Personalizing GUI Agents with Human-like Contrastive Interaction Trace

Anonymous ACL submission

Abstract

Personalized GUI agents show strong potential for assisting users in complex digital environments, yet learning fine-grained user preferences from limited interaction data remains a significant challenge. Existing approaches often rely on manual prompt engineering or synthetic interaction data that fail to capture realistic human behaviors, limiting their robustness and generalization. In this work, we propose PACUT, a novel approach for personalizing GUI agents through human-like contrastive interaction traces. PACUT introduces a multi-agent self-refinement process that iteratively infers and consolidates user profiles from observed GUI interactions, enabling the synthesis of behaviorally realistic yet preference-divergent contrastive traces. Building on this supervision, we fine-tune GUI agents using supervised learning and Direct Preference Optimization to explicitly distinguish preferred behaviors from profile-inconsistent alternatives. To support systematic evaluation, we collect a new benchmark dataset with user-specific mobile GUI interaction traces. Extensive experiments across diverse LLM backbones demonstrate that PACUT consistently outperforms strong baselines and ablation variants.

1 Introduction

Recent advances in large language models (LLMs) have enabled the development of autonomous agents that can mimic human behavior across a wide range of tasks (Zhang et al., 2024b; Jin et al., 2025; Li et al., 2025; Schwarzschild et al., 2024), including reasoning, decision-making, and natural language interaction. However, deploying such agents in real-world settings often requires moving beyond generic capabilities toward **personalized agents** that can adapt to individual users' needs, preferences, and contexts across domains, such as healthcare (Xie et al., 2025; Abbasian et al., 2023), education (Shi et al., 2025; Dinh and Tran, 2023), customer service (Qin et al., 2025), etc. This need for personalization extends to graphical user interface (GUI) agents that support users in interacting with digital systems, as illustrated in Figure 1. Such agents, embedded in mobile platforms (e.g., Doubao's AI Phone (Lei Technology, 2025)) or desktop environments (e.g., Computer Use Agent (Anthropic, 2024)), can streamline interaction by automating routine tasks, learning and recalling user-specific workflows, and inferring user intent from historical behavior, thereby delivering proactive and personalized assistance.

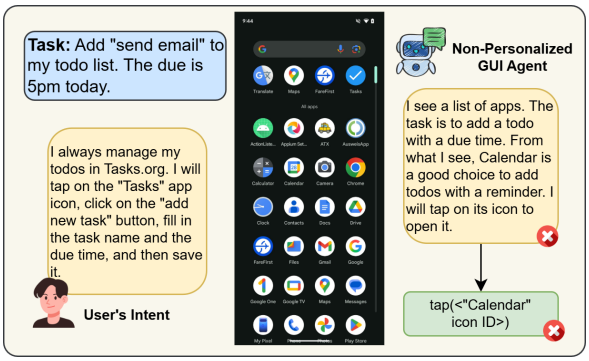

Figure 1: Interaction discrepancies between the GUI agent's execution and the personalized expectation.

Existing personalized GUI agents often rely on carefully crafted role-playing prompts and few-shot examples to simulate user-specific behavior (Park et al., 2024; Qian et al., 2024; Lu et al., 2025). While prompt engineering can be effective in controlled settings, it significantly limits adaptability, particularly when user preferences, tasks, or environmental contexts evolve over time. To address these limitations, substantial research efforts (Rafailov et al., 2023; Zhang et al., 2025c; Wang et al., 2025) have shifted toward data-driven approaches that aim to provide agents with personalization capabilities through learning. In particular, several studies attempt to synthesize reasoning-augmented (Lu et al., 2025; Wang et al., 2025) or

adversarial (Lai et al., 2024) interaction data, such as synthesizing explanations for actions or injecting traces that leave tasks incomplete, to implicitly expose agents to diverse behavioral patterns during training. However, such synthetic data often fails to faithfully capture realistic human behaviors and preferences, leading to unstable learning signals and unreliable personalization performance in real-world deployments.

Inspired by adversarial learning frameworks such as Generative Adversarial Networks (GANs) (Saxena and Cao, 2021) and contrastive learning approaches like Siamese Neural Networks (SNNs) (Chicco, 2021), where models are trained to distinguish highly similar yet semantically distinct examples, we posit that the realism of contrasting examples plays a critical role in personalization. Specifically, the more realistic and human-like the contrasting interaction traces are, the more effectively a model can learn subtle differences in user preferences and interaction strategies. By exposing the agent to synthetic traces that closely approximate authentic human usage patterns, the model is encouraged to focus on fine-grained behavioral cues rather than superficial artifacts, thereby fostering more robust and generalizable personalization across diverse users and contexts.

To this end, we introduce PACUT (**P**ersonalized **A**gent by **C**ontrastive G**U**I **T**races), an approach that automatically synthesizes human-like contrastive traces for personalizing GUI agents. Our approach consists of two main stages. First, in the contrastive trace synthesis stage, PACUT employs a multi-agent self-refinement process to deduce and induce user profiles and interaction dynamics to generate behaviorally realistic yet subtly different GUI traces. Second, in the agent fine-tuning phase, PACUT leverages Direct Preference Optimization (DPO) to optimize the GUI agent to prefer reasoning and action trajectories aligned with a target user profile over the synthesized contrasting alternatives. This approach enables the agent to internalize fine-grained personalization signals without relying on manual prompt engineering or unreliable random perturbations.

Our main contributions are as follows:

- We propose PACUT, a novel approach for personalizing GUI agents by synthesizing human-like contrastive interaction traces.
- We introduce a new dataset of personalized GUI interaction traces, comprising diverse users, tasks, and interaction styles, which serves as a benchmark for training and evaluating personalized GUI agents.
- Experiments demonstrate that PACUT significantly outperforms the baselines and ablation variants in personalized GUI agent benchmarks.

2 Related Work

GUI Agent. Executing tasks on graphical user interfaces through automated interaction has long been studied as a means of reducing user effort and improving efficiency. Early approaches primarily relied on scripts, macros (Yeh et al., 2009), and programming-by-demonstration techniques (Li et al., 2017, 2018) to replay predefined sequences of user interactions. However, these methods often struggle to generalize to evolving and dynamic GUIs. Recent research has shifted toward LLM-based GUI agents, which leverage the planning and reasoning capabilities of LLMs to translate high-level user intents into executable action sequences. Prior work has explored multiple dimensions of effective GUI agent construction, including advanced reasoning (Wang et al., 2023; Wen et al., 2024), enhanced visual understanding (Wang et al., 2024; Jiang et al., 2025), and multi-agent paradigms (Feng et al., 2025; Song et al., 2024; Zhang et al., 2025a).

To enable practical deployment of GUI agents in real-world settings, personalization remains a critical and unresolved challenge. Early efforts such as LearnAct (Liu et al., 2025) aim to enhance mobile GUI agents through few-shot learning, allowing agents to adapt to individual user behaviors from limited examples. More recent work has increasingly focused on data-driven approaches to improve the generalization and robustness of personalized GUI agents. Cai et al. (2025) propose Personalized User Memory-enhanced Alignment (PUMA), which enables large language models to function as personalized web GUI agents by incorporating large-scale synthesized user-specific data and web interaction histories. Similar data-centric personalization frameworks have also been explored by Lu et al. (2025) and Wang et al. (2025), further highlighting the potential of learning-based personalization. However, existing approaches largely rely on synthetic interaction data that is either randomly generated or weakly grounded in

realistic human behavior, which often fails to capture subtle user preferences and interaction dynamics, limiting their effectiveness and reliability in real-world personalization scenarios.

LLM Personalization. Personalization of LLMs has emerged as an important direction for adapting general-purpose LLMs to real-world applications that require user-specific behaviors, preferences, and interaction styles. One prominent line of research investigates LLM personalization in social media and social simulation settings, where prompting models with distinct personas or roles can induce specific personality traits, social dynamics, and decision-making styles (Shanahan et al., 2023; Park et al., 2023; Welch et al., 2022; King and Cook, 2020). Another line of work focuses on character-driven or role-playing agents (Li et al., 2023; Zhang et al., 2025b; Wu et al., 2024), in which LLMs are customized to portray consistent fictional or game characters, enabling long-term coherence and identity preservation in interactive narratives and gameplay environments.

Different from existing LLM personalization approaches that primarily focus on shaping linguistic style, personas, or high-level decision-making behaviors, our work targets personalization in grounded GUI interactions. Specifically, we investigate how personalized preferences manifest in sequential action execution and task completion behaviors within dynamic GUI environments. This setting introduces unique challenges beyond text-based interaction, including long-horizon action planning and subtle user-specific interaction patterns, which are not adequately addressed by prior personalization methods centered on conversational or role-playing contexts.

3 Methodology

As illustrated in Figure 2, PACUT is composed of two main stages. The contrastive trace synthesis stage (Section 3.1) automatically generates human-like contrastive GUI interaction traces through a self-refinement process that induces diverse yet behaviorally realistic user profiles and interaction dynamics. The agent fine-tuning stage (Section 3.2) leverages these synthesized contrastive traces to optimize a personalized GUI agent via Direct Preference Optimization (DPO), enabling the agent to learn user-specific reasoning and action preferences.

3.1 Stage 1: Contrastive Trace Synthesis

In the first stage, our objective is to construct the user profile $\mathcal{P}$ from observed interactions and subsequently synthesize contrastive GUI interaction traces that reflect personalized behavioral variations. Given a specific user, we denote by $\mathcal{T} = \{t_1, \ldots, t_n\}$ a set of GUI interaction traces collected from that user. Each trace t_i is defined as a sequence of state–action pairs conditioned on a task description, i.e.,

$$t_i = \{task_i, ((s_1, a_1), \ldots, (s_m, a_m))\} \quad (1)$$

where $task_i$ represents the natural-language task specification, s_j denotes the GUI state at step j, and a_j denotes the corresponding user action executed on the interface. Details of the GUI state representation and action space are provided in Appendix A.

As GUI state–action pairs are low-level and highly instance-specific, directly modeling user preferences from raw interactions is challenging. We therefore aim to abstract the underlying reasoning that motivates each action, transforming each trace into a sequence of high-level rationales that capture user intent and decision-making patterns. Formally, for each interaction trace t_i, we derive a corresponding reasoning sequence $\mathcal{R}i = (r_1, r_2, \ldots, r_m)$, where each rationale r_j explains why the action a_j is taken under GUI state s_j in the context of the task. To achieve this abstraction in a scalable and reliable manner, we design a multi-agent self-refinement pipeline consisting of four coordinated phases, including a *Reasoner*, an *Analyst*, a *Consolidator*, and a *Judge*. These agents collaboratively infer, critique, and refine the latent reasoning patterns underlying user interactions, progressively constructing a coherent and stable user profile that serves as the foundation for contrastive trace synthesis.

Action Reasoner. The *Action Reasoner* is designed to infer the latent reasoning underlying each GUI interaction trace by mapping low-level state–action sequences to high-level, human-interpretable rationales. To capture long-horizon dependencies and global task intent, the *Reasoner* is provided with the entire interaction trace rather than isolated state–action pairs. In detail, the *Reasoner* is prompted to LLMs to first comprehend the full trace, including the task description and sequential GUI interactions, and then abstract the reasoning behind each action conditioned on the

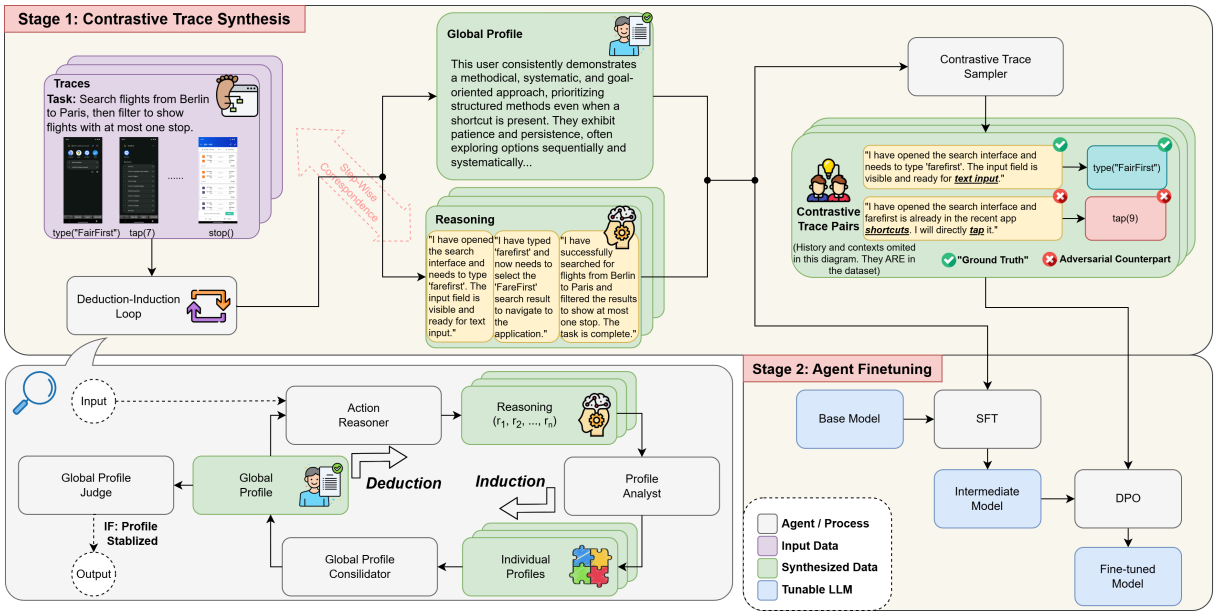

Figure 2: The overview of the PACUT approach.

current user profile. Note that, in the initial iteration, the user profile is empty, and the *Reasoner* relies solely on observable interaction patterns; as the profile is refined over iterations, it serves as additional contextual guidance. This process is formulated as:

$$\mathcal{R}_i^{(k+1)} = f_{\text{reason}}(t_i, \mathcal{P}^{(k)}) \tag{2}$$

where f_{reason} denotes the generation process conditioned on the full trace t_i and the current (Iteration k) user profile $\mathcal{P}^{(k)}$.

Profile Analyst. Given the inferred reasoning sequence $\mathcal{R}_i$, the *Analyst* aims to analyze each interaction trace and induce a user profile conditioned on the specific trace t_i. Rather than focusing on individual actions in isolation, the Analyst aggregates the reasoning signals across the entire trace to infer higher-level personalization attributes. To this end, we prompt the LLM-based *Analyst* to describe salient patterns, including habitual GUI navigation strategies, personality traits, and decision-making tendencies, which are factors commonly associated with personalization in interactive systems (Tam and Ho, 2006; Gajos and Chauncey, 2017). This trace-specific profiling process is formulated as:

$$p_i^{(k+1)} = f_{\text{analyze}}(\mathcal{R}_i^{(k+1)}) \tag{3}$$

where f_{analyze} denotes the LLM generation function that abstracts user-level behavioral attributes from the inferred action rationales.

Global Profile Consolidator. Since user profiles inferred from individual traces may exhibit variation, due to task-specific behaviors, temporary goals, or situational constraints, they may not consistently reflect the user's overarching interaction style. To address this, we introduce the *Consolidator*, which merges the set of per-trace profiles into a unified and stable global profile $\mathcal{P}$ that characterizes the user's consistent habits, preferences, and decision-making patterns across tasks. Importantly, we do not simply average or discard differences among local profiles. Instead, we prompt the LLM-based *Consolidator* to perform a structured comparison, explicitly identifying commonalities while reasoning about potential variability. This allows the *Consolidator* to preserve informative nuances that may reflect adaptive or conditional behaviors, rather than treating them as noise. Such abstraction is critical for capturing both stable traits and context-sensitive strategies. The consolidation process is formalized as:

$$\mathcal{P}^{(k+1)} = f_{\text{consolidate}}(\{p_i^{(k+1)}\}_{i=1}^{n}) \tag{4}$$

where $f_{\text{consolidate}}$ denotes the LLM-guided aggregation function that fuses per-trace user insights into a coherent global profile.

Global Profile Judge. To progressively refine the user profile into a coherent and consistent representation, we iteratively apply the self-refinement loop involving the *Reasoner*, *Analyst*, and *Consolidator*. With each iteration, the global profile $\mathcal{P}$ is

updated to better capture the user's behavioral patterns. However, to avoid unnecessary or redundant updates, it is crucial to determine when this iterative process should terminate. To this end, we introduce a *Judge* that evaluates convergence by comparing successive profiles $\mathcal{P}^{(k+1)}$ and $\mathcal{P}^{(k)}$ along two complementary criteria. The first criterion, consistency, evaluates whether internal contradictions within the profile, such as conflicting habits or preferences, have been resolved or stabilized into a coherent behavioral description, reflecting convergence at a finer (micro) level. The second criterion, stability, assesses whether the semantic content of the profile has ceased to change significantly across iterations, indicating convergence at a global (macro) level.

To assess these criteria, the *Judge* prompts the LLM to perform a comparison between $\mathcal{P}^{(k+1)}$ and $\mathcal{P}^{(k)}$, explicitly assessing semantic changes and internal coherence. The refinement loop terminates when both stability and consistency criteria are satisfied, ensuring that the resulting profile is both robust and interpretable for subsequent contrastive trace synthesis.

Contrastive Trace Generation. Once the user profile $\mathcal{P}$ has converged, we proceed to generate contrastive GUI interaction traces that serve as negative examples for personalization. Inspired by established negative sampling strategies in representation and preference learning (Xu et al., 2022; Yang et al., 2024), we construct contrastive traces by selectively sampling plausible yet profile-inconsistent behaviors. Specifically, we prompt the LLM to generate out-of-character reasoning sequences r^- that intentionally contradict the user-specific preferences encoded in $\mathcal{P}$, while remaining logically valid within the task context. Each such rationale is paired with an alternative action $a^- \neq a$ that diverges from the original user action. To ground these contrastive rationales in executable interactions, we then deploy a state-of-the-art GUI agent (Liu et al., 2025) to roll out full interaction trajectories conditioned on the generated out-of-character reasoning and alternative actions. This process ensures that the synthesized contrastive traces are not only semantically meaningful but also operationally feasible within the GUI environment, yielding high-quality negative examples for subsequent preference-based agent fine-tuning.

3.2 Stage 2: Agent Fine-tuning

Building upon the human-like contrastive traces generated in the previous stage, we then fine-tune the GUI agent to align its reasoning and action selection with a target user profile while explicitly distinguishing preferred behaviors from profile-inconsistent alternatives.

Supervised Fine-Tuning (SFT). We first perform supervised fine-tuning to equip the GUI agent with basic task-solving and interaction capabilities under personalized settings. SFT serves as an initialization step that stabilizes subsequent preference-based optimization by grounding the agent in executable and coherent GUI behaviors. To this end, we leverage the supervised dataset of positive GUI interaction traces that align with the target user profile. Each example is represented as a sequence conditioned on the task description and GUI state, with the model trained to predict the next action given the preceding context. Formally, given a reasoning-enchanced trace $t = \{task, (s_1, r_1, a_1), \ldots, (s_m, r_m, a_m)\}$, the SFT objective minimizes the negative log-likelihood of the demonstrated actions:

$$\mathcal{L}_{\text{SFT}} = -\sum_{t=1}^{m} \log \pi_\theta(r_t, a_t \mid task, s_{\leq t}, r_{<t}, a_{<t}) \tag{5}$$

where π_θ denotes the GUI agent parameterized by θ. By training on profile-consistent interaction traces, SFT enables the agent to acquire foundational personalized behaviors, providing a stable starting point for subsequent contrastive preference optimization.

Direct Preference Optimization (DPO). After supervised fine-tuning, we further optimize the GUI agent using DPO to explicitly encode personalized preferences. DPO enables the model to directly learn from pairwise comparisons between preferred and non-preferred behaviors without requiring an explicit reward model. In our setting, each training instance consists of a preferred GUI interaction trace that aligns with the target user profile and a contrastive trace synthesized to contradict it. Given a task and the corresponding GUI context, DPO encourages the agent to assign a higher likelihood to the preferred reasoning and action trajectory than to the contrasting alternative. In detail, given a reasoning-enchanced trace $t = \{task, (s_1, r_1, a_1), \ldots, (s_m, r_m, a_m)\}$, there

is a negative pair (r_j^-, a_j^-) for each step j. For simplicity, we note $x_t = (task, s_{\leq t}, r_{<t}, a_{<t})$. The DPO objective optimizes the agent policy π_θ from the SFT step by maximizing the relative preference margin between the two:

$$\mathcal{L}_{\text{DPO}} = -\sum_{t=1}^{m} \Big[\log \sigma \Big(\beta \log \frac{\pi_\theta(r_t, a_t \mid x_t)}{\pi_{ref}(r_t, a_t \mid x_t)} - \beta \log \frac{\pi_\theta(r_t^-, a_t^- \mid x_t)}{\pi_{ref}(r_t^-, a_t^- \mid x_t)} \Big) \Big] \tag{6}$$

where $\sigma(\cdot)$ is the sigmoid function, π_{ref} is the frozen model policy after SFT, and β controls the strength of preference optimization.

4 Experiments

We conduct our experiments on mobile GUI environments, as they present rich and diverse user interaction patterns that are well-suited for studying personalization in sequential decision-making. While our evaluation focuses on mobile platforms, the proposed approach is general and can be readily extended to other GUI settings, such as desktop or web interfaces.

4.1 Dataset

Since existing GUI interaction datasets (Rawles et al., 2023; Zhang et al., 2024a; Li et al., 2024) do not provide explicit personalization metadata, we manually collect a benchmark dataset with user-specific interaction traces to enable systematic evaluation of personalized GUI agents. We follow the data collection procedures of prior GUI datasets (Li et al., 2024), while additionally recording user-specific metadata. Details regarding participant recruitment, trace recording, and data anonymization are provided in Appendix B.

In total, the dataset consists of 70 unique tasks spanning 7 mobile apps, including both widely used and less common apps, yielding 420 interaction traces with 3,497 individual actions. For evaluation, we randomly select 40 tasks for training, 20 tasks for in-domain testing (with apps seen during training), and 10 tasks for out-of-domain testing, where the apps does not appear in the training set.

4.2 Evaluation Metrics

We evaluate the performance of GUI agents using two primary metrics. The first metric is action prediction accuracy, a fine-grained measure that assesses the correctness of the agent's sequential decision-making. Specifically, we compare the predicted action type and corresponding parameters (e.g., target GUI element) against the ground-truth actions recorded in the user interaction traces. This metric directly reflects the agent's ability to reproduce user-specific interaction behaviors at the action level.

For a separate study focusing on analyzing the convergence of global user profiles in the contrastive trace synthesis stage (Stage 1), we need to compare the similarity between different profiles. For this, we utilize a second metric named sentence embedding similarity (Reimers and Gurevych, 2019). Since profiles are generated via induction from traces, the exact phrasing may vary. Therefore, we focus on semantic convergence rather than syntactic overlap, measuring the cosine similarity between profile representations in the embedding space.

4.3 Baselines

We evaluate the generalizability of our approach across a diverse set of base LLMs. The selected models vary in scale and architecture, covering both open-weight and proprietary systems, as well as models with and without explicit reasoning capabilities. Specifically, we include GPT-4.1 (nano and mini), Qwen-3 (8B and 30B-A3B), GPT-OSS (20B), and Llama-3 variants (3.1-8B-Instruct and 3.3-70B-Instruct).

We establish two baseline methods for comparison. The first baseline is zero-shot prompting (*Zero-Shot*), the most commonly adopted setting for LLM-based GUI agents, where the model directly predicts actions from task descriptions and GUI states without any personalization. The second baseline is *PAG* (Profile-Augmented Generation) (Richardson et al., 2023), which resembles role-playing approaches by conditioning the agent on an explicitly generated global user profile to guide action inference. To further analyze the contribution of different components in PACUT, we conduct two ablation studies. The first ablation uses *SFT*, where the agent is fine-tuned with supervised learning on personalized traces without contrastive preference optimization. The second variant, *SFT w/ Reas.*, augments supervised fine-tuning with explicit action rationales, but excludes contrastive trace synthesis and DPO. These ablations help isolate the impact of contrastive learning and prefer-

ence optimization in our approach.

In addition, the quality of the converged profile during Stage 1 inherently impacts the quality of synthesized reasoning and contrastive traces. To assess the robustness of the convergence of the global user profile, we designed a separate experiment. Using all traces of one randomly chosen participant, we run the iterative loop specifically with two types of distinct initialization conditions: an empty initial profile (the default) and ten distinct, artificially generated "extreme" profiles (e.g., "hasty," "quitting", detailed in Appendix B). This setup allows us to observe and verify whether the iterative loop converges to a consistent user profile, regardless of the initial prior.

More detailed implementations of the baselines, such as hyperparameters, can be found in Appendix B.

5 Results and Analysis

In this section, we first evaluate the general performance and robustness of the approach. Subsequently, we compare PACUT against baseline methods to validate the necessity of training over prompting. Finally, we present an ablation study to isolate the specific contributions of explicit reasoning and contrastive trace synthesis.

5.1 General Performance

Table 1 demonstrates the overall performance of different LLM backbones across all evaluation settings. Among the evaluated models, GPT-4.1-mini achieves the best performance, reaching 55.63% and 64.36% accuracy on seen and unseen apps, respectively. We attribute this strong performance to its balanced model capacity and robust reasoning ability, which enable effective integration of personalized signals during fine-tuning. Notably, even open-weight models exhibit competitive results: for example, Llama-3.3-70B achieves 53.16% and 62.16% accuracy on seen and unseen apps, demonstrating that our approach generalizes well across different model scales and architectures. This comparative performance across both proprietary and open-source backbones suggests that PACUT is model-agnostic and can effectively enhance personalization regardless of the underlying LLM, thereby opening up promising opportunities for efficient and privacy-preserving on-device personalization in future deployments. Notably, even though GPT-OSS-20B is by design a reasoning model with a different pretraining process, PACUT also demonstrates its effectiveness on this model type, achieving action prediction accuracies of 47.37% on seen apps and 51.37% on unseen apps.

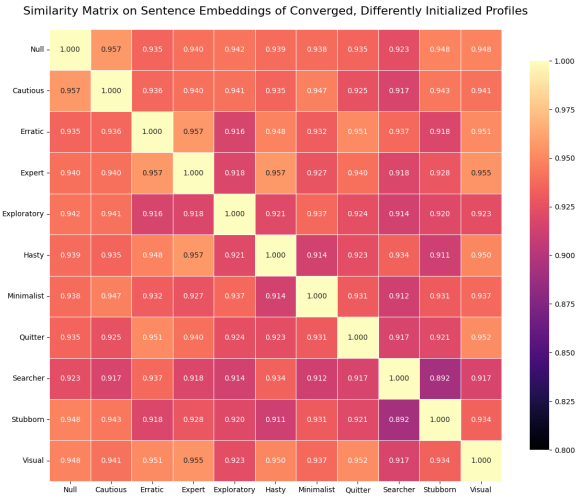

Figure 3: Similarity comparison of the converged profiles. The comparison metric is cosine similarity in the Sentence Embedding space. These converged profiles are originally initialized as different "extreme" profiles. The exact definitions of these initializations are reported in Appendix B. `Null` indicates initialization with an empty profile.

To further assess the robustness of PACUT, we analyze the convergence stability of the global user profile generation in Stage 1 (cf. Section 3.1). As illustrated in Figure 3, even when initialized with disparate or contradictory "extreme" profiles, through the iterative deduction-induction loop, the user profile consistently converges to a high-similarity semantic representation, hinting that the profiles generated from these diverse initializations share high semantic overlap after convergence. This indicates that our iterative mechanism effectively mitigates initial biases (e.g., a "hasty" prior) and grounds the final profile in the actual evidence provided by the interaction traces. Consequently, the framework proves to be highly robust to initialization noises, ensuring that the internalized persona is a faithful reflection of the user's traces rather than an artifact of the prompting strategy.

5.2 Baseline Comparison

Table 1 also presents a comparison between PACUT and baseline approaches, including *Zero-Shot* and *PAG*. Zero-shot prompting yields relatively weak performance, achieving an average accuracy of 33.83% and 37.70% on the Seen and Unseen app

Table 1: Performance of PACUT with results on Seen apps (left) and Unseen apps (right).

Base Model	Zero-Shot	PAG	SFT	SFT w/ Reas.	PACUT
GPT-4.1-nano	25.75 / 31.89	26.32 / 31.45	30.28 / 35.19	32.74 / 37.41	**38.37 / 44.24**
GPT-4.1-mini	39.52 / 44.81	37.65 / 44.65	46.23 / 51.07	51.50 / 60.62	**55.63 / 64.36**
Qwen-3-8B	26.65 / 29.36	27.28 / 30.30	30.91 / 33.78	32.36 / 37.02	**38.92 / 42.82**
Qwen-3-30B-A3B	41.32 / 42.54	40.48 / 45.16	45.54 / 47.91	48.64 / 52.43	**51.22 / 58.94**
GPT-OSS-20B	31.29 / 34.71	31.87 / 35.72	37.25 / 39.7	40.63 / 47.05	**47.37 / 51.37**
Llama-3.1-8B	28.46 / 33.26	27.75 / 32.98	38.31 / 42.82	41.19 / **49.26**	**45.04** / 48.15
Llama-3.3-70B	43.79 / 47.30	42.45 / 45.29	45.49 / 50.25	48.55 / 57.81	**53.16 / 62.16**

settings, respectively. This is primarily because zero-shot agents lack any mechanism to incorporate user-specific preferences, forcing them to rely solely on generic task-solving heuristics that fail to capture personalized interaction patterns. Surprisingly, *PAG* yields even worse performance compared to *Zero-Shot*, with the average accuracy of 33.40% and 37.51% on the Seen and Unseen app settings, respectively. We attribute this to the limited ability of current LLMs to accurately deduce concrete GUI actions in a trace to follow the given profile. Even if the profile accurately describes *what* the user prefers *abstractly*, without fine-tuning, LLMs struggle to map these high-level traits into granular, element-level UI actions.

Nevertheless, PACUT consistently outperforms the better zero-shot baseline across all evaluated backbones, achieving an average accuracy improvement of 14.36%. We attribute this gain to our contrastive example synthesis framework, which provides fine-grained supervision by explicitly contrasting preferred and profile-inconsistent behaviors. Unlike static role-based prompts, contrastive traces enable the agent to learn subtle distinctions in user preferences through preference optimization, leading to more robust and generalizable personalization.

5.3 Ablation Study

We further report ablation study results in Table 1 to analyze the impact of individual components in PACUT. Using SFT only, the GUI agent already exhibits improved personalization performance, achieving an average accuracy of 41.05%. This result suggests that supervised exposure to user-specific interaction traces enables the model to capture basic personalized behaviors, even without explicit preference optimization. Incorporating explicit action reasoning during SFT (*SFT w/ Reas.*) further enhances personalization, yielding an average accuracy of 45.52%. This gain indicates that modeling the latent rationale behind user actions helps the agent better understand and generalize user preferences.

Nevertheless, integrating DPO with contrastive traces (PACUT) yields an additional 4.16% improvement over *SFT w/ Reas.* on average. We attribute this improvement to the core contribution of our framework: human-like contrastive examples explicitly define preference boundaries between profile-consistent and inconsistent behaviors, enabling the agent to learn fine-grained personalization signals through preference-based optimization rather than likelihood maximization alone. For the LLaMA-3.1-8B backbone, SFT with reasoning slightly outperforms PACUT on the Unseen app setting, achieving 49.26% accuracy. We hypothesize that this is due to this small model's slight overfitting to interaction styles in seen apps during the preference optimization step, as PACUT outperforms SFT with reasoning with the same model in the Seen app setting.

6 Conclusion

In this work, we introduce PACUT, an automated approach for synthesizing human-like contrastive examples to enable effective personalization of GUI agents. Specifically, PACUT employs a multi-agent self-refinement process to iteratively infer and consolidate user profiles from interaction traces, producing behaviorally realistic contrastive data. Building on this synthesized supervision, we fine-tune GUI agents using a combination of supervised fine-tuning and Direct Preference Optimization to align action selection with user-specific preferences. Evaluations across diverse LLM backbones demonstrate that PACUT consistently outperforms the baselines and ablation variants, highlighting its effectiveness and generalizability for personalized GUI interaction.

Limitations

We acknowledge three primary limitations in our study. First, our empirical analysis is conducted exclusively on mobile GUI environments, leaving web and desktop GUI platforms unexplored. Nevertheless, we believe our approach can be readily extended to these settings, as the fundamental techniques of PACUT, such as user profile inference, contrastive trace synthesis, and preference-based optimization, are platform-agnostic and rely on abstract state–action representations rather than platform-specific interfaces. Second, we do not evaluate PACUT on extremely large-scale models. This is primarily due to computational constraints and the cost associated with fine-tuning high-capacity proprietary models. Despite this limitation, our results demonstrate strong performance gains even on smaller and mid-sized models, suggesting promising opportunities for on-device deployment and real-time personalization in practical applications. Finally, we acknowledge that the size of our collected benchmark dataset is relatively limited. While sufficient for systematic evaluation, larger-scale and more diverse datasets could further strengthen the analysis. In future work, we plan to expand data collection to cover additional users, tasks, and platforms, as well as to explore online and continual personalization settings.

References

Mahyar Abbasian, Iman Azimi, Amir M Rahmani, and Ramesh Jain. 2023. Conversational health agents: A personalized llm-powered agent framework. *arXiv preprint arXiv:2310.02374*.

Anthropic. 2024. Introducing computer use, a new claude 3.5 sonnet, and claude 3.5 haiku. https://www.anthropic.com/news/3-5-models-and-computer-use. Accessed: 2026-01-04.

Hongru Cai, Yongqi Li, Wenjie Wang, Fengbin Zhu, Xiaoyu Shen, Wenjie Li, and Tat-Seng Chua. 2025. Large language models empowered personalized web agents. In *Proceedings of the ACM on Web Conference 2025*, pages 198–215.

Davide Chicco. 2021. Siamese neural networks: An overview. *Artificial neural networks*, pages 73–94.

Hoa Dinh and Thien Khai Tran. 2023. Educhat: An ai-based chatbot for university-related information using a hybrid approach. *Applied Sciences*, 13(22):12446.

Sidong Feng, Changhao Du, Huaxiao Liu, Qingnan Wang, Zhengwei Lv, Mengfei Wang, and Chunyang Chen. 2025. Breaking single-tester limits: Multi-agent llms for multi-user feature testing. *arXiv preprint arXiv:2506.17539*.

Krzysztof Z Gajos and Krysta Chauncey. 2017. The influence of personality traits and cognitive load on the use of adaptive user interfaces. In *Proceedings of the 22nd international conference on intelligent user interfaces*, pages 301–306.

Wenjia Jiang, Yangyang Zhuang, Chenxi Song, Xu Yang, Joey Tianyi Zhou, and Chi Zhang. 2025. Appagentx: Evolving gui agents as proficient smartphone users. *arXiv preprint arXiv:2503.02268*.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan Arik, Dong Wang, Hamed Zamani, and Jiawei Han. 2025. Search-r1: Training llms to reason and leverage search engines with reinforcement learning. *arXiv preprint arXiv:2503.09516*.

Milton King and Paul Cook. 2020. Evaluating approaches to personalizing language models. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 2461–2469.

Hanyu Lai, Xiao Liu, Iat Long Iong, Shuntian Yao, Yuxuan Chen, Pengbo Shen, Hao Yu, Hanchen Zhang, Xiaohan Zhang, Yuxiao Dong, and 1 others. 2024. Autowebglm: A large language model-based web navigating agent. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 5295–5306.

Lei Technology. 2025. Doubao mobile assistant is here. it suits small manufacturers like nubia. is it unlikely that huawei and xiaomi will pay for it? https://eu.36kr.com/en/p/3576988690782593. Accessed: 2026-01-04.

Cheng Li, Ziang Leng, Chenxi Yan, Junyi Shen, Hao Wang, Weishi Mi, Yaying Fei, Xiaoyang Feng, Song Yan, HaoSheng Wang, and 1 others. 2023. Chatharuhi: Reviving anime character in reality via large language model. *arXiv preprint arXiv:2308.09597*.

Dawei Li, Bohan Jiang, Liangjie Huang, Alimohammad Beigi, Chengshuai Zhao, Zhen Tan, Amrita Bhattacharjee, Yuxuan Jiang, Canyu Chen, Tianhao Wu, and 1 others. 2025. From generation to judgment: Opportunities and challenges of llm-as-a-judge. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 2757–2791.

Toby Jia-Jun Li, Amos Azaria, and Brad A Myers. 2017. Sugilite: creating multimodal smartphone automation by demonstration. In *Proceedings of the 2017 CHI conference on human factors in computing systems*, pages 6038–6049.

Toby Jia-Jun Li, Igor Labutov, Xiaohan Nancy Li, Xiaoyi Zhang, Wenze Shi, Wanling Ding, Tom M

Mitchell, and Brad A Myers. 2018. Appinite: A multi-modal interface for specifying data descriptions in programming by demonstration using natural language instructions. In *2018 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, pages 105–114. IEEE.

Wei Li, William E Bishop, Alice Li, Christopher Rawles, Folawiyo Campbell-Ajala, Divya Tyamagundlu, and Oriana Riva. 2024. On the effects of data scale on ui control agents. *Advances in Neural Information Processing Systems*, 37:92130–92154.

Guangyi Liu, Pengxiang Zhao, Liang Liu, Zhiming Chen, Yuxiang Chai, Shuai Ren, Hao Wang, Shibo He, and Wenchao Meng. 2025. Learnact: Few-shot mobile gui agent with a unified demonstration benchmark. *arXiv preprint arXiv:2504.13805*.

Yuxuan Lu, Jing Huang, Yan Han, Bingsheng Yao, Sisong Bei, Jiri Gesi, Yaochen Xie, Qi He, Dakuo Wang, and 1 others. 2025. Prompting is not all you need! evaluating llm agent simulation methodologies with real-world online customer behavior data. *arXiv preprint arXiv:2503.20749*.

Joon Sung Park, Joseph O'Brien, Carrie Jun Cai, Meredith Ringel Morris, Percy Liang, and Michael S Bernstein. 2023. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th annual acm symposium on user interface software and technology*, pages 1–22.

Joon Sung Park, Carolyn Q Zou, Aaron Shaw, Benjamin Mako Hill, Carrie Cai, Meredith Ringel Morris, Robb Willer, Percy Liang, and Michael S Bernstein. 2024. Generative agent simulations of 1,000 people. *arXiv preprint arXiv:2411.10109*.

Chen Qian, Wei Liu, Hongzhang Liu, Nuo Chen, Yufan Dang, Jiahao Li, Cheng Yang, Weize Chen, Yusheng Su, Xin Cong, and 1 others. 2024. Chatdev: Communicative agents for software development. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15174–15186.

Weicong Qin, Yi Xu, Weijie Yu, Chenglei Shen, Ming He, Jianping Fan, Xiao Zhang, and Jun Xu. 2025. Maps: Motivation-aware personalized search via llm-driven consultation alignment. *arXiv preprint arXiv:2503.01711*.

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.

Christopher Rawles, Alice Li, Daniel Rodriguez, Oriana Riva, and Timothy Lillicrap. 2023. Androidinthewild: A large-scale dataset for android device control. *Advances in Neural Information Processing Systems*, 36:59708–59728.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*.

Chris Richardson, Yao Zhang, Kellen Gillespie, Sudipta Kar, Arshdeep Singh, Zeynab Raeesy, Omar Zia Khan, and Abhinav Sethy. 2023. Integrating summarization and retrieval for enhanced personalization via large language models. *arXiv preprint arXiv:2310.20081*.

Divya Saxena and Jiannong Cao. 2021. Generative adversarial networks (gans) challenges, solutions, and future directions. *ACM Computing Surveys (CSUR)*, 54(3):1–42.

Avi Schwarzschild, Zhili Feng, Pratyush Maini, Zachary Lipton, and J Zico Kolter. 2024. Rethinking llm memorization through the lens of adversarial compression. *Advances in Neural Information Processing Systems*, 37:56244–56267.

Murray Shanahan, Kyle McDonell, and Laria Reynolds. 2023. Role play with large language models. *Nature*, 623(7987):493–498.

Yao Shi, Rongkeng Liang, and Yong Xu. 2025. Educationq: Evaluating llms' teaching capabilities through multi-agent dialogue framework. *arXiv preprint arXiv:2504.14928*.

Zirui Song, Yaohang Li, Meng Fang, Yanda Li, Zhenhao Chen, Zecheng Shi, Yuan Huang, Xiuying Chen, and Ling Chen. 2024. Mmac-copilot: Multi-modal agent collaboration operating copilot. *arXiv preprint arXiv:2404.18074*.

Kar Yan Tam and Shuk Ying Ho. 2006. Understanding the impact of web personalization on user information processing and decision outcomes. *MIS quarterly*, pages 865–890.

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. 2024. Mobile-agent-v2: Mobile device operation assistant with effective navigation via multi-agent collaboration. *Advances in Neural Information Processing Systems*, 37:2686–2710.

Ziyi Wang, Yuxuan Lu, Yimeng Zhang, Jing Huang, and Dakuo Wang. 2025. Customer-r1: Personalized simulation of human behaviors via rl-based llm agent in online shopping. *arXiv preprint arXiv:2510.07230*.

Charles Welch, Chenxi Gu, Jonathan K Kummerfeld, Veronica Perez-Rosas, and Rada Mihalcea. 2022. Leveraging similar users for personalized language modeling with limited data. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1742–1752.

Hao Wen, Yuanchun Li, Guohong Liu, Shanhui Zhao, Tao Yu, Toby Jia-Jun Li, Shiqi Jiang, Yunhao Liu, Yaqin Zhang, and Yunxin Liu. 2024. Autodroid: Llm-powered task automation in android. In *Proceedings of the 30th Annual International Conference on Mobile Computing and Networking*, pages 543–557.

Weiqi Wu, Hongqiu Wu, Lai Jiang, Xingyuan Liu, Hai Zhao, and Min Zhang. 2024. From role-play to drama-interaction: An llm solution. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 3271–3290.

Haojie Xie, Yirong Chen, Xiaofen Xing, Jingkai Lin, and Xiangmin Xu. 2025. Psydt: Using llms to construct the digital twin of psychological counselor with personalized counseling style for psychological counseling. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1081–1115.

Lanling Xu, Jianxun Lian, Wayne Xin Zhao, Ming Gong, Linjun Shou, Daxin Jiang, Xing Xie, and Ji-Rong Wen. 2022. Negative sampling for contrastive representation learning: A review. *arXiv preprint arXiv:2206.00212*.

Zhen Yang, Ming Ding, Tinglin Huang, Yukuo Cen, Junshuai Song, Bin Xu, Yuxiao Dong, and Jie Tang. 2024. Does negative sampling matter? a review with insights into its theory and applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(8):5692–5711.

Tom Yeh, Tsung-Hsiang Chang, and Robert C Miller. 2009. Sikuli: using gui screenshots for search and automation. In *Proceedings of the 22nd annual ACM symposium on User interface software and technology*, pages 183–192.

Chaoyun Zhang, Liqun Li, Shilin He, Xu Zhang, Bo Qiao, Si Qin, Minghua Ma, Yu Kang, Qingwei Lin, Saravan Rajmohan, and 1 others. 2025a. Ufo: A ui-focused agent for windows os interaction. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 597–622.

Haonan Zhang, Run Luo, Xiong Liu, Yuchuan Wu, Ting-En Lin, Pengpeng Zeng, Qiang Qu, Feiteng Fang, Min Yang, Lianli Gao, and 1 others. 2025b. Omnicharacter: Towards immersive role-playing agents with seamless speech-language personality interaction. *arXiv preprint arXiv:2505.20277*.

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

Yadong Zhang, Shaoguang Mao, Tao Ge, Xun Wang, Adrian de Wynter, Yan Xia, Wenshan Wu, Ting Song, Man Lan, and Furu Wei. 2024b. Llm as a mastermind: A survey of strategic reasoning with large language models. *arXiv preprint arXiv:2404.01230*.

Yimeng Zhang, Tian Wang, Jiri Gesi, Ziyi Wang, Yuxuan Lu, Jiacheng Lin, Sinong Zhan, Vianne Gao, Ruochen Jiao, Junze Liu, and 1 others. 2025c. Shopr1: Rewarding llms to simulate human behavior in online shopping via reinforcement learning. *arXiv preprint arXiv:2507.17842*.

A GUI Representation and Action Space Definition

A.1 GUI Representation

We focus on Android GUI interactions as discussed in Section 4. The key challenge for GUI representation is ensuring the interface the LLM perceives resembles what humans observe, while also enabling the LLM to perform similar actions on mobile devices as humans do.

Our pilot study finds that user interactions sometimes target UI element containers (e.g., tapping the "empty" background to dismiss a pop-up menu). Prior works often neglect these elements, focusing only on leaf nodes in the a11y tree. While this simplification is acceptable for GUI agents that focus on task-completion (e.g., a back button can substitute for a background tap), it undermines the fidelity of our work, which focuses on personalization and style learning.

To address this, we traverse the view hierarchy and emit a concise HTML fragment that the agent can reliably interpret. We map Android classes to web-native tags with similar semantics. The detailed mapping is provided in Table 2.

Table 2: Mapping from Android UI classes to HTML tags for agent observation.

Android Class	HTML Tag
TextView	`<p>`
Button Classes	`<button>`
Image Classes	`<img>`
EditText	`<input>`
Others	`<div>`

Building on the mapping, we introduce two specific modifications:

Visibility Based Filtering. We discard UI elements that lack visible information (e.g., texts, content descriptions, etc.) to humans. This gate saves tokens and reduces noise information, which is critical for maintaining attention in context-sensitive prompts.

Container Remapping. We expose interactive containers as `<button>` elements rather than generic `<div>`s. This ensures that container-targeted behaviors observed in human traces (e.g.,

tapping outside a floating window) remain expressible in the agent action space.

A.2 Action Space

The action space of the agent is defined through a series of parameterized function call, as shown in Table 3.

B Experiment Details

B.1 Data Collection Process

This subsection complements the data collection process introduced in Subsection 4.1. In detail, we recruited 6 participants to perform tasks across 7 diverse mobile applications, ranging from widely used ones (such as Google Maps) to less common apps (such as Task.org). This selection ensures the dataset covers both familiar and unfamiliar interaction environments. Recruitment was done through the university's internal Slack workspace and word-of-mouth referrals. At the beginning, we gathered detailed background information from each applicant. This included a short questionnaire on their mobile phone usage habits (familiar mobile OS, average screen time, preferred app categories), self-reported familiarity with each target app, and basic demographics (gender, age, education level, occupation). We use this information to select participants and ensure diversity in their profiles. This information is not stored or linked in the later collected dataset to ensure anonymity. The data collection process involved approximately 27 person-hours from the participants, including task explanation, trace recording exercise with feedback, and the formal trace recording. Participants were specifically informed about the format and content of the data collected and the intended follow-up research use of this data. We asked for explicit confirmation and agreement from the participants. Participants performed tasks based on natural language GUI task descriptions, with all interactions recorded via accessibility logging services. We provided a dedicated device with pre-installed environments and apps. For account-related GUI tasks, we provided dedicated accounts and instructed participants to use them specifically for anonymity. After the recording session, participants were compensated financially in accordance with the local law's guidelines on hourly income, based on the actual duration of the trace recording session.

B.2 Experiment Setup and Details

We refer to two platforms for fine-tuning the involved LLMs in the experiments. For the proprietary LLMs (GPT-4.1-mini and GPT-4.1-nano), we use the OpenAI API for model fine-tuning and inference. For the open-sourced LLMs (Qwen-3-8B, Qwen-3-30B-A3B, GPT-OSS-20B, Llama-3.1-8B-Instruct, and Llama-3.3-70B-Instruct), we use the Tinker platform accordingly. Note that we did not adopt the newest models, such as the GPT 5 series, the Gemini 3 series from Google, or the Llama 4 series, due to restricted access to fine-tuning functionality (GPT 5, Gemini) or general model availability (Llama 4 on Tinker). Still, we experiment with the newest possible models on the corresponding platforms.

The experiments share the same hyperparameter setup: `max tokens` is set to 8192 to allow occasionally longer reasoning output even though it is prompted to be concise; we keep `top_p` as the default and set `temperature` to 0 to restrict the diversity and variance of the LLM generation and allow better reproducibility. For the experiment exclusive parameter, we set `max retry` as 15, which is a relatively lenient limit to allow recovery from occasional errors or internet connection issues. For fine-tuning, we set the `batch size` to 2. For SFT, the `learning rate` is set as 2e-5 with 3 epochs, while for DPO the `learning rate` is 1e-6 with only 1 epoch. The `beta` value for DPO is set as 0.1. All experiment results are produced under the same conditions on the basis of three runs, and the averaged results are reported to smooth the variance of random effects.

B.3 Agent Prompts

We report all the relevant agent prompts in the experiments in this subsection, including the GUI agent under testing (Table 4) as well as the agents used for data synthesizing (Table 5-9, cf. Section 3.1). Note that the Contrastive Trace Sampler shares exactly the same prompt structure as the GUI agent, with the extra information inserted before the final instruction. This is reported in Table 9

B.4 Details on Artificial Initialization Profile

We present the 10 artificially generated "extreme" user profiles used in the convergence verification experiment (cf. Section 4.3). They are reported in Table 10.

Table 3: Agent Action Space Definition

Action Command	Description
`tap(element:int)`	Tap on the UI element with the specified integer ID.
`long_press(element:int)`	Long press on the specified UI element.
`text("string")`	Type the specified UTF-8 string into the focused input.
`enter()`	Press the enter/search key on the virtual keyboard.
`swipe(element:int, "dir")`	Swipe on an element in a direction ("up", "down", "left", "right").
`back()`	Press the system back button.
`home()`	Press the system home button to return to the launcher.
`stop()`	Indicate task completion or impossibility.

Table 4: The prompt used for GUI agents. Note that for the agent variation with an additional user profile in the prompt, it is inserted at the beginning of the user prompt.

Prompt for the GUI Agent

```
<System Prompt>
You are acting as if you are a real human operating an Android device.

You must complete the given task. At each step, first provide your reasoning
in a 'THOUGHT:' block, followed by *exactly ONE* function call on a
new line, with a leading triple-dash (---) and NO extra text.

Your output format MUST be:
THOUGHT:
[Your reasoning for the next action goes here as a plain text paragraph. Explain your choice.]
---[function_call()]

Allowed actions:
---tap(element:int)
---long_press(element:int)
---text("utf8 string")
---enter()
---swipe(element:int, "up"|"down"|"left"|"right")
---back()
---home()
---stop()

Rules:
- Refer only to element ids present in the CURRENT HTML inventory.
- Focus an input before using ---text("...") or ---enter().
- Output MUST be exactly in the 'THOUGHT:' and '---action' format.
- Do not repeat yourself if the last action did not change the HTML contents.

<User Prompt>
Task Instruction: {task_desc}

The interaction history:
{List of matched htmls and reasoning-action pairs}

Current Screen:
{current_html}

Instruction: Based on the history above and the current screen, what is the next thought and action?
```

Table 5: The prompt used for Action Reasoner in the deduction-induction loop.

Prompt for Action Reasoner

```
You are an 'omniscient thought generator' for users trying to complete GUI tasks on Android devices.

You will be given the *entire* sequence of screens and the user's actions for a full episode.

Your task is to generate a JSON object containing the `THOUGHT:` block for *every* step from the
perspective of the user.

There are {} steps. Do not repeat yourself for each step.

When generating the thought for a step (e.g., Step 5), you can and should use your knowledge of
*all* steps (e.g., Steps 1-9) to inform your reasoning.

Your reasoning should explain *why* the given action is being taken at that step to progress toward
the final goal.

If the action is `---stop()`, your thought should be about why the task is complete or why the user
gave up, based on the final screen.

If a profile is provided:
1. If (at least some parts of) the profile can explain the action, use these parts to adjust the
tone and style of your thought.
2. If not at all, do not use the profile, and generate the thought purely based on the sequence of
screens and actions.

<Optional> User Profile:
<Optional> {user_profile}
<Optional> Use the profile above to adjust the style and tone of your thought, and also to help
<Optional> resolve ambiguities in user actions.

Goal: {task_instruction}

Here is the full episode sequence:
{list of html representations and user actions}
```

Table 6: The prompt used for Profile Analyst in the deduction-induction loop.

Prompt for Profile Analyst

```
You are an expert behavior analyst.

You will be given the thought process of a user performing a specific GUI task.

The setup is that the user performs step-by-step GUI actions to complete a task. For each step,
the user observes the screen and performs an action.

Analyze their reasoning style and actions to derive a brief behavioral profile entry.

Focus on the user's action preferences under various conditions rather than interpreting the
specific task success.

These entries should revolve around the user's navigation strategy, personality traits, and
decision-making tendencies.

Write a single concise paragraph.

Each sentence in the paragraph should focus on one of the user's behavioral patterns/personality
traits.

Here is the user's thought process trace:
{List of reasoning}
```

Table 7: The prompt used for Global Profile Consolidator in the deduction-induction loop.

Prompt for Global Profile Consolidator

```
You are an expert behavior analyst. You are compiling a behavioral-psychological profile for
a single user based on observations from multiple tasks.

You are given a list of 'Profile Entries' derived from different tasks.

1. Merge consistent observations (remove redundancy).

2. PRESERVE contradictions (e.g., 'In task A the user was careless, but in task B the user was
very check-heavy'). Do not try to smooth them over.

3. Express confidence based on the frequency of observations (e.g., if 5 entries say careless and
2 say careful, state that the user is predominantly careless but occasionally careful).

4. Output a **single, brief, concise paragraph** that is straight to the point.

Here are the profile entries from different tasks:
{List of profile entries}
```

Table 8: The prompt used for Global Profile Judge in the deduction-induction loop.

Prompt for Global Profile Judge

```
You are an expert behavior analyst. Compare the following two versions of a User Profile (Previous
vs. Current).

Determine if the **semantic meaning** of the profile has changed regarding the user's personality
or habits.
- IGNORE: Phrasing changes, formatting, or ordering.
- FOCUS ON: Personality traits (e.g., cautious vs. careless), recurring habits, and identified
contradictions.

Has the profile **stabilized** (i.e., no new meaningful insights or shifts, and also no more
contradictions (e.g., 'predominantly careless but occasionally careful'))?

Output ONLY a JSON object: {"stable": boolean, "reason": "brief explanation"}

Previous Profile:
{old_profile}

Current Profile:
{new_profile}
```

Table 9: The extra prompt segment for Contrastive Trace Sampler

Extra Prompt Segment for Contrastive Trace Sampler

```
User Profile: {The converged profile of the current user}.

Ground Truth User Action: {The ground truth action}

Constraint: You must still try to complete the task, and this is the most important rule.
BUT your behavior/style should be INCONSISTENT with the User Profile described above.
In addition, on the basis of trying to finish the task, you should choose a different action from the
ground truth.
```

Table 10: Definition of the 10 “extreme” initial artificial user profiles

Profile Name	Profile Description
Cautious	This user exhibits a highly cautious and hesitant behavioral pattern, meticulously verifying every screen element before taking action. They are risk-averse and detail-oriented, often scrolling through entire lists or reading non-essential text to ensure they are on the correct path. Their pace is slow and deliberate, prioritizing accuracy and understanding over efficiency, often double-checking previous steps or pausing for extended periods before committing to a choice.
Erratic	This user's behavior is characterized by unpredictable and seemingly random interactions, lacking a coherent strategy. They fluctuate between rapid tapping and long pauses, often switching between unrelated tasks or screen areas without clear motivation. Their pattern is noisy and inconsistent, making it difficult to discern a logical intent behind their sequence of actions.
Expert	This user is hyper-efficient and assertive, exploiting advanced features like search, gesture shortcuts, and rapid scanning to navigate at high speed. They are confident and knowledgeable, anticipating UI flows and bypassing intermediate steps to reach deep states quickly. Their approach is characterized by high precision, minimal hesitation, and a preference for complex but faster interaction methods over simple scrolling or tapping.
Exploratory	This user demonstrates high curiosity and an exploratory nature, frequently deviating from the direct path to investigate unrelated buttons or menus. They value discovering features over speed, often interacting with 'About', 'Settings', or peripheral tabs just to see what they contain. Their path is winding and non-linear, characterized by a breadth-first search of the interface rather than a depth-first execution.
Hasty	This user demonstrates an impulsive and speed-oriented approach, frequently opting for the first visible element that loosely matches their goal without verification. They are hasty and prone to trial-and-error, preferring to tap rapidly and backtrack rather than reading labels carefully. Their interaction style is characterized by quick, heuristic-based decisions, often leading to navigational errors which they correct through immediate, high-frequency corrective actions.
Minimalist	This user adheres to a strictly minimalist philosophy, interacting only with the absolute minimum number of elements required to complete the task. They systematically ignore all optional features, promotions, or dense information, focusing laser-like on the primary call-to-action. Their efficiency comes not from speed but from a refusal to engage with any UI element that does not directly advance the immediate goal.
Quitter	This user demonstrates extremely low persistence and a low tolerance for friction. Upon encountering any ambiguity, loading delay, or complex form, they are prone to immediately aborting the task or exiting the app. Their interactions are tentative, and they frequently use the 'Home' or 'Back' gesture at the first sign of difficulty, rarely attempting to troubleshoot or explore alternative paths.
Searcher	This user exhibits a strong dependency on search functionality, bypassing navigational hierarchies and category lists entirely. They immediately locate the search bar or 'Find' icon upon entering a screen, preferring to type specific keywords rather than browse. Their behavior is characterized by a lack of engagement with the browsing interface and a high frequency of text input actions.
Stubborn	This user displays a rigid and persistent behavioral pattern, often attempting the same failed action multiple times before considering an alternative. They are resistant to changing strategies, preferring to 'force' a solution through repetition (e.g., repeated tapping of an unresponsive element). Their adaptation to error is slow, showing a preference for brute-force consistency over flexibility.
Visual	This user relies heavily on visual cues, primarily interacting with icons, images, and large buttons while frequently overlooking text-based links or small labels. They scan the screen for graphical representations of their goal, ignoring dense text blocks. This often leads to missed options that are only presented textually, resulting in a navigation path biased towards the graphical elements of the UI.