Enhance Chain Of Action-Planning Reasoning Via Iterative Preference Learning

Anonymous ACL submission

Abstract

VLM-based mobile GUI agents excel in GUI interaction by employing a Chain of Action-Planning Thoughts (CoaT) paradigm, which is like System 2 CoT reasoning. Meanwhile, self-training methods are widely used to optimize the CoT process. However, the lack of diverse CoaT data restricts the agent's output space and limits its generalization ability, which is crucial for the self-training sampling stage. Multiple correct answers in the GUI field also make it challenging to train the process reward model, further hindering the optimization of the CoaT process. To address these problems, we first enhance the diversity of agents' output space through three-stage instruction evolution, then obtain high-quality positive and negative pairs at the CoaT action level using a rule-based value calculation algorithm, and leverage iterative DPO training to optimize the agents' preference between different action types. Experiments are performed on the latest CoaT dataset AITZ, long-trajectory dataset AMEX, and comprehensive dataset AndroidControl. Our agent MobileIPL achieves the SoTA results on AITZ, AMEX, and AndroidControl, while also demonstrating strong generalization performance on the out-domain subsets of AndroidControl.

1 Introduction

VLM-based mobile agents (Wang et al., 2023; Ding, 2024) gain considerable attention due to their ability to seamlessly interact with mobile graphic user interfaces (GUIs) and their potential for handling daily tasks autonomously. Since actions cannot be directly inferred from user instructions, GUI agents must decide based on their thoughts corresponding to current GUIs. This Chain Of Action-Planning Thought (CoaT) pattern is similar to the slow-thinking System 2 paradigm CoT in general domains(Xiong et al., 2024). Recent studies based on self-training (Luong et al., 2024) and solution exploration (Xie et al., 2024) can further enhance the quality of this CoT process in complex tasks such as math and code generation. These methods provide additional insights when the performance of pre-training models plateaus, namely the two-stage training framework. The first stage focuses on learning the generation format and building fundamental capabilities, while the second optimizes the reasoning process for greater diversity and quality beyond the original CoT data.

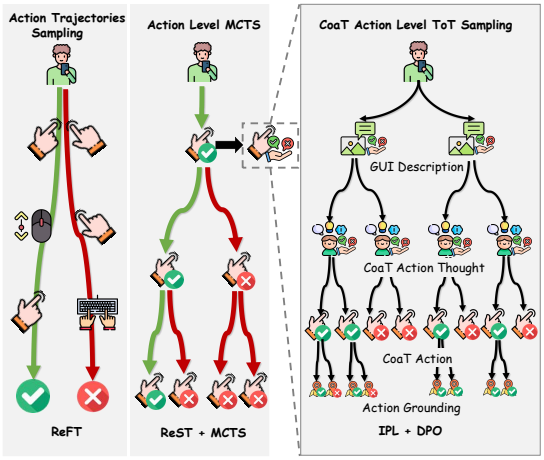

Figure 1: Unlike ReFT, which samples the entire trajectories, and ReST, which performs tree search over all steps in the action space, our method performs sampling at the action level based on CoaT thinking patterns.

However, with the high cost of human annotation, there are no widely available CoaT trajectories in the GUI agent domain (Rawles et al., 2024; Lu et al., 2024), restricting existing agents to Action Models (Lin et al., 2024; Wu et al., 2024a; Shen et al., 2024). At the same time, due to the lack of GUI domain tasks during the pre-training stage, the general VLM's receptive field is limited to regions closely related to the fine-tuning instructions(Zhang et al., 2024b,c). This results in a narrow reasoning space and fixed thinking patterns of agents, further hindering their generalization abilities. The previous GUI agent RL works (Bai

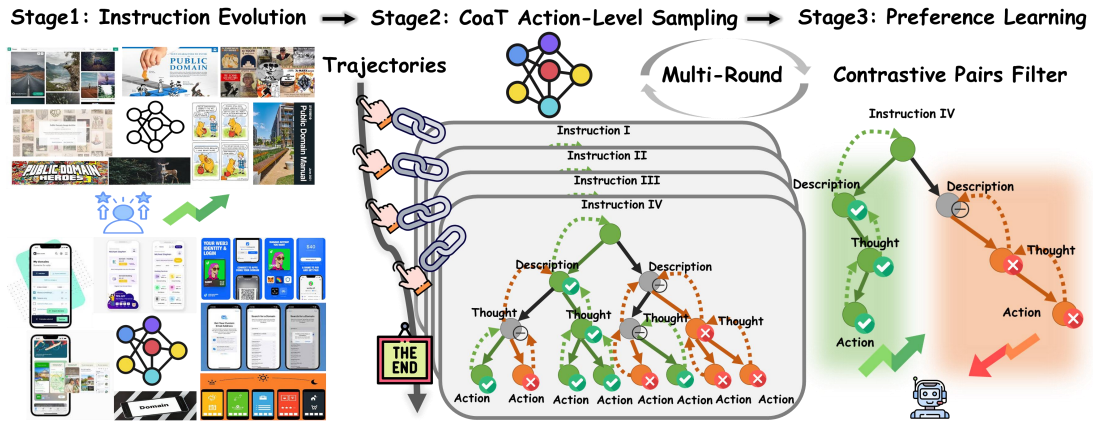

Figure 2: Overview of iterative preference learning framework. The left part presents the process of warm-up fine-tuning a general VLM to a mobile GUI domain agent with basic capabilities. The mid and right parts represent the iterative CoaT action-level sampling and DPO self-training process.

et al., 2024; Wang et al., 2024c) rely primarily on MCTS search over the entire trajectory, neglecting detailed reasoning for individual actions. As shown in Figure 1, the CoaT paradigm has fixed output stages and a specific purpose for each stage, which is different from the unstable CoT output format in general domains. Additionally, the agent's sampling space contains enough potentially correct outputs, but there is still no suitable training method to let the agents fully express this potential during the greedy decoding stage.

To address these limitations, we make the following improvements : (1) **Instruction Evolution**: To prevent the agent from collapsing to regions strongly associated with the instructions, we propose an instruction evolution algorithm (Luo et al., 2024) to construct higher-quality CoaT trajectories and knowledge database on existing datasets such as AITZ, AMEX (Chai et al., 2024), and AndroidControl (Li et al., 2024a). (2) **Action Level Sampling**: Unlike the overall CoT sampling or MCTS (Yu et al., 2024; Putta et al., 2024) over the entire trajectory, we construct tree sampling at the CoaT action level, of which value is computed by predefined rules. (3) **Iterative Preference Learning**: We sample diversified CoaT positive and negative examples by rules and use them for DPO training iteratively. Each stage of the CoaT output has positive and negative examples that are used to train preferences separately. For a completely wrong sampling space, the final ground truth is provided to help the agent to refine the reasoning process.

Overall, our main contributions are as follows:

- We propose an instruction evolution algorithm in the GUI agent domain, which can effectively enhance self-training sampling diversity. We propose higher-quality CoaT datasets named AMEX-CoaT and AndroidControl-CoaT.
- We propose a CoaT Action-level sampling and value calculation method, where the Tree-structure values can be calculated accurately without PRM.
- We propose a GUI agent named MobileIPL, which achieves SoTA results on AITZ, AMEX, and AndroidControl compared to other continuous pre-training agents. MobileIPL achieves improvements of over 6.7%, 3.2%, and 13.4% on out-domain subsets of AndroidControl.

2 Related Work

2.1 GUI Agent

With the rapid development of vision-language models (VLMs), researchers build mobile GUI agents (Yang et al., 2023; Zheng et al., 2024; Qin et al., 2025; Team) and multi-agent frameworks (Ding, 2024; Li et al., 2024b; Wang et al., 2024a) based on closed-source VLMs. Meanwhile, some researchers focus on training agents with stronger element grounding (Cheng et al., 2024; Wu et al., 2024b), page navigation (Niu et al., 2024; Lu et al., 2024; Gou et al., 2024; Wang et al., 2025), GUI understanding (You et al., 2024; Baechler et al., 2024) and task planning capabilities (Zhang et al., 2024d; Nong et al., 2024; Xu et al., 2024; Qinghong Lin et al., 2024; Dorka et al., 2024) based on open-source VLMs. Our method organizes trajectory data into multi-rounds of dialogues based on the CoaT thinking pattern, preventing the agent becomes an action model with limited capabilities.

2.2 Reinforcement Learning

The RL algorithms used to align with human preferences include DPO (Rafailov et al., 2024), IPO (Azar et al., 2024), KTO (Ethayarajh et al., 2023), and PPO (Schulman et al., 2017). ReST-MCTS* (Zhang et al., 2024a) focuses on the higher-quality step reward where the process reward model is important. Xie et al. (2024) labels the preference via MCTS based on feedback from self-evaluation. For GUI agents, Bai et al. (2024) and Wang et al. (2024c) use online trajectory collection to improve the generalization of agents whose process is very slow. Wu et al. (2025) uses DPO training to compare the quality of multiple actions. Our method uses DPO to optimize the agent's detailed thinking process by offline sampling at the coat action level, while the step value is calculated directly by rules without unstable PRMs.

3 Task Formulation

Every GUI trajectory $\mathcal{T}$ contains several images u, actions $\hat{a}$, and a task instruction I, which can be represented as:

$$\mathcal{T} = \{I, u_0, \hat{a}_0, u_1, \hat{a}_1, \cdots, u_n, \hat{a}_n\} \quad (1)$$

We formulate action $\hat{a}_i$ in the CoaT reasoning process as a multi-round dialogue $\hat{a}_i = [s_1, s_2, s_3, s_4]$, where s_i represents description, thought, action-decision, and grounding respectively. So the reasoning process is formulated as:

$$s_1 = Description\{P_1, u_i\} \quad (2)$$

$$s_2 = Thought\{P_2, u_i, I, \hat{a}_0, \cdots, \hat{a}_{i-1}, s_1\} \quad (3)$$

$$s_3 = Action\{P_3, u_i, I, s_1, s_2\} \quad (4)$$

$$s_4 = Grounding\{P_4, u_i, I, s_1, s_2, s_3\} \quad (5)$$

P represents each round of dialogue input prompt, I is the task instruction, u is the current GUI, and $\hat{a}_i$ is the step i history action. When the reasoning process ends, the final s_4 is recorded as $\hat{a}_{n+1}$, and step i moves one step forward on the trajectory $\mathcal{T}$. Without s_i thinking process, the reasoning of action $\hat{a}$ is less accurate, and directly using s_i for SFT makes the agent's thinking pattern over rigid.

4 Methodology

As shown in Figure 2, our method uses a three-stage instruction evolution to enhance the diversity of each round dialogue output, avoiding the agent focusing only on the thinking patterns contained in the CoaT data. CoaT action-level sampling and rule-based value calculation are used to optimize the quality of each round of dialogue output.

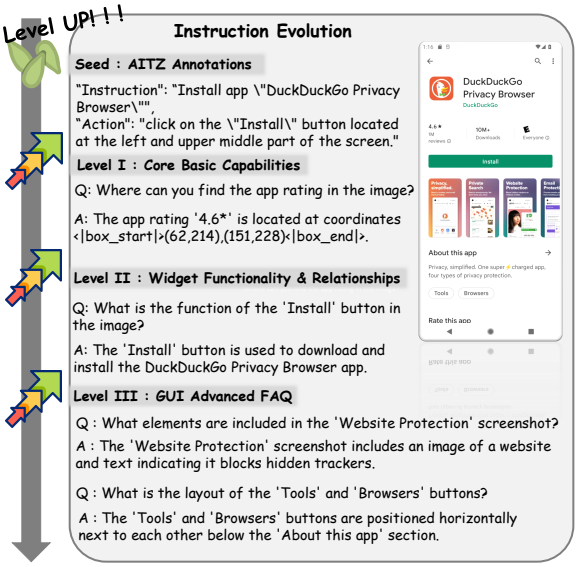

Figure 3: We process a three-stage instruction evolution and knowledge augmentation.

4.1 Instruction Evolution

Unlike mathematical problems, due to the lack of Mobile GUI-specific data during the pre-training stage, agents fail to generate diverse sampling thoughts like general domains. To address this limitation, we supplement the existing training screenshots in the trajectories $\mathcal{T}$ with annotated Q&A through instruction evolution and diversify the rewritten instructions to generate the instruction evolution data $\mathcal{Q}$. Specifically, as shown in Figure 3, the evolution process consists of three levels: **Level I: General GUI Q&A tasks**. Grounding, Reference (Ref), and Page Descriptions are aimed at enhancing the agent's foundational capabilities. These tasks (Liu et al., 2024; Yang et al., 2024) are proven to be the core capabilities of GUI agents during the pre-training. **Level II: Widget caption and relationships**. Descriptions of widget functions and the nested partition relationships between widgets. This task is designed to help the agent understand the relationships between buttons, as previous work (Deng et al., 2024) has found that agents tend to click on the textview, even in scenarios where the textview and the button are separate. **Level III: GUI advanced FAQ**. The advanced FAQ includes more complex Q&A, such as descriptions of part of the structural framework of the page, expectations and predictions about page

navigation triggered by control interactions.

4.2 Warm-up Supervised Fine-tuning

To develop an agent with basic task capabilities and expand the output sampling space, we mix $\mathcal{T}$ and the instruction evolution data $\mathcal{Q}$, then perform supervised fine-tuning (SFT) on $\mathcal{D} = \left\{\mathcal{T}, \mathcal{Q}\right\} = \left\{(u, e)^{(i)}\right\}_{i=1}^{|\mathcal{D}|}$, where u represents the prior knowledge (instructions, screenshot and action history) from $\mathcal{T}$ or the questions from $\mathcal{Q}$. e is the reasoning process from $\mathcal{T}$ or the answer from $\mathcal{Q}$ which is organized into multi-round dialogues. The SFT loss can be computed as:

$$\mathcal{L}_{SFT}(\theta) = -\mathbb{E}_{e\sim\mathcal{D}}[\log \pi_\theta(e|u)].$$

To ensure output diversity, we select an earlier checkpoint with better potential correct space and diverse output to serve as the seed policy model. More details can be seen in Appendix A.

4.3 CoaT Action Level Sampling

After the warm-up SFT stage, agents build basic capabilities for GUIs and can sample outputs based on predefined inputs for each round of dialogue.

Sampling And Value Acquisition. We sample each action on the trajectory along with the CoaT (Zhang et al., 2024c). This allows us to build value functions for different tasks and reduces the pressure on the agent to output such long sentences at once. The K sampling results $(\hat{s}_t|\hat{s}_{1:t-1})^K$ at step t can be expressed as:

$$\hat{s}_t = \left\{(\hat{s}_t^{(k)} \mid \hat{s}_0, \cdots, \hat{s}_{t-1})\right\}_{k=1}^{K} \tag{6}$$

Naturally, the final step in CoaT (the leaf node in the sampling tree) can get a value compared with the ground truth action a^*, which is then propagated back to other intra-nodes. The formula for the value of leaf nodes is as follows:

$$v(s_t) = \begin{cases} 1, & d(s_t, a^*)_{[x,y]} <= d_{min} \\ 0.7 - 0.5\frac{d(s_t,a^*)_{[x,y]}}{d_{max}-d_{min}}, & d(s_t, a^*)_{[x,y]} > d_{min} \\ 0.2 + 0.8F_1, & F_1(s_t, a^*)_{text} \\ 1, & dir(s_t \sim a^*)_{scroll} \\ 0.1, & type(s_t \sim a^*) \\ 0, & else \end{cases} \tag{7}$$

As shown in Figure 11, the value 1 is assigned for a strict action matched (relative grounding box distance less than 0.05 (d_{min}), the F_1 score of input text beyond 0.5 or scroll direction is matched), 0 for unmatched, and other scores are scaled based on the deviation. The recursive calculation formula for intra-nodes is as follows:

$$v(s_{t-1}) = c \cdot \frac{1}{K}\sum_{i=1}^{K} v(s_t^{(k)}) \tag{8}$$

Contrastive Data Filter. After obtaining the sampling tree and node values, we evaluate the quality of the trees and extract contrastive data. The quality of the sampling trees are $\mathcal{R} = \{\alpha, \beta, \gamma\}$, and the number of α, β, γ are calculated as follows:

$$\alpha = \frac{\left|\{\mathcal{S}^{(i)} \mid \forall v_k \in \mathcal{S}^{(i)}, v_k = 1\}\right|}{\sum_{i=1}^{|\mathcal{T}|}|(u,e)^{(i)}|} \tag{9}$$

$$\beta = \frac{\left|\{\mathcal{S}^{(i)} \mid \exists v_k, v_{k'} \in \mathcal{S}^{(i)}, v_k = 1, v_{k'} \neq 1\}\right|}{\sum_{i=1}^{|\mathcal{T}|}|(u,e)^{(i)}|} \tag{10}$$

$$\gamma = \frac{\left|\{\mathcal{S}^{(i)} \mid \forall v_k \in \mathcal{S}^{(i)}, v_k \neq 1\}\right|}{\sum_{i=1}^{|\mathcal{T}|}|(u,e)^{(i)}|} \tag{11}$$

$\mathcal{S}^{(i)}$ and v_k refer to the instruction i sampling tree and the k-th leaf nodes value. α is considered as perfect sampling tree, which can stably output correct thoughts and actions with in-domain trajectories, β represents potential correct trees that can be used to construct contrastive data, and γ denotes sampling trees that require refinement. $\beta + \gamma$ is considered a valid sampling space. In β, actions with a value of 1 and as many diverse action types as possible are extracted as positive samples. In γ, the final ground truth action a^* is used as a positive sample, but the intermediate steps of CoaT are not provided, and the pairs can be represented as:

$$\beta_{pairs} = \langle \hat{s}_t^{(k)} \uparrow, \hat{s}_t^{(k')} \downarrow | \ (\hat{s}_1, \ldots, \hat{s}_{t-1}), \\ v(\hat{s}_t^{(k)}) - v(\hat{s}_t^{(k')}) > 0.3\rangle \tag{12}$$

$$\gamma_{pairs} = \langle a^* \uparrow, \hat{s}_t^{(k)} \downarrow | \ \hat{s}_1, \ldots, \hat{s}_{t-1}\rangle \tag{13}$$

4.4 Iterative Preference Optimization

After CoaT Action-level Sampling, several positive and negative example pairs are collected. During this stage, the agent policy undergoes updates through the above data-pairs, SFT loss, and CoaT-DPO loss (Rafailov et al., 2024). Suppose the agent gets values to pair $\langle +, -\rangle$ at CoaT stage t, which are named s_t^+ and s_t^-; we have the agent performing a comparison for these pairs based on the same

Algorithm 1: Iterative CoaT action-level sampling and DPO self-training.

Input: base VLM π, advanced annotated model R_{SoTA}, step-level trajectory data $\mathcal{T}$, instruction evolution Q&A set $\mathcal{Q}$, number of sampling $\mathcal{K}$, golden action a^*, value function v, the sampled CoaT data D, number of iterations $\mathcal{N}$.

for $i = 1$ to N_0 **do**
  $\mathcal{Q}^* \leftarrow$ instruction_evolution($R_{\text{SoTA}}, \mathcal{T}$) `// instruction evolution by GPT-4o`
  $\mathcal{Q} \leftarrow$ human_evaluation($h, \mathcal{Q}^*$) `// human filter`
end for
$\pi_{S_0} \leftarrow$ Warm-up_SFT($\pi, \mathcal{T}, \mathcal{Q}$) `// fine-tune seed model`
for $n = 1$ to $\mathcal{N}$ **do**
  for $i = 1$ to $|\mathcal{T}|$ **do**
    $D_i \leftarrow$ generate_sampling_thought($\pi_{S_{n-1}}, \mathcal{T}_i, \mathcal{K}$) `// sample CoaT data for reference model`
    $V_i^{leaf} \leftarrow v(D_i, a_i^*)$ `// match and calculate leaf node values using Eq(7)`
    $V_i^{intra} \leftarrow$ recursive_calculate(D_i, V_i^{leaf}) `// recursive intra node values using Eq(8)`
    $D_i^+, D_i^- \leftarrow$ contrastive_data_filter(D_i, V_i) `// filter positive and negative data using Eq(12, 13)`
  end for
  $\pi_{S_n} \leftarrow$ DPO($\pi_{S_{k-1}}, D^+, D^-$) `// DPO self-training reference model`
end for

Output: $\pi_S, D_G, \mathcal{Q}$

thoughts $s_{1:t-1}$, which can be calculated as:

$$\mathcal{L}_{\text{C-DPO}} = -\mathbb{E}_{(s_{1:t-1}, s_t^-, s_t^+)\sim\mathcal{T}_s}\Big[\log\sigma(\beta\log\frac{\pi_\theta(s_t^+|s_{1:t-1})}{\pi_{ref}(s_t^+|s_{1:t-1})} - \beta\log\frac{\pi_\theta(s_t^-|s_{1:t-1})}{\pi_{ref}(s_t^-|s_{1:t-1})})\Big], \tag{14}$$

The final loss combines DPO and SFT losses:

$$\mathcal{L} = \mathcal{L}_{\text{C-DPO}} + c \cdot \mathcal{L}_{\text{SFT}} \tag{15}$$

To further refine the agent's performance post-optimization, we employ the updated agent as the new base agent to continue collecting contrastive CoaT-action level pairs for additional DPO training. This iterative process is maintained until the agent reaches the performance bottleneck on the testset.

5 Experiments

5.1 Dataset And Evaluation

Dataset and Metrics. AITZ (Zhang et al., 2024c) is a high-quality trajectory set filtered and re-annotated from **AITW** (Rawles et al., 2024), containing four subsets: general, install, single, and google apps, which also includes five types of actions: click, input, scroll, press, and stop. It is the first CoaT dataset for the mobile GUI domain, where the CoaT's description, thought, action, grounding, and expectations align with the general agent's standard perception, thought, decision-making, tool call, and reflection. **AMEX** (Chai et al., 2024) uses the same apps and action space as AITZ, but its task instructions are more complex and detailed, with an average trajectory length of 15+. **AndroidControl** includes different task types compared to the previous two datasets. In addition, it contains extra actions such as wait and completely unseen out-of-domain test splits. For metrics, we use **Step.Acc**, consistent with Auto-GUI(Zhang and Zhang, 2023), measures the agent's performance and uses **Action Type** to assess the degree of action type matching. This metric effectively evaluates the model's planning ability. We use Qwen2-VL-7B (Wang et al., 2024b) as the base VLM aligning with baselines. We select CogAgent (Hong et al., 2024), AUTO-GUI, Shpagent, OS-Atlas, UIGround, UI-Tars and Fed-MobileAgent as baseline agents. GUI continuous pre-training agents can be further divided into two categories: (1) training the model as a GUI grounding agent, such as OS-Atlas-7B. (2) training the model as a general GUI agent, such as UI-Tars. More details can be seen in Appendix B.

5.2 Main result

AITZ. As shown in Table 1, except for SCROLL and PRESS, our model achieves SoTA performance on all other metrics. The reason for the lower PRESS Acc. is discussed in Section C and Appendix E. Multiple rounds of DPO improve MobileIPL by more than 10 points (55.40% -> 69.15%) compared to the seed model MobileIPL and ablation model Qwen2-VL-7B (60.36% -> 69.15%). Compared to Falcon-UI, which is pre-trained on three million GUIs, MobileIPL-R6 surpasses a performance difference of 0.05%, approaching the performance of 72B models (72.10%). DPO Round R1 yields the most significant improvement, increasing performance by nearly 10 points (55.40% -> 65.36%). The GUI pre-training Grounding model UIGround underper-

Mode	Model	Atomic								
		SCROLL	CLICK		TYPE		PRESS	STOP	Total	
			type	match	type	match			type	match
ZS	CogAgent-18B	56.41	79.90	51.50	67.40	34.00	48.30	4.76	65.86	44.52
	+CoaT	70.22	88.23	66.15	45.80	21.80	45.95	24.60	72.59	53.28
FT	AUTO-GUI	**74.88**	44.37	12.72	73.00	67.80	49.09	60.12	73.79	34.46
	+CoaT	61.40	74.56	32.20	87.20	81.40	57.70	74.40	82.98	47.69
FT	Qwen2-VL-7B	47.50	81.53	59.72	81.96	73.85	**58.22**	67.39	74.26	60.36
	Qwen2-VL-72B	-	-	-	-	-	-	-	89.60	72.10
PF	UI-Tars-7B	52.50	83.03	64.27	89.97	82.76	61.87	74.35	77.59	65.61
	Falcon-UI-7B	-	-	-	-	-	-	-	**84.70**	69.10
	UIGround-7B	58.22	80.94	58.48	82.56	73.85	58.22	68.78	74.54	60.19
IPL	MobileIPL-Seed	42.83	82.48	53.16	82.56	75.29	56.65	61.82	73.14	55.40
	+R1	45.83	**92.52**	71.12	87.77	81.23	23.49	73.55	78.74	65.36
	+R3	49.83	91.64	71.01	87.97	83.16	49.86	77.93	81.37	68.62
	+R6	51.08	91.73	**71.45**	**88.20**	**83.40**	51.69	**78.17**	81.90	**69.15**

Table 1: **Main results of AITZ dataset.** ZS, FT, PF, and IPL are short for zero-shot, fine-tuning, specific domain pre-training, and iterative preference learning, respectively. '-' represents that the agent or evaluation prompt is not open-sourced. Seed means the seed model for sampling and DPO training.

Model	Training Data	Gmail	Booking	Music	SHEIN	News	CM	ToDo	Signal	Yelp	Overall
SeeClick-7B	AITW+External	28.2	29.4	18.1	20.0	30.0	53.1	30.7	37.1	27.4	30.44
SphAgent-7B	AITW	32.1	45.9	46.1	35.1	48.3	61.1	55.9	43.3	42.9	45.63
SphAgent-7B	AMEX	61.7	68.2	77.7	**72.0**	71.9	64.6	**79.6**	71.3	69.6	70.71
OS-Atlas-7B	AMEX	61.1	73.5	77.9	61.6	75.2	66.4	71.0	75.9	72.0	70.33
UIGround-7B	AMEX	70.9	68.8	72.7	63.7	77.7	67.7	63.7	80.1	67.6	69.12
SphAgent-7B	AITW + AMEX	62.4	68.1	76.3	71.9	68.6	67.3	77.6	66.0	64.1	69.14
Qwen2-VL-7B	AMEX	58.0	70.1	76.6	63.8	79.4	66.8	67.8	80.2	**76.6**	69.01
	+ CoaT	75.9	68.1	77.7	66.2	76.8	66.4	77.5	79.6	65.6	70.93
MobileIPL-7B	AMEX (Seed)	57.0	60.2	68.8	63.1	75.0	50.2	65.6	77.7	62.6	62.19
	+ R1	70.4	72.5	75.8	68.2	83.0	68.1	67.8	80.2	70.6	72.02
	+ R3	77.0	**72.2**	78.6	68.0	**85.8**	70.4	72.6	82.1	74.3	74.14
	+ R4	**77.3**	71.8	**80.0**	68.4	85.3	**71.3**	73.5	**82.1**	71.8	**74.29**

Table 2: **Main results on AMEX**. Seed means the seed model for sampling.

forms its base model Qwen2-VL on AITZ downstream tasks (60.19% < 60.36%), but UI-Tars performs better (65.61% > 60.36%) because of its more various pre-training tasks.

AMEX. As shown in Table 2, except for the SHEIN and ToDo applications, our agent achieves SoTA performance on all other applications, surpassing the previous SoTA pre-training SphAgent by 3.58% (70.71% -> 74.29%). This demonstrates the effectiveness of our method. The CoaT reasoning process significantly improves Qwen2-VL performance (69.01% -> 70.93%), which is nearly on par with the GUI domain pretraining model SphAgent (70.71% $vs.$ 70.93%). Meanwhile, IPL Round R1 yields the most significant improvement, increasing performance by nearly 10 points (62.19% -> 72.02%). R3 and R4 show gradual convergence, and R5 results in negative optimization. Notably, UIGround-7B, which also employs GUI pretraining, fails on the AMEX downstream task, showing equal performance compared to the ablation model Qwen2-VL-7B (69.12% $vs.$ 69.01%).

AndroidControl. Table 3 and Table 4 present the results on the AndroidControl in-domain and out-domain test sets, respectively. In the IDD subset, MobileIPL achieves SoTA in both TYPE and Step.Acc (85.8%, 73.6%), surpassing continual pre-training models of the GUI domain such as OS-Atlas (+2.4%), UI-Tars (+1.1%) and Falcon-UI (+0.9%). However, since these models gain Grounding capabilities during the pretraining phase, MobileIPL still underperforms in element grounding compared to OS-Atlas (-2.8%) and UI-Tars (-4.8%). Similar to AITZ and AMEX, the model achieves the most significant performance improvement in IPL-R1 (63.4% -> 72.0%). In out-domain subsets, the pre-trained model OS-Atlas shows only limited improvements over Qwen2-VL (-0.7%, +2.1%, -7.7%), whereas MobileIPL achieves significant gains (+6.0%, +5.3%, +5.7%).

Mode	Model	Type	Grounding	Step.Acc
ZS	Claude*	74.3	0.0	12.5
	GPT-4o	66.3	0.0	20.8
FT	Aria-UI-7B	-	43.2	10.2
	InternVL-2-4B	84.1	72.7	66.7
	Aguvis-7B	-	-	61.5
	PaLM 2S(full)	-	-	65.5
	Qwen2-VL-7B	81.6	68.5	69.1
PF	Seeclick-7B	82.9	62.9	59.1
	OS-Atlas-7B	85.2	78.5	71.2
	Falcon-UI-7B	-	-	72.7
	UI-Tars-7B	83.7	**80.5**	72.5
IPL	MobileIPL-7B (Seed)	79.7	66.1	63.4
	+ R1	85.5	74.5	72.0
	+ R2	85.8	75.2	72.7
	+ R3	**85.8**	75.7	**73.6**

Table 3: High-level instruction experiment results on **AndroidControl IDD**.

Mode	Model	IDD	app-UN	task-UN	categ-UN
FT	PaLM 2S(full)	65.5	58.7	59.7	58.2
	PaLM 2S(LoRA)	70.8	58.5	59.6	57.4
	Qwen2-VL-7B	69.1	61.4	64.1	62.5
PF	FedMobileAgent	54.7	52.3	51.2	49.2
	SphAgent-7B	69.4	57.1	62.9	50.0
	OS-Atlas-7B	71.2	60.7	66.2	54.8
IPL	MobileIPL-7B (Seed)	63.4	59.1	60.6	59.5
	+ R1	72.0	67.0	69.2	67.9
	+ R2	72.7	67.3	69.2	67.9
	+ R3	**73.6**	**67.4**	**69.4**	**68.2**

Table 4: High-level instruction results on **AndroidControl and three unseen out-domain subsets**.

Additionally, compared to in-domain data, MobileIPL exhibits less performance degradation in the out-domain setting. For example, OS-Atlas experiences a 16.4% drop in the Unseen Category subset (71.2% -> 54.8%), while MobileIPL only decreases by 5.4% (73.6% -> 68.2%).

5.3 Ablation Study

Since our method incorporates the Instruction Evolution (Evo), we conduct an ablation study to analyze the contribution of each component. As shown in Table 5, removing IPL leads to a significant drop in overall performance from 69.2% to 60.4%, demonstrating that IPL plays a crucial role. In contrast, removing Evo results in only a marginal performance decrease (-0.1%), indicating that Evo does not directly enhance the model's performance. We also examine the effects of dialogue (Dia) and history (His) information. Removing historical data leads to a notable decline in the Stop metric (-5.7%), suggesting a strong link between stopping decisions and historical context. Additionally, we remove negative samples from IPL-R1, training the model using only fully correct samples for SFT. This results in a 4.0% performance drop,

Model	Scroll	Click	Type	Press	Stop	Total
MobileIPL-R6	51.1	71.5	83.4	51.7	78.2	69.2
- IPL	46.9	59.4	78.6	55.4	67.2	60.4
- IPL, Evo	47.5	59.7	73.9	58.2	67.4	60.3
- IPL, Evo, Dia.	46.8	55.7	81.8	59.2	71.0	59.2
- IPL, Evo, Dia., His.	47.9	56.8	79.0	56.6	65.3	58.9
MobileIPL-R1	45.8	71.1	81.2	23.5	73.5	65.4
- IPL Negative Data	46.9	61.1	74.2	56.6	72.2	61.4

Table 5: Ablation Study on AITZ.

suggesting that negative samples help the model learn how to reason rather than merely memorize (SFT). Furthermore, compared to training on the original dataset with CoaT data (-IPL), the model's performance improves from 60.4% to 61.4%, due to eliminating noise introduced by CoaT in the original dataset.

6 Discussion and Analysis

6.1 Output space sampling

To evaluate the instruction evolution, we analyze the diversity of the sampling space for **Random 1000 steps**, the standard deviation of encoded embeddings, the dimensionality-reduced distribution, and the distribution of $S^{(i)}$ mentioned in Section 4.3. As shown in Figure 4, the thoughts after instruction evolution exhibit a broader space than direct SFT. Additionally, the embedding standard

Figure 4: The top figures show the t-SNE dimensionality reduction distribution of outputs before and after instruction evolution, while the bottom figures present the distribution of standard deviation and accuracy.

deviation within each tree increases significantly compared to the original data (+ 0.158). The diversified outputs do not negatively impact the agent's reasoning process, while the proportion of action sampling that includes the correct answer improves from 72.7% to 77.9%. The bottom-right subplot reflects the distribution of output accuracy. **Con-**

sistently Correct indicates that all samples for the current step match the golden answer, while **Consistently Error** is the opposite. **Both** represents cases where some samples are correct while others are incorrect, which serves as an ideal source for constructing DPO pairs. Compared to 47% on the evolved data, the agent achieves 68.7% convergence on the original data but exhibits a strong polarization(4%). Three-stage instruction evolution significantly expands the sampling space (from 4% to 31%), enabling the self-training process to be effective. More details can be seen in Appendix D.

6.2 IPL Scaling

Although the total Step.Acc steadily increases with IPL iterations, not all action types follow this upward trend. As shown in Figure 5, from the seed model to the first round of IPL, the accuracy of PRESS drops significantly (58.22% → 23.49%), while CLICK increases (53.26% → 71.12%). Inspired by action equivalence matching in AndroidControl, we attempt to analyze the reasons behind this phenomenon. In fact, multiple actions on the same GUI can result in an equivalent outcome. For example, after entering a search query, PRESS ENTER and directly click on a dropdown suggestion with the same keyword both navigate to the same page. We analyze the proportion of steps where the ground truth action is PRESS, but the model outputs a different action. We find that when PRESS accuracy drops to 23.49%, CLICK accounts for 85.10% of the errors in the lower part of the figure.

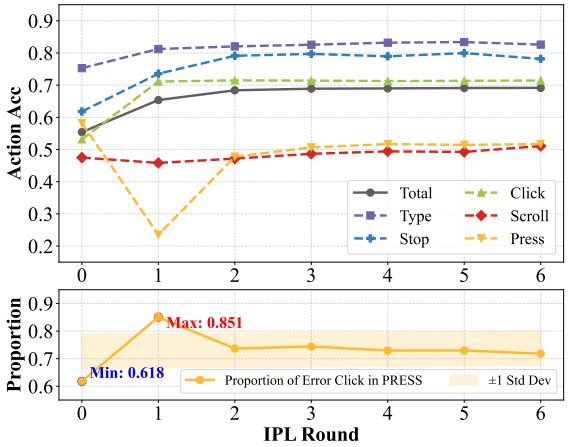

Figure 5: The upper sub-figure shows the changes in **Step.Acc** for action types in AITZ as the number of IPL iterations increases. The lower section presents the proportion of incorrect **CLICK** actions when the ground truth action is **PRESS**.

As more positive PRESS pairs (14.5% -> 18.7%) and less CLICK data (43.3% -> 37.4%) are sampled in subsequent IPL iterations, this error proportion gradually decreases to 73.7%, and PRESS Acc. increases to 47.78%. For detailed statistical data, please refer to Appendix C.

6.3 GUI Continuous Pre-training Agent

As discussed in the previous experimental analysis, continuous pre-training in the GUI domain provides the agents with a stronger base model. However, we still need to explore the compatibility between post-training IPL, instruction evolution, and pre-training. As shown in Figure 6, UI-Tars outperforms Qwen2-VL-7B in all training stages, demonstrating better performance during the instruction evolution phase (62.7% > 55.4%). After four rounds of IPL, UI-Tars Step.Acc improves by 1.4% compared to MobileIPL (69.2% to 70.6%). More importantly, UI-Tars nearly converges after the first round of IPL, significantly reducing the number of sampling and preference learning iterations, thereby keeping the computational cost of post-training within an acceptable range.

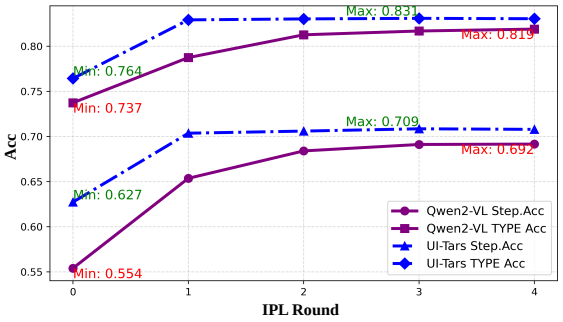

Figure 6: The blue line represents the performance of UI-Tars-7B on AITZ as the seed model and with multi-round IPL training, while the purple line represents Qwen2-VL-7B.

7 Conclusion

In this paper, we propose Mobile Iterative Preference Learning (**MobileIPL**), a self-training GUI agent framework with instruction evolution, CoaT action sampling, and value calculation. We also propose two new datasets: AMEX-CoaT and AndroidControl-CoaT. We extensively validate MobileIPL on AITZ, AMEX, and AndroidControl, demonstrating its effectiveness. Furthermore, the Continuous Pre-training Experiments confirm its mutual reinforcement with pre-training, leading to enhanced performance.

Limitations

Our iterative preference learning method involves multiple rounds of sampling the entire training data, which takes much longer compared to direct SFT, especially when the amount of training data is very large. In our experiments, since some of the continuous pre-training models in the GUI domain do not provide the format and prompt of the training data, the reproduction results may be biased.

Ethics Statement

We have rigorously refined our dataset to remove any elements that could compromise personal privacy, thereby guaranteeing the highest level of protection for individual data. Instruction evolution was completed by AI SoTA close-sourced VLM, to whom we paid the necessary compensation to ensure that the training data was not leaked. The human evaluation of our work was carried out through a meticulously randomized selection of IT professionals. This process ensured a gender-balanced and educationally diverse panel, reflecting a wide spectrum of perspectives and expertise.

References

Mohammad Gheshlaghi Azar, Zhaohan Daniel Guo, Bilal Piot, Remi Munos, Mark Rowland, Michal Valko, and Daniele Calandriello. 2024. A general theoretical paradigm to understand learning from human preferences. In *International Conference on Artificial Intelligence and Statistics*, pages 4447–4455. PMLR.

Gilles Baechler, Srinivas Sunkara, Maria Wang, Fedir Zubach, Hassan Mansoor, Vincent Etter, Victor Cărbune, Jason Lin, Jindong Chen, and Abhanshu Sharma. 2024. ScreenAI: A vision-language model for ui and infographics understanding. *arXiv preprint arXiv:2402.04615*.

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. *arXiv preprint arXiv:2406.11896*.

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

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*.

Shihan Deng, Weikai Xu, Hongda Sun, Wei Liu, Tao Tan, Jianfeng Liu, Ang Li, Jian Luan, Bin Wang, Rui Yan, et al. 2024. Mobile-Bench: An evaluation benchmark for llm-based mobile agents. *arXiv preprint arXiv:2407.00993*.

Tinghe Ding. 2024. MobileAgent: enhancing mobile control via human-machine interaction and sop integration. *arXiv preprint arXiv:2401.04124*.

Nicolai Dorka, Janusz Marecki, and Ammar Anwar. 2024. Training a vision language model as smartphone assistant. *arXiv preprint arXiv:2404.08755*.

Kawin Ethayarajh, Winnie Xu, Dan Jurafsky, and Douwe Kiela. 2023. Human-centered loss functions (halos). Technical report, Technical report, Contextual AI.

Boyu Gou, Ruohan Wang, Boyuan Zheng, Yanan Xie, Cheng Chang, Yiheng Shu, Huan Sun, and Yu Su. 2024. Navigating the digital world as humans do: Universal visual grounding for gui agents. *arXiv preprint arXiv:2410.05243*.

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

Wei Li, William E Bishop, Alice Li, Christopher Rawles, Folawiyo Campbell-Ajala, Divya Tyamagundlu, and Oriana Riva. 2024a. On the effects of data scale on ui control agents. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

Yanda Li, Chi Zhang, Wanqi Yang, Bin Fu, Pei Cheng, Xin Chen, Ling Chen, and Yunchao Wei. 2024b. AppAgent-V2: Advanced agent for flexible mobile interactions. *arXiv preprint arXiv:2408.11824*.

Kevin Qinghong Lin, Linjie Li, Difei Gao, Zhengyuan Yang, Zechen Bai, Weixian Lei, Lijuan Wang, and Mike Zheng Shou. 2024. ShowUI: One vision-language-action model for generalist gui agent. In *NeurIPS 2024 Workshop on Open-World Agents*.

Xiao Liu, Bo Qin, Dongzhu Liang, Guang Dong, Hanyu Lai, Hanchen Zhang, Hanlin Zhao, Iat Long Iong, Jiadai Sun, Jiaqi Wang, et al. 2024. AutoGLM: Autonomous foundation agents for guis. *arXiv preprint arXiv:2411.00820*.

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*.

Run Luo, Haonan Zhang, Longze Chen, Ting-En Lin, Xiong Liu, Yuchuan Wu, Min Yang, Minzheng Wang, Pengpeng Zeng, Lianli Gao, et al. 2024. MMEvol: Empowering multimodal large language models with evol-instruct. *arXiv preprint arXiv:2409.05840*.

Trung Quoc Luong, Xinbo Zhang, Zhanming Jie, Peng Sun, Xiaoran Jin, and Hang Li. 2024. ReFT: Reasoning with reinforced fine-tuning. *arXiv preprint arXiv:2401.08967*.

Runliang Niu, Jindong Li, Shiqi Wang, Yali Fu, Xiyu Hu, Xueyuan Leng, He Kong, Yi Chang, and Qi Wang. 2024. ScreenAgent: A vision language model-driven computer control agent. *arXiv preprint arXiv:2402.07945*.

Songqin Nong, Jiali Zhu, Rui Wu, Jiongchao Jin, Shuo Shan, Xiutian Huang, and Wenhao Xu. 2024. MobileFlow: A multimodal llm for mobile gui agent. *arXiv preprint arXiv:2407.04346*.

Pranav Putta, Edmund Mills, Naman Garg, Sumeet Motwani, Chelsea Finn, Divyansh Garg, and Rafael Rafailov. 2024. AgentQ: Advanced reasoning and learning for autonomous ai agents. *arXiv preprint arXiv:2408.07199*.

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

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. *arXiv e-prints*, pages arXiv–2411.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2024. Direct Preference Optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36.

Christopher Rawles, Alice Li, Daniel Rodriguez, Oriana Riva, and Timothy Lillicrap. 2024. Android in the Wild: A large-scale dataset for android device control. *Advances in Neural Information Processing Systems*, 36.

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

Huawen Shen, Chang Liu, Gengluo Li, Xinlong Wang, Yu Zhou, Can Ma, and Xiangyang Ji. 2024. FalconUI: Understanding gui before following user instructions. *arXiv preprint arXiv:2412.09362*.

Q Team. Qwen2. 5-vl, january 2025. *URL https://qwenlm. github. io/blog/qwen2*.

Bryan Wang, Gang Li, and Yang Li. 2023. Enabling conversational interaction with mobile ui using large language models. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, pages 1–17.

Junyang Wang, Haiyang Xu, Haitao Jia, Xi Zhang, Ming Yan, Weizhou Shen, Ji Zhang, Fei Huang, and Jitao Sang. 2024a. Mobile-Agent-v2: Mobile device operation assistant with effective navigation via multi-agent collaboration. *arXiv preprint arXiv:2406.01014*.

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

Taiyi Wang, Zhihao Wu, Jianheng Liu, Jianye Hao, Jun Wang, and Kun Shao. 2024c. DistRL: An asynchronous distributed reinforcement learning framework for on-device control agents. *arXiv preprint arXiv:2410.14803*.

Wenhao Wang, Zijie Yu, William Liu, Rui Ye, Tian Jin, Siheng Chen, and Yanfeng Wang. 2025. FedMobileAgent: Training mobile agents using decentralized self-sourced data from diverse users. *arXiv preprint arXiv:2502.02982*.

Qinzhuo Wu, Wei Liu, Jian Luan, and Bin Wang. 2025. ReachAgent: Enhancing mobile agent via page reaching and operation. *arXiv preprint arXiv:2502.02955*.

Qinzhuo Wu, Weikai Xu, Wei Liu, Tao Tan, Jianfeng Liu, Ang Li, Jian Luan, Bin Wang, and Shuo Shang. 2024a. MobileVLM: A vision-language model for better intra-and inter-ui understanding. *arXiv preprint arXiv:2409.14818*.

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

Yuxi Xie, Anirudh Goyal, Wenyue Zheng, Min-Yen Kan, Timothy P Lillicrap, Kenji Kawaguchi, and Michael Shieh. 2024. Monte carlo tree search boosts reasoning via iterative preference learning. *arXiv preprint arXiv:2405.00451*.

Weimin Xiong, Yifan Song, Xiutian Zhao, Wenhao Wu, Xun Wang, Ke Wang, Cheng Li, Wei Peng, and Sujian Li. 2024. Watch every step! llm agent learning via iterative step-level process refinement. *arXiv preprint arXiv:2406.11176*.

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*.

Yuhao Yang, Yue Wang, Dongxu Li, Ziyang Luo, Bei Chen, Chao Huang, and Junnan Li. 2024. Aria-UI: Visual grounding for gui instructions. *arXiv preprint arXiv:2412.16256*.

Zhao Yang, Jiaxuan Liu, Yucheng Han, Xin Chen, Zebiao Huang, Bin Fu, and Gang Yu. 2023. AppAgent: Multimodal agents as smartphone users. *arXiv preprint arXiv:2312.13771*.

Keen You, Haotian Zhang, Eldon Schoop, Floris Weers, Amanda Swearngin, Jeffrey Nichols, Yinfei Yang, and Zhe Gan. 2024. Ferret-UI: Grounded mobile ui understanding with multimodal llms. *arXiv preprint arXiv:2404.05719*.

Xiao Yu, Baolin Peng, Vineeth Vajipey, Hao Cheng, Michel Galley, Jianfeng Gao, and Zhou Yu. 2024. ExACT: Teaching ai agents to explore with reflective-mcts and exploratory learning. *arXiv preprint arXiv:2410.02052*.

Dan Zhang, Sining Zhoubian, Ziniu Hu, Yisong Yue, Yuxiao Dong, and Jie Tang. 2024a. ReST-MCTS*: Llm self-training via process reward guided tree search. *arXiv preprint arXiv:2406.03816*.

Jiaqi Zhang, Chen Gao, Liyuan Zhang, Yong Li, and Hongzhi Yin. 2024b. SmartAgent: Chain-of-user-thought for embodied personalized agent in cyber world. *arXiv preprint arXiv:2412.07472*.

Jiwen Zhang, Jihao Wu, Yihua Teng, Minghui Liao, Nuo Xu, Xiao Xiao, Zhongyu Wei, and Duyu Tang. 2024c. Android in the Zoo: Chain-of-action-thought for gui agents. *arXiv preprint arXiv:2403.02713*.

Li Zhang, Shihe Wang, Xianqing Jia, Zhihan Zheng, Yunhe Yan, Longxi Gao, Yuanchun Li, and Mengwei Xu. 2024d. LlamaTouch: A faithful and scalable testbed for mobile ui task automation. *arXiv preprint arXiv:2404.16054*.

Zhuosheng Zhang and Aston Zhang. 2023. You only look at screens: Multimodal chain-of-action agents. *arXiv preprint arXiv:2309.11436*.

Boyuan Zheng, Boyu Gou, Jihyung Kil, Huan Sun, and Yu Su. 2024. GPT-4V(ision) is a generalist web agent, if grounded. *arXiv preprint arXiv:2401.01614*.

A Selection of Seed Policy Model

In our preliminary experimental exploration, we discovered that for the seed policy model, better performance in the SFT phase does not necessarily translate to a higher upper bound in the subsequent IPL phase. This is because as training progresses, the model's output space becomes increasingly aligned with the training data, reducing its diversity in sampling. Consequently, for incorrect instances, the model tends to generate erroneous outputs regardless of the sampling attempts. To address this, we propose a sampling-oriented selection method for the seed policy model, incorporating the following two evaluation metrics:

Sampling Accuracy(Acc_S), which requires the model to hit more correct actions a in the sampled output space $\mathcal{S}$.

$$Acc_S = \frac{\sum_{i=1}^{|\mathcal{T}|} \left| \{ e_j^{(i)} \mid a^{(i)} \sim e_j^{(i)}, e_j^{(i)} \in \mathcal{S}^{(i)} \} \right|}{\sum_{i=1}^{|\mathcal{T}|} |\mathcal{S}^{(i)}|} \quad (16)$$

Sampling Diversity(Div_R), which requires the model to have a more diverse and extensive sampling space. Standard deviation calculation of a single sampled tree $\mathcal{D}ev_{S^{(i)}}$:

$$Dev_{S^{(i)}} = \frac{1}{T} \sum_{t=1}^{T} \text{StdDev} \left(\mathbf{E}(\hat{s}_t^{(k)}) \mid k = 1, \ldots, K \right) \quad (17)$$

Among them, $\mathbf{E}(\hat{s}_t^{(k)})$ represents the representation of the kth sample output of the tth step after the encoder. Calculation of the standard deviation of the set Div_S:

$$Div_R = \frac{1}{N} \sum_{i=1}^{N} Dev_{S^{(i)}} \quad (18)$$

where N is the number of sampled trees in the set $\mathcal{R}$.

B Experiment Setup

Models. Unlike AITZ, we do not compare the CoaT result with the expected page and decide whether to roll back because most actions in real-device scenarios cannot be rolled back without cost. Previous work conducted continual pretraining on Qwen2-VL-7B using GUI domain data, resulting in a stronger base model. In our ablation study, we discuss the impact of continuous pretraining on IPL.

Setup. We conduct hyperparameter searches on AITZ to reproduce the baseline results and find that the optimal learning rate ranges from 3e-5 to 3e-6. Therefore, all baseline fine-tuning experiments adopt this setting. Before IPL, during the instruction evolution stage, we apply LoRA fine-tuning with a LoRA rank of 128. For IPL Stage 1, we use a learning rate between 1e-6 and 1e-7. In subsequent stages, we apply a constant learning rate of 1e-7. The batch size is consistently set to 128. During fine-tuning (including baseline fine-tuning), we enable ViT training, whereas in the IPL phase, we experiment with freezing ViT. For AITZ training, we followed the Falcons' approach, utilizing a maximum 1540×1540 resolution. For other experiments, we reduce the resolution to 1280×720 to optimize computational efficiency. The maximum context length is set to 32K for all experiments. The fine-tuning experiments are conducted for 2 epochs, while IPL training is performed for 1 epoch. Since the large volume of Android control data, we sample 1/5 of the dataset for each IPL training iteration.

CoaT Multi-round Dialogue Prompts.

1. **Page Description.** *Based on the mobile screenshot: Image URL, identify and describe the key elements visible on the screen, including any text, buttons, icons, input fields, or other interactive components.*

2. **CoaT Action Thought.** *Given the task: instruction, and considering the contextual details from the image alongside the full history of previous actions: action history, determine the most logical and effective next step. Focus on providing a clear, actionable, and goal-oriented response to advance the task.*

3. **CoaT Action Description.** *Task: Determine the Most Appropriate Next Step. Based on the previous analysis and the objective, determine the most appropriate next step to achieve the goal. Choose from the following options: - **click**: Select a button or specific UI element by specifying it clearly (e.g., 'click xxx', where 'xxx' is the button name or identifier). - **scroll**: Perform a scrolling action if the required element is not visible, specifying the direction (e.g., 'scroll up', 'scroll down'). - **type**: Input specific text into a field or search bar, specifying the text clearly (e.g., type "content"). - **press**: Interact with device-level buttons such as Home, Back, or Enter, specifying the button (e.g., "press*

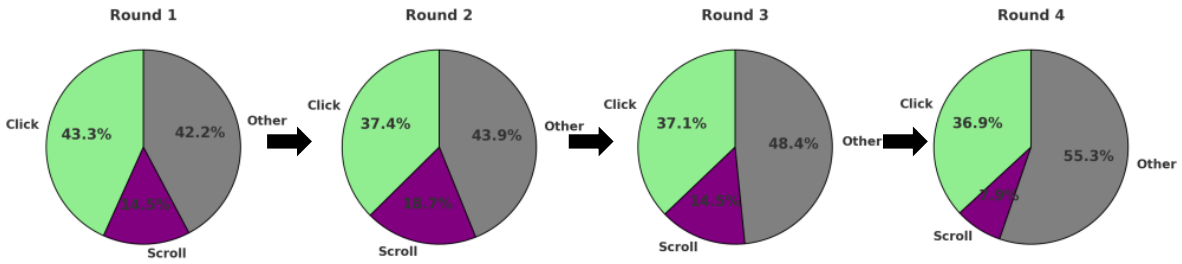

Figure 7: The proportion of CLICK and PRESS actions selected as positive data during the first four rounds of IPL.

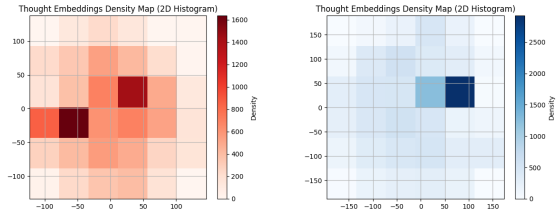

Figure 8: The heatmap at the left represents the sampling before instruction evolution, while the one at the right represents the sampling after instruction evolution.

*Back"). - **stop**: Conclude the task, indicating that the objective has been achieved. Provide the chosen action in the specified format and ensure it aligns with the analysis and the visible UI elements.*

4. **Click Action Grounding.** *As discussed earlier, your task now is to identify the precise screen region coordinates to tap for the action coat action. The coordinates must be integers and strictly within the range of 0 to 1000 for both axes. Please provide your response in the required format: <|box_start|>(top_x, top_y),(bottom_x, bottom_y)<|box_end|>. Ensure your output adheres to these constraints and remains concise.*

Instruction evolution Prompts.

1. **Page Description Annotation.** *I will provide you with a mobile page. Please describe the current page. Your description should include the content of the page and its general functionality. Please note that the descriptions you generate should be of moderate length. Your page description should match the actual image.*

2. **Action Thought Annotation.** ***QUERY**: task, **ACTION HISTORY**: To proceed with the query, your past actions include: action history, **NEXT ACTION**: This is the next action you need to take: coat action, **TASK**: Given the screen and the above information, you have three tasks to do. First, you have to analyze what you have done. Second, you should analyze the screen for relevant details that might pertain to the given query. This includes checking for specific applications, icons, or buttons that are visible and any information or results that are currently displayed on the screen. Tip: If the screen does not have the information you need, you can scroll left or scroll up to try to get the information. Don't answer this logic question by saying that because the provided **NEXT ACTION** is..., therefore, the next action is... You need to think carefully on your own. You must answer the question with suitable lengths and the following format: 'Think: I have done..., Current screen is..., I need to... So the next action is ...' Your final action should be the same as the NEXT ACTION above.*

3. **Q&A Annotation.** *Your goal is to draw inspiration from the given images and image description information to create multiple new questions and answers. This new creation is closely related to the given image and information, but the answers involved should be directly derived from the given information, because UI positions and UI text are one-to-one correspondence. Specifically, you should construct the following three types of questions and answers, a total of 15: 1. the function of some elements in the image. 2. Grounding questions and answers (the coordinates and approximate location of the target in the image). 3. Partial detailed information questions and answers (the structural relationship*

between multiple elements, type, style, etc.). Please try to keep your questions and answers diverse and informative, and ignore the message in the device status bar. Here is the information related to the image: UI positions: {ui positions}, UI text: {ui text}, coat screen desc: {coat screen desc}, Please provide the following information in JSON format with the key questions and answers, and Don't add annotation parsing:

C IPL Scaling Analysis

In the first four rounds of DPO pairs, the number of CLICK positive samples is 1731, 1496, 1485, 1475, while the number of PRESS positive samples is 580, 749, 579, 315. During the first round of IPL, when PRESS accuracy dropped to 10.18%, CLICK has the highest count at 1731. In the following round, CLICK decreased by 235 (1731 $\rightarrow$ 1496), while PRESS increased by 169 (580 $\rightarrow$ 749). After this round of training, PRESS Acc. recovers to 31.07%, leading to a gradual decline in the proportion of this action type in the following round of sampling. In subsequent rounds, their ratio gradually stabilizes. For detailed sampling data changes, please refer to Figure 7.

D CoaT Action Level Sampling

As shown in Figure 8, before instruction evolution, the distribution is highly concentrated, with only 8 points exceeding 1000 (including 3 points above 1200). After instruction evolution, the distribution becomes more balanced, with 20 points exceeding 1000 (including 2 points above 1500).

Potential correct space ratio. The proportion of $|\alpha| + |\beta|$ represents the potential correct space on the training data, and the change of this metric can clearly express the agent's ability to repair and reason out the correct process based on the correct answer.

E Case Study

Unstable annotation preferences. As shown in Figure 9, the left section illustrates two different annotation preferences when searching for an app from the Home Page: **SCROLL UP** and **SCROLL LEFT**, leading to different destination pages. The right part shows the overall preference distribution when annotators need to find an app. In rare cases, the annotation involves clicking on Google Play Store to perform a search. This phenomenon is quite common because, fundamentally, the task completion paths for a UI Agent are diverse. This is also the key difference between online evaluation and offline data evaluation. From this, we observe that RL training on data with unstable preferences performs worse than SFT (e.g., AITZ SCROLL). This is because the DPO pair training method inherently attempts to correct errors in sampled preferences. As a result, the agent oscillates between two decisions when encountering the same GUI and instruction, failing to achieve consistent alignment.

Action Equivalence. Unlike Unstable Annotation Preferences, where different actions lead to different but equivalent pages, the issue here arises from annotators' random labeling habits in the training data, preventing the model from learning a consistent preference. Action Equivalence refers to the phenomenon where multiple actions on the same page can lead to the target page. However, since only one action is annotated as correct, other valid actions are mistakenly treated as incorrect. As shown in Figure 10, after entering a search query, clicking on a suggested item in the recommendation bar, and pressing the Enter key on the keyboard produce the same effect. Similarly, when navigating back, clicking the on-screen back button and pressing the hardware back button yield the same outcome.

Action Level Sampling. As shown in Figure 4, unlike mathematical reasoning, the CoaT process may not exhibit clear logical or computational errors. For a given action, a sampling CoaT data may produce hallucinations (Page Description) due to insufficient detail in the page description or fabricated elements; generate repetitive thoughts (Action Thought) due to neglecting action history; describe the wrong relative position of the correct element (CoaT Action); or misgrounding an element (Grounding), which is then classified as a negative sample. At the same time, outputs with more detailed and accurate descriptions, diversified thoughts, and different ways of describing the same widget are classified as positive samples. Negative examples may be disadvantageous compared to positive examples, for example, because the description of the page is not detailed enough or the positioning of the elements is not accurate enough. At the same time, the wrong process may also give the correct result, but this is a very rare case. In this example, negative samples are generated due to the following three reasons: (1) **Rough page**

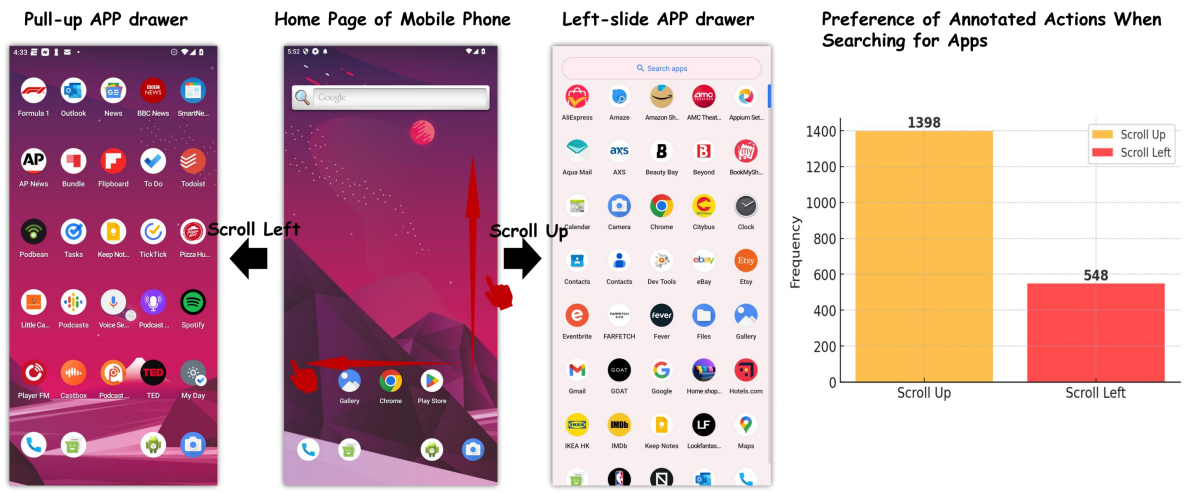

Figure 9: The left figure shows an example of unstable annotation preferences in **AMEX**, while the right figure presents the proportion of this type of annotation.

description: The page contains eight app icons, but the agent's description includes only four apps: Play Store, Gmail, Phone, and YouTube; (2) **Hallucinated Thought:** The agent is unclear about its current page location. In reality, it is on the Home page, but it mistakenly believes it is in the Play Store (e.g., "The Play Store app is already open"). (3) **Fabricated Position and Elements:** The agent generates the action "Click on the 'Spotify' app", even though there is no Spotify icon on the current page. This hallucination may stem from the instruction. Additionally, the Play Store icon should be located at the lower left part of the screen, but the agent incorrectly describes it as being in the middle and lower middle part.

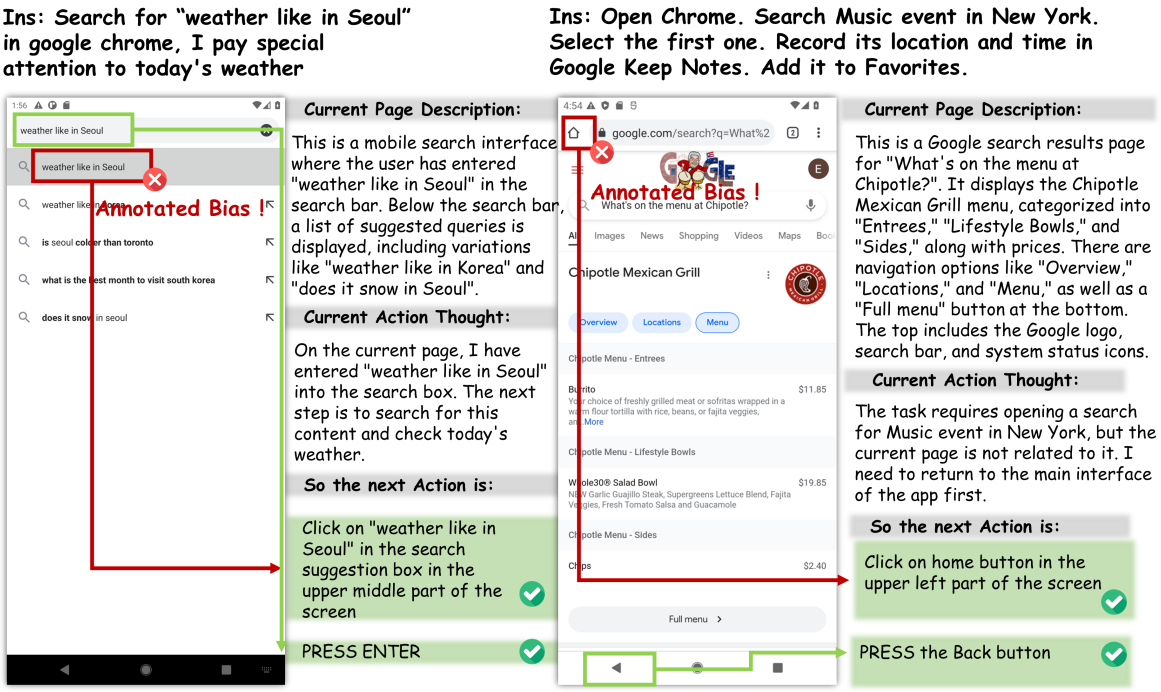

Figure 10: An example from **AITZ** demonstrates that when the task and image are the same, multiple actions may navigate to the same page.

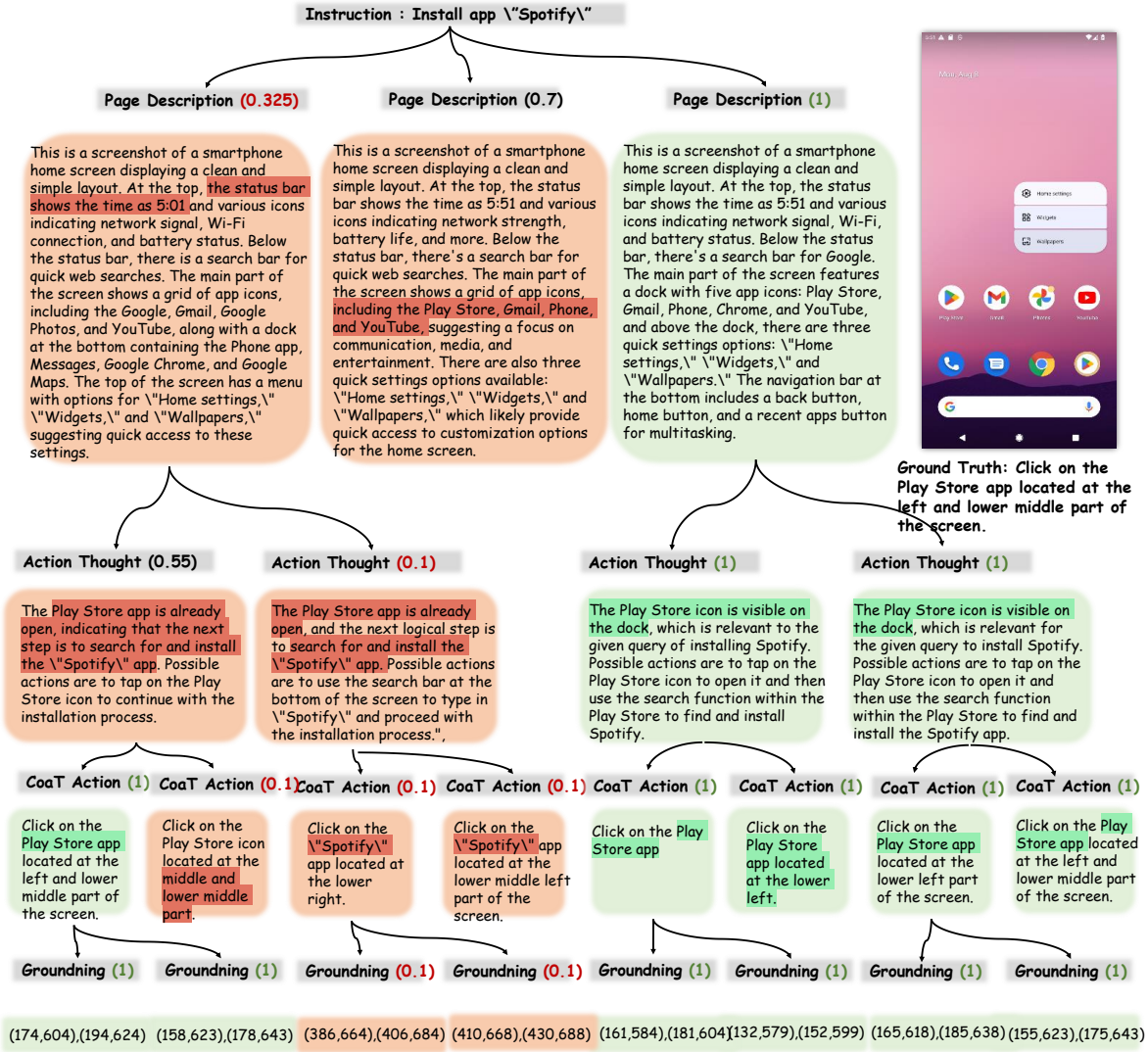

Figure 11: A sampling tree from **AITZ** demonstrates how the value is calculated.