# ANDROID COACH: Improve Online Agentic Training Efficiency with Single State Multiple Actions

**Guo Gan**[Z*] **Yuxuan Ding**[Q] **Cong Chen**[Z] **Yuwei Ren**[Q] **Yin Huang**[Q] **Hong Zhou**[Z✉]

[Z]Zhejiang University [Q]Qualcomm AI Research

## Abstract

Online reinforcement learning (RL) serves as an effective method for enhancing Android agents. However, guiding agents to learn through online interaction is prohibitively expensive due to the high latency of emulators and the sample inefficiency of existing RL algorithms. We identify a fundamental limitation in current approaches: the *Single State Single Action* paradigm, which updates the policy with one-to-one state-action pairs from online one-way rollouts, without fully exploring each costly emulator state. In this paper, we propose ANDROID COACH, a novel framework that shifts the training paradigm to *Single State Multiple Actions*, allowing the agent to sample and utilize multiple actions for a single online state. We enable this without additional emulator overhead by online learning a critic that estimates action values. To ensure the critic serves as a reliable coach, we integrate a process reward model and introduce a group-wise advantage estimator based on the averaged critic outputs. Extensive experiments demonstrate the effectiveness and efficiency of ANDROID COACH: it achieves 7.5% and 8.3% success rate improvements on AndroidLab and AndroidWorld over UI-TARS-1.5-7B, and attains 1.4× higher training efficiency than *Single State Single Action* methods PPO and GRPO at matched success rates.

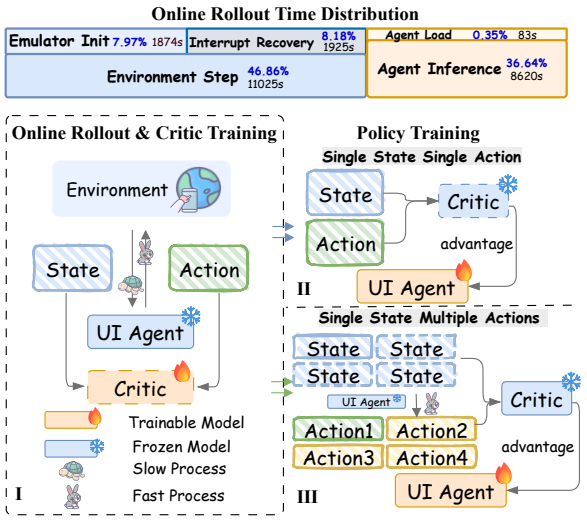



Figure 1: (Top): Online rollout time distribution based on the measured time on 8 parallel environments in training for 80 steps. (Bottom I): The conventional online rollout and critic training loop. The primary bottleneck is the high-latency environmental interaction, while the GUI agent action inference is relatively fast. (Bottom II): Standard agent update with *Single State Single Action* paradigm. Agent updates rely merely on the state-action pairs collected from the online rollout. (Bottom III): ANDROID COACH update with *Single State Multiple Actions* paradigm. We fully leverage each expensive online state by generating multiple actions. The agent is then updated using this data. This approach improves training efficiency by gathering more training samples within the same online interaction cost.

## 1 Introduction

Graphical User Interface (GUI) agent is an application of Vision-Language models (VLMs) in interactive scenarios (Zhang et al., 2025a; Zhang and Zhang, 2024; Yang et al., 2025). When human users provide an instruction, the agent leverages reasoning and function-calling capabilities to autonomously conduct multi-turn interactions to complete the task (Xu et al., 2026; Lian et al., 2025; Xie et al., 2025; Chen et al., 2026). Reinforcement learning (RL) is widely used in agent training, which helps to enhance reasoning and decision-making capability for complex sequential tasks (Lu et al., 2025a). In this paper, we focus on optimizing the reinforcement learning for GUI agent.

Reinforcement learning approaches for GUI agent generally fall into two categories based on

*This work was done during Guo Gan's internship at Qualcomm AI Research, an initiative of Qualcomm Technologies, Inc. ✉Corresponding author: Hong Zhou <zhouhong_zju@zju.edu.cn>. Our code will be available at https://github.com/gguogan/Android_Coach.



*Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13407–13428
July 2-7, 2026 

their interaction paradigm. Offline approaches rely on pre-collected expert trajectories (Sun et al., 2025; Wei et al., 2026). While they avoid frequent environment interactions, the methods are bounded by the data quality and struggle to handle rapid application and GUI updates due to the lack of online exploration (Bai et al., 2024; Lu et al., 2025c). Online methods mitigate these limitations by collecting data through environment interactions and learning from active trial-and-error for better performance (Bai et al., 2025a), but still exhibit other shortcomings in training efficiency.

Online training typically suffers from poor sample efficiency (Lu et al., 2025d; Dong et al., 2025) as shown in Figure 1. First, it requires high-latency emulator interactions, including initialization, recovery and reaction, which is 1.7× greater than the time for model loading and inference. Consequently, the online RL states, which include screenshots and interaction history, are costly to collect. Second, current online RL methods conduct one-to-one state-action rollouts (Zhang et al., 2025b). This means the agent can only sample once with a given state, because the emulator would transition to the next state after the execution. We term this the *Single State Single Action* (*SSSA*) paradigm, like PPO in UI-TARS (Wang et al., 2025) and GRPO in ARPO (Lu et al., 2025a). This paradigm makes it difficult to sufficiently explore the state, since the agent cannot try other actions.

We propose ANDROID COACH, a novel actor-critic framework that adopts *Single State Multiple Actions* (*SSMA*) paradigm to address the problems above: 1) To reduce interaction overhead, we use a critic (*i.e.*, state-action value function Q) to estimate action value, which allows us to get values of more sampled actions without the environment. 2) To sufficiently explore the states, we randomly sample multiple actions given online states and value them with Q, which means the agent can do more exploration without additional emulator overhead.

Reliably evaluating the action value is essential in our paradigm. Our Q is kept updated using the actor online rollout data, which ensures the robustness against the distribution shift typically encountered in offline approaches (Zheng et al., 2025). Meanwhile, we introduce a fine-grained, pretrained process reward model into our framework rather than merely trajectory-level outcome supervision. This makes Q capable of crediting correct steps within a failed trajectory, leading to a better supervision of intermediate steps (Chen et al., 2025a). Besides, tailored to *SSMA* paradigm, we propose a novel advantage estimation method Actor-Critic Leave-one-out(ACLOO), where the baseline is the average Q-value with leave-one-out strategy. Our design reduces estimation variance without effort to train a state-value model, and introduces the relative quality, guiding the agent based on the average level (Konda and Tsitsiklis, 1999; Bai et al., 2024). This is inspired by RLOO (Kool et al., 2019), where the advantage is reward-based while ours uses long-term value.

The overall framework of ANDROID COACH is shown in Figure 2. This is an online actor-critic method with *Single State Multiple Actions* to increase the number of training samples, making full use of the online rollout state within the same interactions. The actor samples multiple actions and constructs state-action pairs as training samples. To do *SSMA* training with fewer interactions, ANDROID COACH does not execute these actions with the emulator, but evaluates them with the critic, where a leave-one-out advantage helps train the actor. For critic training, ANDROID COACH applies the return of online rollout actions as ground truth which is estimated by integrating the process reward and outcome reward. We validate our approach on the AndroidLab (Xu et al., 2025b) and AndroidWorld (Rawles et al., 2025) benchmarks, achieving a 7.5% and 8.3% improvement over the success rate of original UI-TARS (Qin et al., 2025), while outperforming conventional *Single State Single Action* methods in GUI agent RL including PPO (Schulman et al., 2017) and GRPO (Shao et al., 2024) with 1.4x training efficiency.

In summary, our contributions are as follows:

1. We propose ANDROID COACH, a meticulously designed framework that first enables *Single State Multiple Actions* paradigm for efficient online agentic reinforcement learning to the best of our knowledge.

2. We propose an online-trained critic guided by both outcome and process rewards, together with our leave-one-out advantage estimator. Without additional environment overhead, the critic supports *Single State Multiple Actions* paradigm with reliable action advantage.

3. Extensive experiments on dynamic benchmarks demonstrate the training efficiency of ANDROID COACH and the effectiveness of its components in online reinforcement learning.

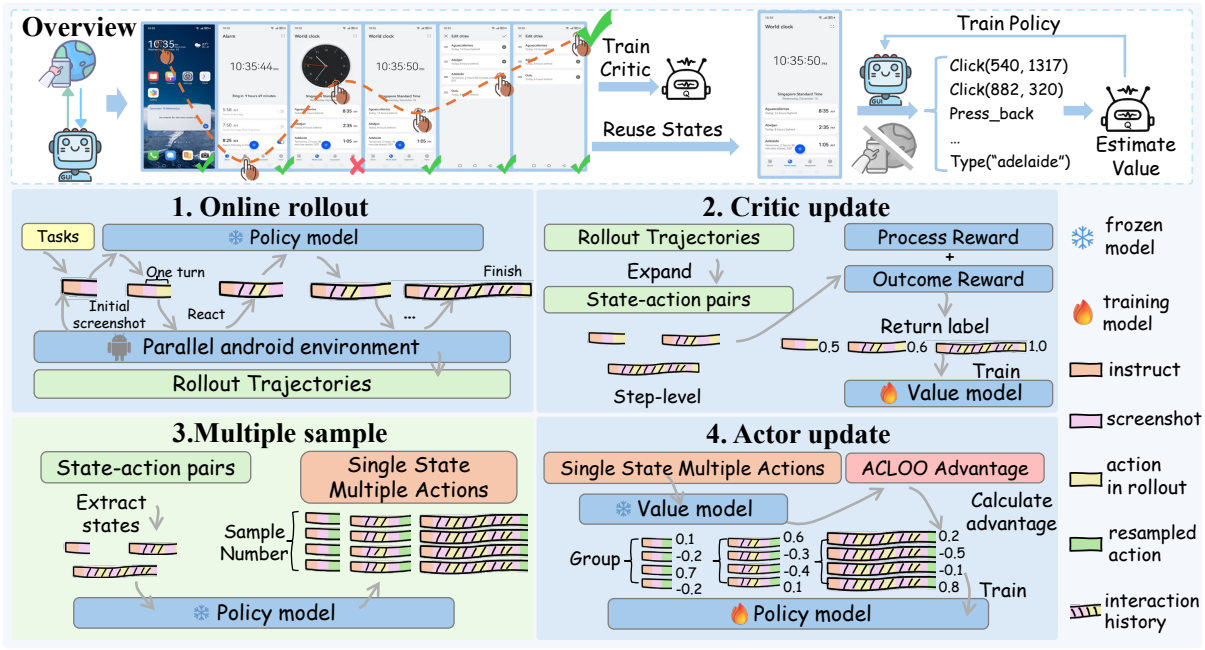



Figure 2: Overview and pipeline for a training step in ANDROID COACH. 1. Online Rollout: Policy interacts with parallel environments to collect complete trajectories. 2. Critic Update: Annotate state-action pairs with returns to train the value model. 3. Multiple sample: Resample multiple actions for each online state. 4. Actor Update: Compute action values and advantages with leave-one-out strategy, then update policy via gradient step.

## 2 Related Work

### 2.1 RL for Training GUI Agents

Existing RL methods for GUI agents generally fall into offline and online categories as shown in Table 1. Offline approaches (Sun et al., 2025; Lu et al., 2025d; Luo et al., 2026) rely on extensive expert or pre-collected datasets. Consequently, such methods are limited by data quality and fail to handle environment updates effectively (Intelligence et al., 2025). In contrast, online training enables continuous improvement through exploration and trial-and-error in real environments (Shi et al., 2025; Ye et al., 2025). Hybrid frameworks like DigiRL (Bai et al., 2024) integrate both phases, while online methods like MobileRL (Xu et al., 2025a), GUI-Shepherd (Chen et al., 2025a), and UI-TARS-2 (Wang et al., 2025) adopt GRPO (Shao et al., 2024) or PPO (Schulman et al., 2017) for online optimization. However, interacting with Android emulator is time-consuming. Although environment parallelization and replay buffers (Lu et al., 2025a) partially alleviate this issue, current online methods remain restricted to *Single State Single Action* paradigm. These approaches generate only one action per state, which fails to fully exploit the expensive state and necessitates more interaction steps for better performance. In this paper, we introduce *Single State Multiple Actions* paradigm for online RL which generates and evaluates multiple actions for each state, significantly enhancing training efficiency under limited interaction budgets.

### 2.2 Advantage Estimation in RL Training

Advantage estimation is a critical component for policy optimization in modern GUI agents. Trajectory-level reward-based approaches (Xu et al., 2025a; Lu et al., 2025a; Luo et al., 2025; Wanyan et al., 2025; Zhang et al., 2026) use GRPO, requiring full rollouts to obtain outcome rewards and compute advantages from averaged returns. Critic-based methods (Chen et al., 2025a; Wang et al., 2025; Bai et al., 2024) can estimate action values without full rollouts, but in online settings the critic is typically trained only with outcome rewards, as process rewards are hard to obtain during interaction. In contrast, offline methods like VEM (Zheng et al., 2025) utilize step-level supervision, but inherit the limitations of offline training. In this paper, we design a critic that incorporates a process reward mechanism for GUI tasks. Furthermore, actor-critic methods using Q-functions (Bai et al., 2025a; Zhou et al., 2024) usually introduce an additional state-value model to reduce variance and stabilize training. To avoid this extra component, we propose an average Q-value baseline with leave-one-out strategy for advantage estimation.

| Training Mode | Base Algorithm | Method | Advantage Estimation | Exploration Paradigm | Sample Efficiency |
|---|---|---|---|---|---|
| Offline | SFT | OS-Genesis (Sun et al., 2025) | Reward-based | Static Data | N/A |
| | Archer (Zhou et al., 2024) | DigiQ (Bai et al., 2025a) | Value-based | Static Data | N/A |
| | Q-Learning (Watkins et al., 1989) | VEM (Zheng et al., 2025) | Value-based | Static Data | N/A |
| | DPO (Rafailov et al., 2023) | UI-TARS-1.5 (Qin et al., 2025) | Pairwise-Preference | Static Data | N/A |
| Hybrid | AWR (Peng et al., 2019) | DigiRL (Bai et al., 2025a) | Value-based | SSSA | Low |
| Online | GRPO (Shao et al., 2024) | MobileRL (Xu et al., 2025a)<br>WebAgent-R1 (Wei et al., 2025)<br>ARPO (Lu et al., 2025a) | Reward-based | SSSA | Low |
| | PPO (Schulman et al., 2017) | UI-TARS-2 (Wang et al., 2025)<br>GUI-Shepherd (Chen et al., 2025a) | Value-based | SSSA | Low |
| **Online** | ACLOO | **ANDROID COACH (Ours)** | **Value-based** | **SSMA** | **High** |

Table 1: Comparison of representative GUI agent training frameworks. ANDROID COACH is the first to achieve efficient *Single State Multiple Actions* (SSMA) exploration in an online setting, significantly improve sample efficiency with limited interaction costs compared to standard *Single State Single Action* (SSSA) approaches. ACLOO refers to Actor-Critic Leave-One-Out we proposed in our method.

### 2.3 RL with Multiple Actions Estimation

Several established RL methods share the core principle of evaluating multiple actions per state for sample efficiency or training stability. Discrete SAC (Christodoulou, 2019) computes the exact expected state value by iterating over all possible actions, yielding zero-variance gradient updates. Similarly, Expected Policy Gradients (EPG) (Ciosek and Whiteson, 2018) eliminates sampling variance entirely via analytic integration for continuous distributions or exhaustive evaluation for discrete ones. However, exact marginalization rather than probabilistic sampling with variance reduction tricks strictly limits these methods to small action spaces (e.g., Atari). Consequently, they are computationally intractable for reasoning VLM-based GUI agents with combinatorially massive action spaces. GRPO (Shao et al., 2024) is a prevalent multiple-rollout method. However, dynamic and irreversible GUI environments prohibit parallel follow-up executions from the exact same state, causing step-level GRPO to fail to acquire long-term value supervision in the absence of value model, while sequence-level GRPO is still limited to Single State Single Action. Given these constraints of dynamic GUI tasks, our method adopts Monte Carlo sampling. To fulfill our core objective of improving online training efficiency via Single State Multiple Actions paradigm without extra interaction overhead, we introduce a learned value model to estimate sample advantages with our group-wise ACLOO baseline design.

## 3 Android Coach

In this section, we first present the **preliminary** knowledge including problem formulation and actor-critic framework (Section 3.1). Then we introduce **how to train a reliable critic** (state-action value function Q) for accurate action value estimation with process reward model and online updates (Section 3.2). Finally, we present **how to leverage the critic** to improve sample efficiency with *Single State Multiple Actions* paradigm and our proposed Actor-Critic Leave-One-Out advantage estimation method (Section 3.3).

### 3.1 Preliminary

**Problem Formulation.** We formulate the Android agent task as a finite-horizon Markov Decision Process (MDP) defined by the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{R})$. Given an instruction $I$, the process begins at state $s_0$ with the initial GUI screenshot. At each timestep $t$, the state $s_t \in \mathcal{S}$ consists of the instruction $I$ and the interaction history. The policy $\pi_\theta(\cdot|s_t)$ parameterized by $\theta$ samples an $a_t$ from action space $\mathcal{A}$, which is constructed by reasoning-based plan and an operation from the GUI operation action space in Appendix A. After execution, the environment transitions to the next state $s_{t+1}$. This interaction loop continues until task completion or a maximum step limit is reached. The agent receives binary rewards $r_t \in \{0, 1\} \sim \mathcal{R}$, including *process reward* for step-wise correctness and *outcome reward* for final success. The objective is to learn the optimal parameters $\theta$ that maximize the expected cumulative discounted return: $J(\pi_\theta) = \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{T} \gamma^t r_t\right]$, where $\gamma$ is the discount factor.

**Actor-Critic Framework.** In the online Actor-Critic method with Q function, the policy is learned concurrently with a state-action value function $Q_\phi(s_t, a_t)$, parameterized by $\phi$. The Critic learns to estimate the expected cumulative reward after taking action $a_t$ in state $s_t$ and following the policy $\pi$ thereafter. This learned Critic $Q_\phi$ then provides

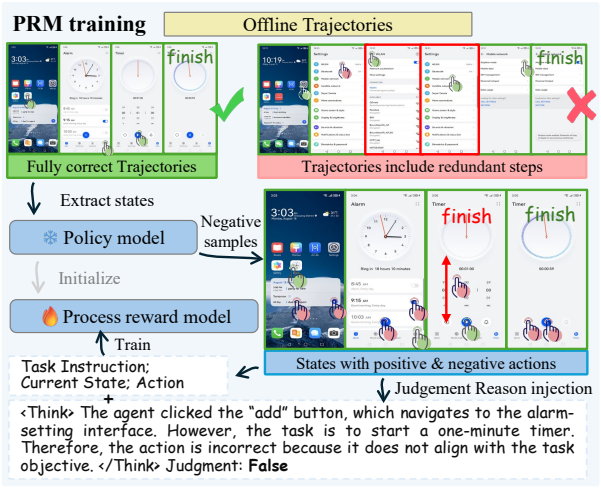



Figure 3: An overview of dataset construction pipeline for process reward model training.

an evaluative signal in the form of an Advantage $A(s_t, a_t)$ to guide the update of Actor policy $\pi_\theta$.

### 3.2 Train a Reliable Coach

We build a reliable state-action value function Q by 1) training a process reward model and 2) online training of the Q incorporating process reward.

**Process Reward Model (PRM).** We first design a PRM to provide step-wise process rewards for intermediate actions. Motivated by the effectiveness of model reasoning (Wei et al., 2022; Guo et al., 2025; Wanyan et al., 2025; Qu et al., 2026; Chen et al., 2025b), we train PRM as a reasoning model that generates analysis prior to judgment. As shown in Figure 3, we construct the training dataset by aggregating trajectories from offline datasets AndroidControl (Li et al., 2024) and supplementary preliminary rollout data, retaining only success and non-redundant trajectories. For each step $(I, s_t, a_t)$, the initial policy conducts single-step generation to produce a reasoning context alongside a predicted action. Each sample is labeled positive if the generated action matches the ground-truth one from the original success trajectory, and negative otherwise. This process yields 20k data points, each structured as $(I, s_t, a_t, \textit{label})$. Then we inject the reason for the process reward label with GPT-4o (OpenAI et al., 2024). More details are provided in Appendix B. We initialize PRM with our initial policy model and perform full-parameter supervised fine-tuning (SFT) with prompt in Figure 10 in Appendix. The PRM parameterized by $\beta$ is updated to optimize the cross-entropy (CE) loss which denoted as $\mathcal{L}_{\text{PRM}}(\beta)$ in Equation 1:

$$\mathcal{L}_{\text{PRM}}(\beta) = -\log P_\beta(y|I, s_t, a_t) \quad (1)$$

**Online Critic Training.** The critic Q-value function $Q_\phi$ shares the same architecture as the policy model, augmented with a value head (von Werra et al., 2020), serving as a coach. We start the RL loop as shown in first two stages of Figure 2, in which the policy model interacts with multiple parallel environments at first to collect a batch of trajectories, $\mathcal{D}$. Upon completion of the batch rollout, the pre-trained PRM assigns step-level process rewards $r_p^t$ based on the intermediate actions, and the outcome verifier (OV) assigns outcome reward ($r_o$) based on the final result. We estimate the target return $R_t$ for each state-action pair using a weighted Monte Carlo estimation that combines these rewards with weight parameters $\omega_p$, $\omega_o$ and discount factor $\gamma$: $R_t = \omega_p \sum_{\tau=t}^{T} \gamma^{T-\tau} r_p^{\tau:T} + \omega_o r_o$. The critic $Q_\phi$ is subsequently updated by minimizing the clipped mean squared error loss between its predictions $Q_\phi(s_t, a_t)$ and the estimated target returns $R_t$ as shown in Equation 2:

$$\mathcal{L}_{\text{Q}}(\phi) = \tfrac{1}{2}\mathbb{E}_t\left[\max\left((Q_\phi - R_t)^2, (\text{clip}(Q_\phi, Q_{old} \pm \epsilon_v) - R_t)^2\right)\right] \quad (2)$$

However, directly training critic together with actor presents a significant challenge in providing reliable value, which is also noted by DigiQ (Bai et al., 2025a). We contend that this issue arises because the value model is poorly prepared for GUI tasks value estimation at the beginning of training, resulting in misleading guidance for the policy updates (Bai et al., 2025a; Wang et al., 2025). Consequently, before online RL, we initialize the model by pre-training it with the PRM dataset, where labels are mapped to binary scores.

### 3.3 RL Guided by Coach

Here we introduce our key designs in actor training: 1) sampling multiple actions and 2) update actor with the Actor-Critic Leave-One-Out advantage.

**Multiple Actions from the Online State.** As shown in third stage of Figure 2, by employing the Q function, we can naturally enhance sample efficiency through the *Single State Multiple Actions* paradigm without additional interactions. Specifically, we reuse every costly state $s_t$ from the online trajectories $\mathcal{D}$, which is collected by the policy $\pi_\theta$ during the online rollout in one training step. We sample a set of $k$ candidate actions $\{a_t^1, ..., a_t^k\}$ for every $s_t$ using the current policy.

| Models | #Params | AndroidLab SR (%) | | | AndroidWorld SR (%) | | | |
|---|---|---|---|---|---|---|---|---|
| | | QD | OP | Average | Easy | Mid | Hard | Average |
| *Proprietary Models* | | | | | | | | |
| Gemini-Pro-1.5 (SoM) (Team et al., 2024) | - | - | - | 16.7 | - | - | - | 22.8 |
| GPT-4o (SoM) (OpenAI et al., 2024) | - | - | - | 31.2 | - | - | - | 34.5 |
| Claude-Sonnet-4 (SoM) (Anthropic, 2025) | - | - | - | 40.6 | - | - | - | **41.0** |
| UI-Genie-Agent (Xiao et al., 2025) | 72B | - | - | **41.2** | - | - | - | - |
| *Open-source 32B/72B Models* | | | | | | | | |
| Qwen2.5VL-32B-Instruct (Bai et al., 2025b) | 32B | 28.4 ±2.1 | 25.8 ±1.1 | 28.5 ±0.4 | 37.2 ±1.9 | 14.8 ±3.2 | 10.5 ±5.3 | 25.9 ±0.9 |
| UI-TARS-72B-DPO (Qin et al., 2025) | 72B | 36.4 ±2.8 | 31.5 ±1.6 | 35.5 ±2.2 | **57.4** ±1.6 | 27.8 ±5.6 | 10.5 ±0.0 | 40.5 ±0.9 |
| *Open-source 7B/8B Models* | | | | | | | | |
| OS-Genesis-7B-AW (Sun et al., 2025) | 7B | 6.8 ±2.8 | 3.6 ±1.2 | 5.1 ±1.9 | 26.8 ±2.5 | 11.1 ±0.0 | 1.8 ±3.0 | 17.8 ±1.3 |
| Qwen2.5-VL-7B-Instruct (Bai et al., 2025b) | 7B | 14.8 ±1.9 | 4.3 ±0.0 | 8.9 ±0.7 | 23.5 ±2.5 | 6.5 ±4.2 | 3.5 ±3.0 | 14.9 ±0.5 |
| AgentCPM-GUI-8B (Zhang et al., 2025b) | 8B | 8.6 ±1.1 | 16.8 ±0.6 | 14.7 ±0.4 | 29.0 ±0.9 | 5.6 ±2.8 | 3.5 ±3.0 | 17.5 ±0.5 |
| *UI-TARS-1.5-7B Model* (Qin et al., 2025) | | | | | | | | |
| Base Model | 7B | 34.0 ±2.8 | 27.6 ±1.2 | 31.9 ±0.7 | 43.7 ±4.1 | 25.9 ±1.6 | 10.5 ±0.0 | 32.8 ±2.3 |
| w/ GRPO (Shao et al., 2024) | 7B | 36.4 ±1.1 | 30.5 ±2.7 | 34.8 ±1.4 | 51.9 ±4.1 | **28.7** ±3.2 | 12.3 ±3.0 | 38.2 ±2.2 |
| w/ PPO (Schulman et al., 2017) | 7B | 38.9 ±3.7 | 29.4 ±0.6 | 35.0 ±1.1 | 50.8 ±1.6 | 26.9 ±1.6 | 14.0 ±3.0 | 37.4 ±1.8 |
| w/ Android Coach | 7B | **42.6** ±1.9 | **33.7** ±0.6 | **39.4** ±0.8 | 56.3 ±2.5 | 27.8 ±2.8 | **17.5** ±3.0 | **41.1** ±1.8 |

Table 2: Success rates of proprietary and open-source models on AndroidWorld and AndroidLab for mobile GUI interaction tasks. QD is the abbreviation for the query detect type, and OP is the abbreviation for the operation type. Standard deviations are reported in gray subscripts for all models except proprietary ones.

**Actor-Critic Leave-One-Out.** As shown in the last stage of Figure 2, to mitigate the high variance associated with policy gradient updates with $Q_\phi$, it is standard practice to subtract a baseline when estimating the advantage $A(s_t, a_t)$. Conventional Actor-Critic methods typically learn a separate state-value function $V_\psi(s_t)$ to serve as this baseline, *i.e.*, $A(s_t, a_t) = Q_\phi(s_t, a_t) - V_\psi(s_t)$. However, within our *SSMA* framework where multiple actions are evaluated for each state, a more direct and potentially more effective baseline is available. Inspired by Reinforce Leave-One-Out (RLOO) (Ahmadian et al., 2024; Kool et al., 2019), we propose the Actor-Critic Leave-One-Out (ACLOO) advantage estimation method. Specifically, given the set of $k$ actions $\{a_t^1, \ldots, a_t^k\}$ sampled *i.i.d.* from the current policy $\pi_\theta(\cdot|s_t)$, we define the ACLOO advantage estimate for action $a_t^i$ as:

$$\hat{A}_t^i = Q_\phi(s_t, a_t^i) - \frac{1}{k-1} \sum_{j \neq i} Q_\phi(s_t, a_t^j) \quad (3)$$

This ACLOO advantage estimation offers two key benefits within our framework: (1) It eliminates the need for a separate value network while effectively reducing variance without bias (proven in Appendix D); (2) This mechanism inherently captures the *relative quality* among the candidate actions at that state, steering the policy towards learning actions that outperform the average. Then the actions and advantages are used in performing gradient updates on the actor policy $\pi_\theta$ with PPO clip surrogate loss:

$$\mathcal{L}_{policy}(\theta) = \mathbb{E}_t\big[\min\big(\tfrac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)} A_t, \text{clip}(\tfrac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}, 1\pm\epsilon) A_t\big)\big] \quad (4)$$

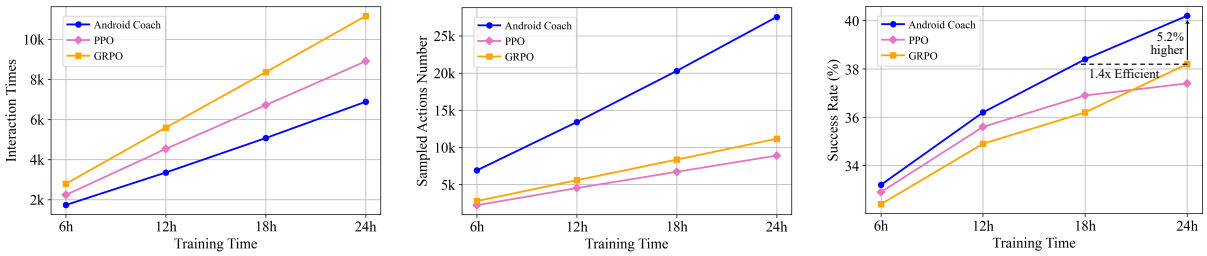
### 3.4 Putting it Together

A pseudocode of our algorithm is provided in Appendix C. Initially, the policy model performs online rollouts in parallel emulators to collect a batch of trajectories. The critic is updated using returns provided by the Process Reward Model and Outcome Verifier with Equation 2. Subsequently, we update the actor by resampling multiple actions for each online state, computing Q-values and advantages with the updated critic, and applying the policy gradient with Equation 4.

## 4 Experiments

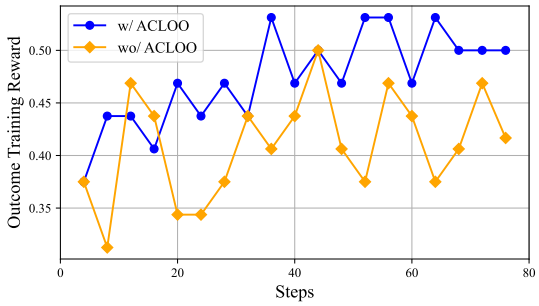
### 4.1 Experiment Setup

**Environment and Benchmarks.** We train the agents in parallel Android emulators running Android 13, where the agent interacts via *uiautomator2*. Following prior work (Xu et al., 2025a; Xiao et al., 2025; Chen et al., 2025a), we evaluate on the AndroidLab (Xu et al., 2025b) and AndroidWorld (Rawles et al., 2025) benchmarks. AndroidLab contains 138 tasks covering both query detection and operation execution, while AndroidWorld includes 116 tasks with easy/medium/hard difficulties and randomized parameters for diverse scenarios. Task success rate (SR) is computed using

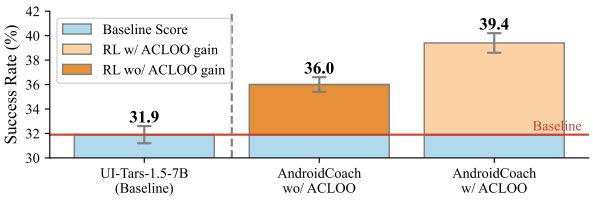



(a) Interaction Times (b) Sampled Actions Number (c) Success Rate

Figure 4: Training efficiency analysis of different methods over training time. We report the relationship between training time and (a) interaction times, (b) sampled actions number, and (c) success rate. The GRPO group size and sample number of ANDROID COACH are both 4. Data is collected on AndroidWorld over UI-TARS-1.5-7B.




(a) Training outcome reward (4-steps average).




(b) Success rate on AndroidLab.

Figure 5: The effect of the ACLOO advantage estimation. (a) The training reward curves. (b) The final success rate gain on AndroidLab.

the built-in rule-based rubrics. Further details are provided in Appendix B.

**Dataset and RL Outcome Verifier.** We construct the training dataset by combining randomized tasks from AndroidWorld with the self-collected AndroidLab tasks to form a collection of 2k tasks. Training outcome reward assignment relies on an Outcome Verifier that uses rule matching for tasks with predefined rules in AndroidWorld and an LLM Judge (GPT-4o) to analyze XML and action trajectories for the others.

**Baselines.** We employ the UI-TARS-1.5-7B (Qin et al., 2025) base model as our starting point. We compare our approach against standard RL baselines for GUI agents including online GRPO and PPO given same RL training time budget, as well as great proprietary and open-source models. To ensure fair comparison, we re-evaluate all open-weight models. We report the mean and standard deviation across three runs.

## 4.2 Main Results

**ANDROID COACH significantly improves the model's performance, making baseline model surpass existing powerful models and methods.** As presented in the Table 2, ANDROID COACH yields substantial performance enhancements for the UI-TARS-1.5-7B. Specifically, it raises the SR on AndroidLab from 31.9% to 39.4% and AndroidWorld from 32.8% to 41.1%. Remarkably, our method enables the model to outperform powerful proprietary models on AndroidWorld such as Claude-Sonnet-4 with Set-of-Mark prompting which achieves 41.0%. These results validate the effectiveness of our RL strategy. While methods like GRPO and PPO also demonstrate performance gains, our approach achieves even stronger results given an identical budget of online training time.

***Single State Multiple Actions* paradigm can increase RL training efficiency compared to *Single State Single Action* methods.** As illustrated in Figure 4a, ANDROID COACH conducts fewer environment interactions under the same training time due to more actions sampling and model updates per online states. However, as shown in Figure 4b, the total number of sampled actions in our approach is substantially higher, which means significantly more samples for policy updates under a fixed training budget. Consequently, our method outperforms PPO by 5.2% given the same training time, and achieves a comparable SR with 1.4× higher efficiency than *SSSA* methods including GRPO and PPO, as illustrated in Figure 4c. These results sug-

| Single State $N$ Actions | Training Time | AndroidLab SR(%) | AndroidWorld SR(%) |
|---|---|---|---|
| *number of N* | | | |
| 1 | 1.00x | 34.8 ±0.7 | 36.8 ±1.3 |
| 2 | 1.22x | 35.5 ±1.2 | 38.5 ±2.2 |
| 4 | 1.62x | **37.0** ±0.7 | 39.1 ±0.5 |
| 8 | 2.18x | 37.0 ±0.0 | **39.4** ±2.0 |

Table 3: Analysis of action rollout times. We report total training time and success rates on AndroidLab and AndroidWorld across different numbers of samples.

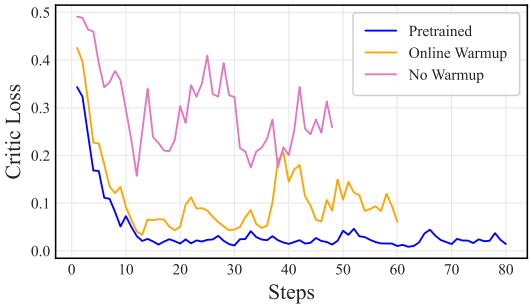



Figure 6: Critic loss with different initialization strategies during joint policy optimization.

| Usage of PRM | SR | ROR | Average Steps |
|---|---|---|---|
| *AndroidLab* | | | |
| wo/ PRM | 37.0 | 83.2 | 12.7 |
| w/ PRM | 39.4 (+2.4) | 89.8 (+6.6) | 11.9 (-0.8) |
| *AndroidWorld* | | | |
| wo/ PRM | 39.1 | 80.0 | 13.0 |
| w/ PRM | 41.1 (+2.0) | 85.1 (+5.1) | 12.5 (-0.5) |

Table 4: Analysis of PRM usage. We report the success rate (SR), reasonable operation ratio (ROR) of steps and average steps on AndroidLab for agent trained with and without process model.

gest that ANDROID COACH provides greater online agent RL training efficiency gains.

### 4.3 Ablation Study

To validate key design choices in our framework, we conduct a set of ablation studies on four key components: the number of action samples, the leave-one-out advantage estimation, the process reward, and the critic initialization strategy.

**Increasing the sample count improves performance with a sub-linear training time increase.** We ablate the number of action samples $N$ for ANDROID COACH without PRM involvement. As shown in Table 3, SR on AndroidLab increases from 34.8% at $N = 1$ to 37.0% at $N = 4$, representing a 6.3% improvement, with AndroidWorld exhibiting a similar trend. However, when $N$ further increases to 8, the performance gain becomes marginal. This can be attributed to the decreasing information gain from additional samples as $N$ grows, leading to a point of diminishing returns. In terms of training time, the cost scales sub-linearly. Specifically, $N = 4$ requires only $1.62\times$ the baseline time, while $N = 8$ requires $2.18\times$, far below the $N\times$ cost typical of standard online methods which adopt *SSSA* paradigm. This highlights the efficacy of decoupling the resampling process from the environment, which enables performance improvements with sub-linear additional cost when increasing the number of samples.

**Incorporating leave-one-out strategy into RL training leads to more stable learning and performant agents.** We assess the effectiveness of the proposed ACLOO strategy by benchmarking it against a vanilla actor-critic implementation without the baseline. To ensure a fair comparison we train all models using identical pipelines and hyperparameters. As demonstrated in Figure 5b removing the ACLOO baseline results in significant underperformance yielding a substantially lower Success Rate of 36.0%. Further analysis of the training curves in Figure 5a reveals that the ACLOO method induces a more stable upward trend in outcome reward throughout training. This underscores the effectiveness of the leave-one-out strategy in stabilizing training and accelerating convergence.

**Pre-training the Q provides a stable initialization, leading to more consistent convergence of its loss during the joint optimization with the policy.** To investigate whether critic pretraining is necessary for policy training, we conduct experiments with three initialization strategies comprising 1) training from scratch without warm-up, 2) warming up via online rollouts, and 3) pre-training on the PRM dataset. Detailed experimental configurations are provided in the Appendix A. As illustrated in the loss curves in Figure 6, utilizing a critic without warm-up directly for policy training results in value divergence. Similarly, restricting the warm-up phase merely to the online environment induces instability during subsequent policy updates. Conversely, pre-training the value model with data aligned to the PRM enables the critic to converge stably when subsequently joint trained with the policy. This highlights the importance of having a well-pretrained critic before the RL phase.

**Using process rewards in training boosts agent performance and leads to more reasonable inter-**

**mediate steps.** We ablate the effect of the PRM as shown in Table 4. The agent trained with the PRM gets higher SR than without PRM on both AndroidLab (39.4% *vs.* 37.0%) and AndroidWorld (41.1% *vs.* 39.1%), confirming its quantitative benefit. We also report another metric taken from the AndroidLab benchmark, the reasonable operation ratio (ROR), which evaluates whether an action is reasonable based on the resulting changes on the screen. With PRM, ROR improves by 6.6% and 5.1% respectively on the two benchmarks, suggesting that incorporating PRM can enhance the reasonableness of actions taken by the agent, ultimately improving the final success rate.

## 5 Conclusion

This work introduces ANDROID COACH, a *Single State Multiple Actions* paradigm reinforcement learning framework to improve online training efficiency. To reduce online interactions for action value estimation, we train a reliable value model to estimate the returns of multiple sampled actions without additional emulator overhead. Specifically, we introduce a fine-grained Process Reward Model to guide the critic online training, and a low-variance group-wise advantage estimator to stabilize policy updates. Extensive experimental results demonstrate that ANDROID COACH significantly enhances online training efficiency while leading to substantial improvement in performance.

## Limitations

Despite the effectiveness of ANDROID COACH, there are several limitations that point to future work. First, we improve sample efficiency mainly from an algorithmic perspective, without optimizing the system architecture for large-scale parallelization; integrating our method into advanced engineering pipelines (Fu et al., 2025) could further boost wall-clock training efficiency. Second, unlike methods such as MobileRL (Xu et al., 2025a) and MAI-UI (Zhou et al., 2025), which apply extensive supervised fine-tuning (SFT) on human annotations before RL, we perform RL directly on a base GUI model. Due to budget constraints, we do not study SFT here, though it could, in principle, raise the performance upper bound. Finally, our reward signal depends, to some extent, on the reliability of the Outcome Verifier: occasional hallucinations in GPT-4o make outcome rewards imperfect. Future research should aim to develop more reliable verification methods for online agentic training that avoid LLM judges' hallucinations and the labor-intensive nature of manually designed rules, while maintaining high precision.

## Acknowledgments

This work is supported by the "Leading Goose + X" Science and Technology Program of Zhejiang Province of China (2025C02104).

## References

Arash Ahmadian, Chris Cremer, Matthias Gallé, Marzieh Fadaee, Julia Kreutzer, Olivier Pietquin, Ahmet Üstün, and Sara Hooker. 2024. Back to basics: Revisiting REINFORCE-style optimization for learning from human feedback in LLMs. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12248–12267, Bangkok, Thailand. Association for Computational Linguistics.

Anthropic. 2025. Introducing claude 4.

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

Hao Bai, Yifei Zhou, Li Li, Sergey Levine, and Aviral Kumar. 2025a. Digi-q: Learning vlm q-value functions for training device-control agents. In *International Conference on Representation Learning*, volume 2025, pages 33183–33203.

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

Cong Chen, Kaixiang Ji, Hao Zhong, Muzhi Zhu, Anzhou Li, Guo Gan, Ziyuan Huang, Cheng Zou, Jiajia Liu, Jingdong Chen, Hao Chen, and Chunhua Shen. 2025a. Gui-shepherd: Reliable process reward and verification for long-sequence gui tasks.

Cong Chen, Mingyu Liu, Chenchen Jing, Yizhou Zhou, Fengyun Rao, Hao Chen, Bo Zhang, and Chunhua Shen. 2025b. Perturbollava: Reducing multimodal hallucinations with perturbative visual training. *arXiv preprint arXiv:2503.06486*.

Kedi Chen, Dezhao Ruan, Yuhao Dan, Yaoting Wang, Siyu Yan, Xuecheng Wu, Yinqi Zhang, Qin Chen, Jie Zhou, Liang He, Biqing Qi, Linyang Li, Qipeng Guo, Xiaoming Shi, and Wei Zhang. 2026. A survey of inductive reasoning for large language models. *Preprint*, arXiv:2510.10182.

Petros Christodoulou. 2019. Soft actor-critic for discrete action settings. *arXiv preprint arXiv:1910.07207*.

Kamil Ciosek and Shimon Whiteson. 2018. Expected policy gradients. *arXiv preprint arXiv:1706.05374*.

Ganqu Cui, Lifan Yuan, Zefan Wang, Hanbin Wang, Wendi Li, Bingxiang He, Yuchen Fan, Tianyu Yu, Qixin Xu, Weize Chen, and 1 others. 2025. Process reinforcement through implicit rewards. *arXiv preprint arXiv:2502.01456*.

Deming Ding, Shichun Liu, Enhui Yang, Jiahang Lin, Ziying Chen, Shihan Dou, Honglin Guo, Weiyu Cheng, Pengyu Zhao, Chengjun Xiao, Qunhong Zeng, Qi Zhang, Xuanjing Huang, Qidi Xu, and Tao Gui. 2026. Octobench: Benchmarking scaffold-aware instruction following in repository-grounded agentic coding. *Preprint*, arXiv:2601.10343.

Guanting Dong, Hangyu Mao, Kai Ma, Licheng Bao, Yifei Chen, Zhongyuan Wang, Zhongxia Chen, Jiazhen Du, Huiyang Wang, Fuzheng Zhang, Guorui Zhou, Yutao Zhu, Ji-Rong Wen, and Zhicheng Dou. 2025. Agentic reinforced policy optimization. *Preprint*, arXiv:2507.19849.

Wei Fu, Jiaxuan Gao, Xujie Shen, Chen Zhu, Zhiyu Mei, Chuyi He, Shusheng Xu, Guo Wei, Jun Mei, Jiashu Wang, Tongkai Yang, Binhang Yuan, and Yi Wu. 2025. Areal: A large-scale asynchronous reinforcement learning system for language reasoning. *Preprint*, arXiv:2505.24298.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, and 1 others. 2025. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645(8081):633–638.

Physical Intelligence, Ali Amin, Raichelle Aniceto, Ashwin Balakrishna, Kevin Black, Ken Conley, Grace Connors, James Darpinian, Karan Dhabalia, Jared DiCarlo, Danny Driess, Michael Equi, Adnan Esmail, Yunhao Fang, Chelsea Finn, Catherine Glossop, Thomas Godden, Ivan Goryachev, Lachy Groom, and 37 others. 2025. $\pi^*_{0.6}$: a vla that learns from experience. *Preprint*, arXiv:2511.14759.

Vijay Konda and John Tsitsiklis. 1999. Actor-critic algorithms. *Advances in neural information processing systems*, 12.

Wouter Kool, Herke van Hoof, and Max Welling. 2019. Buy 4 reinforce samples, get a baseline for free!

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, SOSP '23, page 611–626, New York, NY, USA. Association for Computing Machinery.

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 preprint arXiv:2406.03679*.

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*.

Jiahang Lin, Kai Hu, Binghai Wang, Yuhao Zhou, Zhiheng Xi, Honglin Guo, Shichun Liu, Junzhe Wang, Shihan Dou, Enyu Zhou, Hang Yan, Zhenhua Han, Tao Gui, Qi Zhang, and Xuanjing Huang. 2026. Mmdoc-r1: Training agents for long document visual question answering through multi-turn reinforcement learning. *Preprint*, arXiv:2604.13579.

Ilya Loshchilov and Frank Hutter. 2017. Fixing weight decay regularization in adam. *CoRR*, abs/1711.05101.

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.

Quanfeng Lu, Wenqi Shao, Zitao Liu, Lingxiao Du, Fanqing Meng, Boxuan Li, Botong Chen, Siyuan Huang, Kaipeng Zhang, and Ping Luo. 2025b. Guiodyssey: A comprehensive dataset for cross-app gui navigation on mobile devices. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 22404–22414.

Zhengxi Lu, Yuxiang Chai, Yaxuan Guo, Xi Yin, Liang Liu, Hao Wang, Han Xiao, Shuai Ren, Guanjing Xiong, and Hongsheng Li. 2025c. Ui-r1: Enhancing efficient action prediction of gui agents by reinforcement learning.

Zhengxi Lu, Jiabo Ye, Fei Tang, Yongliang Shen, Haiyang Xu, Ziwei Zheng, Weiming Lu, Ming Yan, Fei Huang, Jun Xiao, and Yueting Zhuang. 2025d. Ui-s1: Advancing gui automation via semi-online reinforcement learning.

Run Luo, Lu Wang, Wanwei He, Longze Chen, Jiaming Li, and Xiaobo Xia. 2025. Gui-r1 : A generalist r1-style vision-language action model for gui agents.

Zhihao Luo, Wentao Yan, Jingyu Gong, Min Wang, Zhizhong Zhang, Xuhong Wang, Yuan Xie, and Xin Tan. 2026. Navimaster: Learning a unified policy for gui and embodied navigation tasks. *Preprint*, arXiv:2508.02046.

Weijian Ma, Ruoxin Chen, Keyue Zhang, Shuang Wu, and Shouhong Ding. 2025. Instruct where the model fails: Generative data augmentation via guided self-contrastive fine-tuning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 39(6):5991–5999.

Weijian Ma, Shizhao Sun, Tianyu Yu, Ruiyu Wang, Tat-Seng Chua, and Jiang Bian. 2026. Thinking with blueprints: Assisting vision-language models in spatial reasoning via structured object representation. *Preprint*, arXiv:2601.01984.

OpenAI, Aaron Hurst, Adam Lerer, Adam P. Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, Aleksander Mądry, Alex Baker-Whitcomb, Alex Beutel, Alex Borzunov, Alex Carney, Alex Chow, Alex Kirillov, Alex Nichol, and 400 others. 2024. Gpt-4o system card. *Preprint*, arXiv:2410.21276.

Xue Bin Peng, Aviral Kumar, Grace Zhang, and Sergey Levine. 2019. Advantage-weighted regression: Simple and scalable off-policy reinforcement learning. *arXiv preprint arXiv:1910.00177*.

Yujia Qin, Yining Ye, Junjie Fang, Haoming Wang, Shihao Liang, Shizuo Tian, Junda Zhang, Jiahao Li, Yunxin Li, Shijue Huang, Wanjun Zhong, Kuanye Li, Jiale Yang, Yu Miao, Woyu Lin, Longxiang Liu, Xu Jiang, Qianli Ma, Jingyu Li, and 16 others. 2025. Ui-tars: Pioneering automated gui interaction with native agents. *Preprint*, arXiv:2501.12326.

Changle Qu, Sunhao Dai, Hengyi Cai, Jun Xu, Shuaiqiang Wang, and Dawei Yin. 2026. Matchtir: Fine-grained supervision for tool-integrated reasoning via bipartite matching.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. *Advances in neural information processing systems*, 36:53728–53741.

Chris Rawles, Sarah Clinckemaillie, Yifan Chang, Jonathan Waltz, Gabrielle Lau, Marybeth Fair, Alice Li, William Bishop, Wei Li, Folawiyo Campbell-Ajala, Daniel Toyama, Robert Berry, Divya Tyamagundlu, Timothy Lillicrap, and Oriana Riva. 2025. Androidworld: A dynamic benchmarking environment for autonomous agents. In *International Conference on Representation Learning*, volume 2025, pages 406–441.

John Schulman, Philipp Moritz, Sergey Levine, Michael I. Jordan, and Pieter Abbeel. 2016. High-dimensional continuous control using generalized advantage estimation. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*.

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

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

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. 2025. Hybridflow: A flexible and efficient rlhf framework. In *Proceedings of the Twentieth European Conference on Computer Systems*, EuroSys '25', page 1279–1297, New York, NY, USA. Association for Computing Machinery.

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. *Preprint*, arXiv:2507.05720.

Jinwei Su, Qizhen Lan, Yinghui Xia, Lifan Sun, Weiyou Tian, Tianyu Shi, Xinyuan Song, Lewei He, and Yang Jingsong. 2026. Difficulty-aware agentic orchestration for query-specific multi-agent workflows. *Preprint*, arXiv:2509.11079.

Qiushi Sun, Kanzhi Cheng, Zichen Ding, Chuanyang Jin, Yian Wang, Fangzhi Xu, Zhenyu Wu, Chengyou Jia, Liheng Chen, Zhoumianze Liu, Ben Kao, Guohao Li, Junxian He, Yu Qiao, and Zhiyong Wu. 2025. OS-genesis: Automating GUI agent trajectory construction via reverse task synthesis. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5555–5579, Vienna, Austria. Association for Computational Linguistics.

Gemini Team, Petko Georgiev, Ving Ian Lei, Ryan Burnell, Libin Bai, Anmol Gulati, Garrett Tanzer, Damien Vincent, Zhufeng Pan, Shibo Wang, Soroosh Mariooryad, Yifan Ding, Xinyang Geng, Fred Alcober, Roy Frostig, Mark Omernick, Lexi Walker, Cosmin Paduraru, Christina Sorokin, and 1118 others. 2024. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *Preprint*, arXiv:2403.05530.

Leandro von Werra, Younes Belkada, Lewis Tunstall, Edward Beeching, Tristan Thrush, Nathan Lambert, Shengyi Huang, Kashif Rasul, and Quentin Gallouédec. 2020. Trl: Transformer reinforcement learning. `https://github.com/huggingface/trl`.

Haoming Wang, Haoyang Zou, Huatong Song, Jiazhan Feng, Junjie Fang, Junting Lu, Longxiang Liu, Qinyu Luo, Shihao Liang, Shijue Huang, Wanjun Zhong, Yining Ye, Yujia Qin, Yuwen Xiong, Yuxin Song, Zhiyong Wu, Aoyan Li, Bo Li, Chen Dun, and 93 others. 2025. Ui-tars-2 technical report: Advancing gui agent with multi-turn reinforcement learning. *Preprint*, arXiv:2509.02544.

Yuyang Wanyan, Xi Zhang, Haiyang Xu, Haowei Liu, Junyang Wang, Jiabo Ye, Yutong Kou, Ming Yan, Fei Huang, Xiaoshan Yang, Weiming Dong, and Changsheng Xu. 2025. Look before you leap: A gui-critic-r1 model for pre-operative error diagnosis in gui automation.

Christopher John Cornish Hellaby Watkins and 1 others. 1989. Learning from delayed rewards.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Jinbiao Wei, Yilun Zhao, Kangqi Ni, and Arman Cohan. 2026. Anchor: Branch-point data generation for gui agents. *arXiv preprint arXiv:2602.07153*.

Zhepei Wei, Wenlin Yao, Yao Liu, Weizhi Zhang, Qin Lu, Liang Qiu, Changlong Yu, Puyang Xu, Chao Zhang, Bing Yin, Hyokun Yun, and Lihong Li. 2025. WebAgent-r1: Training web agents via end-to-end multi-turn reinforcement learning. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 7920–7939, Suzhou, China. Association for Computational Linguistics.

Jinyang Wu, Guocheng Zhai, Ruihan Jin, Jiahao Yuan, Yuhao Shen, Shuai Zhang, Zhengqi Wen, and Jianhua Tao. 2026. Atlas: Orchestrating heterogeneous models and tools for multi-domain complex reasoning. *arXiv preprint arXiv:2601.03872*.

Han Xiao, Guozhi Wang, Yuxiang Chai, Zimu Lu, Weifeng Lin, Hao He, Lue Fan, Liuyang Bian, Rui Hu, Liang Liu, Shuai Ren, Yafei Wen, Xiaoxin Chen, Aojun Zhou, and Hongsheng Li. 2025. Ui-genie: A self-improving approach for iteratively boosting mllm-based mobile gui agents.

Bin Xie, Rui Shao, Gongwei Chen, Kaiwen Zhou, Yinchuan Li, Jie Liu, Min Zhang, and Liqiang Nie. 2025. GUI-explorer: Autonomous exploration and mining of transition-aware knowledge for GUI agent. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5650–5667, Vienna, Austria. Association for Computational Linguistics.

Fangzhi Xu, Hang Yan, Qiushi Sun, Jinyang Wu, Zixian Huang, Muye Huang, Jingyang Gong, Zichen Ding, Kanzhi Cheng, Yian Wang, and 1 others. 2026. Odysseyarena: Benchmarking large language models for long-horizon, active and inductive interactions. *arXiv preprint arXiv:2602.05843*.

Yifan Xu, Xiao Liu, Xinghan Liu, Jiaqi Fu, Hanchen Zhang, Bohao Jing, Shudan Zhang, Yuting Wang, Wenyi Zhao, and Yuxiao Dong. 2025a. Mobilerl: Online agentic reinforcement learning for mobile gui agents.

Yifan Xu, Xiao Liu, Xueqiao Sun, Siyi Cheng, Hao Yu, Hanyu Lai, Shudan Zhang, Dan Zhang, Jie Tang, and Yuxiao Dong. 2025b. AndroidLab: Training and systematic benchmarking of android autonomous agents. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2144–2166, Vienna, Austria. Association for Computational Linguistics.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 41 others. 2025. Qwen3 technical report. *Preprint*, arXiv:2505.09388.

Jiabo Ye, Xi Zhang, Haiyang Xu, Haowei Liu, Junyang Wang, Zhaoqing Zhu, Ziwei Zheng, Feiyu Gao, Junjie Cao, Zhengxi Lu, Jitong Liao, Qi Zheng, Fei Huang, Jingren Zhou, and Ming Yan. 2025. Mobile-agent-v3: Fundamental agents for gui automation. *Preprint*, arXiv:2508.15144.

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*, CHI '25, New York, NY, USA. Association for Computing Machinery.

Yuzhe Zhang, Xianwei Xue, Xingyong Wu, Mengke Chen, Chen Liu, Xinran He, Run Shao, Feiran Liu, Huanmin Xu, Qiutong Pan, and Haiwei Wang. 2026. Don't act blindly: Robust gui automation via action-effect verification and self-correction. *Preprint*, arXiv:2604.05477.

Zhong Zhang, Yaxi Lu, Yikun Fu, Yupeng Huo, Shenzhi Yang, Yesai Wu, Han Si, Xin Cong, Haotian Chen, Yankai Lin, Jie Xie, Wei Zhou, Wang Xu, Yuanheng Zhang, Zhou Su, Zhongwu Zhai, Xiaoming Liu, Yudong Mei, Jianming Xu, and 6 others. 2025b. AgentCPM-GUI: Building mobile-use agents with reinforcement fine-tuning. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 155–180, Suzhou, China. Association for Computational Linguistics.

Zhuosheng Zhang and Aston Zhang. 2024. You only look at screens: Multimodal chain-of-action agents. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 3132–3149, Bangkok, Thailand. Association for Computational Linguistics.

Jiani Zheng, Lu Wang, Fangkai Yang, Chaoyun Zhang, Lingrui Mei, Wenjie Yin, Qingwei Lin, Dongmei Zhang, Saravan Rajmohan, and Qi Zhang. 2025. Vem: Environment-free exploration for training gui agent with value environment model.

Hanzhang Zhou, Xu Zhang, Panrong Tong, Jianan Zhang, Liangyu Chen, Quyu Kong, Chenglin Cai, Chen Liu, Yue Wang, Jingren Zhou, and Steven Hoi. 2025. Mai-ui technical report: Real-world centric foundation gui agents. *Preprint*, arXiv:2512.22047.

Yifei Zhou, Andrea Zanette, Jiayi Pan, Sergey Levine, and Aviral Kumar. 2024. Archer: training language model agents via hierarchical multi-turn rl. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org.

# A Implementation Details

## A.1 ANDROID COACH

**Policy Training.** Our policy models are trained via full-parameter fine-tuning on a single node equipped with 8 NVIDIA A100 GPUs with 80 GB memory. We implement the training pipeline using a modified version of the verl framework (Sheng et al., 2025). To optimize efficiency, we employ a number of parallel emulator environments equal to the batch size during training. Additionally, we leverage the vLLM engine (Kwon et al., 2023) to accelerate inference during the rollout phase, utilize the Fully Sharded Data Parallel (FSDP2) backend for distributed training, and enable BF16 mixed-precision. The critic model is also pretrained alone using FSDP2 with data mentioned in Section B and then co-trained with the policy model. If the Process Reward Model is incorporated, it is deployed together with policy model with vLLM engine. During training, the policy model has access to the complete interaction history. All experiments in the main text are conducted using the same settings as the main experiment, unless otherwise specified. Detailed hyperparameters are listed in Table 5. The action spaces for GUI agent are described in Table 6.

**Outcome Verifier.** The Outcome Verifier functions as the mechanism for assigning outcome rewards during training (Cui et al., 2025; Su et al., 2026). For AndroidWorld tasks detailed in Section B, we utilize the official built-in judge based on defined rules. For AndroidLab tasks, we adopt the DigiRL (Bai et al., 2024) methodology and employ GPT-4o as the Outcome Verifier. We provide the complete trajectory containing compressed XML information (Xu et al., 2025b) and actions for each step to enable accurate task completion judgment with prompt in Figure 9. To validate the reliability of this method, we sample 100 trajectories evaluated by GPT-4o and enlist two graduate students who are fluent in English and majoring in STEM fields as human annotators to label them using identical criteria. Each task is compensated with \$0.5. The agreement rate between human annotations and the rewards assigned by the Outcome Verifier is 92%, demonstrating its reliability.

**PRM Training.** Our Process Reward Model (PRM) is trained via full-parameter Supervised Fine-Tuning (SFT) using verl on a single node with 8 NVIDIA A100 GPUs with 80 GB memory. We employ the FSDP2 backend with a batch size of 32 for 2 epochs. We optimize the model using AdamW (Loshchilov and Hutter, 2017) with $\beta_1 = 0.9$, $\beta_2 = 0.95$, and a weight decay of 0.01. The learning rate is scheduled with a cosine decay strategy, peaking at 1e-5 after a warmup phase covering 10% of the total training steps. To enhance stability, we apply gradient clipping with a maximum norm of 1.0.

## A.2 Reproduction

**PPO and GRPO.** We reproduce the PPO and GRPO algorithms following the identical hardware configuration described in Appendix A.1 which utilizes a single node equipped with eight 80GB NVIDIA A100 GPUs. We employ the native implementations provided by the verl framework (Sheng et al., 2025) and utilize the same Outcome Verifier for training reward assignment. For PPO, the value model is also pretrained using the same dataset with ANDROID COACH using Generalized Advantage Estimation (Schulman et al., 2016) with $\lambda$ set to 1.0 which functions equivalently to the Monte Carlo return. We use a constant learning rate of 1e-6 for the actor and 1e-5 for the value model respectively without applying a KL coefficient. For GRPO we set the group size to 4, with learning rate of 1e-6 for the actor. To ensure fair comparison we maintain a consistent training time budget across these experiments. All third-party artifacts used in this work (VERL, AndroidWorld, AndroidLab) are released under the Apache License 2.0. The full license texts are available in their respective official repositories.

**Evaluation of other models.** Since not all existing baselines report performance on both AndroidLab and AndroidWorld benchmarks or provide the necessary granular data, we conduct a re-evaluation of selected open-weight GUI agent models. For Qwen2.5VL-32B (Bai et al., 2025b), UI-TARS-72B-DPO (Qin et al., 2025), Qwen2.5-VL-7B-Instruct (Bai et al., 2025b), and UI-TARS-1.5-7B (Qin et al., 2025), we adopt the identical input format utilized by ANDROID COACH. Conversely, for OS-Genesis-7B-AW (Sun et al., 2025) and AgentCPM-GUI-8B (Zhang et al., 2025b), we adhere to their officially recommended input formats. Furthermore, we ensure the output action space consistent with the official specifications provided by the model publishers. Besides, due to budget constraints, the results of the proprietary models

are taken from UI-S1 (Lu et al., 2025d) and MobileRL (Xu et al., 2025a), which report performance of gemini-1.5-pro, GPT-4o and Claude-Sonnet-4 under the Set of Marks (SoM) strategy.

# B Benchmarks and Data

## B.1 Environment

Following prior work (Xu et al., 2025a; Xiao et al., 2025; Chen et al., 2025a), we construct our environment for training from AndroidLab (Xu et al., 2025b) and AndroidWorld (Rawles et al., 2025). All experiments are conducted in a controlled emulator environment with a pre-configured Android 13 system at API Level 33 equipped with the complete Google Mobile Services suite. The agent interacts with emulators by reasoning (Wu et al., 2026; Ma et al., 2026, 2025) and generating function-call-like commands, which are subsequently executed via the *uiautomator2* tool. The emulators include applications for commonly used tasks such as bookkeeping, navigation, and calendar management. These tasks cover both execution and querying scenarios. After RL, the agent's performance is evaluated using strict, rule-defined matching criteria (Ding et al., 2026; Lin et al., 2026).

## B.2 Benchmarks

**AndroidLab.** AndroidLab serves as an online benchmark platform designed to evaluate autonomous GUI agents within the Android environment. It comprises 138 tasks spanning nine mobile applications including Zoom, Pi Music Player, Bluecoins and so on. Different from AndroidWorld, there is no randomness in evaluation tasks and initialization scenarios, which means each task has a fixed initial state and expected outcome. Performance metrics include Success Rate (SR) and Reasonable Operation Ratio (ROR).

**AndroidWorld.** AndroidWorld functions as a comprehensive online benchmark for autonomous GUI agents featuring 116 tasks across 20 distinct applications. Task categories encompass audio recording, content editing, gaming, and scheduling. To ensure scenario diversity, these tasks are dynamically generated using variable input parameters and adaptive initialization states. We utilize Success Rate (SR) as the primary metric to evaluate agent performance. We also additionally use the Reasonable Operation Ratio (ROR) metric.

## B.3 Data

**AndroidControl.** AndroidControl represents a large-scale dataset consisting of 15,283 demonstrations of everyday tasks involving 833 Android applications. We utilize AndroidControl exclusively to construct our Process Reward training dataset, which is detailed in subsequent paragraphs.

**Policy Training Tasks.** The reinforcement learning phase requires only unsupervised task instructions. We construct our task pool by leveraging the AndroidLab and AndroidWorld environments. Specifically, we automatically generate candidate tasks based on accessible applications in AndroidLab and synthesize additional tasks using randomized parameters within AndroidWorld. Following a manual verification process to ensure feasibility and strictly exclude overlaps with the test set, we compile a final dataset of 2,000 training tasks.

**Process Reward Model Training Dataset.** We compile our dataset from AndroidControl and pre-collected online trajectories, filtering for successful and non-redundant sequences via GPT-4o and manual verification. For each state, we generate eight candidate reasoning-action pairs using UI-TARS-1.5-7B. Candidates are labeled positive if their actions align with the ground truth while mismatches are labeled negative. Following GUIOdyssey (Lu et al., 2025b), we consider coordinates correct if they fall within a distance of 14% of the screen width from the ground truth. We further synthesize high-quality reasoning components for process judgment using GPT-4o. The final dataset comprises 20k samples with a balanced 1:1 ratio between positive and negative examples, as shown in Figure 7. Subsequently, we adapt these data points to pre-train the Q function by mapping the

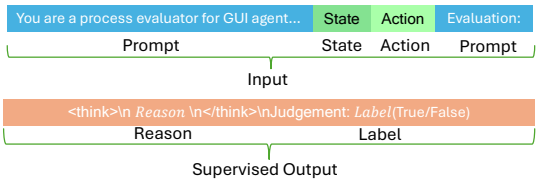



Figure 7: Process Reward Model training data format.

**License.** All third-party artifacts used in this work are released under the Apache License 2.0, and our use is consistent with their standard open-source terms for academic research.

| Component | Hyperparameter | Value |
|---|---|---|
| Data | Max Prompt Length | 32768 |
| Data | Max Response Length | 512 |
| Data | Train Batch Size | 8 |
| Actor/Policy | Strategy (Parallelism) | FSDP2 |
| Actor/Policy | PPO Micro Batch Size/GPU | 1 |
| Actor/Policy | Learning Rate (LR) | 1e-6 |
| Actor/Policy | Gradient Clipping | 1.0 |
| Actor/Policy | Clip Ratio | 0.2 |
| Rollout & Sampling | Sampling Temperature | 1.0 |
| Rollout & Sampling | Max New Tokens | 512 |
| Rollout & Sampling | Max Turns | 25 |
| Rollout & Sampling | Max Pixels | 1270180 |
| Rollout & Sampling | Min Pixels | 256 |
| Reward | Process Reward Weight $\omega_p$ | 0.2 |
| Reward | Outcome Reward Weight $\omega_o$ | 1 |
| Reward | Discount factor $\lambda$ | 0.95 |
| Critic | Learning Rate (LR) | 1e-5 |
| Critic | Clip Range Value | 0.5 |
| Critic | Warmup Ratio | 0.1 |

Table 5: Main hyperparameters in ANDROID COACH.

| **Action** | **Definition** |
|---|---|
| Click(x, y) | Clicks at coordinates (x, y). |
| Scroll(x1, y1, x2, y2) | Scrolls from (x1, y1) to (x2, y2). |
| Drag(x1, y1, x2, y2) | Drags from (x1, y1) to (x2, y2). |
| Type(content) | Types the specified content. |
| Wait() | Pauses for a brief moment. |
| Finished(content) | Marks the task as complete. |
| LongPress(x, y) | Long presses at (x, y). |
| PressBack() | Presses the "back" button. |
| PressHome() | Presses the "home" button. |
| PressEnter() | Presses the "enter" key. |

Table 6: Operation action space for GUI agent in ANDROID COACH.

**Prompt for ANDROID COACH**

**Task Description** 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**
Thought: ...
Action: ...
**Action Space**
click(start_box='<|box_start|>(x1,y1)<|box_end|>')
long_press(start_box='<|box_start|>(x1,y1)<|box_end|>')
type(content='xxx')
scroll(start_box='<|box_start|>(x1,y1)<|box_end|>', end_box='<|box_start|>(x2,y2)<|box_end|>')
open_app(app_name='')
press_home()
press_back()
finished(content='') # Submit the task regardless of whether it succeeds or fails.
**Note**
- Use English in Thought part.
- First summarize your previous actions, then write a small plan and finally summarize your next action (with its save target element) in one sentence in Thought part.
Mobile and UI Agent Interaction History: {interaction_history}

Figure 8: Prompt for ANDROID COACH. This prompt is consistent with the official prompt provided by UI-TARS-1.5-7B.

**Prompt for Outcome Verifier**

**Task Overview:**
You are an expert evaluator for determining the success of GUI tasks. You will be provided with the following information:
1. The task description.
2. Mobile and UI Agent Interaction History including the step-by-step page state in compressed XML format and the agent's action for each step.
**Scoring Rule:**
You need to judge if the UI Agent completed the task based on the interaction trajectory. You should return True or False according to your judgment.
**Output Format:**
<analysis> [Your analysis] </analysis>
</ans> [Your judgment] </ans>
**Current Task Information:**
Task Description: {instruction}
Mobile and UI Agent Interaction History: {interaction_history}

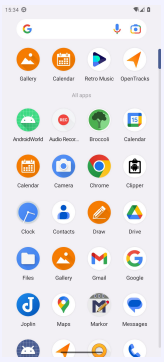
Figure 9: Prompt for outcome verifier.

**Prompt for Process Reward Model**

**Role Definition:** You are a meticulous evaluator for an Android GUI automation agent. Your primary mission is to analyze the agent's reasoning and proposed action in the context of a given task and the current user interface. You must determine if the agent's action is a correct and logical step towards completing the task, and judge whether the operation conforms to Android system specifications.
**Input Data:** You will be provided with: 1. Instruction: The high-level goal. 2. Screenshot: A visual representation of the current GUI state. 3. Agent's Thought and Action: The reasoning process and the specific `Action` intended.
**Evaluation Criteria:** Your output format should be `<think>...thought process...</think> judgment:True or False`.
**Return `True` if:** The proposed `Action` is logical, relevant, and productive based on a correct interpretation of the `Screenshot`.
**Return `False` if:** The action is incorrect (illogical, misinterpretation of UI, redundant, or counter-productive).

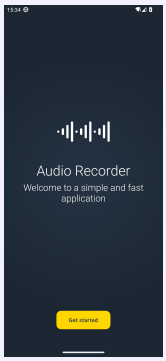
---

**Example 1 (Correct Action):**
**Instruction:** Record an audio clip and save it with name "F3tb_presentation.m4a" using the Audio Recorder app.
**Screenshot:** [Home screen with "Audio Recorder" icon visible.]
**Agent's Thought and Action:** Thought: I need to open the app. I see the "Audio Recorder" icon. I will tap it. Action: `click(start_box='(635,520)')`
**Evaluation:** <think> Agent's Logic Analysis: The agent correctly identifies the first step and locates the icon. Action Validation: The action is the most direct and logical step. </think> judgment:True



Example 1 Screenshot

**Example 2 (Incorrect Action):**
**Instruction:** Record an audio clip and save it with name "F3tb_presentation.m4a" using the Audio Recorder app.
**Screenshot:** [Inside Audio Recorder app. "Get Start" button visible.]
**Agent's Thought and Action:** Thought: I don't see the app icon, so I must be in the wrong place. I need to go back. Action: `press_back()`
**Evaluation:** <think> Agent's Logic Analysis: The agent used flawed reasoning; it failed to recognize it is already in the app. Action Validation: `press_back()` exits the app, which is counter-productive. Correct action is clicking get start. </think> judgment:False




Example 2 Screenshot

---

**Now, evaluate the following scenario:**
**Instruction:** {instruction}
**Agent's Thought and Action:** {agent_thought_and_action}
**Evaluation:**

Figure 10: Prompt for Process Reward Model.

**Prompt for GPT-4o to Generate Reason for PRM Training**

**Task Overview:**
You are an expert evaluator for an Android GUI Agent. Your mission is to analyze the agent's behavior at each step and determine if its intended action is correct and logical for accomplishing a given task.

**1. Context**
The agent interacts with an android operating system. It receives a high-level instruction from a user and attempts to complete it by performing a sequence of actions on the device's GUI. The process is sequential: observe screen → generate thought/action → execute action → repeat. The agent's interaction is restricted to predefined actions:
click(start_box='[x1, y1, x2, y2]'), long_press(start_box='<|box_start|>(x1,y1)<|box_end|>'), scroll(start_box='<|box_start|>(x1,y1)<|box_end|>', end_box='<|box_start|>(x2,y2)<|box_end|>'), drag(start_box='<|box_start|>(x1,y1)<|box_end|>', type(content='xxx'), open_app(app_name=''), press_back(), press_home(), finished(content='xxx')

**2. Your Task**
You will be provided with four pieces of information for a single step: instruction, screenshot, historical actions, ground truth action, agent's thought and action and ground truth judgment. Your task is to critically evaluate the agent's step using an analytical process: 1) Understand the instruction. 2) Analyze the History and Screenshot. 3) Evaluate the agent's action given the ground truth judgment. Output **1 (Correct/Reasonable Action)** if: Direct progress, logical sub-task, handling obstacles, sensible exploration, appropriate action type, error correction. Output **0 (Incorrect/Unreasonable Action)** if: Contradicts instruction, illogical action, flawed reasoning, no progress, inefficient action, wrong action type, perpetuating errors.

**3. Output Format**
Your response MUST be a valid JSON object with two keys:
1. "result": Integer (1 for Correct, 0 for Incorrect)
2. "reason": Concise string explaining your judgment

**4. Examples**

**Example 1:** Correctly Handling an Obstacle (Correct)
**Instruction:** Open a new private tab in the browser.
**Agent's Thought and Action:** Thought: A system dialog has appeared asking to set the default browser. This is not required for my current task. I will dismiss this dialog by clicking 'Cancel' to continue. Action: click(start_box='(700, 1300)')
**Expected Output:** {"result": 1, "reason": "The agent correctly identified a system dialog as an obstacle not relevant to the main task. Its decision to dismiss the dialog is a logical and efficient way to handle the interruption and proceed with the instruction."}

**Example 2:** Error Based on Historical Context (Incorrect)
**Instruction:** Delete the contact "John Smith".
**Agent's Thought and Action:** Thought: A confirmation pop-up has appeared to delete the contact. I need to confirm the deletion to complete the task. Action: click(start_box='(600, 1200)')
**Expected Output:** {"result": 0, "reason": "The action perpetuates a previous error. The agent is about to delete 'John Appleseed', but the instruction was to delete 'John Smith'. This stems from an incorrect selection in a previous step, and proceeding would fail the task."}

**Scoring Rule:**
You need to judge if the agent's current action is correct and logical based on the instruction, screenshot, historical context, and the agent's reasoning.

**Output Format:**
<analysis> [Your analysis] </analysis>
</ans> [Your judgment] </ans>

**Your Turn:**
Instruction:{instruction}. Historical Actions:{history}. Ground truth action:{ground_truth}. Agent's Thought and Action:{agent_thought_and_action}. Ground Truth judgment:{judgment}.
**Your Output:**

Figure 11: Prompt for GPT-4o to generate reason for PRM Training.

## C Pseudocode

**Algorithm 1** Android Coach Framework

**Initialize:** Initial actor parameters $\theta$, initial critic parameters $\phi$.
**Given:** Instruction Pool $\mathcal{I}$, process reward model(PRM), outcome reward verifier(OV).
**for** each iteration **do**
    **# Phase 1: Actor Data Collection**
    **for** each Android step $t$ **do**
        Given $\mathcal{I}$, execute action $a_t \sim \pi_\theta(\cdot|s_t)$ and store finished trajectory $\tau$. ▷ Online Interaction
    **end for**
    **# Phase 2: Assign Returns**
    **for** each trajectory $\tau$ **do**
        $R_{\text{outcome}} \leftarrow \text{OV}(\tau, \text{Instruction})$
        **for** step $t \leftarrow T$ **downto** 1 in trajectory $\tau$ **do**
            $r_t^p \leftarrow \text{PRM}(a_t, s_t)$
            $R_t \leftarrow$ MC estimation $\left(r_p^{t:T}, r_{\text{O}}\right)$
            Add $(s_t, a_t, R_t)$ to replay buffer $\mathcal{D}$.
        **end for**
    **end for**
    **# Phase 3: Update Critic**
    **for** each critic step **do**
        Sample batch of $(s_t, a_t, R_t) \sim \mathcal{D}$.
        Update $\phi$ by clipped MSE loss in Equation 2.
    **end for**
    **# Phase 4: Update Actor**
    **for** each actor step **do**
        Sample batch of states $\{s\} \sim \mathcal{D}$
        Generate $K$ responses: $\{a_1, \ldots, a_K\} \sim \pi_\theta(\cdot|s)$ ▷ Single State Multiple Actions
        Compute advantages $\hat{A}(s, a_i)$ by ACLOO:
            $Q_i \leftarrow Q_\phi(s, a_i)$
            $\hat{A}(s, a_i) \leftarrow Q_i - \frac{1}{k-1} \sum_{j \neq i} Q_j$
        Update $\theta$ by PPO loss in Equation 4.
    **end for**
**end for**

## D Lemma

Let $s_t$ be the current state. We sample $k$ independent and identically distributed (i.i.d.) actions from our policy $\pi_\theta(\cdot|s_t)$:

$$a^{(1)}, a^{(2)}, \ldots, a^{(k)} \sim \pi_\theta(\cdot|s_t)$$

For each $i$-th sample $a^{(i)}$, we compute its Q-value $Q(s_t, a^{(i)})$ and define a leave-one-out baseline $b_i$:

$$b_i = \frac{1}{k-1} \sum_{j \neq i} Q(s_t, a^{(j)})$$

The advantage for the $i$-th sample is:

$$\hat{A}^{(i)} = Q(s_t, a^{(i)}) - b_i$$

While the PPO $L^{CLIP}$ objective is inherently biased for stabilization, we justify our choice of the ACLOO estimator by proving that it is statistically sound. Specifically, it is a low-variance estimator that does not introduce any bias when applied to the standard policy gradient theorem. This demonstrates its validity as a high-quality advantage signal. The policy gradient estimator for this sample is:

$$g_i = \hat{A}^{(i)} \nabla_\theta \log \pi_\theta(a^{(i)}|s_t)$$

We will now prove that this estimator $g_i$ is **unbiased** and has **reduced variance**.

### D.1 Proof of Unbiasedness

**Lemma 1** (Unbiased Estimator)**.** *The policy gradient estimator $g_i$ is an unbiased estimator of the true policy gradient $\nabla_\theta J(\theta)$.*

*Proof.* We prove that the expected value of our estimator $g_i$ is equal to the true policy gradient $\nabla_\theta J(\theta)$.

The true policy gradient is defined as:

$$\nabla_\theta J(\theta) = \mathbb{E}_{a \sim \pi_\theta}[Q(s_t, a) \nabla_\theta \log \pi_\theta(a|s_t)]$$

The expectation of our estimator $g_i$ is taken over all $k$ i.i.d. samples:

$$\mathbb{E}[g_i] = \mathbb{E}\left[\left(Q(s_t, a^{(i)}) - b_i\right) \cdot \nabla_\theta \log \pi_\theta(a^{(i)}|s_t)\right]$$

By linearity of expectation, we split this into two terms:

$$\mathbb{E}[g_i] = \mathbb{E}[Q(s_t, a^{(i)}) \nabla_\theta \log \pi_\theta(a^{(i)}|s_t)] - \mathbb{E}[b_i \cdot \nabla_\theta \log \pi_\theta(a^{(i)}|s_t)]$$

**1. Analyzing the first term:** Since $a^{(i)}$ is a sample drawn from $\pi_\theta(\cdot|s_t)$, the first term is, by definition, the true policy gradient:

$$\mathbb{E}[Q(s_t, a^{(i)}) \nabla_\theta \log \pi_\theta(a^{(i)}|s_t)] = \nabla_\theta J(\theta)$$

**2. Analyzing the second term (the bias term $B$):**

$$B = \mathbb{E}[b_i \cdot \nabla_\theta \log \pi_\theta(a^{(i)}|s_t)]$$

The key insight is that our $k$ samples are i.i.d.

- The baseline $b_i = \frac{1}{k-1} \sum_{j \neq i} Q(s_t, a^{(j)})$ is a random variable that depends only on the samples $\{a^{(j)}\}_{j \neq i}$.
- The gradient term $\nabla_\theta \log \pi_\theta(a^{(i)}|s_t)$ is a random variable that depends only on the sample $a^{(i)}$.

Because $a^{(i)}$ is **statistically independent** of $\{a^{(j)}\}_{j\neq i}$, the random variables $b_i$ and $\nabla_\theta \log \pi_\theta(a^{(i)}|s_t)$ are also **statistically independent**.

For independent random variables $X$ and $Y$, $\mathbb{E}[XY] = \mathbb{E}[X]\mathbb{E}[Y]$. Therefore:

$$B = \mathbb{E}[b_i] \cdot \mathbb{E}[\nabla_\theta \log \pi_\theta(a^{(i)}|s_t)]$$

We now compute the expectation of the gradient term:

$$\begin{aligned}
\mathbb{E}[\nabla_\theta \log \pi_\theta(a^{(i)}|s_t)] &= \sum_a \pi_\theta(a|s_t) \cdot \nabla_\theta \log \pi_\theta(a|s_t) \\
&= \sum_a \pi_\theta(a|s_t) \cdot \frac{\nabla_\theta \pi_\theta(a|s_t)}{\pi_\theta(a|s_t)} \\
&= \sum_a \nabla_\theta \pi_\theta(a|s_t) \\
&= \nabla_\theta \left( \sum_a \pi_\theta(a|s_t) \right) \\
&= \nabla_\theta(1) = 0
\end{aligned}$$

Substituting this result back into the bias term $B$:

$$B = \mathbb{E}[b_i] \cdot 0 = 0$$

**3. Conclusion:** The bias term is zero. Thus, the expectation of our estimator is the true policy gradient:

$$\mathbb{E}[g_i] = \nabla_\theta J(\theta) - 0 = \nabla_\theta J(\theta)$$

This proves the estimator is **unbiased**. □

### D.2 Proof of Variance Reduction (via Shift-Invariance)

**Lemma 2** (Variance Reduction)**.** *The advantage estimator $\hat{A}^{(i)}$ is invariant to an arbitrary constant shift $C$ added to the Q-function, which centers the advantage estimates and reduces variance.*

*Proof.* We prove that $\hat{A}^{(i)}$ is shift-invariant. Let $Q'(s,a) = Q(s,a) + C$ be the shifted Q-function for any constant $C \in \mathbb{R}$.

The new advantage $\hat{A}'^{(i)}$ is:

$$\hat{A}'^{(i)} = Q'(s_t, a^{(i)}) - b'_i$$

First, we compute the new baseline $b'_i$ using the shifted $Q'$-values:

$$\begin{aligned}
b'_i &= \frac{1}{k-1} \sum_{j\neq i} Q'(s_t, a^{(j)}) \\
&= \frac{1}{k-1} \sum_{j\neq i} \left( Q(s_t, a^{(j)}) + C \right) \\
&= \left( \frac{1}{k-1} \sum_{j\neq i} Q(s_t, a^{(j)}) \right) + \left( \frac{1}{k-1} \sum_{j\neq i} C \right) \\
&= b_i + \frac{1}{k-1} \left( (k-1) \cdot C \right) \\
&= b_i + C
\end{aligned}$$

Now, substitute $Q'$ and $b'_i$ back into the expression for $\hat{A}'^{(i)}$:

$$\begin{aligned}\hat{A}'^{(i)} &= Q'(s_t, a^{(i)}) - b'_i \\ &= \Big(Q(s_t, a^{(i)}) + C\Big) - (b_i + C) \\ &= Q(s_t, a^{(i)}) - b_i \\ &= \hat{A}^{(i)}\end{aligned}$$

**Conclusion:** Since $\hat{A}'^{(i)} = \hat{A}^{(i)}$, the advantage estimator is invariant to any constant shift $C$. This demonstrates that $\hat{A}^{(i)}$ measures the relative quality of $a^{(i)}$ compared to the average of its peers, effectively centering the advantage values. This centering property dramatically **reduces the variance** of the gradient estimator $g_i$. □