages its own context, and what has to change in GRPO when the verifier returns *heterogeneous* rewards (Observation 1) and the decision that actually compresses context, the span fold, is a *minority* of the data (Observation 2)?

[Observation 1] Heterogeneous verifier signals need an explicit comparison scale. Our verifier scores four components of a ConAct response: format, action type, action parameters, and folding range. The standard recipe normalizes their weighted sum within the group with one shared standard deviation, so a high-dispersion component moves the advantage far more than a low-dispersion one, even under equal weights (Section 3.1). In our groups the format component is saturated (mean ≈ 0.999) while folding is not (0.33 at initialization), so the weights written into the objective are not the weights the update obeys.

[Observation 2] Memory compression needs controlled exposure, or RL collapses it. A *span fold* compresses several past steps and is the operation that actually shrinks the context; a *step fold* records one step, and span folds are only 22.7% of the annotated folds. Step folds are also nearly always correct under the SFT policy, so under natural sampling most prompt groups soon have zero reward variance: the share of informative groups falls from about 70% to 23–38% within one epoch, the surviving signal pushes the policy towards the majority class, and the deep-fold rate on held-out states collapses from 17.7% to 0.2%; normalization alone cannot fix a behavior that the policy almost never samples in the first place.

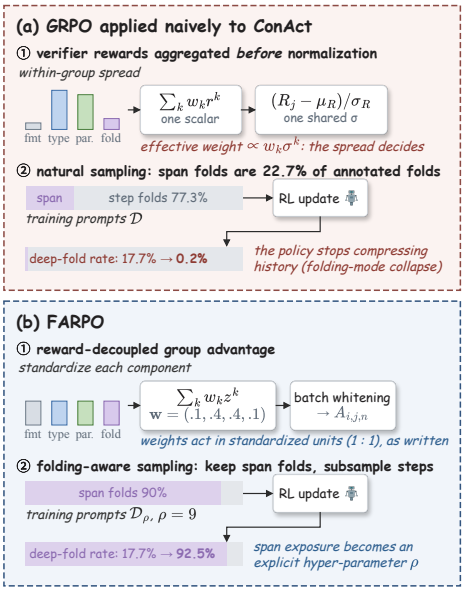


Figure 2: **Why GRPO needs adapting for policy-managed memory. (a)** Naive GRPO: one shared standard deviation lets the highest-spread component dominate the advantage; natural sampling collapses folding to step-only folds. **(b)** FARPO: each component is standardized before weighting; span exposure is a sampling ratio ρ.

We introduce **F**olding-**A**ware **R**eward-Decoupled **P**olicy **O**ptimization (**FARPO**), a GRPO-style post-training recipe for long-horizon mobile GUI agents that manage their own context, which answers the two observations at two levels: within a prompt group it standardizes each verifier component

[1] Appendix A reviews mobile GUI agents, memory management, and multi-reward policy optimization.

separately, as in GDPO (Liu et al., 2026c), and only then applies the preference weights, which makes the advantage group-centered and scale-invariant (Section 3); across prompts it keeps every span fold and subsamples $\lceil N_{\text{span}}/\rho \rceil$ step folds, making span exposure an explicit hyper-parameter ρ.

Starting from MemGUI-8B-SFT, FARPO with ρ=9 yields MemGUI-8B-RL (Figure 1): 48.4% Pass@3 and 39.0% information-retention rate (IRR) on MemGUI-Bench, 2.4× and 2.6× the Qwen3-VL-8B-Instruct backbone, 35% and 29% above the SFT policy, and above Qwen3-VL-235B-Thinking; the gain falls on the memory-intensive tasks (+13.9 Pass@3), and on the out-of-distribution MobileWorld the success rate doubles over the backbone (9.4% → 19.7%). Sweeping seven ratios reveals a bimodal footprint (the policy folds spans almost always or almost never, and ρ decides which), and adding the two mechanisms one at a time shows that each improves on the last: reward decoupling carries most of the in-distribution gain, folding-aware sampling most of the out-of-distribution gain.

Contributions.

- We identify two ways the standard GRPO recipe fails for policy-managed memory: aggregate-then-normalize weights each verifier component by its within-group spread, and natural sampling starves the minority span folds.
- We propose FARPO, which standardizes each component before weighting and turns span exposure into a single ratio ρ, and prove that the estimator is group-centered and rescaling-invariant.
- MemGUI-8B-RL is the best open-data 8B model on MemGUI-Bench and out-of-distribution MobileWorld; a ratio sweep, a stepwise ablation, offline metrics, and case studies attribute the gain to each mechanism.

2 Preliminaries: RL over Context-as-Action Decisions

The ConAct interface. At interaction step t a ConAct policy observes $S_t = (G, H_t, M_t, L_t, I_t)$: the goal G, the *Folded Action History* H_t, the *Folded UI State* M_t, the *Recent Step Record* L_t, and the current screenshot I_t (Liu et al., 2026a). It emits one structured response

$$y_t = (\tau_t, \phi_t, a_t, o_t, \iota_t), \tag{1}$$

with reasoning τ_t, a folding command $\phi_t = ([s_t, e_t], z_t)$ that replaces the history interval $[s_t, e_t]$ by the summary z_t, a UI or memory action a_t (`click`, `type`, `swipe`, …, `memory_add/update/delete`), a UI observation o_t, and the next intent ι_t. A *step fold* has $e_t = s_t$ and records one step; a *span fold* has $e_t > s_t$ and compresses several steps into one entry (Figure 3); the boxed response of the trained policy on a MemGUI-Bench task shows both. Over a long horizon the raw history grows with t and span folds are what keep S_t bounded, so the folding decision is the object of post-training here.

Annotation-based group sampling. We post-train on the annotated states of MemGUI-3K. Each example (x_i, y_i^*) pairs a prompt built from a recorded state with its reference response; a rollout policy π_{old} samples a group of G=5 responses $\{y_{i,j}\}_{j=1}^{G}$ per prompt, and a rule-based verifier scores each against y_i^*. The reward measures agreement with an annotated decision rather than task success, so training needs no device interaction; task success is measured downstream on the benchmarks.

A ConAct response of MemGUI-8B-RL *MemGUI-Bench task 091 · step 23 · first attempt · abridged*

thinking … GoPro's Wikipedia page shows the founding year (2002) and founder (Nick Woodman); steps 13–22 are a complete sub-task, so deep consolidation is appropriate; store the facts, then move on to the Insta360 Ace Pro.

folding
```
{"range": [13, 22], "summary": "[Steps 13-22] Searched for the GoPro Wikipedia
entry, opened the first result, and found: founded in 2002 by Nick Woodman."}
```

tool_call
```
{"name": "mobile_use", "arguments": {"action": "memory_add", "memory_id":
"gopro_company", "content": "Founded: 2002, Founder: Nick Woodman"}}
```

ui_observation GoPro Wikipedia page: "an American technology company founded in 2002 by Nick Woodman" …

Effect: steps 13–22 collapse into one history entry; the facts persist in memory and reach the final message (step 80).

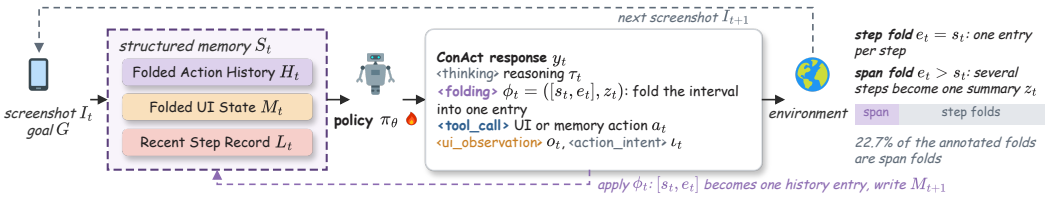


Figure 3: **The ConAct loop.** The policy reads the goal, the screenshot and its memory S_t and emits a folding command with the UI or memory action; both rewrite S_t and the environment returns I_{t+1}.

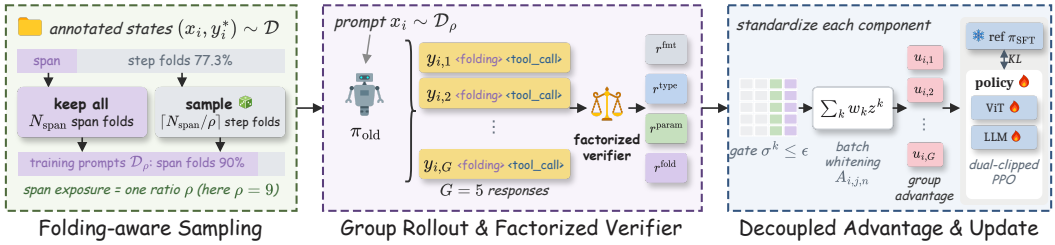


Figure 4: **Overview of the FARPO pipeline.** Folding-aware sampling keeps every span fold and subsamples step folds, so span exposure is one ratio ρ; per prompt, the rollout policy samples G ConAct responses that the factorized verifier scores on four components; FARPO standardizes each component within the group before weighting and updates the policy with dual-clipped PPO.

Factorized verifier. The verifier returns a vector $\mathbf{r}_{i,j} \in [0,1]^4$ of four components, in the order of Table 1: r^{fmt} is the fraction of schema checks passed, r^{type} checks the operation family, and, conditional on a correct type, r^{param} scores the arguments with action-specific rules (Appendix B.5). For an annotated fold with predicted interval p and reference g of lengths ℓ_p, ℓ_g,

$$r^{\text{fold}}_{i,j} = \text{IoU}(p,g) \cdot \frac{\min(\ell_p, \ell_g)}{\max(\ell_p, \ell_g)}, \qquad (2)$$

so an exact interval scores 1, a missing or invalid one 0, and partial overlap is discounted by the length mismatch; the summary text is not scored (its content is judged downstream by IRR), and states without an annotated fold receive a constant r^{fold}=1 that group centering removes; Appendix B.5 lists the rules of every component.

Table 1: **Verifier components** and their mean reward on the SFT policy's first rollout batches: format is saturated, folding is not and depends on the training corpus (natural vs. ρ=9).

Component	What is scored	Init. mean	
		natural	ρ=9
r^{fmt}	schema checks passed	0.999	1.000
r^{type}	operation family	0.86	0.83
r^{param}	arguments, given the type	0.77	0.74
r^{fold}	fold interval, Equation (2)	0.69	0.32

3 FARPO

FARPO combines *reward decoupling* (how candidates are compared *within* a prompt group; Sections 3.1 and 3.2) with *folding-aware sampling* (which decisions the policy sees *across* prompts; Section 3.3); Figure 4 and Algorithm 1 summarize the pipeline.

3.1 Why Aggregation Order Matters

We call the usual multi-reward recipe *scalar GRPO*: form the weighted sum $R_{i,j} = \mathbf{v}^\top \mathbf{r}_{i,j}$ and normalize it within the group. Let $\boldsymbol{\mu}_i$ be the group mean of the component vectors, $\boldsymbol{\delta}_{i,j} = \mathbf{r}_{i,j} - \boldsymbol{\mu}_i$, $\boldsymbol{\Sigma}_i = G^{-1} \sum_j \boldsymbol{\delta}_{i,j} \boldsymbol{\delta}_{i,j}^\top$ the empirical covariance, and $\mathbf{D}_i = \text{diag}(\sigma_i^1, \ldots, \sigma_i^K)$ the marginal standard deviations (K=4). Scalar GRPO assigns

$$A^{\text{GRPO}}_{i,j} = \frac{\mathbf{v}^\top \boldsymbol{\delta}_{i,j}}{\sqrt{\mathbf{v}^\top \boldsymbol{\Sigma}_i \mathbf{v} + \epsilon}}. \qquad (3)$$

Table 2: **Aggregation order on a constructed group** (G=5, w_1=w_2=$\frac{1}{2}$, σ^1:σ^2 = 10:1): responses 2 and 5 differ only in the low-spread component; scalar GRPO separates them by two standard deviations, FARPO ties them at zero.

j	r_j^1	r_j^2	R_j	A_j^{GRPO}	u_j^{FARPO}
1	0	0.50	0.250	−1.14	−0.56
2	0	0.55	0.275	−1.03	0
3	0.5	0.45	0.475	−0.11	−0.56
4	1	0.55	0.775	+1.26	+1.12
5	1	0.45	0.725	+1.03	0
σ^k	0.447	0.045	0.219		

Table 3: **What each estimator does with a prompt group** (Equations (3) and (4); Propositions 1 and 2 in Appendix B); $\mathbf{C}_i$ is the within-group correlation matrix.

	Scalar GRPO (aggregate, then normalize)	**FARPO** (normalize, then aggregate)
Weight of standardized component z^k	$v^k\sigma_i^k/(\sqrt{\mathbf{v}^\top\mathbf{\Sigma}_i\mathbf{v}}+\epsilon)$: follows the within-group spread	w_k for every active component: the weight as written
Constant component ($\sigma_i^k \le \epsilon$)	stays inside the shared denominator	gated out (p_i^k=0)
Rescaling $r^k \to c_k r^k + b_k$	changes the ordering	ordering and values unchanged (ϵ=0)
Group centering / variance	exact / 1	exact / $\mathbf{w}^\top\mathbf{C}_i\mathbf{w}$ (correlations kept)
Prompt prior over span folds	natural share (22.7%)	$\rho/(\rho{+}1)$, chosen (Section 3.3)

Writing $\delta_{i,j}^k = \sigma_i^k z_{i,j}^k$ in standardized coordinates, the coefficient of $z_{i,j}^k$ is proportional to $v^k\sigma_i^k$ (Proposition 1, Appendix B.2). With the shared denominator, two components with equal weights but a 10:1 ratio of within-group spread contribute 10:1 per standardized unit; Table 2 makes this concrete. In ConAct groups the format component is almost always saturated while the folding component varies widely, exactly the structure of Table 2.

3.2 Reward-Decoupled Group Advantages

FARPO standardizes each component before aggregation (the reward-decoupled normalization of GDPO (Liu et al., 2026c)). With a diagonal operator $\mathbf{P}_i^\epsilon = \mathrm{diag}(p_i^1, \ldots, p_i^K)$,

$$p_i^k = \begin{cases} (\sigma_i^k + \epsilon)^{-1}, & \sigma_i^k > \epsilon, \\ 0, & \sigma_i^k \le \epsilon, \end{cases} \qquad u_{i,j} = \mathbf{w}^\top \mathbf{P}_i^\epsilon \boldsymbol{\delta}_{i,j} = \sum_k w_k\, p_i^k \big(r_{i,j}^k - \mu_i^k\big), \tag{4}$$

where ϵ=10^{-6}, the zero branch gates out constant components, and $\mathbf{w} = (0.1, 0.4, 0.4, 0.1)^\top$ weights format, type, parameters, and folding *in standardized units*. The response advantage $u_{i,j}$ is then broadcast to the response tokens and whitened across the valid tokens of the batch,

$$A_{i,j,n} = m_{i,j,n}\, \frac{u_{i,j} - \bar{u}_m}{\sqrt{s_m^2 + \epsilon_b}}, \qquad \epsilon_b = 10^{-8}, \tag{5}$$

with token mask $m_{i,j,n}$, token-weighted batch mean $\bar{u}_m$, and Bessel-corrected variance s_m^2.

Proposition 2 (Appendix B.3) gives the properties summarized in Table 3: $u_{i,j}$ is exactly group-centered for any $\epsilon \ge 0$, invariant to per-component affine rescaling for $\epsilon = 0$, and has within-group variance $\mathbf{w}^\top\mathbf{C}_i\mathbf{w}$ (marginal scale removed, cross-component correlation kept); batch whitening preserves the response ordering.

Policy objective. The actor minimizes a critic-free, dual-clipped PPO surrogate on $A_{i,j,n}$ (Schulman et al., 2017; Ye et al., 2020; Yu et al., 2025b) with a KL penalty to the frozen SFT policy π_{ref} (ϵ_{lo}=0.2, ϵ_{hi}=0.3, dual-clip constant c=3, β=0.01); Equation (10) in Appendix C.1 gives the loss.

3.3 Folding-Aware Sampling

Normalization cannot create signal for a decision that the policy rarely faces. Among the quality-filtered annotations of MemGUI-3K, folds split into span folds $\mathcal{D}_{\mathrm{span}}$ ($e > s$) and step folds $\mathcal{D}_{\mathrm{step}}$ ($e = s$) with $N_{\mathrm{span}}/(N_{\mathrm{span}} + N_{\mathrm{step}}) = 22.7\%$. Step folds are also easy: the SFT policy already reproduces most of them, so their groups often carry zero advantage. FARPO retains every span example and draws m_ρ step examples uniformly without replacement (fixed seed):

$$\mathcal{D}_\rho = \mathcal{D}_{\mathrm{span}} \cup \mathrm{Sample}(\mathcal{D}_{\mathrm{step}}, m_\rho), \qquad m_\rho = \min\{N_{\mathrm{step}}, \lceil N_{\mathrm{span}}/\rho \rceil\}; \tag{6}$$

Algorithm 1 FARPO: one training iteration

Require: SFT policy π_{SFT}, corpus $\mathcal{D} = \mathcal{D}_{\mathrm{span}} \cup \mathcal{D}_{\mathrm{step}}$, ratio ρ, group size G, weights $\mathbf{w}$, gate ϵ

$\pi_\theta \leftarrow \pi_{\mathrm{SFT}}$; $\pi_{\mathrm{ref}} \leftarrow \pi_{\mathrm{SFT}}$ (frozen)

$\mathcal{D}_\rho \leftarrow \mathcal{D}_{\mathrm{span}} \cup \mathrm{Sample}(\mathcal{D}_{\mathrm{step}}, m_\rho)$, $m_\rho = \min\{N_{\mathrm{step}}, \lceil N_{\mathrm{span}}/\rho \rceil\}$ ▷ *folding-aware sampling, (6)*

for each rollout batch $\{(x_i, y_i^*)\}$ drawn uniformly from $\mathcal{D}_\rho$; $\pi_{\mathrm{old}} \leftarrow \pi_\theta$ **do**

  for each prompt i **do**

    sample $y_{i,1}, \ldots, y_{i,G} \sim \pi_{\mathrm{old}}(\cdot \mid x_i)$; $\mathbf{r}_{i,j} \leftarrow \textsc{Verify}(y_{i,j}, y_i^*) \in [0,1]^4$

    for each component k: $\mu_i^k, \sigma_i^k \leftarrow$ group mean and population standard deviation of $\{r_{i,j}^k\}_{j=1}^G$

    $a_{i,j}^k \leftarrow \mathbb{1}[\sigma_i^k > \epsilon]\,(r_{i,j}^k - \mu_i^k)/(\sigma_i^k + \epsilon)$; $u_{i,j} \leftarrow \sum_k w_k a_{i,j}^k$ ▷ *reward decoupling*

  $A_{i,j,n} \leftarrow$ masked batch whitening of $u_{i,j}$ broadcast to the response tokens, (5)

  update θ with the dual-clipped PPO surrogate and the KL penalty, (10), for one PPO epoch

uniform prompt sampling from $\mathcal{D}_\rho$ then has span prior $\alpha_\rho = N_{\mathrm{span}}/(N_{\mathrm{span}} + m_\rho) \approx \rho/(\rho+1)$. This changes the training prior without importance correction (Appendix B.4). In the natural corpus $\rho_{\mathrm{nat}} \approx 0.29$; we sweep $\rho \in \{\frac{1}{4}, 2, 3, 4, 9, \infty\}$ ($\rho{=}\frac{1}{4}$ reverts to the natural ratio); the final model, MemGUI-8B-RL, uses $\rho{=}9$ (Section 4.4).

4 Experiments

4.1 Setup

Initialization and data. All runs start from MemGUI-8B-SFT, a Qwen3-VL-8B-Instruct (Bai et al., 2025) model supervised on MemGUI-3K (Liu et al., 2026a) (2,956 trajectories, 26 applications; no task overlap with MemGUI-Bench). RL prompts are the quality-filtered annotated states of the training split; 295 held-out trajectories (5,941 states) serve only for validation. Each prompt yields $G{=}5$ responses at temperature 1.0 in batches of 128 prompts; AdamW at 10^{-6}, one PPO epoch per batch, veRL (Sheng et al., 2025) (Appendix C.1). MemGUI-8B-RL is the step-100 checkpoint of $\rho{=}9$ (a second seed reproduces its offline metrics, Table 11); the ratio sweep trains one epoch over each $\mathcal{D}_\rho$ (90–398 steps), whereas the estimator ablation fixes 100 steps and matches corpus size.

Benchmarks. MemGUI-Bench contains 128 long-horizon mobile tasks (36 golden steps on average; 115 memory-intensive, spanning up to four applications) (Liu et al., 2026b); a 40-task subset (MemGUI-Bench-40) is our development set. Following its protocol we allow three attempts and report Pass@k (solved within k attempts) and the information-retention rate IRR (task-critical facts preserved in the final output). MobileWorld GUI-only (Kong et al., 2025) is our out-of-distribution setting: 117 long-horizon tasks (28 steps on average) in a different environment and application set, deterministic checkers, one attempt; we report solved tasks and success rate (SR).

4.2 Main Results

FARPO improves the SFT policy on both benchmarks. MemGUI-8B-RL ($\rho{=}9$) reaches 48.4% Pass@3 (62 of 128 tasks) and 39.0% IRR (Table 4), 2.4× and 2.6× the backbone (20.3%, 15.1%) and 35% and 29% above MemGUI-8B-SFT, with Pass@1 at 30.5%; the GRPO recipe from the same start gains 3.2 points (Table 5). It is the best 7–8B result, above every end-to-end model except the 235B ConAct agent, and between the two Gemini-2.5-Pro frameworks (M3A 47.7%, Agent-S2 49.2%). On MobileWorld (Table 5) it solves 23 of 117 tasks (19.7%), twice the backbone's 11 and the strongest open-data result on the GUI-only split (Table 6; the open-weight models above it train on far larger closed corpora).

Gains concentrate on memory-intensive tasks. By golden-step difficulty (Table 4) the gain over SFT is 29.2 Pass@3 points on Easy and 9.6 on Medium tasks; on Easy the 8B policy matches the zero-shot 235B ConAct agent (68.8%), and on Hard (41+ golden steps) the two ConAct policies are within two tasks (13 vs. 11 of 38) while the full recipe still adds 21 points over the backbone; RL sees this regime least, 88% of the annotated training states lie within the first 40 steps of their trajectory, and its dominant failure, losing the thread of the procedure, is not what the fold verifier scores (Appendix D.3). The 115 memory-intensive tasks carry the whole gain (Pass@3 31.3→45.2%; standard tasks unchanged, Figure 10); by application count (Table 10) the gain is 10.7 / 14.3 / 14.7 points on one-, two- and three-application tasks.

Table 4: **Results on MemGUI-Bench (128 tasks)** by golden-step difficulty (Easy 1–20, Medium 21–40, Hard 41+ golden steps): Pass@k and information-retention rate (IRR, %). Baselines as reported by the MemGUI-Agent and MemGUI-Bench papers (agentic frameworks use Gemini-2.5-Pro). **Best** / second-best among end-to-end models ≤8B. Δ: SFT+FARPO over the Qwen3-VL-8B-Instruct backbone (gains over MemGUI-8B-SFT: Figure 10; by application count: Table 10).

Agent	Memory	Easy (48)			Medium (42)			Hard (38)			Overall (128)		
		P@1	P@3	IRR	P@1	P@3	IRR	P@1	P@3	IRR	P@1	P@3	IRR
Agentic frameworks													
Agent-S2 (Agashe et al., 2025)	memory agent	41.7	64.6	38.7	19.0	42.9	42.5	18.4	36.8	36.9	27.3	49.2	39.5
M3A (Rawles et al., 2025)	memory agent	39.6	47.9	31.7	35.7	50.0	48.7	21.1	44.7	36.4	32.8	47.7	39.3
T3A (Rawles et al., 2025)	memory agent	31.2	45.8	–	16.7	45.2	–	18.4	34.2	–	22.7	42.2	29.6
End-to-end models, 32B and larger													
Qwen3-VL-235B-Thinking (Bai et al., 2025)	action-thought	18.8	47.9	21.8	28.6	47.6	38.8	26.3	44.7	28.2	24.2	46.9	30.0
Qwen3-VL-235B-Instruct (Bai et al., 2025)	action-thought	27.1	39.6	26.5	21.4	40.5	28.8	21.1	28.9	32.2	23.4	36.7	29.2
Qwen3-VL-32B-Instruct (Bai et al., 2025)	action-thought	35.4	47.9	39.7	14.3	23.8	22.4	7.9	21.1	13.8	20.3	32.0	25.0
GUI-Owl-1.5-32B (Xu et al., 2026)	action-thought	22.9	37.5	18.4	7.1	14.3	21.3	0.0	5.3	15.3	10.9	20.3	18.4
MemGUI-Agent-235B (Liu et al., 2026a)	ConAct (zero-shot)	41.7	68.8	40.8	35.7	61.9	52.3	34.2	55.3	46.5	37.5	62.5	46.8
End-to-end models, 7–8B													
UI-Venus-7B (Gu et al., 2025)	action-thought	14.6	20.8	6.9	0.0	0.0	1.2	0.0	0.0	0.0	5.5	7.8	2.6
UI-TARS-1.5-7B (Qin et al., 2025)	multi-turn	8.3	16.7	9.3	0.0	0.0	1.7	0.0	0.0	0.9	3.1	6.2	3.8
GUI-Owl-1.5-8B (Xu et al., 2026)	action-thought	22.9	31.2	15.7	2.4	2.4	16.5	7.9	10.5	14.2	11.7	15.6	15.5
Qwen3-VL-8B-Instruct (Bai et al., 2025)	action-thought	18.8	35.4	20.1	4.8	14.3	16.0	2.6	7.9	9.5	9.4	20.3	15.1
+ FARPO (ρ=9, direct RL)	ConAct	27.1	41.7	29.5	9.5	21.4	30.1	2.6	2.6	19.5	14.1	23.4	26.5
MemGUI-8B-SFT (Liu et al., 2026a)	ConAct	25.0	39.6	20.5	**23.8**	33.3	36.5	**21.1**	**34.2**	**32.6**	23.4	35.9	30.2
+ FARPO (ρ=9) = MemGUI-8B-RL	ConAct	**47.9**	**68.8**	**48.0**	**23.8**	**42.9**	**39.6**	15.8	28.9	29.6	**30.5**	**48.4**	**39.0**
Δ *SFT+FARPO vs. Qwen3-VL-8B-Instruct*		+29.1	+33.4	+27.9	+19.0	+28.6	+23.6	+13.2	+21.0	+20.1	+21.1	+28.1	+23.9

Table 5: **Initialization controls and the stepwise ablation** (RL rows: 100 steps, G=5, $\mathbf{w} = (0.1, 0.4, 0.4, 0.1)$; *scalar* = aggregate then normalize, *decoupled* = Equation (4); *natural* corpora truncated to the ρ=9 corpus size). Each ablation row adds one mechanism to the row above; Δ rows give the increments. **Best** / second-best per column among RL rows from MemGUI-8B-SFT; green superscripts: gain over MemGUI-8B-SFT.

Policy	MemGUI-Bench (128)			MobileWorld	
	P@1	P@3	IRR	Solved	SR
Initialization controls					
Qwen3-VL-8B-Instruct	9.4	20.3	15.1	11	9.4
+ decoupled / ρ=9 (direct RL)	14.1	23.4	26.5	18	15.4
MemGUI-8B-SFT	23.4	35.9	30.2	21	17.9
One mechanism at a time, from MemGUI-8B-SFT (100 steps, size-matched)					
scalar / natural (GRPO recipe)	23.4	39.1	35.7	18	15.4
+ reward decoupling (decoupled / natural)	29.7	45.3	38.5	20	17.1
Δ *vs. GRPO recipe*	+6.3	+6.2	+2.8	+2	+1.7
+ folding-aware sampling (decoupled / ρ=9 = FARPO)	**30.5**$^{+7.1}$	**48.4**$^{+12.5}$	**39.0**$^{+8.8}$	**23**$^{+2}$	**19.7**$^{+1.8}$
Δ *vs. + reward decoupling*	+0.8	+3.1	+0.5	+3	+2.6

Table 6: **MobileWorld GUI-only SR (%):** 117 tasks, one attempt, out-of-distribution; baselines from MemGUI-Agent; AT = action-thought; **best** / second open-data model; Δ as in Table 4.

Agent	Context	SR
Open-weight models		
GUI-Owl-1.5-32B	AT	43.9
MAI-UI-235B-A22B	AT	39.7
GUI-Owl-1.5-8B	AT	38.2
MAI-UI-8B	AT	27.5
Qwen3-VL-235B-Thinking	AT	14.5
Qwen3-VL-8B-Instruct	AT	9.4
MemGUI-Agent-235B	ConAct	29.1
Open-data models		
OpenMobile-8B	AT	17.7
ClawGUI-2B	AT	17.1
OpenMobile-7B	AT	14.8
ScaleCUA-7B	AT	7.7
MemGUI-8B-RL (ρ=9)	ConAct	**19.7**
Δ *SFT+FARPO vs. Qwen3-VL-8B-Instruct*		+10.3

4.3 Estimator and Sampling Ablation

Design. Table 5 adds the two mechanisms one at a time from the same SFT checkpoint with a fixed budget of 100 steps: the GRPO recipe (*scalar* estimator, aggregate then normalize, on the *natural* corpus), then reward decoupling (Equation (4)), then folding-aware sampling (ρ=9), which is FARPO; natural corpora are truncated to the ρ=9 corpus size.

Findings: each mechanism adds to the last. The GRPO recipe improves on SFT in distribution (39.1% Pass@3) but loses three MobileWorld tasks (18 vs. 21 of 117). Reward decoupling (the GDPO recipe) adds +6.2 Pass@3, +6.3 Pass@1 and +2.8 IRR (45.3 / 29.7 / 38.5) and recovers two MobileWorld tasks: the type and parameter rewards regain

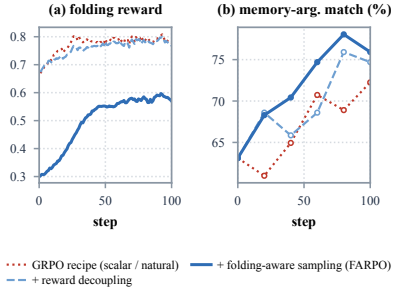


Figure 5: **Ablation dynamics**: **(a)** smoothed folding reward, **(b)** memory-argument match on the held-out states (Figure 11).

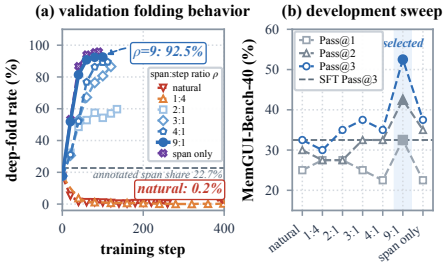


Figure 6: **Folding exposure is bimodal.** **(a)** Validation deep-fold rate during training (natural corpora collapse to step-only folding, $\rho \geq 3$ to span-only, ρ=2 in between); **(b)** MemGUI-Bench-40 Pass@k across the sweep, best at ρ=9.

Table 7: **Span-to-step ratio sweep** (one epoch over $\mathcal{D}_\rho$ from MemGUI-8B-SFT, decoupled estimator). Deep-fold: greedy validation rate at the last step; span at deploy: span share of the folds emitted on MemGUI-Bench-40; MW: MobileWorld tasks solved of 117. **Best** / second of the RL rows; green superscripts: gain over MemGUI-8B-SFT.

Corpus $\mathcal{D}_\rho$	Steps (1 ep.)	Deep-fold	MemGUI-Bench-40 P@1	P@2	P@3	Span at deploy	MW solved
MemGUI-8B-SFT (no RL)	–	17.7	30.0	30.0	32.5	13.3	21
natural (all folds)	264	0.2	25.0	30.0	32.5	2.3	22
$\rho = \frac{1}{4}$ ($\to$ nat.)	398	0.4	27.5	27.5	30.0	1.4	17
$\rho = 2$	135	59.7	27.5	27.5	35.0	77.3	21
$\rho = 3$	120	86.5	25.0	32.5	37.5	91.1	22
$\rho = 4$	112	89.5	22.5	32.5	35.0	93.9	**25**
$\rho = 9$ (ours)	100	92.5	**32.5**$^{+2.5}$	**42.5**$^{+12.5}$	**52.5**$^{+20.0}$	92.5	23^{+2}
$\rho = \infty$ (span only)	90	95.9	22.5	35.0	37.5	97.7	18

the weight that the dispersion-dominated advantage (Section 3.1) took from them, and memory-argument match on held-out states rises stepwise (63.1 $\to$ 72.3 $\to$ 74.7, Figure 5). Folding-aware sampling adds +3.1 Pass@3 in distribution but the larger step out of distribution, +3 MobileWorld tasks: it alone changes the deployment footprint from step to span folds (Table 8), and its folding reward, flat for both natural-corpus runs, keeps rising through the epoch. Direct RL from the backbone reaches only 23.4% Pass@3 (Appendix D.1): RL amplifies a ConAct policy rather than replacing its supervision.

4.4 Folding Exposure as a Hyper-parameter

The trained policy collapses to one folding mode, and ρ decides which. Figure 6a tracks the deep-fold rate on validation states, which is 17.7% for the SFT policy (annotated share: 22.7%). Under natural sampling or ρ=$\frac{1}{4}$ the rate drops to 1% within 100 steps and to 0.2–0.4% by the end of the epoch; under $\rho \geq 3$ it rises above 85% (ρ=9: 92.5%); ρ=2 (59.7%) lies between the two modes.

Performance is non-monotonic in ρ. On the development set (Figure 6b, Table 7), ρ=9 gives the best Pass@1/2/3 and ρ=4 the most MobileWorld tasks (25/117); span-only training hurts MobileWorld (18/117) and the natural corpus does not improve over SFT (32.5 Pass@3). ρ=9 was selected on this subset and is among the two best ratios on MobileWorld. Offline validation metrics improve for every ratio (Figure 7; memory-trigger F1 70.5$\to$74–77%, memory-argument match 63$\to$72–79%; all logged metrics: Appendix D.2).

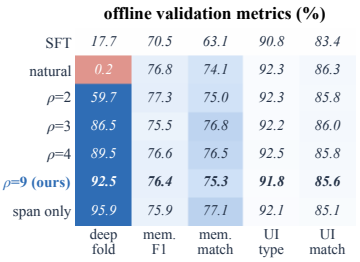


Figure 7: **Offline validation metrics** of the sweep (final checkpoint, greedy, 5,941 held-out states); color = change vs. MemGUI-8B-SFT, blue up, red down; the full set of logged metrics is in Table 11.

4.5 Training Signal, Failure Modes, and a Case Study

Folding-aware sampling keeps the signal alive. With natural corpora the share of zero-advantage groups (five identical rewards, hence no gradient) grows from 25–33% to 62–77% over the epoch, so two thirds of a late batch is wasted; with ρ=9 it stays at 13–15% (Figure 9). At deployment (Table 8) natural-corpus policies emit 2–7% span folds and folding-aware policies 92–98%; MemGUI-8B-RL folds 21 steps per span (SFT: 11) and writes 1.68 memory entries per attempt (SFT: 1.26).

Failure modes shift, and a case shows how. MemGUI-Eval's failure labels (Figure 12, Appendix D) show fewer failed attempts overall (272$\to$225; GRPO recipe: 238) and far fewer output-memory hallucinations and knowledge-deficiency failures (53$\to$34, 38$\to$22); the remaining failures concentrate on process-memory hallucination. The history bars of Figure 8 make the mechanism



Figure 8: **One task, five policies** (MemGUI-Bench task 091, first attempts). Bars show the folded history at the end of each run, replayed from the folding directives (purple: span entries, grey: step entries, white: never folded; ◇ memory writes, red: repeated actions); quotes are copied from the stored responses. Only MemGUI-8B-RL solves the task, with a span fold per sub-task and eight facts kept in memory; the baselines forget, loop, or stop short of the message (more tasks: Appendix E).

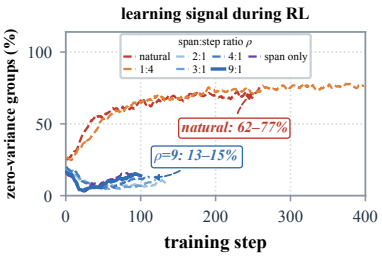


Figure 9: Share of zero-advantage prompt groups during training (9-step moving average): step-heavy corpora saturate within the epoch, span-heavy corpora keep the share below 25% at every step.

Table 8: **Deployment footprint on MemGUI-Bench** (all attempts, stored trajectories): span share of the emitted folds (%), mean span length (steps), and memory writes per attempt. Both ρ=9 policies fold at span level in over 90% of the cases, both natural-corpus policies in under 7%.

Policy	Span share (%)	Span len.	Mem. writes/att.
MemGUI-8B-SFT	13.3	10.9	1.26
scalar, natural (GRPO recipe)	1.7	15.3	1.72
decoupled, natural	6.8	16.9	1.53
decoupled, $\rho = 9$ (ours)	91.7	21.5	1.68
direct RL from Instruct	98.3	36.3	1.91

visible: the baselines re-read, loop, or fold one step at a time and end with dozens of history entries, whereas MemGUI-8B-RL closes each sub-task with a span fold, ends its 80 steps with a two-line history, and answers from memory.

5 Conclusion

For long-horizon mobile GUI agents, policy-managed memory breaks the naive GRPO recipe in two ways: heterogeneous verifier rewards acquire dispersion-dependent weights when aggregated before normalization, and minority span folds let RL collapse to step-only folding. FARPO answers both with reward-decoupled group advantages and folding-aware sampling; MemGUI-8B-RL is the best open-data 8B model on MemGUI-Bench and out-of-distribution MobileWorld, and folding exposure is a bimodal, non-monotonic hyper-parameter. Two directions follow: a verifier that scores what a summary preserves, and online RL on device rollouts with the same estimator and sampler.

Limitations. We train for one epoch at a fixed ratio ρ and evaluate 8B policies on Android; annealing ρ, deeper trajectory states, larger backbones, and desktop or web agents are next steps.

AI USE STATEMENT

Generative AI tools assisted with research-code development, log parsing and data aggregation, implementation-to-manuscript consistency checks, literature search, proof editing, manuscript drafting and revision, diagram construction, and LaTeX formatting. The MemGUI-3K corpus uses model-generated trajectories and model-assisted filtering as documented in the MemGUI-Agent work, and MemGUI-Eval uses LLM judges as specified by the benchmark. All reported numbers were computed from stored training logs and evaluation records by code that ships with the source; none were produced by an AI tool. The authors verified the implementation, mathematical claims, citations, figures, numerical results, and final text, and take full responsibility for them.

ETHICS STATEMENT

This work trains and evaluates mobile GUI agents in emulated Android environments (MemGUI-Bench) and a containerized benchmark (MobileWorld); no human subjects, personal data, or live accounts are involved, and the recipients and accounts that appear in tasks are the benchmarks' test fixtures. The training corpus MemGUI-3K is a public research dataset, and the released code, model, data, and evaluation logs are redacted of credentials and internal identifiers. Agents that operate real devices can take irreversible actions such as sending messages or making purchases; the released policy is a research model evaluated in sandboxes, and any deployment should keep a confirmation step before such actions.

REPRODUCIBILITY STATEMENT

The anonymized release at https://memgui-rl-anonymous.github.io/ contains the FARPO training code with the launch script of every reported run, the MemGUI-8B-RL checkpoint, the RL prompt corpus with its training and held-out split, and the stored MemGUI-Bench and MobileWorld trajectories and judge outputs behind every row of Tables 4 to 8; all reported numbers are computed from these records. Algorithms 1 and 2 give the two estimators line by line; Table 9 and Appendix C.1 list every hyper-parameter, the policy objective, the KL estimator, and the corpus construction; Appendix B.5 lists the verifier rules and Appendix B.6 the response schema; Appendix C.2 states the evaluation protocol. Propositions 1 and 2 are proved in Appendices B.2 and B.3 under the stated assumptions.

REFERENCES

Saaket Agashe, Kyle Wong, Vincent Tu, Jiachen Yang, Ang Li, and Xin Eric Wang. Agent S2: A compositional generalist-specialist framework for computer use agents. *arXiv preprint arXiv:2504.00906*, 2025.

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, et al. Qwen3-VL technical report. *arXiv preprint arXiv:2511.21631*, 2025.

Changlong Gao, Zhangxuan Gu, Yulin Liu, Xinyu Qiu, Shuheng Shen, Yue Wen, Tianyu Xia, Zhenyu Xu, Zhengwen Zeng, Beitong Zhou, et al. UI-Venus-1.5 technical report. *arXiv preprint arXiv:2602.09082*, 2026.

Zhangxuan Gu, Zhengwen Zeng, Zhenyu Xu, Xingran Zhou, Shuheng Shen, Yunfei Liu, Beitong Zhou, Changhua Meng, Tianyu Xia, Weizhi Chen, et al. UI-Venus technical report: Building high-performance UI agents with RFT. *arXiv preprint arXiv:2508.10833*, 2025.

Wenyi Hong, Weihan Wang, Qingsong Lv, Jiazheng Xu, Wenmeng Yu, Junhui Ji, Yan Wang, Zihan Wang, Yuxiao Dong, Ming Ding, et al. CogAgent: A visual language model for GUI agents. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14281–14290, 2024.

Quyu Kong, Xu Zhang, Zhenyu Yang, Nolan Gao, Chen Liu, Panrong Tong, Chenglin Cai, Hanzhang Zhou, Jianan Zhang, Liangyu Chen, et al. MobileWorld: Benchmarking autonomous mobile agents in agent-user interactive and MCP-augmented environments. *arXiv preprint arXiv:2512.19432*, 2025.

Ruoran Li, Xinghua Zhang, Haiyang Yu, Shitong Duan, Xiang Li, Wenxin Xiang, Chonghua Liao, Xudong Guo, Yongbin Li, and Jinli Suo. MemPO: Self-memory policy optimization for long-horizon agents. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 23286–23301. Association for Computational Linguistics, 2026. doi: 10.18653/v1/2026.findings-acl.1166.

Guangyi Liu, Gao Wu, Congxiao Liu, Pengxiang Zhao, Liang Liu, Mading Li, Qi Zhang, Mengyan Wang, Liang Guo, and Yong Liu. MemGUI-Agent: An end-to-end long-horizon mobile GUI agent with proactive context management. *arXiv preprint arXiv:2606.19926*, 2026a.

Guangyi Liu, Pengxiang Zhao, Yaozhen Liang, Qinyi Luo, Shunye Tang, Yuxiang Chai, Weifeng Lin, Han Xiao, WenHao Wang, Siheng Chen, et al. MemGUI-Bench: Benchmarking memory of mobile GUI agents in dynamic environments. In *Proceedings of the ACM International Conference on Multimedia*, 2026b. arXiv:2602.06075.

Shih-Yang Liu, Xin Dong, Ximing Lu, Shizhe Diao, Peter Belcak, Mingjie Liu, Min-Hung Chen, Hongxu Yin, Yu-Chiang Frank Wang, Kwang-Ting Cheng, Yejin Choi, Jan Kautz, and Pavlo Molchanov. GDPO: Group reward-decoupled normalization policy optimization for multi-reward RL optimization. In *International Conference on Machine Learning (ICML)*, 2026c. arXiv:2601.05242.

Zhengxi Lu, Jiabo Ye, Fei Tang, Yongliang Shen, Haiyang Xu, Ziwei Zheng, Weiming Lu, Ming Yan, Fei Huang, Jun Xiao, et al. UI-S1: Advancing GUI automation via semi-online reinforcement learning. *arXiv preprint arXiv:2509.11543*, 2025.

Zhengxi Lu, Yuxiang Chai, Yaxuan Guo, Xi Yin, Liang Liu, Hao Wang, Han Xiao, Shuai Ren, Pengxiang Zhao, Guangyi Liu, et al. UI-R1: Enhancing efficient action prediction of GUI agents by reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2026a. arXiv:2503.21620.

Zhengxi Lu, Fei Tang, Guangyi Liu, Kaitao Song, Xu Tan, Jin Ma, Wenqi Zhang, Weiming Lu, Jun Xiao, Yueting Zhuang, and Yongliang Shen. UI-Copilot: Advancing long-horizon GUI automation via tool-integrated policy optimization. *arXiv preprint arXiv:2604.13822*, 2026b.

Yujia Qin, Yining Ye, Junjie Fang, Haoming Wang, Shihao Liang, Shizuo Tian, Junda Zhang, Jiahao Li, Yunxin Li, Shijue Huang, et al. UI-TARS: Pioneering automated GUI interaction with native agents. *arXiv preprint arXiv:2501.12326*, 2025.

Chris Rawles, Sarah Clinckemaillie, Yifan Chang, Jonathan Waltz, Gabrielle Lau, Marybeth Fair, Alice Li, William Bishop, Wei Li, Folawiyo Campbell-Ajala, et al. AndroidWorld: A dynamic benchmarking environment for autonomous agents. In *International Conference on Learning Representations*, 2025.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. HybridFlow: A flexible and efficient RLHF framework. In *Proceedings of the Twentieth European Conference on Computer Systems*, 2025.

Yucheng Shi, Wenhao Yu, Zaitang Li, Yonglin Wang, Hongming Zhang, Ninghao Liu, Haitao Mi, and Dong Yu. MobileGUI-RL: Advancing mobile GUI agent through reinforcement learning in online environment. *arXiv preprint arXiv:2507.05720*, 2025.

Weiwei Sun, Miao Lu, Zhan Ling, Kang Liu, Xuesong Yao, Yiming Yang, and Jiecao Chen. Scaling long-horizon LLM agent via context-folding. In *Proceedings of the Forty-Third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=lNRgWoGfYg.

Junyang Wang, Haiyang Xu, Haitao Jia, Xi Zhang, Ming Yan, Weizhou Shen, Ji Zhang, Fei Huang, and Jitao Sang. Mobile-Agent-v2: Mobile device operation assistant with effective navigation via multi-agent collaboration. *Advances in Neural Information Processing Systems*, 37:2686–2710, 2024.

Zhenhailong Wang, Haiyang Xu, Junyang Wang, Xi Zhang, Ming Yan, Ji Zhang, Fei Huang, and Heng Ji. Mobile-Agent-E: Self-evolving mobile assistant for complex tasks. *arXiv preprint arXiv:2501.11733*, 2025.

Haiyang Xu, Xi Zhang, Haowei Liu, Junyang Wang, Zhaozai Zhu, Shengjie Zhou, Xuhao Hu, Feiyu Gao, Junjie Cao, Zihua Wang, et al. Mobile-Agent-v3.5: Multi-platform fundamental GUI agents. *arXiv preprint arXiv:2602.16855*, 2026.

Yifan Xu, Xiao Liu, Xinghan Liu, Jiaqi Fu, Hanchen Zhang, Bohao Jing, Shudan Zhang, Yuting Wang, Wenyi Zhao, and Yuxiao Dong. MobileRL: Online agentic reinforcement learning for mobile GUI agents. *arXiv preprint arXiv:2509.18119*, 2025.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2022.

Deheng Ye, Zhao Liu, Mingfei Sun, Bei Shi, Peilin Zhao, Hao Wu, Hongsheng Yu, Shaojie Yang, Xipeng Wu, Qingwei Guo, et al. Mastering complex control in MOBA games with deep reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 6672–6679, 2020.

Hongli Yu, Tinghong Chen, Jiangtao Feng, Jiangjie Chen, Weinan Dai, Qiying Yu, Ya-Qin Zhang, Wei-Ying Ma, Jingjing Liu, Mingxuan Wang, et al. MemAgent: Reshaping long-context LLM with multi-conv RL-based memory agent. *arXiv preprint arXiv:2507.02259*, 2025a.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, et al. DAPO: An open-source LLM reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*, 2025b.

Zijian Zhou, Ao Qu, Zhaoxuan Wu, Sunghwan Kim, Alok Prakash, Daniela Rus, Jinhua Zhao, Bryan Kian Hsiang Low, and Paul Pu Liang. Mem1: Learning to synergize memory and reasoning for efficient long-horizon agents. *arXiv preprint arXiv:2506.15841*, 2025.

Contents of the appendix

A Related Work . . . 13
B Method Details . . . 13
B.1 Algorithms . . . 13
B.2 Scalar GRPO in Standardized Coordinates . 14
B.3 Properties of Decoupled Advantages . . . 14
B.4 What the Sampling Mixture Implies . . . 15
B.5 Verifier Details . . . 15
B.6 Prompt and Response Schema . . . 15
C Training and Evaluation Setup . . . 16
C.1 Training Configuration . . . 16
C.2 Evaluation Protocol . . . 16
D Additional Results . . . 17
D.1 Results by Application Count and Task Type . . . 17
D.2 Offline Memory Metrics . . . 17
D.3 Failure Labels . . . 18
D.4 Training Curves of the Ratio Sweep 19
E Case Studies . . . 20

A Related Work

Mobile GUI agents and their memory. End-to-end mobile agents map screenshots to actions (Hong et al., 2024; Qin et al., 2025; Gu et al., 2025; Gao et al., 2026; Xu et al., 2026), while agentic frameworks add planning, grounding, and memory modules around a backbone (Agashe et al., 2025; Wang et al., 2024; 2025); cross-application, memory-intensive benchmarks show that neither family reliably retains information over long horizons (Kong et al., 2025; Liu et al., 2026b). MemGUI-Agent makes memory management part of the policy output through ConAct (Section 2) (Liu et al., 2026a); UI-Copilot delegates retrieval and calculation to a copilot model (Lu et al., 2026b); we keep ConAct unchanged and study how RL post-training changes its use.

Learned context and memory management. ReAct interleaves reasoning with actions (Yao et al., 2022); later work learns what to keep: MemAgent updates a compact memory over document segments with RL (Yu et al., 2025a), Mem1 consolidates memory and reasoning jointly (Zhou et al., 2025), MemPO trains self-written memory for long-horizon agents (Li et al., 2026), and context-folding agents branch into sub-tasks and summarize them (Sun et al., 2026). Here memory operations are interleaved with grounded device actions inside one response, and the verifier scores the *structure* of a folding decision against annotations rather than end-task success.

Policy optimization with multiple rewards. PPO uses a clipped surrogate (Schulman et al., 2017); GRPO replaces the critic with within-group comparisons (Shao et al., 2024), and DAPO adds asymmetric clipping and dynamic sampling (Yu et al., 2025b). Rule-based multi-component rewards are common in GUI RL (Lu et al., 2026a; 2025), usually aggregated into one scalar before normalization. GDPO shows that normalizing the aggregate collapses distinct reward combinations onto identical advantages and normalizes each reward separately before weighting and batch normalization (Liu et al., 2026c). We use the same per-component standardization for the four ConAct components (Section 3.2). Online mobile RL such as MobileGUI-RL and MobileRL optimizes task success through environment interaction (Shi et al., 2025; Xu et al., 2025); our annotation-based verification is complementary and needs no device interaction during training.

B Method Details

B.1 Algorithms

Algorithm 1 (Section 3.3) lists one FARPO training iteration; Algorithm 2 lists the matched scalar baseline of Table 5, which differs only in the order of aggregation and normalization. Both use population standard deviations within a group, the same ϵ gate, the same weights, and the same masked batch whitening. Classic GRPO (Shao et al., 2024) additionally uses a Bessel-corrected group standard deviation, no zero-variance gate, and no batch whitening; the matched baseline holds these choices fixed so that the comparison isolates the aggregation order.

Algorithm 2 Matched scalar baseline (aggregate, then normalize)

1: lines 1–6 as in Algorithm 1
2: $R_{i,j} \leftarrow \sum_k w_k r^k_{i,j}$ ▷ aggregate first
3: $\mu_i, \sigma_i \leftarrow$ group mean and population standard deviation of $\{R_{i,j}\}_{j=1}^{G}$
4: $u_{i,j} \leftarrow \mathbb{1}[\sigma_i > \epsilon]\,(R_{i,j} - \mu_i)/(\sigma_i + \epsilon)$ ▷ one shared denominator
5: lines 10–11 as in Algorithm 1

B.2 Scalar GRPO in Standardized Coordinates

Proposition 1 (Dispersion-dependent standardized sensitivity)**.** *Conditioned on the empirical group moments, the coefficient of the standardized coordinate $z^k_{i,j}$ in Equation (3) is*

$$\gamma^{\mathrm{GRPO}}_{i,k} = \frac{v^k \sigma^k_i}{\sqrt{\mathbf{v}^\top \mathbf{\Sigma}_i \mathbf{v}} + \epsilon}. \tag{7}$$

For active components k and ℓ the coefficient ratio is $v^k\sigma^k_i : v^\ell\sigma^\ell_i$. The raw reward-space weights remain $\mathbf{v}$, and normalization preserves the within-group ordering of the scalar score.

Proof. The scalar group mean is $\mathbf{v}^\top \boldsymbol{\mu}_i$, so $R_{i,j} - \mathbf{v}^\top \boldsymbol{\mu}_i = \mathbf{v}^\top \boldsymbol{\delta}_{i,j}$ with population variance $G^{-1}\sum_j (\mathbf{v}^\top \boldsymbol{\delta}_{i,j})^2 = \mathbf{v}^\top \mathbf{\Sigma}_i \mathbf{v}$. Substituting $\delta^k_{i,j} = \sigma^k_i z^k_{i,j}$ into Equation (3) gives $A^{\mathrm{GRPO}}_{i,j} = \sum_k \gamma^{\mathrm{GRPO}}_{i,k} z^k_{i,j}$ with the stated coefficients. A Bessel correction multiplies every coefficient in a group by the same factor $\sqrt{G/(G-1)}$ and the additive stabilizer perturbs only the shared denominator, so neither changes the ratios. □

Worked example. Table 2 (Section 3.1) is a constructed arithmetic example, not a training result: two active components with equal weights, one with ten times the within-group spread of the other. A second constructed group with all four ConAct components, $\mathbf{r}_j = (1, 1, r^{\mathrm{param}}_j, r^{\mathrm{fold}}_j)$ with $r^{\mathrm{param}} = (0.95, 0.85, 0.90, 0.80, 0.90)$ and $r^{\mathrm{fold}} = (1, 0.4, 0, 1, 0.4)$: behaves the same way: format and action type are constant and gated out, parameters vary little ($\sigma = 0.05$) and folding varies a lot ($\sigma = 0.39$), so scalar GRPO lets the folding component outweigh the parameter component 1.9:1 although $\mathbf{w}$ prefers parameters 4:1, and it assigns a positive advantage ($+0.29$, second best) to the response with the worst parameters; the reward-decoupled estimator ranks that response lowest (-0.51) while response 1 stays best under both estimators.

B.3 Properties of Decoupled Advantages

Proposition 2 (Centering, scale invariance, and residual correlation)**.** *For any $\epsilon \geq 0$, $G^{-1}\sum_j u_{i,j} = 0$. Let $\mathcal{A}_i = \{k : \sigma^k_i > \epsilon\}$ be the active set. For $\epsilon = 0$, $u_{i,j}$ is invariant to per-component transformations $r'^k = c_k r^k + b_k$ with $c_k > 0$, and its within-group variance is $\mathrm{Var}_j[u_{i,j}] = \mathbf{w}^\top_{\mathcal{A}_i} \mathbf{C}_{i,\mathcal{A}_i} \mathbf{w}_{\mathcal{A}_i}$, where $\mathbf{C}_i = \mathbf{D}_i^{-1}\mathbf{\Sigma}_i\mathbf{D}_i^{-1}$ is the within-group correlation matrix.*

Proof of Proposition 2. Let row j of $\mathbf{\Delta}_i$ be $\boldsymbol{\delta}^\top_{i,j}$, so $\mathbf{u}_i = \mathbf{\Delta}_i \mathbf{P}^\epsilon_i \mathbf{w}$. Since $\sum_j \boldsymbol{\delta}_{i,j} = \mathbf{0}$, the group mean of $u_{i,j}$ is exactly zero for any $\epsilon \geq 0$. On the active coordinates with $\epsilon = 0$, $\mathbf{P}^0_i = \mathbf{D}_i^{-1}$. For $\mathbf{\Lambda} = \mathrm{diag}(c_1, \ldots, c_K)$ with $c_k > 0$, centering removes the offsets b_k, $\boldsymbol{\delta}'_{i,j} = \mathbf{\Lambda}\boldsymbol{\delta}_{i,j}$ and $\mathbf{D}'_i = \mathbf{\Lambda}\mathbf{D}_i$; positive scaling preserves the zero-variance coordinates and hence the active set, so $(\mathbf{D}'_i)^{-1}\boldsymbol{\delta}'_{i,j} = \mathbf{D}_i^{-1}\boldsymbol{\delta}_{i,j}$ and $u'_{i,j} = u_{i,j}$. Finally, $G^{-1}\sum_j u^2_{i,j} = \mathbf{w}^\top \mathbf{D}_i^{-1}\big(G^{-1}\sum_j \boldsymbol{\delta}_{i,j}\boldsymbol{\delta}^\top_{i,j}\big)\mathbf{D}_i^{-1}\mathbf{w} = \mathbf{w}^\top \mathbf{C}_i \mathbf{w}$ on the active coordinates. For finite ϵ and a component that stays active after rescaling, the deviation from invariance is $\left|\frac{c_k\delta^k}{c_k\sigma^k+\epsilon} - \frac{\delta^k}{\sigma^k+\epsilon}\right| = \frac{|\delta^k|\,\epsilon\,|c_k-1|}{(c_k\sigma^k+\epsilon)(\sigma^k+\epsilon)}$, which vanishes as $\epsilon \to 0$ and is small whenever $\sigma^k, c_k\sigma^k \gg \epsilon$. □

B.4 What the Sampling Mixture Implies

With the span prior α_ρ of Section 3.3, for any per-prompt function $f_\theta(x)$ fixed independently of the subset construction, with class averages $\mathcal{J}_{\text{span}}$ and $\mathcal{J}_{\text{step},\mathcal{S}_\rho}$,

$$\mathbb{E}_{x\sim\text{Unif}(\mathcal{D}_\rho)}[f_\theta(x)] = \alpha_\rho\,\mathcal{J}_{\text{span}}(\theta) + (1-\alpha_\rho)\,\mathcal{J}_{\text{step},\mathcal{S}_\rho}(\theta), \tag{8}$$

and each eligible step example is included with probability m_ρ/N_{step}, so $\mathbb{E}_{\mathcal{S}_\rho}[\mathcal{J}_{\text{step},\mathcal{S}_\rho}] = \mathcal{J}_{\text{step}}$. This identity describes prompt exposure for a prompt-averaged surrogate: the sampler changes the empirical prompt prior without importance correction, and the logged training batches confirm the realized priors (Table 7). The implemented loss averages over response tokens and batch whitening couples responses across prompts, which act on top of this prior. Because responses have different lengths, $\bar{u}_m = \sum_{i,j} L_{i,j}u_{i,j} / \sum_{i,j} L_{i,j}$ can be non-zero even though every group mean of u is zero (for example advantages $(-1, 1)$ with lengths $(1, 3)$ give $\bar{u}_m = \frac{1}{2}$); whitening preserves the ordering of the valid responses.

B.5 Verifier Details

This section lists the rules behind the four components of Section 2. The applicable format checks are reasoning tags, a valid action JSON with the required arguments, UI-observation tags, intent tags, and, from step two onward, a folding JSON with a two-endpoint range and a summary field. The action-type comparison maps `swipe`↔`scroll` and `terminate`↔`complete`. Parameter rules by type:

- **click, long press**: Euclidean distance between the predicted point and the annotated target on the 0–1000 normalized screen, threshold 140 (or point-in-box when a box is annotated); a long press without a duration receives $0.8\times$ the coordinate score.
- **swipe**: direction agreement; **system button**: button agreement; **wait**: 1 for a positive duration, 0.5 for any other duration value.
- **type, answer, open**: exact normalized match or token-F1 ≥ 0.5.
- **memory add/update**: $0.3 \cdot [\text{F1(id)} \geq 0.5] + 0.7 \cdot \text{F1(content)}$; **memory delete**: $[\text{F1(id)} \geq 0.5]$.

Folding intervals are inclusive integer ranges; an optional summary-similarity term and depth bonus of the verifier are disabled in all reported runs, and the outer exception handler sets the semantic components to 0 on parsing failures. A logged scalar $0.1r^{\text{fmt}} + 0.4r^{\text{type}} + 0.4r^{\text{param}}$ (rescaled by 0.9 and combined with $0.1r^{\text{fold}}$ when a fold is annotated) is used for monitoring only; the estimator consumes the four raw components.

B.6 Prompt and Response Schema

All policies use the ConAct system prompt of MemGUI-Agent unchanged. The box below reproduces the response schema that the verifier parses; the full prompt, including the action space and memory-operation arguments, is given in the MemGUI-Agent paper.

ConAct response schema (abridged from the MemGUI-Agent system prompt)

```
<thinking> step-by-step reasoning about the goal, the folded history, the stored UI
facts and the current screen </thinking>
<folding> {"range": [start_step, current_step], "summary": "compressed description of
the folded steps"} </folding>  (from step 2; range [t,t] records step t, [s,t] with
s<t compresses steps s..t)
<tool_call> {"name": "mobile_use", "arguments": {"action": click | long_press | swipe
| type | answer | system_button | wait | terminate | memory_add | memory_update |
memory_delete, ...}} </tool_call>
<ui_observation> facts visible on the current screen </ui_observation>
<action_intent> what the next step should achieve </action_intent>
```

Table 9: **Training configuration of MemGUI-8B-RL (ρ=9).** The ratio sweep changes only the corpus $\mathcal{D}_\rho$ (hence the steps per epoch); the estimator ablation fixes 100 steps and truncates natural corpora to the ρ=9 size. Reward weights: format, action type, action parameters, folding.

Setting	Value	Setting	Value
Initialization	MemGUI-8B-SFT (Qwen3-VL-8B)	Advantage estimator	decoupled (Equation (4))
Rollouts per prompt G	5	Reward weights $\mathbf{w}$	0.1 / 0.4 / 0.4 / 0.1
Sampling	temperature 1.0, top-p 1.0	Component gate / stabilizer ϵ	10^{-6} (population std)
Prompt / response length	16,384 / 4,096 tokens	Batch whitening ϵ_b	10^{-8} (Bessel-corrected)
Prompts per rollout batch	128	PPO clipping	ϵ_{lo}=0.2, ϵ_{hi}=0.3, dual c=3
PPO epochs per batch	1	KL penalty	low-variance estimator, β=0.01
Optimizer	AdamW, lr 10^{-6}, wd 0.01	Loss averaging	valid response tokens
Warm-up / gradient clip	0 / 1.0	Folding summary scored	no (range only)
Span-to-step ratio ρ	9 (all span folds kept)	Folding depth bonus	0 (disabled)
Epochs / optimizer steps	1 / 100	Validation	every 20 steps, greedy, 5,941 states
Hardware	8 GPUs, FSDP actor, vLLM rollouts	Seeds (filter / data / rollout)	42 / 1 / 1

C Training and Evaluation Setup

C.1 Training Configuration

Table 9 lists the configuration of MemGUI-8B-RL; the reward weights $\mathbf{w}$, the group size G, and the PPO constants were fixed before the first run and are shared by every RL row of the paper, so the sweep and the ablation vary one factor at a time.

Policy objective. With likelihood ratio $q_{i,j,n} = \pi_\theta(y_{i,j,n} \mid x_i, y_{i,j,<n})/\pi_{\text{old}}(\cdot)$, the actor of Section 3.2 minimizes a critic-free, dual-clipped PPO surrogate (Schulman et al., 2017; Ye et al., 2020; Yu et al., 2025b) with a KL penalty to the frozen SFT policy π_{ref}:

$$\ell^{\text{clip}}_{i,j,n} = \min\big(q_{i,j,n}A_{i,j,n},\ \text{clip}(q_{i,j,n}, 1-\epsilon_{\text{lo}}, 1+\epsilon_{\text{hi}})A_{i,j,n}\big), \tag{9}$$

$$\ell^{\text{dual}}_{i,j,n} = \begin{cases} \max(\ell^{\text{clip}}_{i,j,n},\ cA_{i,j,n}), & A_{i,j,n} < 0, \\ \ell^{\text{clip}}_{i,j,n}, & A_{i,j,n} \geq 0, \end{cases} \qquad \mathcal{L}(\theta) = -\langle \ell^{\text{dual}} \rangle_m + \beta\, \langle \widehat{d}_{\text{KL}}(\pi_\theta, \pi_{\text{ref}}) \rangle_m, \tag{10}$$

where $\langle\cdot\rangle_m$ is the mean over valid tokens, the constants are those of Section 3.2 and Table 9, and $\widehat{d}_{\text{KL}}$ is the low-variance estimator below.

KL estimator and validation. The KL penalty is the low-variance estimator $\widehat{d}_{\text{KL}} = \text{clip}(\exp(h) - h - 1, -10, 10)$ with $h = \text{clip}(\log \pi_{\text{ref}} - \log \pi_\theta, -20, 20)$, an implementation safeguard rather than an exact sequence-level divergence. Validation decodes greedily (n=1, temperature 0) on 5,941 states from 295 held-out trajectories every 20 steps.

Corpus construction. The filter first keeps examples marked reasonable in the dataset metadata, then classifies the annotated fold: a valid range with $e > s$ is a span fold and $e = s$ a step fold; no-fold and invalid-range states are dropped from mixed corpora. The realized step-fold share of every corpus, measured in the logged training batches, matches its target: 77.4% for the natural corpus (hence $\rho_{\text{nat}} \approx 0.29$), $1/(\rho{+}1)$ for the folding-aware corpora (33.3 / 25.0 / 20.0 / 10.0% for ρ=2/3/4/9), and 0 for the span-only corpus. The size-matched natural corpora of the estimator ablation sample span and step examples at the natural prior until they reach the size of the ρ=9 corpus.

C.2 Evaluation Protocol

Every policy is run on all 128 MemGUI-Bench tasks for up to three attempts under the benchmark's protocol, and MemGUI-Eval judges each attempt from the stored trajectory and final output; an attempt without a verdict counts as a failure (at most two such attempts per run). MobileWorld GUI-only runs one attempt per task against the benchmark's deterministic checkers.

D Additional Results

D.1 Results by Application Count and Task Type

Table 10 gives the full Pass@k and IRR breakdown by the number of applications a task spans for every agent of Table 4, and Figure 10 plots the SFT-to-FARPO gains by application count and task type that Section 4.2 discusses. One reading of the Hard tier of Table 4 complements the main text: both ConAct 8B policies exceed every end-to-end model up to 32B there (28.9 and 34.2 vs. at most 21.1 Pass@3), whereas direct RL from the backbone falls to 2.6, so the capacity to carry the longest tasks comes from the ConAct supervision, and FARPO adds to it on the two shorter tiers.

Table 10: **Results on MemGUI-Bench by the number of applications a task spans** (28/56/34/10 tasks span 1/2/3/4 applications): Pass@k and information-retention rate (IRR, %). Same rows, sources, and Δ (SFT+FARPO over the backbone) as Table 4; **Best** / second-best among end-to-end models ≤8B. MemGUI-8B-RL leads every end-to-end model up to 32B on one-, two-, and three-application tasks and both Qwen3-VL-235B models on two-application tasks.

Agent	Memory	1 app (28)			2 apps (56)			3 apps (34)			4 apps (10)			Overall (128)		
		P@1	P@3	IRR	P@1	P@3	IRR	P@1	P@3	IRR	P@1	P@3	IRR	P@1	P@3	IRR
Agentic frameworks																
Agent-S2 (Agashe et al., 2025)	memory agent	50.0	78.6	51.7	19.6	35.7	37.6	26.5	52.9	38.9	10.0	30.0	33.3	27.3	49.2	39.5
M3A (Rawles et al., 2025)	memory agent	46.4	64.3	31.7	28.6	41.1	43.8	29.4	44.1	35.9	30.0	50.0	37.5	32.8	47.7	39.3
T3A (Rawles et al., 2025)	memory agent	42.9	60.7	26.7	16.1	37.5	33.3	23.5	38.2	30.2	0.0	30.0	11.2	22.7	42.2	29.6
End-to-end models, 32B and larger																
Qwen3-VL-235B-Thinking (Bai et al., 2025)	action-thought	32.1	60.7	35.3	23.2	42.9	28.5	23.5	44.1	29.3	10.0	40.0	32.0	24.2	46.9	30.0
Qwen3-VL-235B-Instruct (Bai et al., 2025)	action-thought	35.7	50.0	37.7	21.4	35.7	27.9	20.6	32.4	31.8	10.0	20.0	15.0	23.4	36.7	29.2
Qwen3-VL-32B-Instruct (Bai et al., 2025)	action-thought	46.4	64.3	40.0	10.7	19.6	14.2	17.6	32.4	34.1	10.0	10.0	31.9	20.3	32.0	25.0
GUI-Owl-1.5-32B (Xu et al., 2026)	action-thought	35.7	60.7	18.9	1.8	7.1	19.8	5.9	11.8	14.7	10.0	10.0	22.5	10.9	20.3	18.4
MemGUI-Agent-235B (Liu et al., 2026a)	ConAct (zero-shot)	64.3	89.3	73.7	30.4	57.1	46.8	35.3	58.8	42.5	10.0	30.0	20.7	37.5	62.5	46.8
End-to-end models, 7–8B																
UI-Venus-7B (Gu et al., 2025)	action-thought	21.4	28.6	6.7	1.8	1.8	2.7	0.0	2.9	1.5	0.0	0.0	0.0	5.5	7.8	2.6
UI-TARS-1.5-7B (Qin et al., 2025)	multi-turn	14.3	21.4	11.1	0.0	1.8	1.8	0.0	2.9	4.0	0.0	0.0	2.9	3.1	6.2	3.8
GUI-Owl-1.5-8B (Xu et al., 2026)	action-thought	39.3	50.0	22.7	5.4	7.1	14.3	2.9	5.9	15.6	0.0	0.0	11.0	11.7	15.6	15.5
Qwen3-VL-8B-Instruct (Bai et al., 2025)	action-thought	28.6	42.9	19.6	1.8	12.5	9.6	8.8	17.6	17.4	0.0	10.0	**31.5**	9.4	20.3	15.1
+ FARPO (ρ=9, direct RL)	ConAct	32.1	53.6	31.8	8.9	12.5	25.7	11.8	17.6	23.9	0.0	**20.0**	31.3	14.1	23.4	26.5
MemGUI-8B-SFT (Liu et al., 2026a)	ConAct	39.3	64.3	36.3	21.4	32.1	34.1	17.6	23.5	24.8	**10.0**	**20.0**	17.1	23.4	35.9	30.2
+ FARPO (ρ=9) = MemGUI-8B-RL	ConAct	**64.3**	**75.0**	**60.9**	**23.2**	**46.4**	**34.4**	**23.5**	**38.2**	**40.1**	0.0	**20.0**	28.5	**30.5**	**48.4**	**39.0**
Δ *SFT+FARPO vs. Qwen3-VL-8B-Instruct*		+35.7	+32.1	+41.3	+21.4	+33.9	+24.8	+14.7	+20.6	+22.7	0.0	+10.0	−3.0	+21.1	+28.1	+23.9

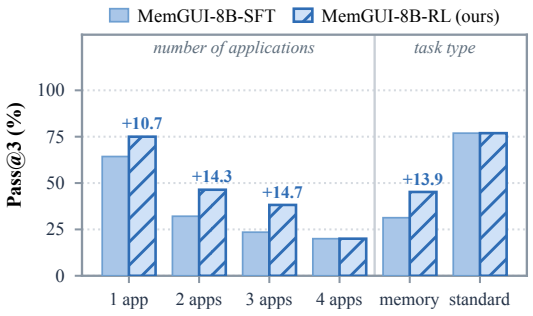


Figure 10: Pass@3 of MemGUI-8B-SFT vs. MemGUI-8B-RL by the number of applications a task spans and by task type (memory-intensive vs. standard), with the FARPO-only gains over SFT annotated above the bars.

D.2 Offline Memory Metrics

Table 11 and Figure 7 report the validation metrics logged during training at the final checkpoint of every run. A predicted fold matches the annotation when both endpoints are within two steps of the reference; *depth-type accuracy* is agreement on whether the fold is a span or a step; UI-action match uses the same coordinate, direction, and text rules as the verifier; memory-trigger F1 treats the presence of a memory operation as the positive class. Beyond the improvements reported in Section 4.4, one metric moves the other way: folding-range agreement tracks the training prior rather than downstream success, dropping from 81.6% to 63.4% for ρ=9 and to 58–69% for the other span-heavy policies, which nevertheless solve more benchmark tasks. Range agreement falls

Table 11: **Offline memory and action metrics on the held-out validation states** (295 trajectories, 5,941 states, greedy decoding, final checkpoint). Range agreement uses a two-step endpoint tolerance; *deep-fold rate* is the share of predicted folds with $e > s$ (annotated share on these states: 23.2%, in the training corpus: 22.7%); match accuracies require type and argument agreement.

Policy	Steps	Folding			Memory ops		UI actions	
		deep-fold rate	range acc.	depth-type acc.	trigger F1	match acc.	type acc.	match acc.
MemGUI-8B-SFT	0	17.7	81.6	71.6	70.5	63.1	90.8	83.4
Decoupled estimator, ratio sweep (one epoch)								
natural	264	0.2	83.5	73.9	76.8	74.1	92.3	86.3
$\rho = \frac{1}{4}$ (reverts to natural)	398	0.4	83.4	73.7	76.5	79.0	92.4	86.7
$\rho = 2$	135	59.7	70.5	50.4	77.3	75.0	92.3	85.8
$\rho = 3$	120	86.5	66.1	34.5	75.5	76.8	92.2	86.0
$\rho = 4$	112	89.5	69.0	33.1	76.6	76.5	92.5	85.8
$\rho = 9$ (MemGUI-8B-RL)	100	92.5	63.4	32.0	76.4	75.3	91.8	85.6
$\rho = \infty$ (span only)	90	95.9	58.8	29.1	75.9	77.1	92.1	85.1
Stepwise ablation (100 steps, size-matched)								
scalar, natural (GRPO recipe)	100	0.1	83.7	74.0	75.1	72.3	91.9	85.5
decoupled, natural	100	0.8	83.8	74.2	74.1	74.7	92.2	85.8
decoupled, $\rho = 9$ (second seed)	100	92.0	63.3	31.7	75.8	75.9	91.8	85.4
direct RL from Qwen3-VL-8B-Instruct (ρ=9)	100	95.8	61.3	29.8	73.1	72.9	90.7	82.7
Qwen3-VL-8B-Instruct before RL	0	13.2	84.6	70.5	60.4	42.7	84.8	75.8

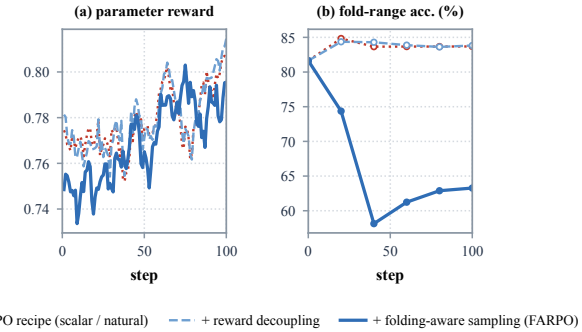


Figure 11: **Further training dynamics of the estimator×sampling ablation**, complementing Figure 5: **(a)** smoothed parameter reward, **(b)** validation fold-range accuracy (greedy, every 20 steps), which drops for the ρ=9 policy while its benchmark results improve.

because these policies predict spans where the annotator recorded steps; boundary agreement is thus not what makes the memory policy useful, exposure to span compression is, which argues for a verifier that scores what a summary preserves rather than where it ends. The direct-RL row shows that post-training a policy that was never supervised on ConAct improves every metric but stays well below the SFT-initialized runs on UI-action match (82.7 vs. 85–87) and memory-trigger F1 (73.1 vs. 74–77), a gap that RL alone does not close.

D.3 Failure Labels

Process-memory hallucination (losing track of the procedure) is the one label that RL leaves in place (134→133 attempts); it concentrates on the longest tasks: on the Hard tier of Table 4 it accounts for 45 of the 88 failed attempts of MemGUI-8B-SFT and 51 of the 85 of MemGUI-8B-RL, and the two policies solve 13 and 11 of the 38 tasks. Two facts explain why the RL stage adds little there. The RL prompts under-cover the regime: 60.6% of the annotated training states of MemGUI-3K lie within steps 1–20 of their trajectory, 27.0% within 21–40, and only 12.4% beyond step 40, so deep-history folding decisions are rarely sampled, the same exposure problem that ρ addresses for span folds, now along the depth axis. And the fold verifier scores where a fold ends, not whether its summary preserves the state of the procedure, so the failure that dominates these tasks carries no training signal; a verifier that scores what a summary preserves (Section 5) and a depth-aware sampler are the corresponding next steps.

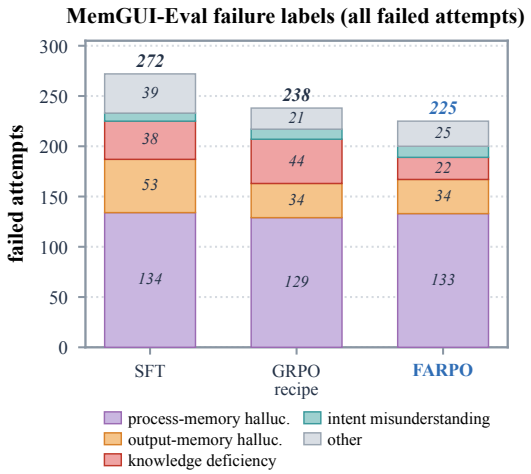


Figure 12: **MemGUI-Eval failure labels** over all failed attempts of MemGUI-8B-SFT, the GRPO recipe (scalar/natural), and MemGUI-8B-RL (three attempts per task, 128 tasks).

D.4 Training Curves of the Ratio Sweep

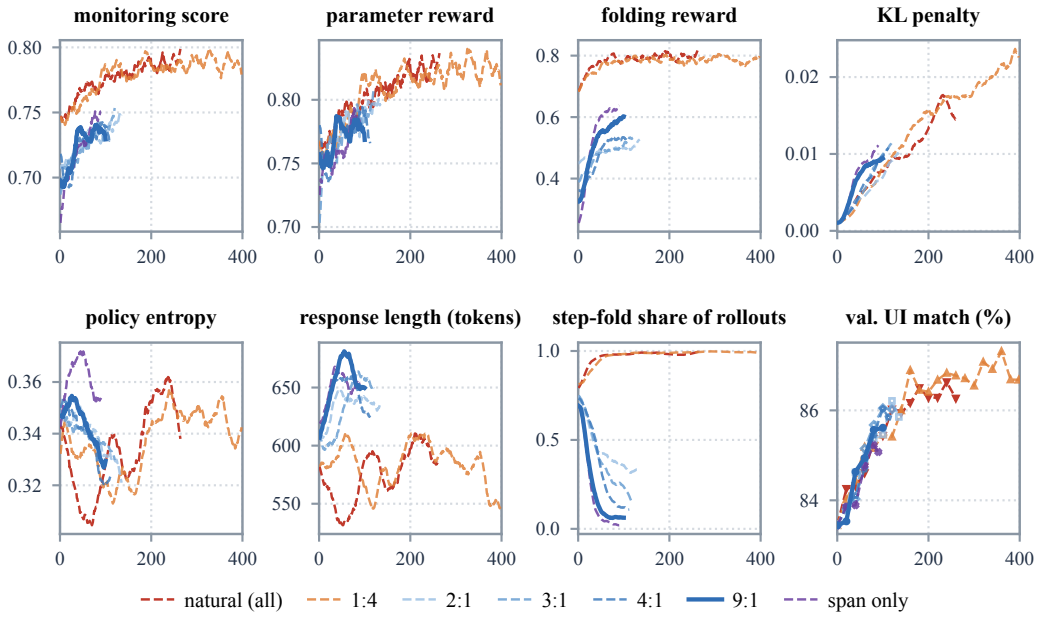


Figure 13: **Training and validation curves of the ratio sweep** (training metrics smoothed over 15 steps; warm colors: step-heavy corpora, blue ramp: increasing span exposure). The deep-fold rate and the zero-variance share are Figure 6a and Figure 9; the format reward is saturated for every run and is therefore not shown.

Figure 13 shows the remaining logged metrics of the seven runs; three points complement Section 4.4. First, the folding reward of the step-heavy corpora (natural, $\rho=\frac{1}{4}$) starts near 0.70 and reaches its plateau of about 0.80 within 50 steps, whereas the span-heavy corpora start at 0.28–0.45 and are still improving (0.51–0.63) when their epoch ends: the span-fold decision is where learning signal remains. Second, span exposure lengthens the responses (628–654 tokens at the end of training vs. 551–585 for the step-heavy corpora), the cost of emitting span summaries, and the step-fold share of the rollouts falls from about 0.7 to 0.33, 0.19, 0.12, 0.06, and 0.02 in the order of ρ (and rises to 0.99

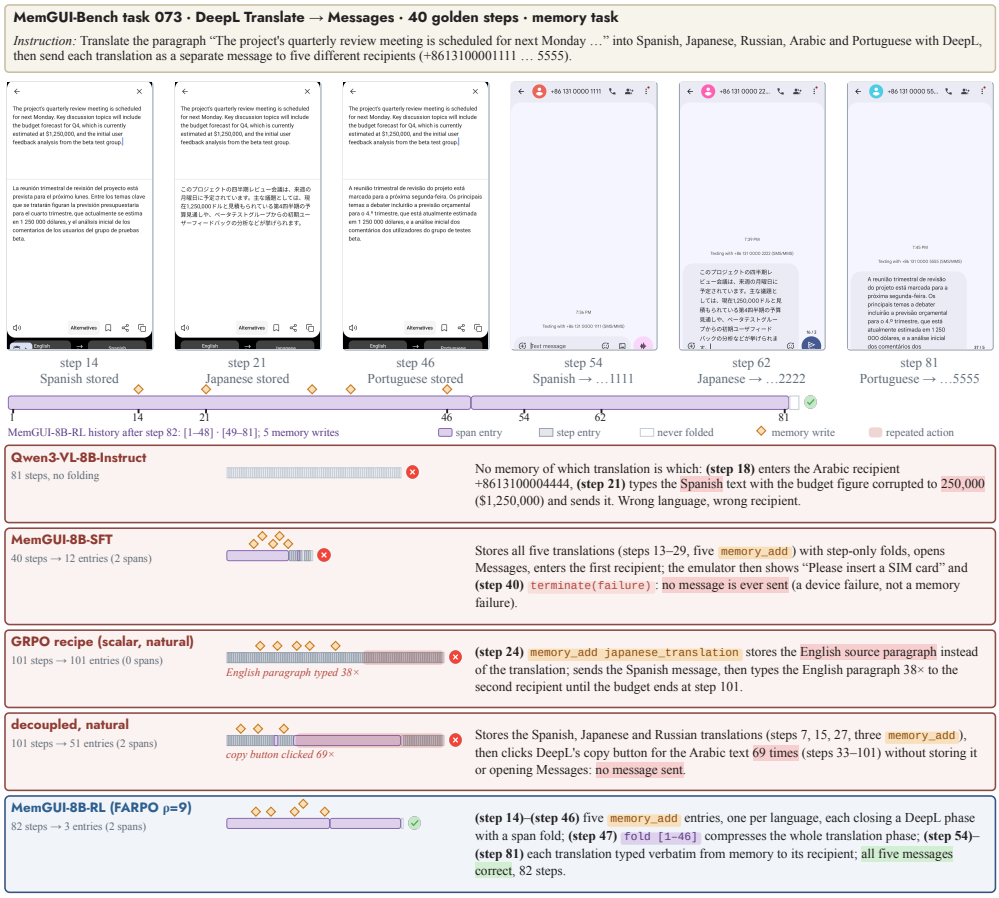


Figure 14: **Case study, task 073** (DeepL Translate → Messages, 40 golden steps).

for the two step-heavy corpora). Third, policy entropy stays within 0.32–0.35 and the KL penalty below 0.025 for every ratio, so no run collapses or drifts far from the SFT policy within one epoch.

E Case Studies

Figures 14 to 16 complement Figure 8 with three further MemGUI-Bench tasks, one per page. Each figure shows the MemGUI-8B-RL trajectory as a strip of screenshots (the step under each screenshot is the step at which it was observed) and, below it, one row per policy with the MemGUI-Eval verdict, the folded history at the end of the attempt, and the evidence from the stored trajectory. The history bars are replayed from the folding directives of every step on a common step scale: purple blocks are span entries, grey cells step entries, white cells steps that were never folded; diamonds mark memory writes and red bands runs of the same repeated action. Quotes are copied from the model responses, and every other policy is shown on its first attempt; the icons of the schematic and case figures are by Icons8.

Task 073 (Figure 14). A memory-output task: five translations must be produced in DeepL and delivered to five different recipients. The GRPO recipe stores the *English source* under the key `japanese_translation` and re-types it until its budget ends; the decoupled/natural policy stores three correct translations and then clicks DeepL's copy button 69 times for the fourth; the base model sends the Spanish text to the Arabic recipient with a corrupted amount. The SFT policy stores all five translations but its emulator reported a missing SIM card when the first chat opened and it terminated, a device failure rather than a memory failure. Only MemGUI-8B-RL delivers all five messages, closing each translation with a span fold and ending with a three-entry history.

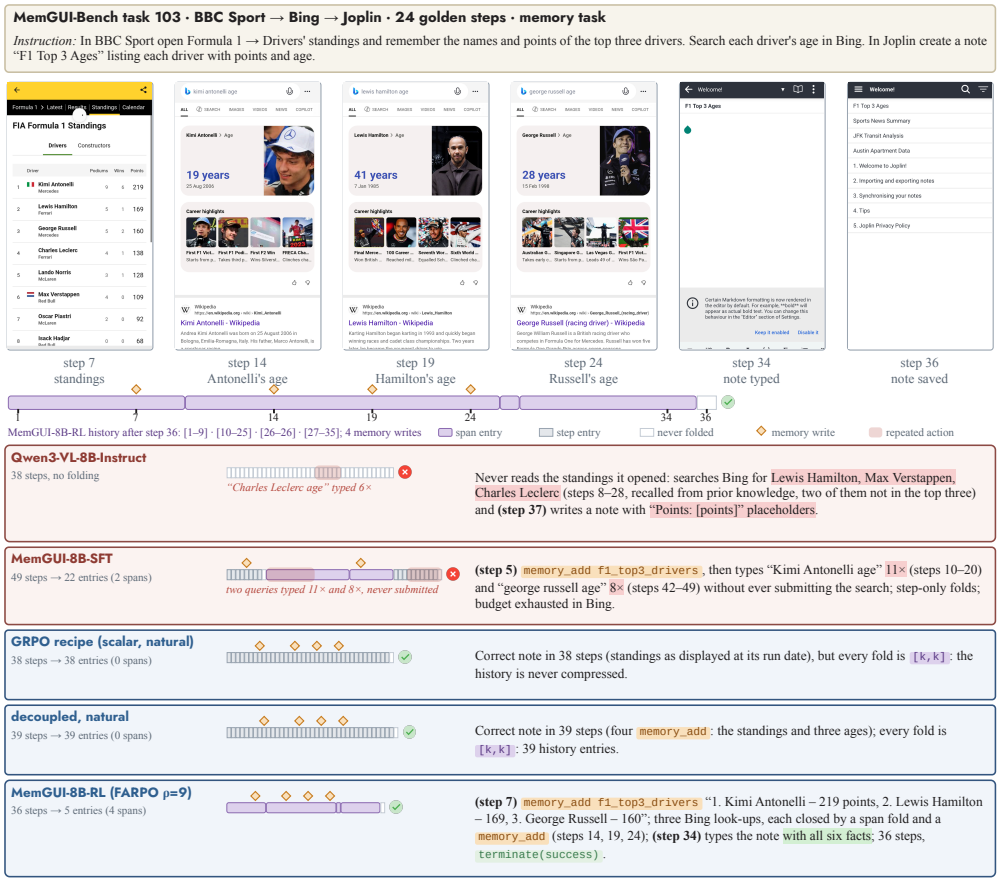


Figure 15: **Case study, task 103** (BBC Sport → Bing → Joplin, 24 golden steps).

Task 103 (Figure 15). A task that all three RL policies solve and neither the base model nor the SFT policy does: the base model never reads the standings it opened and searches for drivers it recalls from prior knowledge, while the SFT policy loops on an unsubmitted search. The RL policies differ in *how* they solve it: the two natural-corpus policies fold one step at a time (38 and 39 history entries), whereas MemGUI-8B-RL closes each Bing look-up with a span fold (five entries).

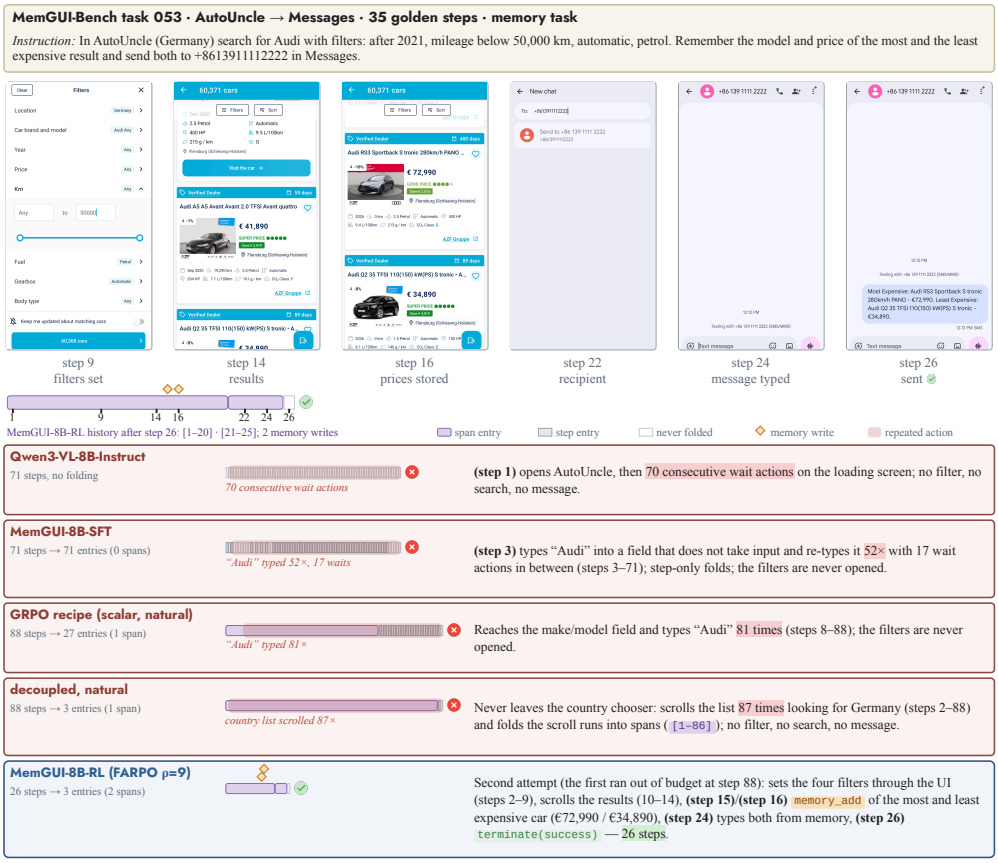


Figure 16: **Case study, task 053** (AutoUncle → Messages, 35 golden steps); the MemGUI-8B-RL row is its second attempt.

Task 053 (Figure 16). The failure mode that remains dominant after RL (Figure 12): all four baselines stall for 68–88 steps, three of them typing the same word or waiting on an unresponsive text field and the decoupled/natural policy scrolling a country list, and MemGUI-8B-RL also fails its first attempt before solving the task on the second, which goes through the filter UI and takes 26 steps.