Experience-driven Multi-turn Reinforcement Learning for GUI Agents

Zhengxi Lu[1,2*], **Jiabo Ye**[2], **Fei Tang**[1], **Yongliang Shen**[1†], **Haiyang Xu**[2‡], **Ziwei Zheng**[1], **Weiming Lu**[1], **Ming Yan**[2], **Fei Huang**[2], **Jun Xiao**[1], **Yueting Zhuang**[1]

[1]Zhejiang University [2]Tongyi Lab, Alibaba Group

{zhengxilu, syl}@zju.edu.cn shuofeng.xhy@alibaba-inc.com

Abstract

GUI agents have demonstrated remarkable progress in automating complex user interface interactions. However, training such agents for long-horizon tasks remains challenging. Single-turn reinforcement learning conditions on expert histories during training but self-generated histories during deployment, causing distribution mismatch. Online multi-turn methods eliminate this gap via environment interaction but suffer from sparse rewards and prohibitive costs. We propose **E**xperience-driven **M**ulti-turn **P**olicy **O**ptimization (**EMPO**), which leverages expert trajectories as environment experiences for on-policy multi-turn training. The agent constructs self-generated history throughout rollouts; when actions match expert experiences, the trajectory provides valid state transitions, and a Patch Module recovers mismatched steps to maintain on-policy rollouts. EMPO further incorporates discounted future rewards and dual-level advantage estimation to capture long-horizon dependencies. We also propose **AndroidControl-Real**, an evaluation metric strongly correlated with real-world performance (R^2=0.934). With only 1K public trajectories as RL experiences, our method achieves substantial gains over the base model (e.g., +12.0% on AndroidWorld and +23.8% on AITW) and achieves competitive performance against strong baselines such as UI-TARS-7B and GPT-4o, demonstrating better generalization than prior single-turn RL approaches. Code available: https://github.com/X-PLUG/MobileAgent/tree/main/UI-S1.

1 Introduction

Graphical User Interface (GUI) automation aims to enable agents to interact with digital environments through vision-based perception and action (Hu et al., 2025; Zhang et al., 2025a; Wang et al., 2025a; Liu et al., 2025a; Chen et al., 2026). Given a task like *"Delete the Wednesday alarm."*, mobile or computer use agents make sequential decisions, maintain memory across turns, and adapt to stochastic feedback from the dynamic screens.

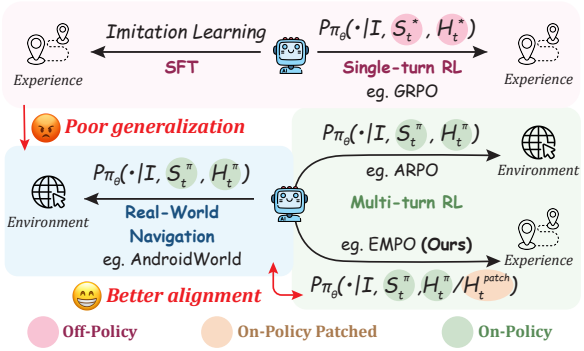

Figure 1: **Overview of three training paradigms for long-horizon GUI tasks**. SFT and single-turn RL (e.g., GRPO) condition on off-policy expert histories H_t^* from experience, whereas multi-turn RL methods (e.g., ARPO and our EMPO) construct on-policy, self-generated histories H_t^π or H_t^{patch} during rollout, yielding better alignment with real-world GUI interactions.

Reinforcement learning (RL) has emerged as an effective training paradigm for vision-based GUI agents (Bai et al., 2024; Lu et al., 2025b; Tang et al., 2025a; Ye et al., 2025; Du et al., 2025; Lu et al., 2026a). While single-turn RL (Lu et al., 2025b; Luo et al., 2025a; Liu et al., 2025c) primarily targets step-wise optimization via Group Relative Policy Optimization (GRPO) (Shao et al., 2024), recent multi-turn RL methods (e.g., ARPO (Lu et al., 2025a)) instead optimize rule-based rewards over full trajectories, yielding better generalization in real-world interactive tasks. We attribute it to the closer alignment between multi-turn RL and evaluation: **they both conduct multi-turn interactions driven by on-policy (self-generated) history**, as shown in Figure 1 and Table 1.

However, training multi-modal LLMs to operate

*Work done during intern at Tongyi Lab, Alibaba Group.
†Corresponding author.
‡Project Lead.

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 9474–9496
July 2-7, 2026

Method	Environment	History policy	Rollout policy	Reward Signals	Generalization	Stability
SFT	😡 Static	😡 Off-policy	😡 Off-policy	😁 Fine-Grained	😡 Poor	😁 Stable
GRPO	😡 Static	😡 Off-policy	😁 On-policy	😁 Fine-Grained	😐 Moderate	😐 Moderate
ARPO	😁 Dynamic	😁 On-policy	😁 On-policy	😡 Sparse	😁 Excellent	😡 Unstable
EMPO	😐 Static*	😁 On-policy	😁 On-policy	😁 Fine-Grained	😁 Excellent	😐 Moderate

Table 1: Comparison of GUI agent training paradigms. **Static environment**: Expert data (* denotes experiences). **Dynamic environment**: Real GUI devices. **Policy**: whether the content is generated by current policy model. **Generalization**: real-world performance (e.g., AITW). **Stability**: training efficiency and environment stability.

as autonomous GUI agents in interactive environments faces unique challenges: **1) Sparse and delayed rewards**: feedbacks are often received only at task completion, resulting in inefficient training for complex tasks; **2) Limited data diversity**: scaling to new environments or tasks requires extensive engineering effort for verification and simulation, which can be more labor-intensive than manually curating diverse, high-quality trajectories. **3) Operational instability**: real-world applications include CAPTCHAs, authentication prompts, and network failures that introduce frequent interruptions.

These challenges motivate our central question: *can we achieve on-policy multi-turn training without online interaction?* The key insight is that what truly matters for multi-turn learning is not environment dynamics, but on-policy history construction. The agent must condition on its own generated actions and reasoning traces during training.

Building on this insight, we propose **Experience-driven Multi-turn Policy Optimization (EMPO)**, which leverages expert trajectories as environmental experiences for multi-turn training. The screenshot sequences in trajectories serve as stable environmental states, while the agent constructs on-policy history by generating its own actions and reasoning at each step. When predictions match expert experiences, rollouts proceed along the trajectory; when divergence occurs, a Patch Module substitutes the expert action to recover the rollout, ensuring subsequent steps remain accessible for learning. Unlike sparse online rewards, expert trajectories provide fine-grained step-level feedback, enabling more informative learning signals.

To better capture these long-horizon signals of current actions, we incorporate discounted future rewards and weighted step- and episode-level advantages into policy optimization. *For efficient multi-turn evaluation*, we propose `AC-Real`, which demonstrates a stronger correlation with AndroidWorld (R^2=0.934), as shown in Figure 6. Trained with EMPO on 1K trajectories, UI-S1-7B achieves SOTA among all open-source 7B models on multi-turn benchmarks. Notably, UI-S1-7B improves success rates by +12.0 on AndroidWorld and +23.8 on AITW-Gen compared to its base model (i.e., Qwen2.5VL-7B). We also validate that EMPO does not compromise single-turn capabilities (e.g., +1.9 on SS-Pro and +7.1 on GUI Odyssey).

Table 1 summarizes the key comparisons between our approach and existing training paradigms, and our contributions are listed below,

- We introduce a training paradigm **EMPO** that efficiently trains GUI Agents' multi-turn capabilities. We treat pre-collected trajectories as training experiences and design a Patch Module to extend the on-policy rollout turns.
- We incorporate discounted future returns and dual-level advantages into policy optimization to help complete multi-step tasks.
- We propose `AC-Real`, a simple yet efficient metric for better real-world evaluation.

2 Related Work

GUI Agents with Reinforcement Learning. Recent advances in multimodal models have catalyzed significant progress in GUI automation (Hu et al., 2025; Zhang et al., 2025a; Wang et al., 2025a; Tang et al., 2025b; Liu et al., 2025a; Ye et al., 2025; Wu et al., 2026b). AGUVIS (Xu et al., 2024), OS-Atlas (Wu et al., 2024), SeeClick (Cheng et al., 2024), and UI-TARS (Qin et al., 2025) leverage millions of annotated GUI elements to achieve impressive single-step accuracy, but suffer from limited OOD generalization. Inspired by the success of DeepSeek-R1 (Guo et al., 2025), recent works (Lu et al., 2025b; Luo et al., 2025a; Liu et al., 2025c) have begun applying GRPO (Shao et al., 2024) and achieve improved GUI task completion rates. However, these single-turn RL methods optimize actions independently, leading to poor multi-turn performance.

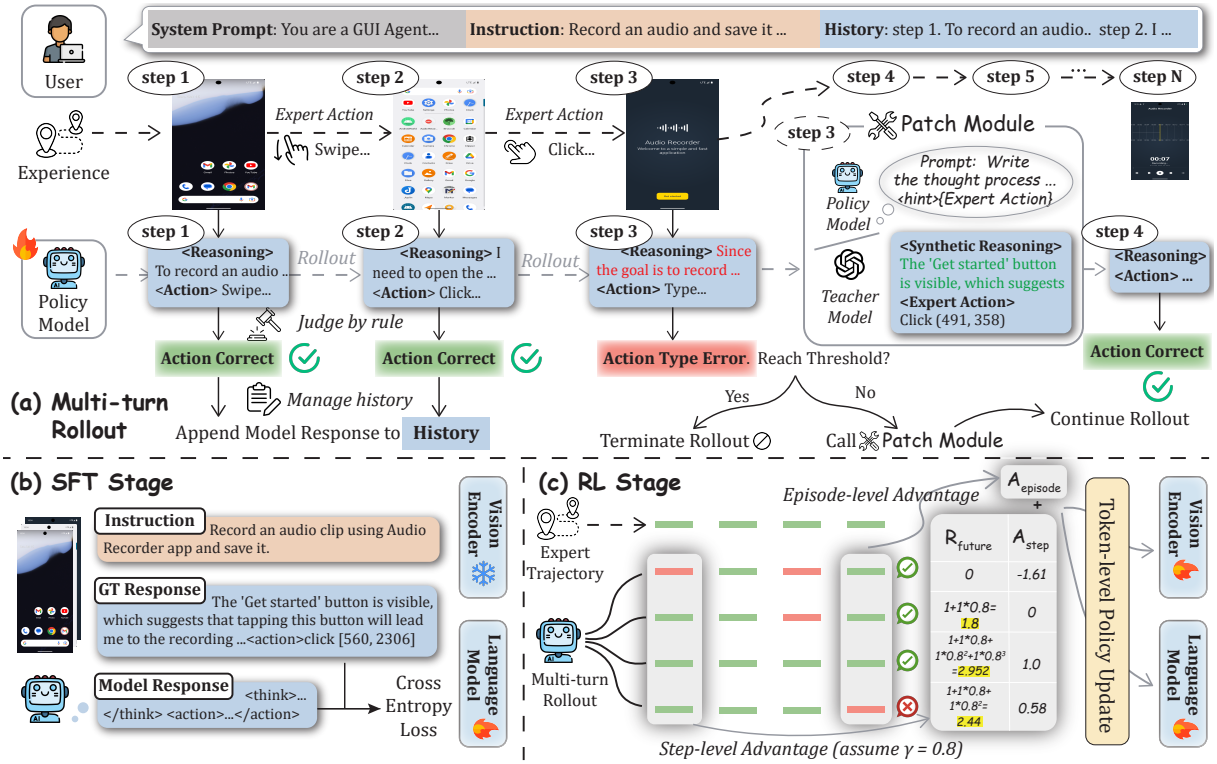

Figure 2: **Overview of EMPO**, which consist of (a) experience-driven multi-turn rollouts for RL training, (b) supervised fine-tuning (SFT) stage optimized with next-token cross-entropy loss, and (c) RL stage that computes discounted future rewards and performs dual-level advantage estimation.

Multi-turn Reinforcement Learning. Recognizing the limitations of single-step optimization, recent work has explored multi-turn RL through online environment interaction (Feng et al., 2025; Wang et al., 2025b; Dong et al., 2025; Wu et al., 2026a; Lu et al., 2026b). ARPO (Lu et al., 2025a) and MobileGUI-RL (Shi et al., 2025) extend multi-turn GRPO to computer or mobile with trajectory-aware advantages, but struggle with reward sparsity, engineering costs and training instability.

3 Method

Our approach EMPO consists of three key parts. (1) **Experience-driven Multi-turn Rollout** (Section 3.2) simulates online interaction dynamics without environmental dependencies; (2) **Patch Module** (Section 3.3) adaptively extends the on-policy history; (3) **Policy Optimization** (Section 3.4) optimizes agents through a hierarchical reward structure and dual-level advantages.

3.1 Problem Formulation

We formulate GUI automation as a multi-turn sequential decision-making problem. Given a high-level instruction I, the agent must interact with the GUI devices to complete the specified goal through a sequence of actions. At each time step t, the agent observes current state $S_t \in \mathcal{S}$ (current screenshot) and maintains a history of past interactions:

$$H_t = \{(S_1, a_1, \mathcal{T}_1), \ldots, (S_{t-1}, a_{t-1}, \mathcal{T}_{t-1})\}$$

where a_i represents the executed action (available action space shown in Table 8 and 9) and $\mathcal{T}_i$ captures the agent's reasoning process at step i. The agent then generates action sequences: $a_T, \mathcal{T}_T \sim \prod_{t=1}^{T} P_{\pi_\theta}(a_t, \mathcal{T}_t \mid S_t, H_t, I)$ where π_θ denotes current policy. The environment $\mathcal{E}$ transitions to the next state according to $S_{t+1} = \mathcal{E}(S_t, a_t)$ and rollout continues until task finish or failure.

History Construction. Single-turn RL trains on experiences where each step conditions on off-policy generated history (labeled with *):

$$H_t^* = \{(S_1^*, a_1^*), \ldots, (S_{t-1}^*, a_{t-1}^*)\}$$

Multi-turn RL instead requires agent to condition on its own generated rollouts (labeled with $^\pi$):

$$H_t^\pi = \{(S_1, a_1^\pi, \mathcal{T}_1^\pi), \ldots, (S_{t-1}, a_{t-1}^\pi, \mathcal{T}_{t-1}^\pi)\}$$

A notable performance gap exists: **Multi-turn RL consistently outperforms Single-turn RL**. We

provide a possible explanation as follows, with theoretical analysis in Appendix E.

Training with H^* induces a distributional bias in the gradient estimation, whereas constructing H^π reduces this bias and better aligns with the evaluation distribution.

3.2 Experience-driven Multi-turn Rollout

Experience. Experiences are defined as thousands of diverse, human-collected trajectories $\tau^* = \{(S_1^*, a_1^*), \ldots, (S_T^*, a_T^*)\}$. They provide optimal decision sequences across tasks and also implicitly capture the underlying environment dynamics.

Rollout. We sample N rollouts from current policy model π. The i-th candidate trajectory is

$$\tau^i = \{(S_1^i, a_1^i), \ldots, (S_T^i, a_T^i)\}, \quad i = 1, \ldots, N \tag{1}$$

To reduce bias between training and inference, agent maintains history H_t^π generated by current policy, serving as subsequent step's condition:

$$H_t^i = \{(S_1^i, a_1^i, \mathcal{T}_1^i), \ldots, (S_{t-1}^i, a_{t-1}^i, \mathcal{T}_{t-1}^i)\} \tag{2}$$

At each step, π generates action a_t^i based on this self-generated history. Environment state S steps on when actions match experiences (as defined in Appendix C):

$$S_{t+1}^i = \begin{cases} S_{t+1}^* & \text{if } r_{\text{format}} \cdot r_{\text{type}} \cdot r_{\text{acc}} = 1 \\ \text{None} & \text{otherwise} \end{cases} \tag{3}$$

However, when actions diverge, simple termination would prevent learning from the remaining trajectory steps, resulting in insufficient training.

3.3 Patch Module for Trajectory Recovery

To improve the data utilization against early termination, we introduce Patch Module $\mathcal{P}$. When a mismatch occurs at step t, the module replaces the incorrect action with the experience action a_t^* and generates synthetic reasoning $\mathcal{T}_t^{\text{patch}}$. The patched components are then integrated into the history, allowing the rollout to continue with $H_{t+1}^{patch} = H_t \cup \{(S_t, a_t^*, \mathcal{T}_t^{\text{patch}})\}$ (as detailed in Appendix D). ϵ is defined as the maximum threshold of patching times. We explore three patching strategies below: **1) Thought-Free Patch** simply injects the experience action without reasoning. **2) Off-Policy Thought Patch** uses a teacher model $\mathcal{M}_0$ (e.g., Gemini or Claude) to generate high-quality reasoning. **3) On-Policy Thought Patch** uses current policy model $\mathcal{M}$ with experience action as hints to generate reasoning content (prompt in Figure 23).

3.4 Policy Optimization

Rule-based reward. For each step t, we compute a composite reward (definitions in Appendix C):

$$\begin{aligned} r_t = 0.1 \cdot r_{\text{format}} + 0.4 \cdot \mathbb{I}_{[r_{\text{format}}=1]} \cdot r_{\text{type}} \\ + 0.5 \cdot \mathbb{I}_{[r_{\text{format}} \cdot r_{\text{type}}=1]} \cdot r_{\text{acc}} \end{aligned} \tag{4}$$

where r_{format}, r_{type}, and r_{acc} evaluate format, action type, and exact match accuracy respectively.

Future reward. Step-level rewards are insufficient for multi-turn tasks. Consider two rollouts, A and B, that both click correctly and successfully move to the next state, receiving the same reward of 1. However, **their reasoning quality differs**: A contains richer observation of the environment, which facilitate ultimate task completion; B doesn't maintain these information, and finally fails to follow the instruction (e.g., *answering a question incorrectly*). To capture such long-horizon signals, we compute discounted future returns:

$$R_t^i = \sum_{k=t}^{t_{\text{end}}} \gamma^{k-t} r_k^i, \qquad (t \le t_{\text{end}} \le T) \tag{5}$$

where γ ($0 < \gamma < 1$) controls how strongly future outcomes influence current decisions, t_{end} denotes the final step of the natural (unpatched) segment, and T is the last step of the full trajectory (with patching). In this way, A receives a higher return than B, providing a sharper and more informative training signal for policy optimization.

Dual-level advantage. Inspired by GiGPO (Feng et al., 2025), we introduce dual-level advantages that capture both immediate and future impacts:

$$A(a_t^i) = \omega_s \cdot \underbrace{\frac{R_t^i - \mu_t}{\sigma_t}}_{\texttt{step-level}} + \omega_e \cdot \underbrace{\frac{R(\tau^i) - \mu_\tau}{\sigma_\tau}}_{\texttt{episode-level}} \tag{6}$$

where $R(\tau^i)$ represents the total trajectory return and is computed as R_T^i. μ_t and σ_t denote the mean and standard deviation over all rollouts at step t.

Size	Methods	Single-turn Benchmarks									Multi-turn Benchmarks						
		ScreenSpot		AC-High			GUI Odyssey			*Avg*	AC-Real		AITW		Wob	And.	*Avg*
		V2	*Pro*	*TM*	*GR*	*SR*	*TM*	*GR*	*SR*		*PG*	*TSR*	*Gen*	*Web*			
3B	Vanilla	85.0	16.1	47.8	46.5	38.9	37.4	26.5	26.7	*41.7*	3.4	1.4	13.8	6.2	14.3	5.0	*7.4*
	SFT	**86.7**	**18.7**	53.6	53.8	42.5	36.2	28.7	23.2	*42.8*	6.2	3.7	22.9	6.8	13.4	5.8	*9.8*
	Δ	*+1.7*	*+2.6*	*+5.8*	*+7.3*	*+3.6*	*-1.2*	*+2.2*	*-3.5*	*+1.1*	*+2.8*	*+2.3*	*+9.1*	*+0.6*	*-0.9*	*+0.8*	*+2.4*
	GRPO	85.0	17.2	56.9	54.3	45.1	39.6	24.8	32.1	*44.9*	8.1	4.2	21.3	11.3	21.5	6.3	*12.1*
	Δ	*+0.0*	*+1.1*	*+9.1*	*+7.8*	*+6.2*	*+2.2*	*-1.7*	*+5.4*	*+3.2*	*+4.7*	*+2.8*	*+7.5*	*+5.1*	*+7.2*	*+1.3*	*+4.7*
	DAPO	85.2	18.0	58.9	56.2	46.3	43.2	24.6	34.6	*46.0*	8.0	4.2	23.4	10.8	21.7	8.2	*12.7*
	Δ	*+0.2*	*+1.9*	*+11.1*	*+9.7*	*+7.4*	*+5.8*	*-1.9*	*+7.9*	*+4.3*	*+4.6*	*+2.8*	*+9.6*	*+4.6*	*+7.4*	*+3.2*	*+5.3*
	EMPO	85.7	18.0	60.3	**56.5**	45.6	43.4	32.5	35.7	*46.3*	14.7	6.5	31.5	**18.7**	22.4	13.1	*17.8*
	Δ	*+0.7*	*+1.9*	*+12.5*	*+10.0*	*+6.7*	*+6.0*	*+6.0*	*+9.0*	*+4.6*	*+11.3*	*+5.1*	*+17.7*	*+12.5*	*+8.1*	*+8.1*	*+10.4*
	EMPO†	**86.7**	18.2	**61.7**	54.4	**48.7**	**45.1**	**33.7**	**37.8**	***47.9***	**16.9**	**7.1**	**35.4**	16.3	**25.7**	**14.2**	***19.3***
	Δ	*+1.7*	*+2.1*	*+13.9*	*+7.9*	*+9.8*	*+7.7*	*+7.2*	*+11.1*	*+6.2*	*+13.5*	*+5.7*	*+21.6*	*+10.1*	*+11.4*	*+9.2*	*+11.9*
7B	Vanilla	89.0	28.7	62.2	72.5	52.7	67.4	56.3	52.4	*55.7*	16.8	9.1	50.5	28.8	54.0	14.9	*29.0*
	SFT	**90.1**	29.6	66.8	**74.3**	56.1	56.9	61.5	43.2	*54.7*	17.0	9.3	58.9	28.5	46.7	21.7	*30.4*
	Δ	*+1.1*	*+0.9*	*+4.6*	*+1.8*	*+3.4*	*-10.5*	*+5.2*	*-9.2*	*-1.0*	*+0.2*	*+0.2*	*+8.4*	*-0.3*	*-7.3*	*+6.8*	*+1.4*
	GRPO	88.4	29.2	69.7	68.2	59.0	62.5	50.2	48.7	*56.3*	18.3	10.5	54.6	24.7	53.3	15.7	*29.5*
	Δ	*-0.6*	*+0.5*	*+7.5*	*-4.3*	*+6.3*	*-4.9*	*-6.1*	*-3.7*	*+0.6*	*+1.5*	*+1.4*	*+4.1*	*-4.1*	*-0.7*	*+0.8*	*+0.5*
	DAPO	89.2	29.8	71.4	68.6	61.3	68.7	54.2	53.9	*58.6*	18.9	10.4	57.6	25.2	58.3	18.7	*31.5*
	Δ	*+0.2*	*+1.1*	*+9.2*	*-3.9*	*+8.6*	*+1.3*	*-2.1*	*+1.5*	*+2.9*	*+2.1*	*+1.3*	*+7.1*	*-3.6*	*+4.3*	*+3.8*	*+2.5*
	EMPO	89.7	30.2	77.6	71.3	66.8	74.5	58.9	56.3	*60.8*	30.6	16.0	70.2	36.3	57.6	30.4	*40.2*
	Δ	*+0.7*	*+1.5*	*+15.4*	*-1.2*	*+14.1*	*+7.1*	*+2.6*	*+3.9*	*+5.1*	*+13.8*	*+6.9*	*+19.7*	*+7.5*	*+3.6*	*+15.5*	*+11.2*
	EMPO†	**90.1**	**30.6**	**79.9**	73.4	**68.2**	**76.3**	**61.7**	**59.5**	***62.1***	**32.4**	**16.3**	**74.3**	**40.2**	**60.9**	**34.0**	***43.0***
	Δ	*+1.1*	*+1.9*	*+17.7*	*+0.9*	*+15.5*	*+8.9*	*+5.4*	*+7.1*	*+6.4*	*+15.6*	*+7.2*	*+23.8*	*+11.4*	*+6.9*	*+19.1*	*+14.0*

Table 2: **Method Comparison on Single-turn and Multi-turn Benchmarks.** All RL methods are fine-tuned on the same 1K trajectories from AndroidControl using Qwen2.5VL series. EMPO† denotes EMPO initialized with SFT as a cold start. **Wob** refers to MiniWob++, and **And.** refers to AndroidWorld. ***Avg*** denotes the average success rate across benchmarks. In each column, the highest value is in **bold**, and the second highest is underlined. Δ reports the absolute increment over the Vanilla baseline, with improvements highlighted in green and degradations in red.

Training Objective. Then our EMPO optimizes the policy through the following objective:

$$\mathcal{J}_{EMPO}(\theta) = \mathbb{E}_{\substack{\{\tau^i\}_{i=1}^{N} \overset{\mathcal{P}}{\sim} \pi_{\theta_{\text{old}}}(\cdot|I) \\ \{o_{i,t}\}_{t=1}^{T} \sim \tau^i}} \frac{1}{K} \sum_{i=1}^{N} \sum_{t=1}^{T} \sum_{k=1}^{|o_{i,t}|} \min\big(\rho(\theta) A(a_t^i), \text{clip}(\rho(\theta), 1 \pm \epsilon) A(a_t^i)\big) - \beta\, D_{KL}(\pi_\theta \,\|\, \pi_{\text{ref}})$$

where $\overset{\mathcal{P}}{\sim}$ indicates our Patch Module-enhanced rollout, K is the total number of tokens, $\rho(\theta) = \frac{\pi_\theta(o_{i,t,k}|I, o_{i,t,<k})}{\pi_{\theta_{\text{old}}}(o_{i,t,k}|I, o_{i,t,<k})}$ is the importance sampling ratio, and β controls the KL penalty strength. To ensure effective learning, we enforce minimum advantage variance: $\sigma(\{A(a_t^i)\}) > \eta$, performing dynamic sampling until this diversity threshold is met.

4 Experiment

4.1 Experiment Setup

Baselines. We compare against three training paradigms using the same dataset in Table 2: **1) SFT** on about 5K expert trajectories, **2) Single-turn RL**: vanilla GRPO (Shao et al., 2024) and vanilla DAPO (Yu et al., 2025) conditioning on ground-truth history, and **3) Multi-turn RL**: ours EMPO conditioning on self-generated history. Our final model UI-S1-7B uses EMPO with only 1K RL samples and SFT as a cold start. We also compare UI-S1-7B with other advancing models in Table 3 and Table 4, including GPT-4o (Hurst et al., 2024), MobileGUI-7B (Shi et al., 2025), AgentCPM-GUI-8B (Zhang et al., 2025b), UI-TARS-7B (Qin et al., 2025) (all models are detailed in Table 14).

Single-turn Benchmarks Single-turn tasks evaluate the grounding capability and GUI Understanding capability of the end-to-end GUI model in conversations without historical context. We use ScreenSpot-V2 (Cheng et al., 2024) and ScreenSpot-Pro (Li et al., 2025) to evaluate the grounding ability. We also adopt AndroidControl-High (Li et al., 2024) and GUI Odyssey (Lu et al., 2024), for comprehensive GUI understanding evaluation under a high-level instruction. The action type match accuracy (TM), grounding accuracy rate (GR) and step success rate (SR) are reported.

Models	AC-Real		AITW		Wob	And.
	PG	*TSR*	*Gen*	*Web*	*SR*	*SR*
Closed-source Models						
Gemini-Pro-1.5*	–	–	–	–	–	22.8
Claude-CU	–	–	–	–	–	27.9
GPT-4o*	–	–	–	–	62.0	**34.5**
Open-source Models						
Qwen2-VL-2B	2.0	1.0	3.7	2.6	20.8	0.0
ShowUI-2B	6.8	2.6	9.5	6.6	27.1	7.0
UI-R1-3B	8.4	4.1	17.8	9.2	26.1	8.2
UI-R1-7B	16.9	10.8	48.7	23.3	45.2	15.1
GUI-Rise-2B	–	–	–	–	30.6	10.4
OS-Genesis-7B	7.6	3.0	14.5	7.8	19.8	17.4
OS-Atlas-7B	14.3	8.6	45.6	17.9	35.2	12.1
Qwen2.5VL-7B	17.4	9.8	49.0	20.0	54.0	22.0
AgentCPM-8B†	17.1	10.6	58.6	15.2	37.8	16.4
Aguvis-72B	–	–	–	–	**66.0**	26.1
MobileGUI-7B	–	–	65.3	22.7	–	30.0
UI-TARS-7B	28.1	14.0	64.9	28.1	58.7	33.0
Ours 7B Models						
UI-S1-7B	**32.4**	**16.3**	**74.3**	**40.2**	60.9	34.0

Table 3: **Model Comparison on Multi-turn Benchmarks.** * denotes results using Set-of-Marks (SoM). † refers to AgentCPM-GUI-8B. Claude-CU refers to Claude-Computer-Use. The best and second results are in **bold** and underline respectively.

Models	SS		AC-High			GUI Odyssey		
	V2	*Pro*	*TM*	*GR*	*SR*	*TM*	*GR*	*SR*
Closed-source Models								
GPT-4o	18.3	0.8	66.3	0.0	20.8	34.3	0.0	3.3
Claude-CU	83.0	17.1	63.7	0.0	12.5	60.9	0.0	3.1
Open-source Models								
OS-Atlas-4B	71.9	3.7	49.0	49.5	22.8	49.6	34.6	20.3
OS-Atlas-7B	84.1	18.9	57.4	54.9	29.8	60.4	39.7	27.0
Qwen2.5VL-3B	80.9	28.7	47.8	46.5	38.9	37.4	26.5	26.7
Qwen2.5VL-7B	89.0	28.7	62.2	72.5	52.7	67.4	56.3	52.4
SeeClick	55.1	1.1	82.9	62.9	59.1	71.0	52.4	53.9
UI-R1-3B	85.4	17.8	57.9	55.7	45.4	52.2	34.5	32.5
UI-R1-7B	90.0	33.5	72.4	62.8	54.2	67.1	41.3	43.5
GUI-R1-3B	85.0	28.6	58.0	56.2	46.6	54.8	41.5	41.3
GUI-R1-7B	88.2	31.3	71.6	65.6	51.7	65.5	43.6	38.8
OS-Genesis-7B	–	–	65.9	–	44.4	11.7	–	3.6
Aguvis-7B	81.8	22.9	65.6	–	54.2	26.7	–	13.5
NaviMaster-7B	–	–	72.9	–	54.0	64.4	–	36.9
PAL-UI-3B	–	–	60.4	58.7	49.3	56.7	36.9	34.6
PAL-UI-7B	–	–	71.3	70.5	57.8	65.1	46.8	41.7
UI-AGILE-3B	88.6	37.9	78.6	60.7	56.8	–	–	–
UI-AGILE-7B	**92.1**	**44.0**	80.1	61.9	60.6	–	–	37.0
AgentCPM-8B†	–	–	77.7	–	69.2	90.8	–	75.0
UI-TARS-7B	91.6	35.7	**83.7**	**80.5**	**72.5**	**94.6**	**90.1**	**87.0**
Ours 7B Models								
UI-S1-7B	90.1	30.6	79.9	73.4	68.2	76.3	61.7	59.5

Table 4: **Model Comparison on Single-turn Benchmarks.** SS refers to ScreenSpot. † refers to AgentCPM-GUI-8B. The highest value is **bolded**, the second and third are underlined.

Multi-turn Benchmarks. To evaluate end-to-end task completion requiring sequential reasoning, we introduce **AndroidControl-Real (AC-Real)**, an efficient proxy for online evaluation built on AndroidControl-Test (Li et al., 2024). Unlike AC-High which conditions on ground truth at each step, AC-Real continues with the model's own outputs, terminating only upon action mismatch. We report Progress (PG) as the average task completion ratio and Task Success Rate (TSR) as the proportion of fully completed tasks (as detailed in Appendix E). To demonstrate GUI agents' real-world performance, we also evaluate UI-S1-7B on dynamic environments including AndroidWorld (116 tasks) (Rawles et al., 2024), AITW-Gen (300 filtered tasks), AITW-Web (150 filtered tasks) (Bai et al., 2024; Shi et al., 2025), and MiniWob++ (92 tasks) (Liu et al., 2018).

4.2 Main Results

Method Comparison. The comparison between training paradigms in Table 2 reveals critical insights: **EMPO consistently outperforms all baselines on both single-turn and multi-turn benchmarks.** While SFT, GRPO and DAPO achieve modest gains on AC-High, AC-Real, and AndroidWorld, they lead to performance degradation on GUI Odyssey, AITW-Gen, and MiniWob++ relative to the base model, indicating limited generalization to complex or multi-turn GUI scenarios. In contrast, EMPO delivers substantial gains on multi-turn benchmarks, with absolute improvements of 10.4% for the 3B model and 11.2% for the 7B model, demonstrating its strong capability in long-horizon GUI task execution. Importantly, these gains do not come at the expense of single-turn performance. For instance, EMPO improves accuracy by 3.9% on the out-of-distribution GUI Odyssey benchmark and by 1.5% on ScreenSpot-Pro, outperforming DAPO, which yields only 1.5% and 1.1% improvements, respectively.

Model Comparison on Multi-turn Benchmarks. As shown in Table 3, UI-S1-7B establishes a new state-of-the-art among 7B/8B open-source models across all evaluated multi-turn benchmarks. Compared to Qwen2.5VL-7B, UI-S1-7B achieved substantial improvements: +19.1% on AndroidWorld and +23.8% on AITW-Gen. Remarkably, our UI-S1-7B outperforms strong baselines such as MobileGUI-7B and also delivers competitive results on AndroidWorld (34.0%) compared with significantly larger open-source models like Aguvis-72B (26.1%), as well as closed-source models such as GPT-4o (34.5%). Despite its enhanced planning and reasoning for UI navigation, UI-S1-7B exhibits limitations in tasks requiring precise nu-

merical computation, as detailed in Table 15.

Model Comparison on Single-turn Benchmarks. Table 4 demonstrates that UI-S1-7B maintains competitive single-turn performance. Compared to the base model, UI-S1-7B achieves consistent improvements, with gains of +15.5% on AC-High SR and +7.1% on GUI Odyssey SR. Notably, although models trained with single-turn RL (e.g., AgentCPM-GUI-8B) excel on single-turn tasks, they struggle with multi-turn execution (only 16.4% on AndroidWorld). This performance gap can be attributed to two primary factors: **(1) a mismatch between the training and the evaluation dynamics**, particularly regarding whether the the historical context is on-policy or not (see Appendix E); and **(2) overfitting to local reward signals**, leading to ignorance of global training objectives (as evidenced by the ablation of γ in Figure 10 and Table 10).

4.3 Ablation Studies and Analysis

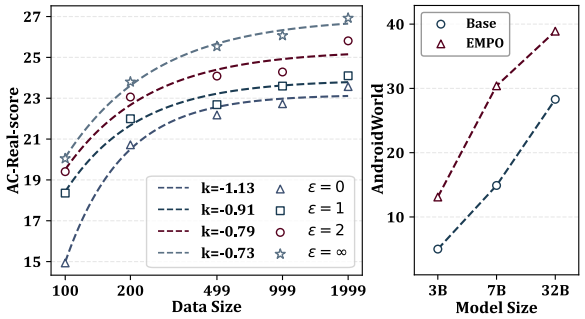

Figure 3: **Scaling Performance. Left**: Training data size scaling based on Qwen2.5VL-7B; **Right**: Model size scaling of Qwen2.5-VL series.

Data Scaling. Figure 3 (Left) reveals the data scaling performance of EMPO across different patch configurations. The performance follows an exponential scaling law $y = A + B \cdot e^{C+kx}$, where the scaling coefficient k increases with ϵ from -1.13 to -0.73. This indicates that larger ϵ values not only improve absolute performance but also enhance data efficiency, enabling more effective learning from each training sample.

Model Size Scaling. As shown in Figure 3 (Right), our framework exhibits strong scalability. When scaling from 3B to 32B parameters, EMPO consistently expands its performance improvements over the Qwen2.5-VL baseline.

Generalization Analysis. To evaluate EMPO's generalization ability, we extend our training framework to computer-use scenarios and conduct cross-domain validation in Table 5. When training Qwen2.5VL-7B on 1K AgentNet (Wang et al., 2025a) data (computer use), EMPO yields larger improvements over the base model *(+14.2)* than GRPO *(+8.6)* on the in-domain (ID) OSWorld benchmark (Xie et al., 2024). For OOD web-use validation, the GRPO-trained model exhibits degradation *(-2.9)*, whereas EMPO achieves consistent gains *(+2.5)*. Similarly, when trained on AndroidControl (Li et al., 2024), EMPO delivers stable improvements across both ID and OOD settings, while GRPO shows only slight gains.

Method	OSWorld①		And.②		Wob③	
	Acc	Δ	*Acc*	Δ	*Acc*	Δ
Vanilla	2.3	–	14.9	–	54.0	–
*Training on **AgentNet** (domain ①)*						
GRPO	10.9	*+8.6*	16.3	*+1.4*	51.1	*-2.9*
EMPO	17.1	*+14.2*	15.9	*+1.0*	56.5	*+2.5*
*Training on **AndroidControl** (domain ②)*						
GRPO	3.9	*+1.6*	15.7	*+0.8*	53.3	*-0.7*
EMPO	4.1	*+1.8*	30.4	*+15.5*	57.6	*+3.6*

Table 5: **Cross-Domain Generalization Validation.** OSWorld refers to OSWorld-Verified (15 steps). We categorize benchmark and dataset domains into Computer Use (①), Mobile Use (②), and Web Use (③), where domain ③ serves as OOD validation.

Method	Config	AC-Real			AW
		PG	TSR	Score	SR
On-policy patch	ϵ=∞	**34.4**	**17.8**	**26.1**	**34.5**
On-policy patch	ϵ=1	32.9	16.7	24.8	32.8
Off-policy patch	ϵ=∞	31.8	13.3	22.6	24.0
Off-policy patch	ϵ=1	29.5	12.0	20.8	24.6
Thought-free patch	ϵ=∞	**34.4**	17.0	25.7	**34.5**
Thought-free patch	ϵ=1	32.4	16.3	24.4	34.0
w/o dynamic sampling	η=0	30.3	14.7	22.5	33.0
w/o cold start	/	30.6	16.0	23.3	30.4
w/o Patch Module	ϵ=0	29.6	15.0	22.3	30.0
w/o episode-level adv	ω_e=0	33.5	15.8	24.7	31.2
w/o step-level adv	ω_s=0	25.8	11.2	18.5	21.3
w/o future reward	γ=0	28.7	13.4	21.1	27.6
w/o multi-turn rollout	/	20.6	10.3	15.7	18.6

Table 6: **Ablation Study Results.** All experiments are done using 1K training samples on Qwen2.5VL-7B.

Ablation on EMPO. We present ablation results in Table 6. For the Patch Module, the on-policy thought patch and a larger ϵ consistently lead to better performance, as more on-policy content is better aligned with the evaluation distribution. Other ablations on key designs are conducted using the thought-free patch with $\epsilon = 1$. EMPO outperforms all ablation variants on both AC-Real and AW, val-

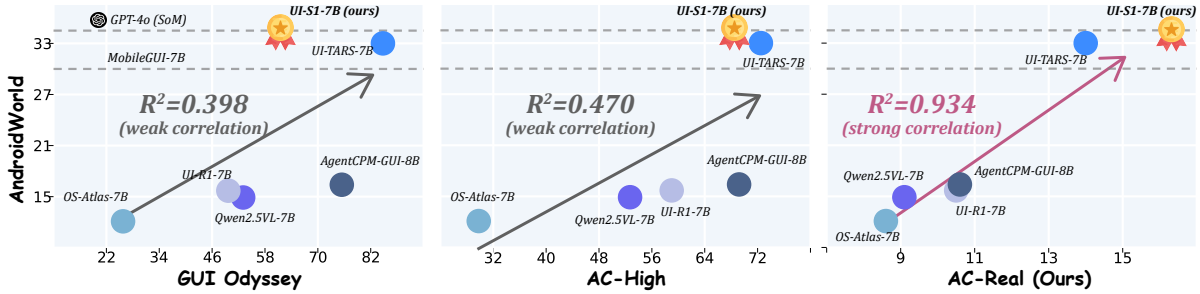

Figure 4: **Metric comparison. Left & Middle**: Prior GUI Odyssey and AC-High demonstrate weak correlation (R^2=0.398 & 0.470) with dynamic metric AndroidWorld (AW). **Right**: Our proposed AC-Real shows stronger correlation (R^2=0.934), while ours UI-S1-7B achieves superior performance on all the four metrics.

Figure 5: **Case Comparison across Training Paradigms (SFT, DAPO, and EMPO) on the Vanilla Model (Qwen2.5VL-7B).** The task is a cross-application, multi-step instruction from AndroidWorld: *"Create a file in Markor, called receipt.md with the transactions from the receipt.png. Use Simple Gallery to view the receipt. Please enter transactions in csv format including the header 'Date, Item, Amount'."*

idating the essential of our choices, including the Patch Module, dynamic sampling, future reward, dual-level advantage, and multi-turn rollout.

4.4 Analysis of AC-Real

Figure 4 compares AC-Real with existing GUI benchmarks, including GUI Odyssey, AC-High, and AndroidWorld. AC-Real exhibits stronger correlation with AndroidWorld (R^2 = 0.934) than AC-High (R^2 = 0.470) and GUI Odyssey (R^2 = 0.398). This advantage stems from the fact that both AC-Real and AndroidWorld evaluate full task completion over multi-turn interactions, whereas other methods assess agent performance under single-turn dialogue settings. As further evidenced by Figure 7, AC-Real could serve as an effective proxy for real-world evaluation, bridging the critical gap between fast but less realistic offline metrics and accurate yet costly online testing.

5 Case Study

Figure 5 presents a challenging AndroidWorld case that requires information retention and coherent decision-making under different training paradigms. We observe the following behaviors: **(1) The vanilla and GRPO models exhibit action–thought inconsistency.** For instance, the GRPO model prematurely terminates after planning to navigate to a subsequent application, likely due to neglecting global training objectives. **(2) The SFT model loses critical information and performs redundant actions**, such as attempting to create a file that already exists. **(3) In contrast, our model consistently retains essential information throughout the 12-step interaction**, correctly recording *"2023-03-23, Monitor Stand, $33.22"* in

CSV format. These results highlight our model's ability to learn robust multi-turn behaviors with coherent reasoning–action alignment. Additional qualitative examples are provided from Figure 15 to Figure 20, with a detailed analysis in Appendix J.

6 Conclusion

In this work, we present Experience-driven Multi-turn Policy Optimization (EMPO), a novel training paradigm which simulates stable multi-turn RL for GUI automation agents without environmental dependencies. Experimental evaluation shows that our UI-S1-7B achieves state-of-the-art results among open-source 7B-scale models and substantial improvements across both single-turn and multi-turn benchmarks. Our findings highlight the promise of EMPO as an effective and scalable training framework for real-world GUI agents.

Limitations

Under our current training setup, trajectories are relatively short, with most consisting of 5 to 15 interaction steps. As a result, more complex tasks requiring over 20 steps remain challenging. Moreover, tasks that demand long-horizon memory are also un-solved for the limited context window of 7B-scale vision–language models. Future work may focus on these limitations by incorporating explicit memory mechanisms or extending the effective context length through architectural or training-time improvements.

Acknowledgment

This work was supported by National Natural Science Foundation of China (No. 62506332), National Natural Science Foundation of China (No. 62436007), and "Pioneer" and "Leading Goose" R&D Program of Zhejiang (NO. 2026C02A1223).

References

Anthropic. 2024. Developing a computer use model. https://www.anthropic.com/news/developing-computer-use.

Hao Bai, Yifei Zhou, Jiayi Pan, Mert Cemri, Alane Suhr, Sergey Levine, and Aviral Kumar. 2024. Digirl: Training in-the-wild device-control agents with autonomous reinforcement learning. *Advances in Neural Information Processing Systems*, 37:12461–12495.

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, and 1 others. 2025. Qwen2. 5-vl technical report. *arXiv preprint arXiv:2502.13923*.

Yuxiang Chai, Siyuan Huang, Yazhe Niu, Han Xiao, Liang Liu, Dingyu Zhang, Shuai Ren, and Hongsheng Li. 2024. Amex: Android multi-annotation expo dataset for mobile gui agents. *arXiv preprint arXiv:2407.17490*.

Tongbo Chen, Zhengxi Lu, Zhan Xu, Guocheng Shao, Shaohan Zhao, Fei Tang, Yong Du, Kaitao Song, Yizhou Liu, Yuchen Yan, Wenqi Zhang, Xu Tan, Weiming Lu, Jun Xiao, Yueting Zhuang, and Yongliang Shen. 2026. Knowu-bench: Towards interactive, proactive, and personalized mobile agent evaluation. *Preprint*, arXiv:2604.08455.

Kanzhi Cheng, Qiushi Sun, Yougang Chu, Fangzhi Xu, Yantao Li, Jianbing Zhang, and Zhiyong Wu. 2024. Seeclick: Harnessing gui grounding for advanced visual gui agents. *arXiv preprint arXiv:2401.10935*.

Guanting Dong, Hangyu Mao, Kai Ma, Licheng Bao, Yifei Chen, Zhongyuan Wang, Zhongxia Chen, Jiazhen Du, Huiyang Wang, Fuzheng Zhang, and 1 others. 2025. Agentic reinforced policy optimization. *arXiv preprint arXiv:2507.19849*.

Yong Du, Yuchen Yan, Fei Tang, Zhengxi Lu, Chang Zong, Weiming Lu, Shengpei Jiang, and Yongliang Shen. 2025. Test-time reinforcement learning for gui grounding via region consistency. *arXiv preprint arXiv:2508.05615*.

Lang Feng, Zhenghai Xue, Tingcong Liu, and Bo An. 2025. Group-in-group policy optimization for llm agent training. *arXiv preprint arXiv:2505.10978*.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, and 1 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Xueyu Hu, Tao Xiong, Biao Yi, Zishu Wei, Ruixuan Xiao, Yurun Chen, Jiasheng Ye, Meiling Tao, Xiangxin Zhou, Ziyu Zhao, and 1 others. 2025. Os agents: A survey on mllm-based agents for general computing devices use. *arXiv preprint arXiv:2508.04482*.

Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, and 1 others. 2024. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*.

Kaixin Li, Ziyang Meng, Hongzhan Lin, Ziyang Luo, Yuchen Tian, Jing Ma, Zhiyong Huang, and Tat-Seng Chua. 2025. Screenspot-pro: Gui grounding for professional high-resolution computer use. *arXiv preprint arXiv:2504.07981*.

Wei Li, William Bishop, Alice Li, Chris Rawles, Folawiyo Campbell-Ajala, Divya Tyamagundlu, and Oriana Riva. 2024. On the effects of data scale on computer control agents. *arXiv e-prints*, pages arXiv–2406.

Shuquan Lian, Yuhang Wu, Jia Ma, Yifan Ding, Zihan Song, Bingqi Chen, Xiawu Zheng, and Hui Li. 2025. Ui-agile: Advancing gui agents with effective reinforcement learning and precise inference-time grounding. *arXiv preprint arXiv:2507.22025*.

Kevin Qinghong Lin, Linjie Li, Difei Gao, Zhengyuan Yang, Shiwei Wu, Zechen Bai, Weixian Lei, Lijuan Wang, and Mike Zheng Shou. 2024. Showui: One vision-language-action model for gui visual agent. *Preprint*, arXiv:2411.17465.

Evan Zheran Liu, Kelvin Guu, Panupong Pasupat, Tianlin Shi, and Percy Liang. 2018. Reinforcement learning on web interfaces using workflow-guided exploration. *arXiv preprint arXiv:1802.08802*.

Guangyi Liu, Pengxiang Zhao, Liang Liu, Yaxuan Guo, Han Xiao, Weifeng Lin, Yuxiang Chai, Yue Han, Shuai Ren, Hao Wang, and 1 others. 2025a. Llm-powered gui agents in phone automation: Surveying progress and prospects. *arXiv preprint arXiv:2504.19838*.

Tao Liu, Chongyu Wang, Rongjie Li, Yingchen Yu, Xuming He, and Bai Song. 2025b. Gui-rise: Structured reasoning and history summarization for gui navigation. *arXiv preprint arXiv:2510.27210*.

Yuhang Liu, Pengxiang Li, Congkai Xie, Xavier Hu, Xiaotian Han, Shengyu Zhang, Hongxia Yang, and Fei Wu. 2025c. Infigui-r1: Advancing multimodal gui agents from reactive actors to deliberative reasoners. *arXiv preprint arXiv:2504.14239*.

Zikang Liu, Junyi Li, Wayne Xin Zhao, Dawei Gao, Yaliang Li, and Ji-rong Wen. 2025d. Pal-ui: Planning with active look-back for vision-based gui agents. *arXiv preprint arXiv:2510.00413*.

Fanbin Lu, Zhisheng Zhong, Shu Liu, Chi-Wing Fu, and Jiaya Jia. 2025a. Arpo: End-to-end policy optimization for gui agents with experience replay. *arXiv preprint arXiv:2505.16282*.

Quanfeng Lu, Wenqi Shao, Zitao Liu, Fanqing Meng, Boxuan Li, Botong Chen, Siyuan Huang, Kaipeng Zhang, Yu Qiao, and Ping Luo. 2024. Gui odyssey: A comprehensive dataset for cross-app gui navigation on mobile devices. *arXiv preprint arXiv:2406.08451*.

Zhengxi Lu, Yuxiang Chai, Yaxuan Guo, Xi Yin, Liang Liu, Hao Wang, Han Xiao, Shuai Ren, Guanjing Xiong, and Hongsheng Li. 2025b. Ui-r1: Enhancing efficient action prediction of gui agents by reinforcement learning. *arXiv preprint 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. 2026a. Ui-copilot: Advancing long-horizon gui automation via tool-integrated policy optimization. *Preprint*, arXiv:2604.13822.

Zhengxi Lu, Zhiyuan Yao, Jinyang Wu, Chengcheng Han, Qi Gu, Xunliang Cai, Weiming Lu, Jun Xiao, Yueting Zhuang, and Yongliang Shen. 2026b. Skill0: In-context agentic reinforcement learning for skill internalization. *arXiv preprint arXiv:2604.02268*.

Run Luo, Lu Wang, Wanwei He, and Xiaobo Xia. 2025a. Gui-r1: A generalist r1-style vision-language action model for gui agents. *arXiv preprint arXiv:2504.10458*.

Zhihao Luo, Wentao Yan, Jingyu Gong, Min Wang, Zhizhong Zhang, Xuhong Wang, Yuan Xie, and Xin Tan. 2025b. Navimaster: Learning a unified policy for gui and embodied navigation tasks. *arXiv preprint arXiv:2508.02046*.

Yujia Qin, Yining Ye, Junjie Fang, Haoming Wang, Shihao Liang, Shizuo Tian, Junda Zhang, Jiahao Li, Yunxin Li, Shijue Huang, and 1 others. 2025. Ui-tars: Pioneering automated gui interaction with native agents. *arXiv preprint arXiv:2501.12326*.

Christopher Rawles, Sarah Clinckemaillie, Yifan Chang, Jonathan Waltz, Gabrielle Lau, Marybeth Fair, Alice Li, William Bishop, Wei Li, Folawiyo Campbell-Ajala, and 1 others. 2024. Androidworld: A dynamic benchmarking environment for autonomous agents. *arXiv preprint arXiv:2405.14573*.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Y Wu, and 1 others. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.

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

Qiushi Sun, Kanzhi Cheng, Zichen Ding, Chuanyang Jin, Yian Wang, Fangzhi Xu, Zhenyu Wu, Chengyou Jia, Liheng Chen, Zhoumianze Liu, and 1 others. 2024. Os-genesis: Automating gui agent trajectory construction via reverse task synthesis. *arXiv preprint arXiv:2412.19723*.

Fei Tang, Zhangxuan Gu, Zhengxi Lu, Xuyang Liu, Shuheng Shen, Changhua Meng, Wen Wang, Wenqi Zhang, Yongliang Shen, Weiming Lu, and 1 others. 2025a. Gui-g^2: Gaussian reward modeling for gui grounding. *arXiv preprint arXiv:2507.15846*.

Fei Tang, Haolei Xu, Hang Zhang, Siqi Chen, Xingyu Wu, Yongliang Shen, Wenqi Zhang, Guiyang Hou, Zeqi Tan, Yuchen Yan, and 1 others. 2025b. A survey on (m) llm-based gui agents. *arXiv preprint arXiv:2504.13865*.

Gemini Team, Petko Georgiev, Ving Ian Lei, Ryan Burnell, Libin Bai, Anmol Gulati, Garrett Tanzer, Damien Vincent, Zhufeng Pan, Shibo Wang, and 1 others. 2024. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint arXiv:2403.05530*.

Peng Wang, Shuai Bai, Sinan Tan, Shijie Wang, Zhihao Fan, Jinze Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, and 1 others. 2024. Qwen2-vl: Enhancing vision-language model's perception of the world at any resolution. *arXiv preprint arXiv:2409.12191*.

Xinyuan Wang, Bowen Wang, Dunjie Lu, Junlin Yang, Tianbao Xie, Junli Wang, Jiaqi Deng, Xiaole Guo, Yiheng Xu, Chen Henry Wu, and 1 others. 2025a. Opencua: Open foundations for computer-use agents. *arXiv preprint arXiv:2508.09123*.

Zihan Wang, Kangrui Wang, Qineng Wang, Pingyue Zhang, Linjie Li, Zhengyuan Yang, Xing Jin, Kefan Yu, Minh Nhat Nguyen, Licheng Liu, and 1 others. 2025b. Ragen: Understanding self-evolution in llm agents via multi-turn reinforcement learning. *arXiv preprint arXiv:2504.20073*.

Jinyang Wu, Shuo Yang, Changpeng Yang, Yuhao Shen, Shuai Zhang, Zhengqi Wen, and Jianhua Tao. 2026a. Spark: Strategic policy-aware exploration via dynamic branching for long-horizon agentic learning. *arXiv preprint arXiv:2601.20209*.

Zheng Wu, Pengzhou Cheng, Zongru Wu, Lingzhong Dong, and Zhuosheng Zhang. 2026b. Gem: Gaussian embedding modeling for out-of-distribution detection in gui agents. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 33989–33997.

Zhiyong Wu, Zhenyu Wu, Fangzhi Xu, Yian Wang, Qiushi Sun, Chengyou Jia, Kanzhi Cheng, Zichen Ding, Liheng Chen, Paul Pu Liang, and 1 others. 2024. Os-atlas: A foundation action model for generalist gui agents. *arXiv preprint arXiv:2410.23218*.

Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh J Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, and 1 others. 2024. Osworld: Benchmarking multimodal agents for open-ended tasks in real computer environments. *Advances in Neural Information Processing Systems*, 37:52040–52094.

Yiheng Xu, Zekun Wang, Junli Wang, Dunjie Lu, Tianbao Xie, Amrita Saha, Doyen Sahoo, Tao Yu, and Caiming Xiong. 2024. Aguvis: Unified pure vision agents for autonomous gui interaction. *arXiv preprint arXiv:2412.04454*.

Jiabo Ye, Xi Zhang, Haiyang Xu, Haowei Liu, Junyang Wang, Zhaoqing Zhu, Ziwei Zheng, Feiyu Gao, Junjie Cao, Zhengxi Lu, and 1 others. 2025. Mobile-agent-v3: Foundamental agents for gui automation. *arXiv preprint arXiv:2508.15144*.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, and 1 others. 2025. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*.

Chi Zhang, Zhao Yang, Jiaxuan Liu, Yanda Li, Yucheng Han, Xin Chen, Zebiao Huang, Bin Fu, and Gang Yu. 2025a. Appagent: Multimodal agents as smartphone users. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems*, pages 1–20.

Zhong Zhang, Yaxi Lu, Yikun Fu, Yupeng Huo, Shenzhi Yang, Yesai Wu, Han Si, Xin Cong, Haotian Chen, Yankai Lin, and 1 others. 2025b. Agentcpm-gui: Building mobile-use agents with reinforcement fine-tuning. *arXiv preprint arXiv:2506.01391*.

A Notation Definition

Symbol	Description
$\mathcal{M}_0$	Assist model used for thought patching
$\mathcal{M}$	Policy model
prompt	Prompt for thought generation (Appendix **??**)
a_t	Predicted action at step t
a_t^*	Expert action at step t
$\mathcal{T}_t$	Thought representation at step t
$\mathcal{F}$	Patch function that corrects action and thought
I	High-level GUI task instruction
S_t	Current observation (e.g., screenshot) at step t
H_t	History up to step t including $(S, a, \mathcal{T})$
r_{format}	Binary score for correct output format
r_{type}	Binary score for predicted action type
r_{acc}	Binary score for exact action
r_t	Step-wise reward at time t
$\mathbb{I}_{[\cdot]}$	Indicator function (equals 1 if condition is true)
γ	Discount factor for return computation
R_t^i	Discounted return of i-th trajectory at step t
N	Number of trajectories sampled in a batch
$A^S(\boldsymbol{a}_t^i)$	Step-level advantage for action $\boldsymbol{a}_t^i$
$A^E(\boldsymbol{\tau}^i)$	Episode-level advantage for trajectory τ^i
$R(\tau^{(j)})$	Episode return of trajectory j
t_{end}	Last step of a natural trajectory segment
T	Last step index of a trajectory
$\sigma(\cdot)$	Standard deviation function
ω_e	Weight of episode-level advantages
ω_s	Weight of step-level advantages
$A(\boldsymbol{a}_t^i)$	Combined group-in-group advantage
K	Total number of tokens in the current batch
$o_{i,t}$	Model output tokens at step t of trajectory i
$o_{i,t,k}$	k-th token of $o_{i,t}$
q	Conditioning input (e.g., instruction and history)
$\rho(\theta)$	Importance sampling ratio between policies
θ	Current policy parameters
θ_{old}	Policy parameters before update (rollout policy)
π_{ref}	Reference policy for KL regularization
β	Coefficient for KL divergence penalty
η	Minimum standard deviation threshold in DAPO

Table 7: Notation Definition in Section 2.

B Action Space

Action Type	Description
`click`	Tap a specific coordinate (x, y) on the screen.
`long_press`	Press and hold at (x, y) for a specified duration.
`swipe`	Perform a swipe gesture from (x_1, y_1) to (x_2, y_2).
`answer`	Output a textual answer to the task.
`type`	Enter text into the currently focused input field.
`system_button`	Trigger a system-level button (e.g., Home, Back).
`open`	Launch an APP on the device.
`wait`	Pause execution for a given number of seconds to allow UI changes.
`terminate`	Stop execution and report task success or failure.

Table 8: Action space in AndroidWorld automation.

Action Type	Description
`key`	Press one or multiple keyboard keys in order, and release them then.
`type`	Input a text string into the active text field.
`mouse_move`	Move the cursor to a target screen coordinate (x, y).
`click`	Perform a left-click at coordinate (x, y).
`drag`	Hold the left mouse button at (x_1, y_1) and drag to (x_2, y_2).
`right_click`	Perform a right-click at (x, y).
`middle_click`	Perform a middle-click at coordinate (x, y).
`double_click`	Perform a double-click at (x, y).
`scroll`	Scroll the mouse wheel upward or downward.
`wait`	Pause execution for a specified number of seconds.
`terminate`	Terminate the task and output the final success or failure status.

Table 9: Action space in OSWorld automation.

C Reward Definition

- **Type Reward (r_{type})**

  $r_{\text{type}} \in \{0, 1\}$ indicates whether the predicted action type matches the ground-truth action type. Let a^{pred} and a^{gt} denote the predicted and ground-truth action types, respectively. The type reward is defined as $r_{\text{type}} = \mathbb{I}[a^{\text{pred}} = a^{\text{gt}}]$.

- **Accuracy Reward (r_{acc})**

  $r_{\text{acc}} \in \{0, 1\}$ evaluates whether the predicted action is accurate given the ground-truth action, conditioned on the action type being correct and after coordinate normalization. Let $\mathbf{p}^{\text{pred}}$ and $\mathbf{p}^{\text{gt}}$ denote the predicted and ground-truth coordinates, respectively.

  - **Wait / Terminate**
    The prediction is accurate if the action type exactly matches: $r_{\text{acc}} = \mathbb{I}[a^{\text{pred}} = a^{\text{gt}}]$.
  - **System Button**
    Let $\text{button}^{\text{pred}}$ and $\text{button}^{\text{gt}}$ denote the predicted and ground-truth system button names. The prediction is accurate if the button names match in a case-insensitive manner: $r_{\text{acc}} = \mathbb{I}[\text{button}^{\text{pred}} = \text{button}^{\text{gt}}]$.
  - **Type / Answer / Key / Open.** Let $\text{text}^{\text{pred}}$ and text^{gt} denote the predicted and ground-truth input strings. The prediction is accurate if the texts match under relaxed string matching: $r_{\text{acc}} = \mathbb{I}[\text{text}^{\text{pred}} \sim \text{text}^{\text{gt}}]$.
  - **Swipe**
    Let $\mathbf{p}_1^{\text{pred}}, \mathbf{p}_2^{\text{pred}}$ denote the start and end points of the predicted swipe, and $\text{dir}(\cdot, \cdot)$ be the function that infers swipe direction. The prediction is accurate if the inferred swipe direction matches the ground truth: $r_{\text{acc}} = \mathbb{I}[\text{dir}(\mathbf{p}_1^{\text{pred}}, \mathbf{p}_2^{\text{pred}}) = \text{dir}^{\text{gt}}]$.
  - **Click / Long Press**
    Let $\mathcal{B}^{\text{gt}}$ denote the enlarged ground-truth bounding box and ϵ be a fixed distance threshold. The prediction is accurate if the predicted point falls inside the bounding box or is sufficiently close to the ground-truth point: $r_{\text{acc}} = \mathbb{I}[\mathbf{p}^{\text{pred}} \in \mathcal{B}^{\text{gt}} \vee \|\mathbf{p}^{\text{pred}} - \mathbf{p}^{\text{gt}}\|_2 \leq \epsilon]$.

D Algorithm

Our experience-driven multi-turn rollout algorithm with a Patch Module is as follows,

Algorithm 1 Multi-turn Rollout

Input:
$\pi_{\theta_{\text{old}}}$: initial policy model
$\tau^* = \{(S_1^*, a_1^*), \ldots, (S_T^*, a_T^*)\}$: experience trajectory
Output: $\tau = \{(S_1, a_1), (S_2, a_2), \ldots\}$: rollout trajectory
Initialize $H_1 \leftarrow \varnothing$, $\tau \leftarrow \varnothing$, $c \leftarrow 0$, $S_1 \leftarrow S_1^*$
for $t = 1$ **to** T **do**
  $a_t, \mathcal{T}_t \sim \pi_{\theta_{\text{old}}}(\cdot \mid S_t, H_t)$
  $a_t^*, S_{t+1}^* \sim \tau^*$
  `# Patch Module:`
  if $a_t = a_t^*$ **then**
  else if $c < \epsilon$ **then**
    $a_t^{\text{patch}}, \mathcal{T}_t^{\text{patch}} \leftarrow \mathcal{F}(a_t, \mathcal{T}_t)$
    $c \leftarrow c + 1$
  else
    $\tau \leftarrow \tau \cup (S_t, a_t)$
    break
  end if
  if $S_{t+1} = \text{NONE}$ **then**
    break
  end if
  $S_{t+1} \leftarrow S_{t+1}^*$
  $H_{t+1} \leftarrow H_t \cup \{(S_t, a_t^{\text{patch}}, \mathcal{T}_t^{\text{patch}})\}$
  $\tau \leftarrow \tau \cup (S_t, a_t^{\text{patch}})$
  $H_t \leftarrow H_{t+1}$, $S_t \leftarrow S_{t+1}$
end for
Output: τ

E Theoretical Analysis

Setup. Let $\pi(\cdot \mid \theta)$ denote the training policy and $\mu(\cdot \mid \theta)$ the rollout (inference) policy. Given a high-level instruction I, the deployment objective is

$$\mathcal{J}(\theta) = \mathbb{E}_{I \sim p_{\mathcal{I}}}\Big[\mathbb{E}_{(a_{1:T}, \mathcal{T}_{1:T}) \sim \mu(\cdot \mid I)}\big[R(I, a_{1:T}, \mathcal{T}_{1:T})\big]\Big].$$

Single-turn Training Mismatch. In single-turn (ST) training, each step conditions on off-policy histories H_t^*, yielding

$$\widehat{\nabla_\theta \mathcal{J}}_* = \mathbb{E}_{I \sim p_{\mathcal{I}}, a_t \sim \pi(\cdot \mid I, S_t, H_t^*)} \Big[\nabla_\theta \log \pi(a_t \mid I, S_t, H_t^*)\, R(I, a_{1:T}, \mathcal{T}_{1:T})\Big].$$

Evaluation, however, uses self-generated histories H_t^π: $a_t^\pi = P_\mu(\cdot \mid I, S_t, H_t^\pi)$, so that

$$\widehat{\nabla_\theta \mathcal{J}}_* \neq \nabla_\theta \mathbb{E}_{(a_{1:T}^\pi, \mathcal{T}_{1:T}^\pi) \sim \mu}[R(I, a_{1:T}^\pi, \mathcal{T}_{1:T}^\pi)].$$

Equivalently,

$$\arg\max_\theta \mathbb{E}_{a_{1:T} \sim \pi}[R(I, a_{1:T})] \;\neq\; \arg\max_\theta \mathbb{E}_{a_{1:T}^\pi \sim \mu}[R(I, a_{1:T}^\pi)]$$

which illustrates the biased gradient and deployment gap.

Multi-turn Training Alignment. Multi-turn (MT) training conditions on self-generated histories H_t^π at each step, producing full trajectories $(a_{1:T}^\pi, \mathcal{T}_{1:T}^\pi) = P_\mu(\cdot \mid I)$ and the gradient estimator

$$\widehat{\nabla_\theta \mathcal{J}}_\pi = \mathbb{E}_{I \sim p_{\mathcal{I}}}\Big[\sum_{t=1}^{T} \nabla_\theta \log \mu(a_t^\pi \mid I, S_t, H_t^\pi)\, R(I, a_{1:T}^\pi, \mathcal{T}_{1:T}^\pi)\Big].$$

By aligning training histories with rollout histories, MT training better approximates the deployment objective:

$$\arg\max_\theta \mathbb{E}_{(a_{1:T}^\pi, \mathcal{T}_{1:T}^\pi) \sim \mu}[R(I, a_{1:T}^\pi)] \approx \arg\max_\theta \mathcal{J}(\theta),$$

reducing the train–inference mismatch.

Conclusion. Single-turn training suffers from biased gradients due to off-policy histories H_t^*, whereas multi-turn training uses self-generated histories H_t^π, leading to more consistent and stable optimization toward deployment-time performance.

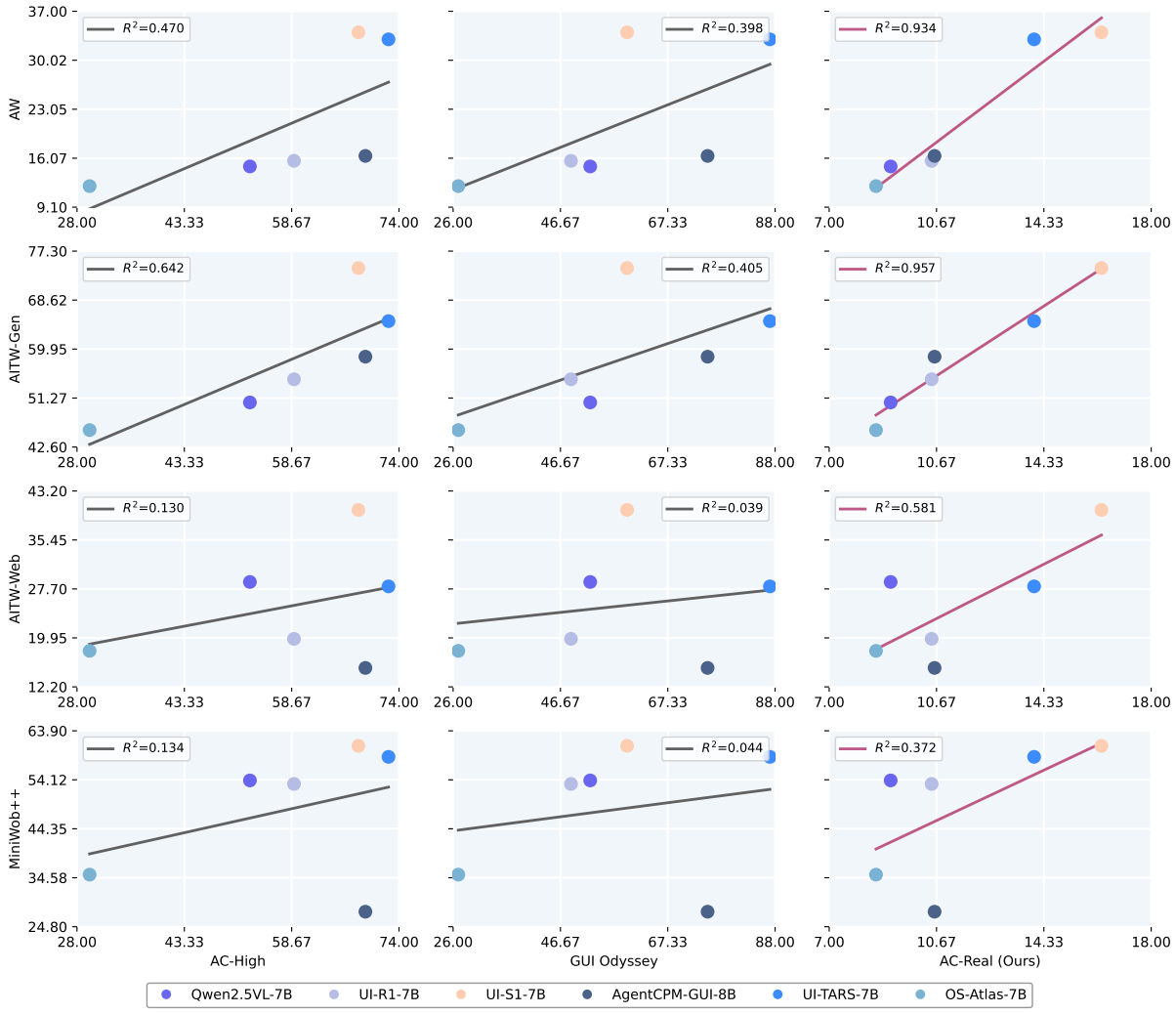

Figure 6: **Overall Comparisons of Dynamic Metrics (AW, AITW-Gen, AITW-Web, MiniWob++) with Static Metrics (AC-High, GUI Odyssey, `AC-Real`). Left**: AC-High demonstrates weak correlation with dynamic benchmarks. **Middle**: GUI Odyssey demonstrates weak correlation with dynamic benchmarks. **Right**: Ours proposed `AC-Real` shows stronger correlation.

F AC-Real

Definition Let N be the total number of tasks. For the i-th task, let s_i denote the number of successful steps, and t_i denote the total number of steps in its expert trajectory. We define the following metrics of `AC-Real`: $\text{PG} = \frac{1}{N}\sum_{i=1}^{N}\frac{s_i}{t_i}$, $\text{TSR} = \frac{1}{N}\sum_{i=1}^{N}\mathbb{I}[s_i = t_i]$, and $\text{Score} = \frac{\text{PG+TSR}}{2}$. Here, $\mathbb{I}[\cdot]$ is the indicator function, which equals 1 if the condition inside the brackets is true and 0 otherwise.

[Exp1] Alignment with Real-world Metrics We also compare more metrics with `AC-Real` in Figure 6, which demonstrates `AC-Real`'s strong correlation with real-world metrics.

For the linear regression analyses in Figure 6, the coefficient of determination, denoted as R^2, is defined as $R^2 = 1 - \frac{\text{SS}_{\text{res}}}{\text{SS}_{\text{tot}}}$, where SS_{res} (Residual Sum of Squares) is $\text{SS}_{\text{res}} = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2$, and SS_{tot} (Total Sum of Squares) is $\text{SS}_{\text{tot}} = \sum_{i=1}^{n}(y_i - \bar{y})^2$. Here, n is the number of observations; y_i is the observed value of the dependent variable for the i-th data point; $\hat{y}_i$ is the corresponding predicted value from the regression model; and $\bar{y}$ is the mean of all observed values. The R^2 metric ranges from 0 to 1 and represents the proportion of variance in the dependent variable explained by the independent variable(s)—higher values indicate a better fit.

[Exp2] Multi-dimension Validation Figure 7 validates `AC-Real` as an effective proxy for real-world evaluation. We compare three evaluation paradigms across efficiency (inverse time cost), diversity (number of tasks), and correlation with

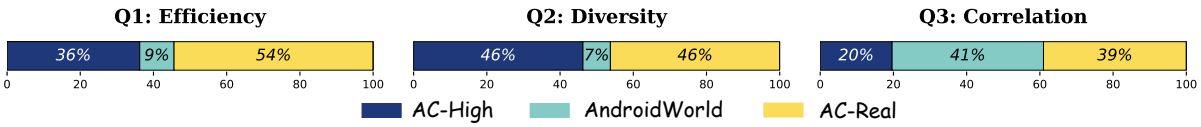

Figure 7: **Metric Comparison across Three Dimensions**: efficiency (inverse rollout time cost), diversity (number of tasks), and correlation (vs real-world performance).

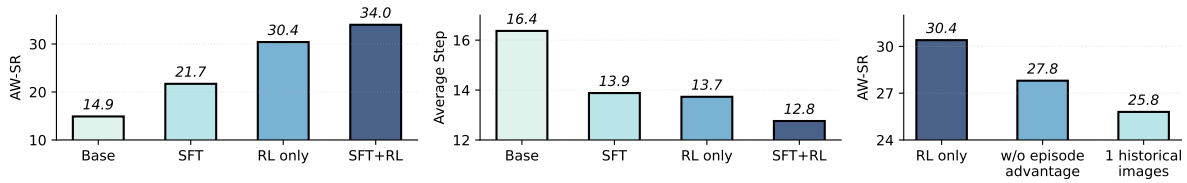

Figure 8: **Left**: Performance of different training paradigm combinations. **Middle**: Average steps to complete AndroidWorld tasks. **Right**: Ablations on episode advantages and historical images.

real-world performance. `AC-Real` achieves the highest correlation with AndroidWorld (R^2=0.934), substantially outperforming AC-High (R^2=0.470) while requiring minimal evaluation time. This strong correlation confirms that maintaining model-generated history during evaluation accurately captures the multi-turn dynamics. The metric fills a critical gap between fast but unrealistic static evaluation and accurate but expensive dynamic testing.

Setting	AC-Real		
	PG	TSR	Score
γ=0.0, ω_e=0.0, η=0.1	22.2	11.0	16.6
γ=0.0, ω_e=0.0, η=0.3	22.3	10.8	16.6
γ=0.0, ω_e=0.0, η=0.5	21.7	11.4	16.6
γ=0.0, ω_e=0.5, η=0.1	22.7	12.2	17.5
γ=0.0, ω_e=0.5, η=0.3	23.3	12.5	17.9
γ=0.0, ω_e=0.5, η=0.5	22.5	10.2	16.4
γ=0.0, ω_e=1.0, η=0.1	20.6	11.8	16.2
γ=0.0, ω_e=1.0, η=0.3	22.2	11.2	16.7
γ=0.0, ω_e=1.0, η=0.5	22.8	12.1	17.5
γ=0.5, ω_e=0.0, η=0.1	26.8	13.8	20.3
γ=0.5, ω_e=0.0, η=0.3	26.1	14.6	20.4
γ=0.5, ω_e=0.0, η=0.5	27.0	13.9	20.5
γ=0.5, ω_e=0.5, η=0.1	26.9	14.2	20.6
γ=0.5, ω_e=0.5, η=0.3	27.3	14.0	20.7
γ=0.5, ω_e=0.5, η=0.5	27.5	14.5	21.0
γ=0.5, ω_e=1.0, η=0.3	<u>27.9</u>	**15.4**	**21.7**
γ=0.5, ω_e=1.0, η=0.1	26.5	<u>14.8</u>	20.7
γ=0.5, ω_e=1.0, η=0.5	**28.4**	14.5	<u>21.5</u>

Table 10: Ablation on γ (future reward discount), ω_e (advantage weight), and η (DAPO threshold), with $\epsilon = 0$, data size of 1000, and one training epoch. The highest value is in **bold**, the second highest is <u>underlined</u>.

G Other Ablations

Hyper-parameter We conduct an ablation study to determine the optimal values for key hyperparameters. As detailed in Table 10, we explore different settings for $\gamma \in \{0, 0.5\}$, $\omega \in \{0, 0.5, 1\}$, and $\eta \in \{0.1, 0.3, 0.5\}$. Based on the empirical results, we adopt $\gamma = 0.5$, $\omega_e = 1$, and $\eta = 0.3$ for the final training configuration.

Training Paradigm. We also conduct ablation studies on training paradigms in Figure 8. Combining SFT with EMPO outperforms either method alone, achieving 34.0% on AndroidWorld compared to 30.4% for EMPO only and 21.7% for SFT only. The combined approach also reduces average task completion steps (middle panel), eliminating redundant actions with better planning. Additional ablations (right panel) confirm that both episode-level advantages and maintaining multiple historical images contribute to performance, validating our training setup. More ablations about the hyper-parameter and the reward design are shown in Appendix G. We also conduct experiments on 3B and 32B models to investigate the effect of model scale and demonstrate the generalization capability of our method (as shown in Figure 3 (Right)). Detailed task analysis and case study are shown in Appendix J.

Impact of Patch Threshold. We present the results of patch strategies across different data scales and thresholds in Table 11. The patch threshold ϵ controls how many mismatches are recovered before termination. Results demonstrate that increasing ϵ consistently improves both `AC-Real` and AndroidWorld metrics. With 1000 training sam-

ϵ	AC-Real			AW
	PG	TSR	Score	
Thought-Free Patch				
0	26.3	14.3	20.3	21.0
1	27.9	15.1	21.5	24.0
2	29.1	16.5	22.8	25.4
∞	30.4	16.7	23.6	25.6
Off-Policy Thought Patch				
0	26.3	14.3	20.3	21.0
1	24.0	12.9	18.5	19.7
2	28.1	14.9	21.5	25.0
∞	30.2	13.3	21.8	24.0
On-Policy Thought Patch				
0	26.3	14.3	20.3	21.0
1	28.7	15.3	22.0	25.0
2	29.4	16.0	22.7	24.9
∞	30.3	17.1	23.7	26.9

ϵ	AC-Real			AW
	PG	TSR	Score	
Thought-Free				
0	28.0	14.8	21.4	27.2
1	28.5	15.7	22.1	29.1
2	31.6	16.5	24.1	31.5
∞	33.8	17.0	25.4	30.8
Off-Policy Thought Patch				
0	28.0	14.8	21.4	27.2
1	28.5	12.5	20.5	25.0
2	30.0	13.5	21.8	26.0
∞	30.5	14.0	22.3	24.0
On-Policy Thought Patch				
0	28.0	14.8	21.4	27.2
1	31.0	15.2	23.1	28.2
2	32.0	16.7	24.4	29.8
∞	33.2	17.2	25.2	31.5

ϵ	AC-Real			AW
	PG	TSR	Score	
Thought-Free Patch				
0	29.6	15.0	22.3	30.0
1	32.4	16.3	24.4	34.0
2	32.6	16.8	24.7	33.9
∞	34.4	17.0	25.7	34.5
Off-Policy Thought Patch				
0	29.6	15.0	22.3	30.0
1	29.5	12.0	20.8	24.6
2	31.6	12.6	22.1	25.3
∞	31.8	13.3	22.6	24.0
On-Policy Thought Patch				
0	29.6	15.0	22.3	30.0
1	32.9	16.7	24.8	31.9
2	33.1	17.4	25.3	32.8
∞	34.4	17.8	26.1	34.5

Table 11: Performance comparison for different ϵ values with varying data sizes (200, 500, 1000 from left to right). Each table shows results for AC-Real and AW under three patching strategies.

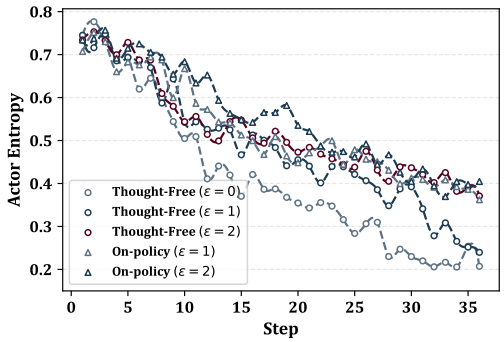

Figure 9: Actor entropy during training process with different patch method and threshold.

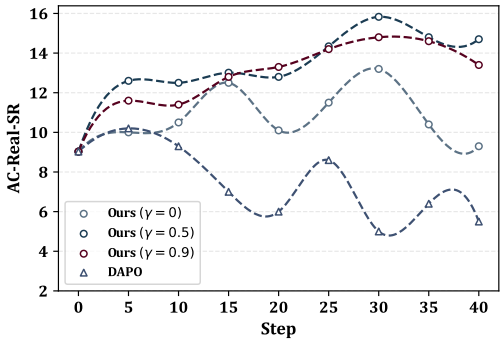

Figure 10: Comparison of EMPO (with different γ) and Single RL (DAPO) during training.

ples, AC-Real-Score increases from 22.3 (ϵ=0) to 25.7 (ϵ=∞) for Thought-Free Patch, representing a 15% relative improvement. This gain stems from increased exposure to later trajectory steps, as higher ϵ values enable learning from previously inaccessible trajectory segments. Figure 9 reveals that larger ϵ values maintain greater policy entropy during training, indicating more diverse exploration and preventing premature convergence. We select ϵ=1 as optimal, achieving 34.0% on AndroidWorld while minimizing computational overhead.

Comparison of Patch Methods. Three patching strategies exhibit distinct trade-offs between performance and efficiency (from Figure 13). On-Policy Thought Patch achieves the highest AC-Real scores (26.1 at ϵ=∞) by maintaining reasoning consistency with the policy model. Thought-Free Patch delivers competitive performance (25.7) with significantly lower computational cost, requiring no additional inference for synthetic reasoning generation. Off-Policy Thought Patch underperforms (22.6) due to distribution mismatch between the auxiliary model's reasoning style and the policy model's expectations. Based on these results and efficiency considerations, we adopt Thought-Free Patch with ϵ=1 for our final configuration.

Discount Factor Analysis. The results in Figure 10 demonstrate the importance of future reward discounting in EMPO. Our approach increases the task success rate during training steps while traditional Single-turn RL (DAPO) exhibits opposite behavior. This divergence highlights a fundamental difference: EMPO's historical context continuity enables effective multi-turn paradigms learning, while single-turn RL ignores long-horizon training signals. Among different γ in our setting, per-

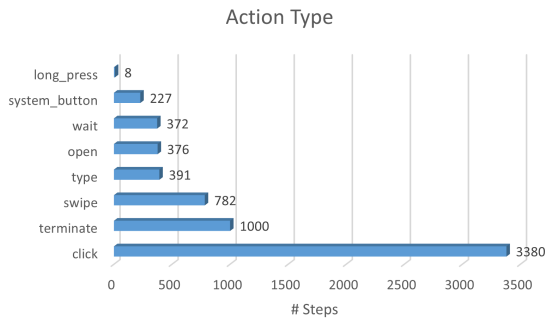

Figure 11: Action Type Distribution of Training Data.

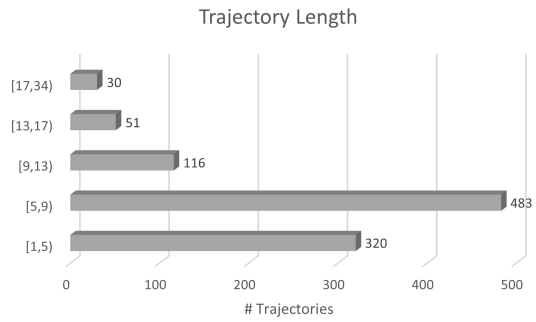

Figure 12: Trajectory Length of Training Dataset.

formance peaks at γ=0.5. Setting γ=0 (no future rewards) yields the worst results, confirming that long-horizon optimization is essential. We further conduct ablation studies on the terminal step t_{end} used to compute future rewards in Equation 5, with results reported in Table 12.

t_{end}	$\epsilon = 0$	$\epsilon = 1$	$\epsilon = 2$	$\epsilon = \infty$
T	25.6	27.9	27.7	27.4
Ours	25.6	28.0	28.9	28.4

Table 12: Ablation on t_{end} in Equation 5 with AndroidWorld success rate reported.

H Training Details

Training Parameters. Our UI-S1-7B is first Supervised Fine-Tuned (SFT) on Qwen2.5VL-7B, trained on AndroidControl-Train (Li et al., 2024) and Amex (Chai et al., 2024), then optimized using EMPO with the thought-free patch mechanism. The parameters are detailed in Table 13.

Parameter	Value
train_batch_size	32
max_prompt_length	12288
γ (future reward discount)	0.5
ω (advantage weight)	1.0
ϵ (patch threshold)	1
η (DAPO threshold)	0.3
historical images	2
learning rate	1×10^{-6}
ppo_mini_batch_size	32
fixed_num_mini_batches	4
ppo_micro_batch_size_per_gpu	1
kl_loss_coef	1×10^{-4}
n_gpus_per_node	8
nnodes	4
total_epochs	5

Table 13: Key Training Hyper-parameters

Training Dataset. Figures 11 and 12 illustrate the distributions of action types and trajectory lengths across 1,000 training trajectories, respectively. Among action types, `CLICK` is the most prevalent, followed by `TERMINATE`, which consistently serves as the final action in all successfully completed trajectories, and `SWIPE`. In terms of trajectory length, most trajectories comprise between 5 and 9 interaction steps.

Training Hours. We analyze the training overhead of different patch methods, measured in GPU hours, as depicted in Figure 13. Although the on-policy method yields a slight improvement in `AC-Real` performance, it incurs a significant 2.3-fold increase in training time compared to other approaches. To strike a balance between effectiveness and efficiency, we therefore select the thought-free patch method for our final model.

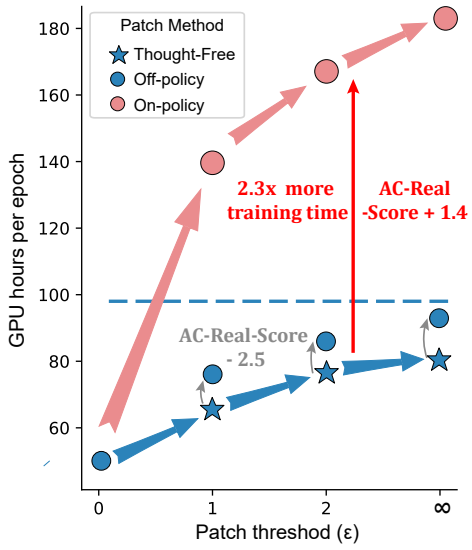

Figure 13: Training GPU hours of different patch methods and patch threshold.

I Baseline

We list all the baseline models in Table 14.

Model	Size	*Conference*
General Models		
Qwen2-VL (Wang et al., 2024)	2B	*Tech Report*
Qwen2.5VL (Bai et al., 2025)	3B/7B/32B	*Tech Report*
Closed-Source Models		
Gemini-Pro-1.5 (Team et al., 2024)	–	*Tech Report*
Claude-CU (Anthropic, 2024)	–	*Tech Report*
GPT-4o (Hurst et al., 2024)	–	*Tech Report*
GUI-Specific SFT Models		
SeeClick (Cheng et al., 2024)	–	*ACL'24*
ShowUI (Lin et al., 2024)	2B	*CVPR'25*
OS-Genesis (Sun et al., 2024)	7B	*ACL'25*
OS-Atlas (Wu et al., 2024)	4B/7B	*ICLR'25*
Aguvis (Xu et al., 2024)	7B/72B	*ICML'25*
UI-TARS (Qin et al., 2025)	7B	*Tech Report*
GUI-Specific RL Models		
UI-R1 (Lu et al., 2025b)	3B	*AAAI'26*
GUI-R1 (Luo et al., 2025a)	3B/7B	*Preprint*
AgentCPM-GUI (Zhang et al., 2025b)	8B	*EMNLP'25*
MobileGUI (Shi et al., 2025)	7B	*Preprint*
PAL-UI (Liu et al., 2025d)	3B/7B	*Preprint*
UI-AGILE (Lian et al., 2025)	3B/7B	*Preprint*
NaviMaster (Luo et al., 2025b)	7B	*Preprint*
GUI-Rise (Liu et al., 2025b)	2B	*NeurIPS'25*

Table 14: Baseline models used in Table 3 and Table 4. Claude-CU refers to Claude-Computer-Use.

Tags	Qwen2.5VL-7B			UI-S1-7B		
	Easy	Medium	Hard	Easy	Medium	Hard
ui_understanding	0.00	0.00	0.00	0.17$_{+0.17}$	0.20$_{+0.20}$	0.14$_{+0.14}$
data_edit	0.09	0.00	0.00	0.64$_{+0.55}$	0.14$_{+0.14}$	0.00$_{+0.00}$
data_entry	0.00	0.11	0.00	0.07$_{+0.07}$	0.10$_{-0.01}$	0.00$_{+0.00}$
game_playing	0.00	–	–	0.00$_{+0.00}$	–	–
information_retrieval	0.14	0.00	0.00	0.43$_{+0.29}$	0.11$_{+0.11}$	0.00$_{+0.00}$
math_counting	0.00	0.00	0.00	0.00$_{+0.00}$	0.33$_{+0.33}$	0.00$_{+0.00}$
memorization	0.00	0.00	0.00	0.00$_{+0.00}$	1.00$_{+1.00}$	0.00$_{+0.00}$
multi_app	0.00	0.00	0.00	0.00$_{+0.00}$	1.00$_{+1.00}$	0.00$_{+0.00}$
parameterized	0.09	0.09	0.06	0.41$_{+0.32}$	0.18$_{+0.09}$	0.11$_{+0.05}$
repetition	0.00	0.00	0.20	0.50$_{+0.50}$	0.00$_{+0.00}$	0.20$_{+0.00}$
requires_setup	0.00	0.00	0.00	0.67$_{+0.67}$	0.00$_{+0.00}$	0.00$_{+0.00}$
screen_reading	0.08	0.00	0.11	0.50$_{+0.42}$	0.33$_{+0.33}$	0.11$_{+0.00}$
search	0.00	0.00	0.00	0.73$_{+0.73}$	0.20$_{+0.20}$	0.00$_{+0.00}$
transcription	0.00	0.00	0.00	0.00$_{+0.00}$	0.50$_{+0.50}$	0.00$_{+0.00}$
untagged	0.40	0.00	–	0.80$_{+0.40}$	0.00$_{+0.00}$	–
verification	0.86	–	–	1.00$_{+0.14}$	–	–

Table 15: Mean AndroidWorld success rate comparison between Qwen2.5VL-7B and UI-S1-7B across tags and difficulty levels, with improvement indicated (positive, negative, no change).

J Result Analysis

J.1 Task Analysis

As detailed in Table 15, our UI-S1-7B model demonstrates substantial performance gains over the Qwen2.5VL-7B baseline across a majority of task categories. The improvements are particularly pronounced in tasks requiring multi-step interactions and complex comprehension, such as `multi_app` (+1.00), `search` (+0.73 on Easy), `requires_setup` (+0.67 on Easy), and `complex_ui_understanding`. These results strongly suggest its enhanced planning and reasoning capabilities for navigating complex user interfaces. Nevertheless, challenges persist in domains that demand specialized skills. For instance, both models struggle with `game_playing` and `math_counting` tasks. We attribute this to the inherent limitations of small-scale vision-language models in handling precise numerical computation and abstract logical reasoning. We also showcase a failure case of `math_counting` in Figure 20. In this case, while UI-S1-7B was able to remember the numbers it encountered, it made an error at step 11, calculating 9*10*9*5*5 as 2250.

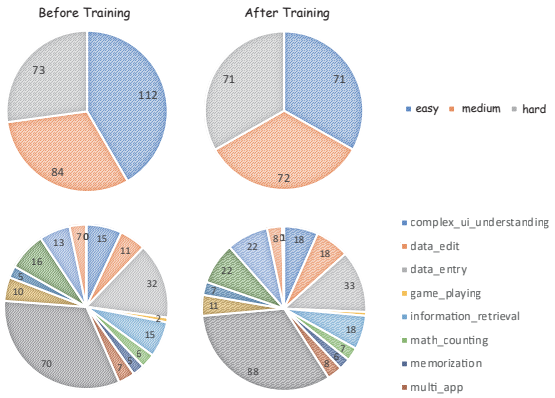

Figure 14: AndroidWorld task error count distribution.

We also display the error distribution. From the **difficulty** perspective (Figure 14), the most substantial improvement was observed in 'easy' tasks, where UI-S1-7B achieved a remarkable reduction of 41 errors compared to the base model. Following this, a moderate but significant performance gain was noted for 'medium' difficulty tasks. In stark contrast, the model's advantage diminished considerably for 'hard' tasks, showing only a marginal improvement with a reduction of two errors.

From a task **classification** perspective (Figure 14), while the proportional distribution of errors across different task categories remained largely consistent between the two models, UI-S1-7B demonstrated marked advancements in several key functional areas, such as `scree_reading`, `search`, `transcription`, `data_edit`, and `parameterized`.

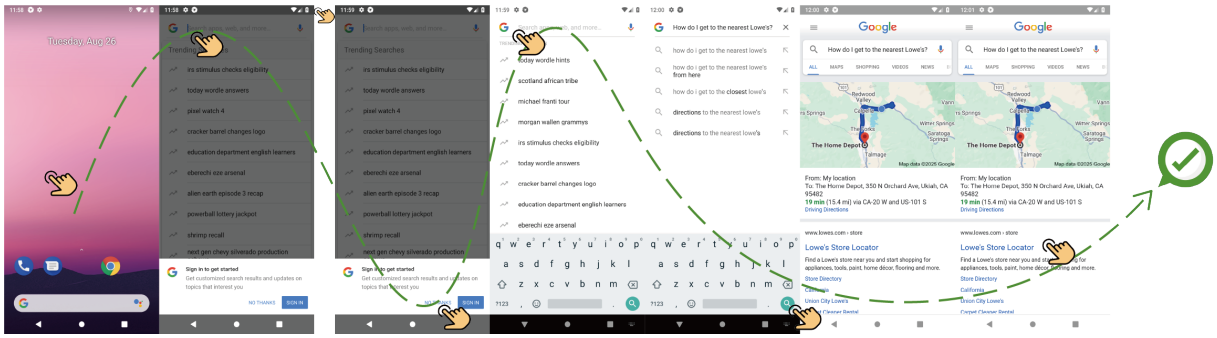

Figure 15: A successful task case encountering *sign in notes* in **AITW-Gen**. The instruction is *"How do I get to the nearest Lowe's?"*.

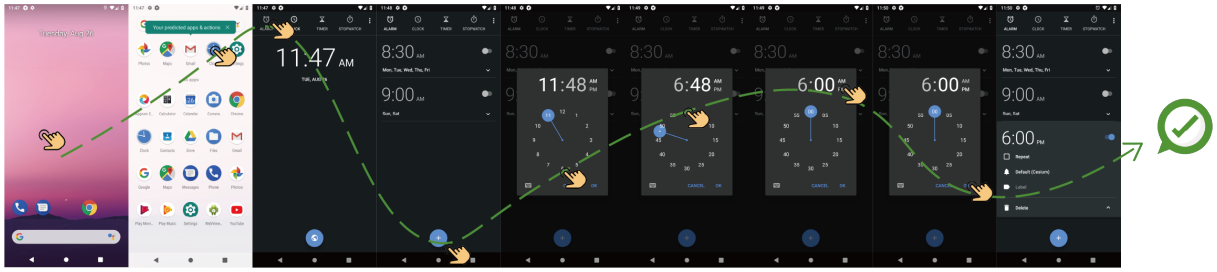

Figure 16: A successful task case in **AITW-Gen**. The instruction is *"Set an alarm for 6pm"*.

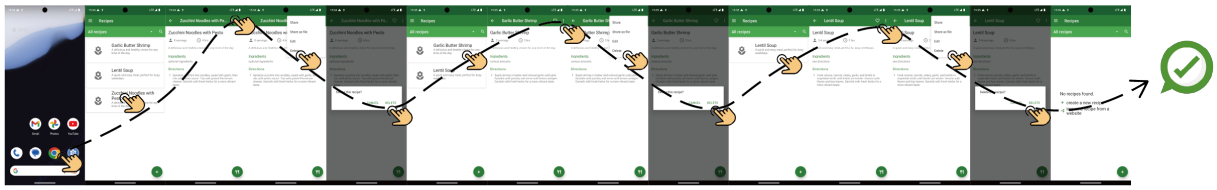

Figure 17: A successful task case in **AndroidWorld**. The instruction is *"Delete the following recipes from Broccoli app: Zucchini Noodles with Pesto, Garlic Butter Shrimp, Lentil Soup."*

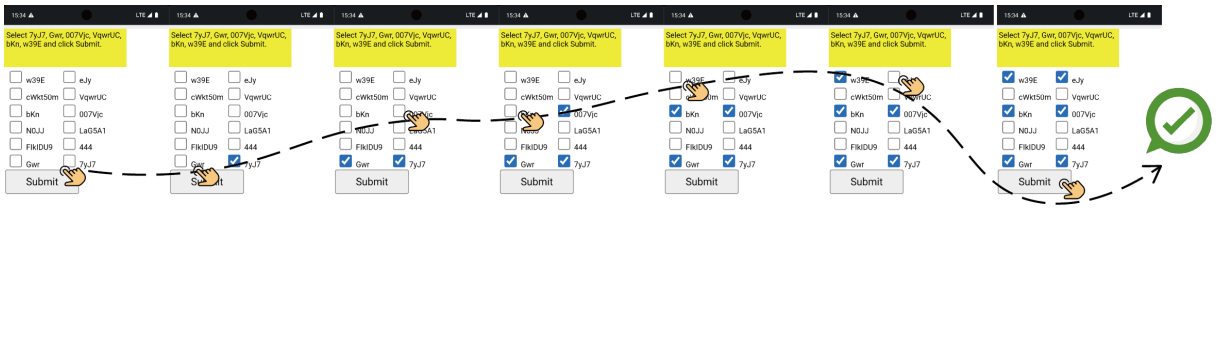

Figure 18: A successful task case in **MiniWob++**. The instruction is *"Follow the instructions shown on the top of the screen: Select 7yJ7, Gwr, 007Vjc, VqwrUC, bKn, w39E and click Submit."*

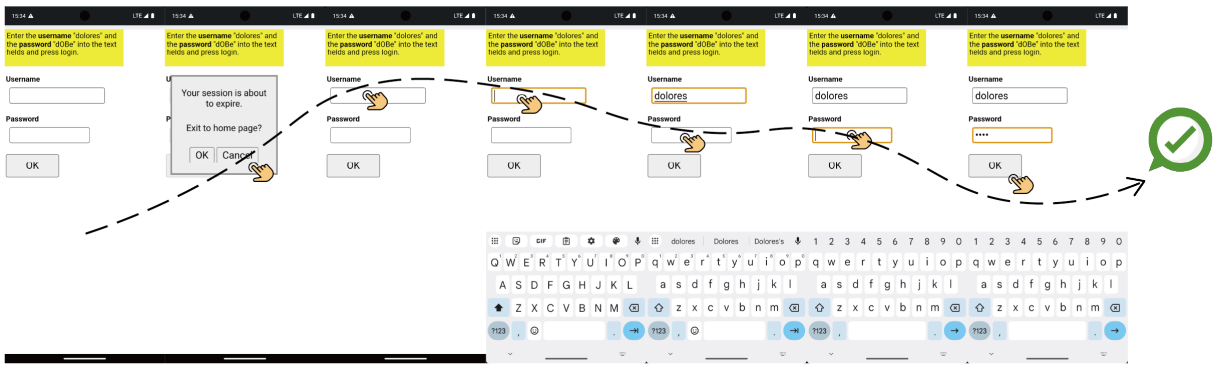

Figure 19: A successful task case in **MiniWob++**: *"Follow the instructions shown on the top of the screen: Enter the username dolores and the password dOBe into the text fields and press login."*.

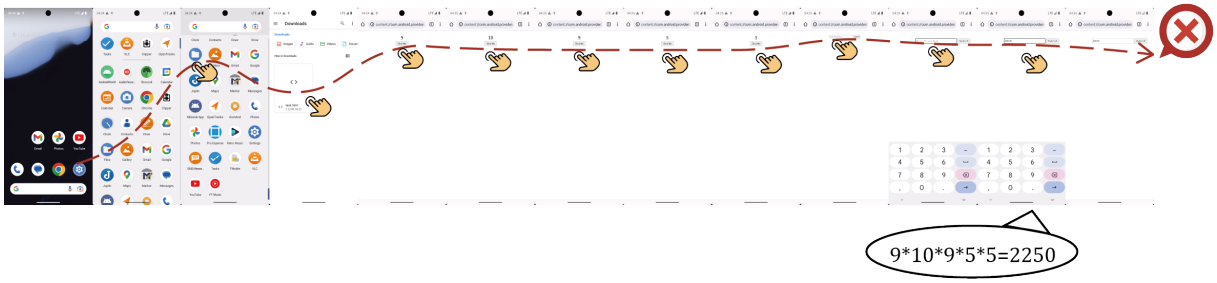

Figure 20: A failed task case in **AndroidWorld**. The instruction is *"Open the file task.html in Downloads in the file manager; when prompted open it with Chrome. Then click the button 5 times, remember the numbers displayed, and enter their product in the form."*.

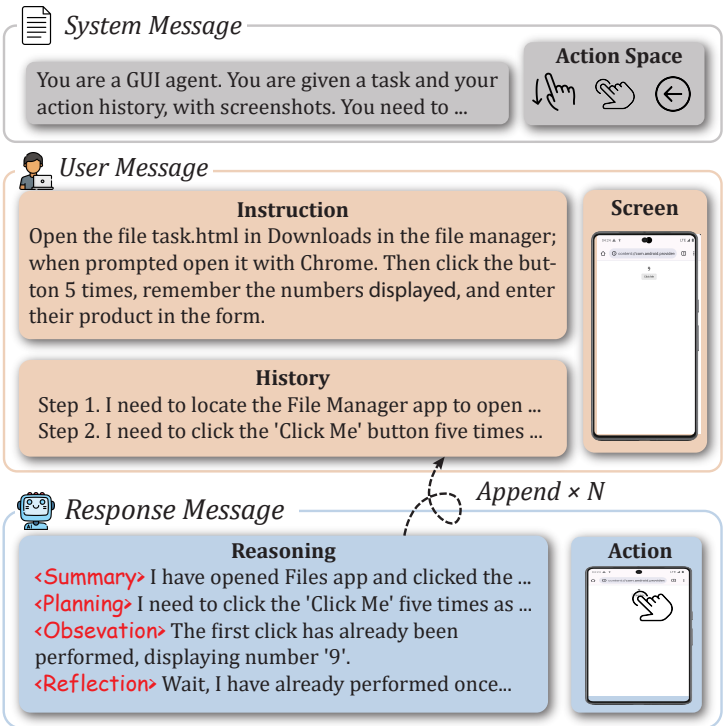

Figure 21: **Overview of the multi-turn interaction flow.** The system message defines the available action space, the user message provides the task instruction, action history, and current screen, and the response message contains the agent's reasoning and final action.

System prompt:
You are a GUI agent. You are given a task and your action history, with screenshots. You need to perform the next action to complete the task.
Output Format

```
<think> ... </think>
<action> ... </action>
```

Action Space
You can perform the following actions:
- key: Perform a key event on the mobile device using adb's 'keyevent' syntax.
- click: Click the point on the screen with specified (x, y) coordinates.
- long_press: Press the point on the screen with specified (x, y) coordinates for a specified number of seconds.
- swipe: Swipe from starting point with specified (x, y) coordinates to endpoint with specified (x2, y2) coordinates.
- type: Input the specified text into the activated input box.
- answer: Output the specified answer.
- system_button: Press the specified system button: Back, Home, Menu, or Enter.
- open: Open an application on the device specified by text.
- wait: Wait for a specified number of seconds for changes to occur.
- terminate: Terminate the current task and report its completion status: success or failure.

The arguments you can use are:
- coordinate: (x, y): The x and y pixels coordinates from the left and top edges.
- coordinate2: (x, y): The x and y pixels coordinates from the left and top edges for the endpoint of a swipe.
- text: Text input required by actions like 'key', 'type', 'answer', and 'open'.
- time: The time in seconds required by actions like 'long_press' and 'wait'.
- button: System buttons available for pressing: Back, Home, or Enter. Possible values: Back, Home, Menu, Enter.
- status: The completion status of a terminated task. Possible values: success, failure.

Format your output as a JSON object with the selected action and its arguments at the same level.
Example Output

```
<think>...</think>
<action>{"action": "key", "text": "<value>"}
```

Note
- Planing the task and explain your reasoning step-by-step in 'think' part.
- Write your action in the 'action' part according to the action space.
- If the query asks a question, please answer the question through the answer action before terminating the process.
- Swipe the screen to find the File Manager app if needed.

User prompt:
User Instruction: USER INSTRUCTION
Assistant prompt:
HISTORY RESPONSES
HISTORY IMAGES

Figure 22: **Prompt template for end-to-end GUI Agent.**

System prompt:
End-to-End Model Thought Integration
Integration Requirements

- Write the thought process from a global goal, the action history, thought history and screenshot history.
- The reasoning logic must satisfy:
  - Begin by reviewing the global task objective.
  - Inherit the context and decisions from historical steps.
  - Incorporate the manager's planning logic.
  - Derive actions that fully align with the operator's output.

Output Format

`<think> [A coherent reasoning process, reflecting task decomposition, environmental observation, and iterative decision-making] </think>`

Output Example

`<think> The current task requires checking the order status of DeepSeek. Access to the official website and locating the login entry have been completed. Based on the page loading result, the login form is ready. Authentication information needs to be filled: the username has already been entered as "DeepSeek," and now... </think>`

Key Design Notes

- Explicitly require the global task objective to ensure the end-to-end model always anchors to the core goal.
- Enforce structured historical records to prevent information loss.
- Logic consistency mechanism.
- The thought process should naturally connect historical conclusions with the current manager's planning.
- Transform the manager's planning into autonomous decisions phrased as "According to the requirements, determine..."
- Translate operator actions into imperative statements phrased as "Execute..."
- Do not mention any coordinates in <think> ... </think>.

Global Task Objective
USER INSTRUCTION
- If this isn't the target app for your operation, you can use open operation to navigate to the correct application.
- You can use Next Action Hint to guide the think process, but within the think section, you must conceal the fact that hints were received.
- Please integration the thought of current manager and operation into <think> ... </think> in English.
Assistant prompt:
HISTORY RESPONSES
HISTORY IMAGES

Figure 23: **Prompt template for thought generation.**

System prompt:
You're an expert in evaluating whether the Screenshot successfully completes the Task.
==========Examples==========
Task: Open the settings. Q: What should I expect to see on the screenshot if I've opened the settings? A: I should expect to see I'm in the settings app. The screenshot shows the home screen of a mobile device, with various app icons displayed, including the settings app icon, but the settings app is not opened.
Status: failure Screenshot: SCREENSHOT
Task: Find hotels in Washington DC Q: What should I expect to see on the screenshot if I've searched for hotels in Washington, DC? A: I should expect to see I'm in a search results page for hotels in Washington, DC. The screenshot shows a Google search page with the search field populated with the query "hotels in washington dc" and a list of suggested searches related to hotels in Washington, DC, but it does not show any search results for hotels in Washington, DC.
Status: failure Screenshot: SCREENSHOT
Task: What's a good restaurant in Portland? Q: What should I expect to see on the screenshot if I've searched for a good restaurant in Portland? A: I should expect to see I'm in a search results page for a good restaurant in Portland. The screenshot shows a Google search page with a search input field for "good restaurant in portland" and a map results preview showing business locations near Portland, like "Li Pigeon", "Portland City Grill", and "Higgins".
Status: success Screenshot: SCREENSHOT
Task: What's on the menu at In-N-Out? Q: What should I expect to see on the screenshot if I've searched for the menu at In-N-Out? A: I should expect to see a menu page for In-N-Out, including product names, thumbnails and prices. The screenshot shows a Google search page with a search input field for "In-N-Out menu" and some page snippets of In-N-Out indicating potential menu items, but does not actually show the actual menu.
Status: failure Screenshot: SCREENSHOT
Task: What's the news in Suriname? Q: What should I expect to see on the screenshot if I've searched for the news in Suriname? A: I should expect to see some news in Suriname, such as someone did something or some accident happens in Suriname. The screenshot shows a Google search page with a search input field for "Suriname news today" and some page snippets indicating potential news items, but does not actually show the news.
Status: failure Screenshot: SCREENSHOT
Task: What's the weather like in Chicago? Q: What should I expect to see on the screenshot if I've searched for the weather in Chicago? A: I should expect to see some exact values like temperature, humidity, wind speed, and weather condition in Chicago. The screenshot shows a Google search page with a search input field for "weather in Chicago" and some page snippets indicating potential weather information. Although one page snippet contains some weather information, the information is not comprehensive enough to determine the weather in Chicago.
Status: failure Screenshot: SCREENSHOT
Task: Set an alarm for 6pm. Q: What should I expect to see on the screenshot if I've set an alarm for 6pm? A: I should expect to see some alarms including a 6pm alarm activated in the clock app. The screenshot shows an attempt to set an alarm for 6pm in the clock app, but the alarm is not set yet.
Status: failure Screenshot: SCREENSHOT

Figure 24: **Prompt for GPT-4o to evaluate MiniWob++ Task**