ExpeLAgent: Experiential Learning for Personalized Mobile GUI Task Execution

Anonymous ACL submission

Abstract

Recent advances in Multimodal Large Language Models (MLLMs) have fueled growing interest in mobile agents capable of autonomously completing complex GUI tasks. However, existing methods mainly focus on isolated task execution and lack effective learning from cross-task interaction history. Recent experiential learning frameworks introduce skill libraries and memory banks, yet they organize experience around tasks rather than users, limiting their ability to model personal preferences and handle semantically ambiguous instructions. Consequently, user behavior data remains underutilized, hindering personalization and execution accuracy. To bridge these gaps, we propose a novel **Expe**riential **L**earning **Agent** (**ExpeLAgent**) framework that learns to anticipate a user's preferred workflow and adjust its actions accordingly. Specifically, to help ExpelAgent better capture user preference, we design a context engineering workflow that extracts structured user preference profiles from historic interaction data across user queries and applications. To improve the cross-task generalizability, we design a Universal Skill module that distills reusable and transferable sub-routines from the pool of historical user interactions. Moreover, to address the data scarcity in cold-start scenarios, we propose a Shared Memory mechanism that dynamically recalls the anonymized interaction instances extracted from the most similar users and boosts the performance via in-context learning. Experiments show that ExpeLAgent outperforms baselines. User profiling improves personalization by 9.5%. Universal skill extraction increases task success by 26.5%, and in cold-start scenarios, the shared memory mechanism enhances success by 11.1% and personalization by 6.7%. These results demonstrate the potential of experiential learning in enhancing future personalized mobile agents. Code is available at `https://anonymous.4open.science/r/ExpeLAgent-6CE1/`.

1 Introduction

Recent advances in Multimodal Large Language Models (MLLMs) (Kuang et al., 2025; Zhao et al., 2025) have driven growing interest in mobile GUI agents, which autonomously handle complex user queries through multi-step graphical user interface interactions (Ge et al., 2025; Chen et al., 2025). A mobile GUI agent receives a user instruction, perceives the current screen, and produces a sequence of actions to accomplish the stated goal. However, executing user queries accurately requires more than parsing the current instruction: it depends on a user's intent, preferences, and operational routines, which are accumulated in long-term interaction history. For example, an instruction like "order a cup of coffee" leaves the brand, drink size, and payment method entirely unspecified. Advancing mobile agents from reactive instruction followers into experiential learners that continuously adapt to individual users is therefore essential for both personalization and execution accuracy.

Despite recent progress, three key gaps remain unresolved. First, most mobile GUI agents (Hong et al., 2024; Cheng et al., 2024) focus on isolated task execution and discard the behavioral signals accumulated in historical interactions, leaving user behavior data largely unexploited and making personalization fundamentally infeasible. Second, recent attempts to incorporate experience and memory—through executable skill libraries (Wang et al., 2024a; Zheng et al., 2025), evolving memory banks (He et al., 2026; Sun et al., 2026), hierarchical multimodal skills (Li et al., 2024b; Xie et al., 2025), or context-aware memory modules (Yang et al., 2025a; Li et al., 2024a)—organize experience around tasks rather than users, and thus fail to disentangle individual preferences from collective behavior. Even studies that leverage personal data (Wu et al., 2025; Yang et al., 2025b; Lyu et al., 2026) lack structured preference modeling and do

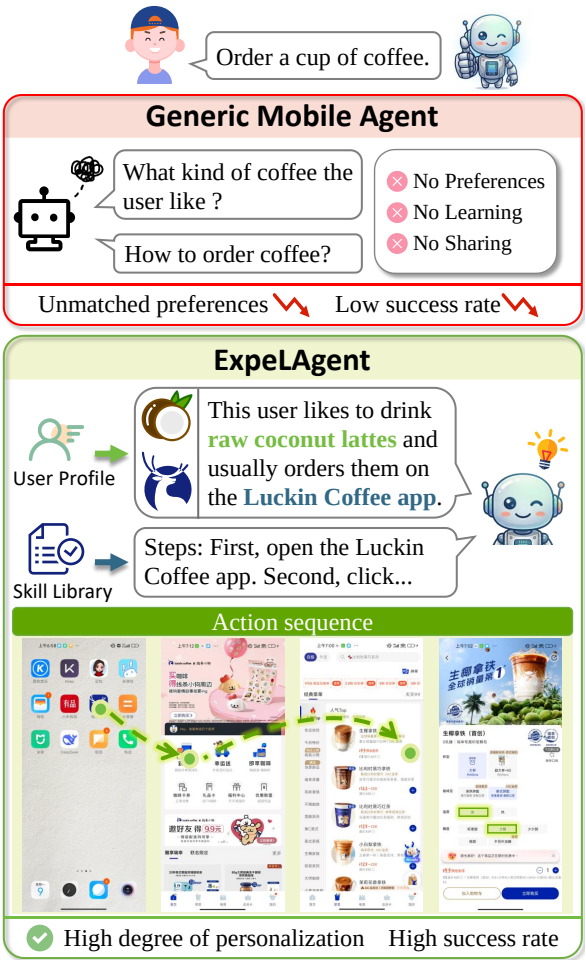

Figure 1: Comparison between a generic mobile GUI agent and ExpeLAgent.

not abstract reusable skills, limiting cross-task generalization. Third, most prior approaches depend heavily on abundant historical records and degrade sharply in cold-start scenarios.

To address these challenges, we propose **ExpeLAgent**, a framework that learns to anticipate a user's preferred workflow and adjust its actions accordingly. To accurately capture user preferences and habits, we design a context-engineering workflow that produces a **User Profile** by extracting structured preference representations from historical interaction data, supplying explicit individual intent at inference time. To bridge the gap of missing reusable skill abstraction, we design a **Universal Skill** module that distills transferable sub-routines from the pool of historical user interactions, enabling the agent to retrieve and compose matching action sequences for new tasks rather than re-deriving them from scratch. To address cold-start challenges, we propose a **Shared Memory** mechanism that dynamically recalls anonymized interaction instances from the most similar users and enhances performance via in-context learning, ensuring reliable execution when personal history is sparse. As illustrated in Fig. 1, when given an ambiguous instruction such as "order a cup of coffee," a generic agent lacks both user preference awareness and reusable skill knowledge, leading to execution failure. ExpeLAgent overcomes this limitation by leveraging a User Profile and a Universal Skill library to achieve high personalization and task success. And, when personal history is insufficient, recalling interactions from similar users via Shared Memory

Our main contributions are as follows:

We propose a context-engineering method that extracts structured User Profiles from cross-application historical interactions, modeling users at three levels: App Preferences, Content Interests, and Operation Habits.

We design a Universal Skill module that distills reusable and transferable sub-routines from historical user interactions, improving task success rates and cross-task generalization.

We propose a Shared Memory mechanism that dynamically recalls interaction instances from the similar users via in-context learning, leveraging collective experience to ensure stable performance in cold-start scenarios.

2 Related Work

2.1 Multi-Agent Systems

Multi-agent systems decompose complex reasoning tasks into specialized roles. Mobile GUI agents, in particular, require tightly coupled capabilities such as perception, planning, decision-making, and reflection, and architectures that decouple these functions into specialized components have been shown to improve both efficiency and task execution (Guo et al., 2024). Beyond GUI settings, frameworks such as MetaGPT (Hong et al., 2023) and DyLAN (Liu et al., 2024) demonstrate the benefits of structured role decomposition and dynamic routing. In mobile environments, the Mobile-Agent series progressively extends this paradigm: from decoupled perception and planning (Wang et al., 2024b), to planning–decision–reflection collaboration (Wang et al., 2024b), and more recently to hierarchical self-evolution (Wang et al., 2025) and unified multi-platform agents built on foundation models (Ye et al., 2025; Alibaba Tongyi Lab, 2025). While these frameworks successfully scale

GUI agents to increasingly complex tasks, they decompose reasoning at the task level rather than the user level, and therefore lack mechanisms for accumulating and exploiting user-specific experience across sessions.

2.2 Personalized GUI Agents

Driven by rapid advances in MLLMs (Kuang et al., 2025; Zhao et al., 2025), autonomous GUI agents have demonstrated strong capabilities in interface understanding and task execution (Lee et al., 2024; Hong et al., 2024; Li et al., 2024b; Shao et al., 2024; Li et al., 2025; Zhang et al., 2025a). However, most existing methods optimize for task success while neglecting user preferences and interaction history (Xie et al., 2025; Hong et al., 2024; Cheng et al., 2024), which fundamentally limits personalization. Recent work has begun to address this gap, yet each line of effort remains partial: IFRAgent (Wu et al., 2025) models intent flows from short-term context but ignores long-term history; FingerTip (Yang et al., 2025b) leverages recent interactions yet lacks structured preference modeling; and PersonalAlign (Lyu et al., 2026) extracts preferences from individual user histories but does not exploit cross-user shared experience, leaving it vulnerable to data sparsity. As a result, current personalized agents still lack both structured user representations and reusable cross-task skills, and they degrade sharply in cold-start scenarios.

2.3 Experience-Driven Learnable Agents

Existing mobile agents mainly focus on isolated task execution, overlooking experience accumulated from past interactions. Recent work enables agents to learn from experience through executable skill libraries (Wang et al., 2024a), reusable APIs (Zheng et al., 2025), evolving memory banks (He et al., 2026; Sun et al., 2026), and hierarchical multimodal skills (Li et al., 2024b).

However, these methods organize experience around tasks rather than users, limiting personalization on ambiguous instructions. Based on the insight that interaction history is a rich source of personalizable knowledge once abstracted, we propose ExpeLAgent, an experiential learning framework that learns and adapts to user-preferred workflows. It comprises three components: to model user preferences, we propose a User Profile that distills cross-application history into app preferences, content interests, and operation habits; to enable cross-task reuse, we propose a Universal Skill module that abstracts action sequences into parameterized templates with symbolic placeholders; and to alleviate cold-start, we propose a Shared Memory that retrieves records from profile-similar users via in-context learning.

3 Problem Formulation

A personalized mobile GUI task is defined as $\mathcal{X}_u = \langle I, \mathcal{A}, \mathcal{O}, \mathcal{H}_u \rangle$, where I is the user instruction, and $\mathcal{O}$, $\mathcal{A}$ are the screen observation and action spaces (see Appendix A). For each user u, $\mathcal{H}_u = \{h_1, \ldots, h_z\}$ records historical interactions, where $h_i = (\tau_i, l_i, I^i, \mathcal{O}^i, \mathcal{A}^i)$ contains the timestamp, location, instruction, and the observation–action trajectory $(\mathcal{O}^i, \mathcal{A}^i)$ of length T_i.

The mobile GUI agent generates the next action conditioned on the instruction, user history, current observation, and past actions:

$$a_{t+1} = f(I, \mathcal{H}_u, o_t, \mathcal{A}_{1:t}), \tag{1}$$

where $f(\cdot)$ denotes the agent.

The objective of mobile GUI agents is to understand user intentions and accomplish tasks through device interactions. As personal assistants, they should also remember user preferences or operation habits—for instance, a user saying "Hi, ExpeLAgent, order me a cup of milk tea" should not need to specify the app, flavor, or other details. This reflects the interaction experience users truly expect and also motivates the design of ExpeLAgent.

4 Method

4.1 Overview

As illustrated in Fig. 2, ExpeLAgent is a multi-agent framework designed to jointly improve task success rate and personalization through experiential learning. To support experiential learning, ExpeLAgent incorporates three core modules: a **User Profile** (§4.2) that models individual user preferences and behavioral patterns, **Universal Skills** (§4.3) that capture transferable procedural knowledge distilled from cross-user interactions, and a **Shared Memory** (§4.4) that retrieves relevant experiences from similar users to alleviate cold-start issues. During execution, these modules are collaboratively integrated into a three-stage **Personalized GUI Task Execution** pipeline (§4.5).

4.2 User Profiling via Context Engineering

To better understand users, we propose a user profiling approach based on context engineering.

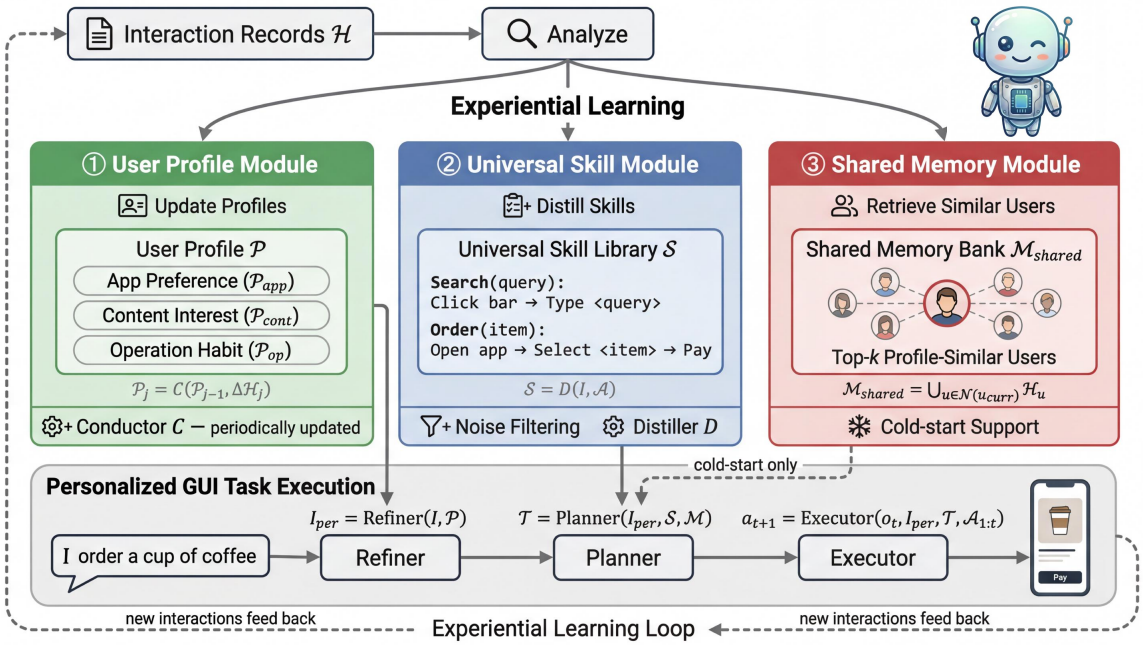

Figure 2: Overview of the proposed **ExpeLAgent** framework, including three complementary modules: a **User Profile Module** capturing app preferences, content interests, and operation habits; a **Universal Skill Module** that abstracts action sequences into a reusable skill library; and a **Shared Memory Module** that aggregates histories from profile-similar users to support cold start. At inference, a Refiner–Planner–Executor pipeline performs personalized GUI task execution, and new interactions feed back into $\mathcal{H}$.

Structured User Profile. The user profile $\mathcal{P}$ is represented in natural language, so that it can be directly consumed by the agent while remaining interpretable to humans. $\mathcal{P}$ unfolds along three complementary dimensions: (i) **app preference** $\mathcal{P}_{\text{app}}$, capturing stable mappings from task context to app choice; (ii) **content interest** $\mathcal{P}_{\text{cont}}$, abstracting semantic topics that recur across time; and (iii) **operation habit** $\mathcal{P}_{\text{op}}$, distilling the user's decision and execution habits during interaction. Notably, user behavior reflects both stable long-term preferences and consistent inertial patterns (e.g., habitually selecting top-ranked results or familiar apps).

Construction and Update. Rather than updating online at every interaction, $\mathcal{P}$ is refreshed once per K=20 interactions to stabilize profile estimation:

$$\mathcal{P}_j = C(\mathcal{P}_{j-1},\, \Delta\mathcal{H}_j), \tag{2}$$

here, C denotes the Conductor, which takes the latest interaction batch and the current profile as input to generate the updated profile $\mathcal{P}_j$ using the prompt described in Appendix C.1, while incorporating auxiliary statistical summaries from Appendix B. To handle temporal preference drift, when newly observed preferences conflict with existing ones, the Conductor replaces outdated preferences with more recent behavioral patterns, ensuring that the profile remains consistent and up to date.

4.3 Universal Skill Distiller

Existing GUI agents lack a mechanism to transform interaction experience into long-term capability: even after successfully completing a task, the execution process is rarely reusable in future interactions. We propose Universal Skill Distillation, which canonicalizes raw GUI trajectories into parameterized procedural templates, enabling a single successful execution to be retained as a transferable procedural skill.

For mobile GUI scenarios, extracting high-quality and reusable skills from real user interactions is inherently challenging. Raw action trajectories often contain substantial noise, including redundant clicks, accidental operations, and navigation loops. To address this issue, rule-based filtering is first applied to remove low-level interaction noise, such as null-effect actions, short-horizon rollback behaviors, consecutive duplicate operations, and meta-control actions (e.g., `wait` and `finished`). After deterministic cleaning, an LLM performs intent-level semantic filtering to

determine whether the trajectory can genuinely fulfill the user intent. Only high-quality samples are retained for subsequent skill distillation.

For validated samples, a Distiller converts the action sequence into a parameterized skill template $\mathcal{S}$ with the prompt provided in Appendix C.2:

$$\mathcal{S} = D(I, \mathcal{A}), \tag{3}$$

here, D denotes the Distiller. For example, the trajectory corresponding to the instruction "search for Deep Learning tutorials" can be distilled into the reusable skill:

`Search(query): Click search bar` $\rightarrow$ `Type <query>` $\rightarrow$ `Click search button`.

The generated skill is then validated through schema-level checking and further evaluated by an LLM-based quality assessment module. Low-quality skills are discarded. Finally, cosine similarity is computed between the new skill and existing skills in the library. Based on predefined thresholds, the candidate skill is classified as redundant, mergeable, or novel, corresponding to discard, merge, and insertion operations, respectively.

4.4 Shared Memory

Existing methods typically rely on sufficient personal interaction history, which limits their effectiveness in cold-start scenarios. To address this limitation, we introduce a shared memory module.

It is observed that users with similar profiles tend to exhibit overlapping intent patterns and operation preferences, motivating the use of cross-user interaction histories as surrogate demonstrations. Specifically, the top-k most similar users are retrieved based on cosine similarity over profile embeddings, and their interaction histories are aggregated into a shared memory bank:

$$\mathcal{M}_{\text{shared}}^{(u_{\text{curr}})} = \bigcup_{u \in \mathcal{N}(u_{\text{curr}})} \mathcal{H}_u, \tag{4}$$

where $\mathcal{N}(u_{\text{curr}})$ denotes the set of top-k users most similar to the current user u_{curr}. The detailed implementation process is provided in Appendix D. Note that shared memory operates on de-identified user data; privacy-preserving mechanisms are beyond the scope of this work and will be explored in future research.

4.5 Personalized GUI Task Execution

During personalized task execution, ExpeLAgent proceeds through three steps: instruction personalization, action planning, and operation, driven by accumulated individual and collective experience.

Instruction Personalization. To better tailor I to what the user truly wants, the Refiner queries $\mathcal{P}$ to align the instruction with the user's preferences:

$$I_{\text{per}} = \text{Refiner}(I, \mathcal{P}). \tag{5}$$

Action Planning. Given I_{per}, the matching skills $\mathcal{S}$ are retrieved from the Universal Skill library, together with the top-k relevant histories $\mathcal{M}$personal (or $\mathcal{M}$shared in cold-start scenarios). The Planner then constructs an action plan $\mathcal{T}$:

$$\mathcal{T} = \text{Planner}(I_{\text{per}}, \mathcal{S}, \mathcal{M}). \tag{6}$$

Action Execution. The Executor reasons over $\mathcal{T}$ step by step to predict each atomic action conditioned on o_t and $\mathcal{A}_{1:t}$:

$$a_{t+1} = \text{Executor}(o_t, I_{\text{per}}, \mathcal{T}, \mathcal{A}_{1:t}). \tag{7}$$

The prompts are provided in Appendix C.3, C.4, and C.5.

5 Experiments

We conduct extensive experiments covering overall performance against GUI agent baselines, the impact of base models, component ablations under both cold-start and normal settings, computational efficiency, and online evaluation on physical Android devices. Our experiments address three core questions: whether ExpeLAgent sustains high success rates across diverse scenarios, whether it accurately follows user preferences and habits, and whether each module, including the user profile, universal skills, and shared memory, contributes meaningfully to overall performance and cold-start robustness.

5.1 Experiment Setup

Benchmark We evaluate personalized GUI agents using the FingerTip 20K benchmark (Yang et al., 2025b), which contains one month of daily mobile usage data from 95 participants. Each participant recorded their intention before every interaction. For each user, we sort all records chronologically and sample test instances randomly from the most recent 20% of records, yielding 250 test samples in total; the earlier 80% serves as history for experiential learning, intent retrieval, and action sequence retrieval at test time. To streamline evaluation, we pre-distill universal skills and user profiles for all participants using DeepSeek-V3 (DeepSeek-AI et al., 2024).

Method	*TA*	*SR*	*CSR*	*GA*	*TS*	*PDR*
Single-agent methods						
AppAgent	0.68	0.41	0.41	0.38	0.40	0.87
AppAgentX	0.71	0.42	0.40	0.42	0.41	0.85
HIM-Agent	0.68	0.36	0.41	0.46	0.37	0.89
IFRAgent	0.67	0.35	0.40	0.45	0.34	0.85
Multi-agent methods						
Mobile-Agent-v2	0.65	0.32	0.36	0.39	0.32	0.78
Mobile-Agent-v3	0.67	0.37	0.40	0.40	0.35	0.85
Mobile-Agent-v3.5	0.71	0.38	0.42	0.44	0.38	0.89
Mobile-Agent-E	0.70	0.40	0.39	0.38	0.37	0.83
ExpeLAgent (Ours)	**0.74**	**0.46**	**0.47**	**0.48**	**0.43**	**1.12**

Table 1: Comparison with representative mobile GUI agent baselines on the FingerTip 20K benchmark. All methods are implemented using the Qwen3-VL-8B base model. Bold indicates the best value per column; underline indicates second-best.

Implementation Details We deploy ExpeLAgent on a diverse set of MLLMs to comprehensively assess its effectiveness. To support future on-device deployment under resource constraints, we include lightweight open-source models with no more than 8B parameters: **Qwen3-VL-8B** (Bai et al., 2025), **Qwen3-VL-2B** (Bai et al., 2025), **GUI-OWL-7B** (Ye et al., 2025), **GUI-OWL-2B** (Ye et al., 2025), **MAI-UI-8B** (Microsoft AI, 2025), and **MAI-UI-2B** (Microsoft AI, 2025). We also evaluate on state-of-the-art closed-source models, including **GPT-4.5** (OpenAI, 2025), **Claude Sonnet 4.6** (Anthropic, 2026), **GLM-4V** (GLM et al., 2024), and its lightweight variant **GLM-4.6V-Flash** (Team et al., 2026). We disable thinking mode to reduce inference latency and token consumption. During execution, we retrieve skills and historical records using **Qwen3-Embedding-0.6B** (Zhang et al., 2025b) with cosine similarity, returning the top-k ($k = 2$) results.

Metrics ExpeLAgent is evaluated from two perspectives: task execution and personalization. Task execution includes four metrics: *TA* for action-type correctness, *SSR* for step-level success rate, *CSR* for exponentially weighted step success ($\alpha = 0.6$) that penalizes early failures more heavily, and *GA* for click accuracy within ground-truth bounding boxes. Personalization is measured using *TS*, which evaluates trajectory alignment via longest common subsequence (LCS), and *PDR*, which compares similarity to the target user's trajectory against trajectories from other user clusters.

Baselines We compare ExpeLAgent with representative single-agent and multi-agent GUI baselines: **AppAgent** (Zhang et al., 2025a) and **AppAgentX** (Jiang et al., 2025), which adopt exploration-based skill learning; **HIM-Agent** (Lyu et al., 2026) and **IFRAgent** (Wu et al., 2025), which focus on intent understanding and instruction following; and the Mobile-Agent series (Wang et al., 2024b; Ye et al., 2025; Alibaba Tongyi Lab, 2025; Wang et al., 2025), the state of the art in multi-agent collaboration. All methods are evaluated with the same base model, Qwen3-VL-8B, for a fair comparison.

5.2 Overall Performance

Table 1 compares ExpeLAgent with representative baselines. ExpeLAgent achieves the best results across all metrics, demonstrating strong effectiveness in both task execution and personalization. For task completion, it attains the highest *SR*, *CSR*, and *GA*, with relative gains of 9.5%, 11.9%, and 4.3% over the strongest baselines. Notably, baselines exhibit a clear planning–execution gap—e.g., AppAgentX and Mobile-Agent-v3.5 both reach *TA* of 0.71 but drop to *SR* of 0.42 and 0.38, respectively—whereas ExpeLAgent substantially narrows it, suggesting that accumulated experience better translates high-level intent into executable actions. The advantage is most pronounced on *PDR*, where ExpeLAgent reaches 1.12, outperforming the strongest baseline by 25.8%; competitive baselines on *TA* or *CSR* plateau on *PDR*, indicating limited personalization despite strong task-level performance. Overall, ExpeLAgent jointly improves execution and personalization by transforming past interactions into user-aligned policies rather than treating tasks independently.

Base Model	*TA*	*SR*	*CSR*	*GA*	*TS*	*PDR*
General-purpose MLLMs						
GPT-4.5	0.75	0.35	0.31	0.25	0.34	0.81
Claude Sonnet 4.6	0.72	0.22	0.15	0.10	0.26	0.62
GLM-4V	0.75	**0.47**	0.46	0.43	0.44	1.00
GLM-4.6V-Flash	0.72	0.46	0.41	0.36	0.43	0.99
Qwen3-VL-8B	0.74	0.46	**0.47**	**0.48**	0.43	**1.12**
Qwen3-VL-2B	0.63	0.28	0.31	0.36	0.30	0.97
GUI-specialized MLLMs						
GUI-OWL-7B	**0.77**	0.42	0.38	0.46	**0.47**	1.08
GUI-OWL-2B	0.65	0.16	0.14	0.18	0.26	0.81
MAI-UI-8B	0.71	0.21	0.17	0.20	0.27	0.77
MAI-UI-2B	0.63	0.06	0.07	0.20	0.15	0.61

Table 2: Performance of ExpeLAgent with different MLLMs on the FingerTip 20K benchmark. All entries use the same ExpeLAgent framework; only the underlying foundation model varies.

Setting	*SR*	*CSR*	*TS*	*PDR*
Full	**0.46**	**0.47**	**0.43**	**1.12**
w/o User Profile	0.43	0.42	0.41	0.98
w/o Universal Skill	0.40	0.36	0.40	1.04
w/o User Profile & Universal Skill	0.34	0.35	0.34	0.95

Table 3: Ablation study on user profile and universal skill. All methods are implemented using the Qwen3-VL-8B base model.

Setting	*SR*	*CSR*	*TS*	*PDR*
Full	**0.42**	**0.41**	**0.41**	**0.97**
w/o Shared Memory	0.40	0.37	0.38	0.91
w/o Universal Skill	0.40	0.35	0.37	0.96
w/o Shared Memory & Universal Skill	0.36	0.34	0.34	0.90

Table 4: Ablation study on cold-start performance. All methods are implemented using the Qwen3-VL-8B base model.

5.3 Base Model Analysis

Table 2 evaluates ExpeLAgent across base models. Among general-purpose models, Qwen3-VL-8B performs best (*CSR* 0.47, *GA* 0.48, *PDR* 1.12); GUI-OWL-7B leads the specialized category on *TA* (0.77) and *TS* (0.47), showing that domain-specific pre-training benefits action-type prediction but does not consistently outperform strong general models. Smaller 2B base models underperform substantially, likely because lengthy planning contexts exceed their context-processing capacity. Closed-source models offer no clear advantage: Claude Sonnet 4.6 scores *GA* 0.10 and *PDR* 0.62, and GPT-4.5 achieves only *SR* 0.35, both attributable to limited exposure to Chinese GUI environments. These results indicate that long-term experiential learning contributes more to performance than model scale or GUI-specific pre-training.

5.4 Cold-Start Evaluation

Table 4 evaluates ExpeLAgent under cold-start settings without prior user interaction data. The full model achieves the best performance across all metrics, demonstrating the effectiveness of both universal skills and shared memory. Removing *Shared Memory* causes the largest drops in *TS* and *PDR*, indicating that peer-user experience is the primary source of personalization in cold-start scenarios. In contrast, removing *Universal Skill* mainly reduces *CSR* while leaving *SR* unchanged, suggesting that skills mainly stabilize early-step execution. Removing both modules degrades all metrics, confirming their complementary roles in improving task execution and personalization under cold-start conditions.

5.5 Ablation Study

Table 3 isolates the contributions of the user profile and universal skills in normal (non-cold-start) settings. Removing the user profile (*w/o User Profile*) causes the sharpest drop in *PDR* (1.12 $\rightarrow$ 0.98), confirming that explicit preference modeling is the key driver of personalized behavior. Removing universal skills (*w/o Universal Skill*) primarily hurts task execution: *SR* falls from 0.46 to 0.40 and *CSR* from 0.47 to 0.36, while *PDR* remains relatively high (1.04), since the user profile still supplies preference signals. Removing both degrades all metrics to their lowest values, demonstrating that the two components address distinct bottlenecks and are jointly necessary for full performance.

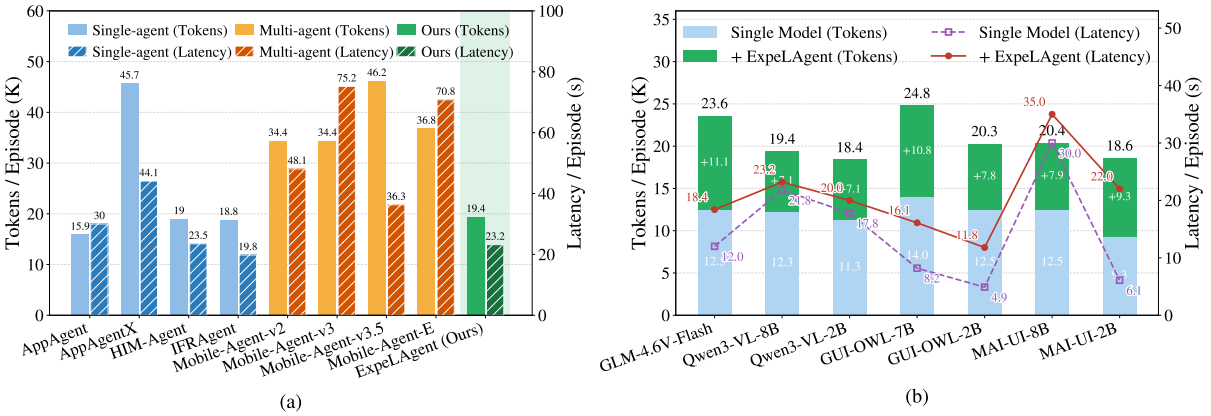

Figure 3: Efficiency evaluation of ExpeLAgent. (a) Comparison of token consumption and latency per episode against baselines. (b) The computational overhead introduced by integrating our framework across various MLLMs.

Benchmark	Method	*SR* (%)↑	*ASR*↓	*PDR*↑
FingerTip 20K	GLM-4.6V	40.5	1.59	1.45
	Aguvis-7B	20.0	1.64	1.24
	UI-TARS-1.5	35.5	1.53	1.28
	ExpeLAgent	**50.5**	**1.51**	**1.51**
SPA-Bench	GLM-4.6V	39.5	1.55	–
	Aguvis-7B	18.5	1.62	–
	UI-TARS-1.5	33.5	1.50	–
	ExpeLAgent	**45.0**	**1.48**	–

Table 5: Online evaluation results on FingerTip 20K and SPA-Bench. ExpeLAgent uses Qwen3-VL-8B as the base model.

5.6 Efficiency Analysis

Execution efficiency is a key bottleneck for deploying GUI agents on real-world mobile devices. Figure 3 reports token consumption and inference latency on a single NVIDIA A100 80GB GPU. As shown in Figure 3(a), ExpeLAgent matches single-agent baselines and clearly outperforms other multi-agent systems, owing to a single-step execution per episode during both instruction personalization and action planning. Figure 3(b) further shows that the overhead is modest: with Qwen3-VL-8B as the backbone, token input increases by only 7.1K and latency by 1.4s, indicating that the performance gains come without prohibitive cost.

5.7 Online Evaluation

We further conduct an online evaluation in which ExpeLAgent completes user tasks on a physical smartphone via the Android Debug Bridge; a real device is required because most Chinese apps enforce integrity checks that fail in emulators. We report *Success Rate* (*SR*), judged by an MLLM from the executed actions and screenshots, the *Average Step Ratio* (*ASR*, executed vs. reference steps) as an efficiency measure, and reuse *PDR* for personalization. Besides FingerTip 20K, we also evaluate on the Chinese single-app subset of SPA-Bench (Chen et al., 2024); since it provides no user history, we report only *SR* and *ASR*. As shown in Table 5, ExpeLAgent achieves the best results across all metrics on both benchmarks. On FingerTip 20K, it improves *SR* by over 10 absolute points while attaining the lowest *ASR* and highest *PDR*. The advantage carries to SPA-Bench, where it still leads by 5.5 *SR* points with the best *ASR*, demonstrating the broad applicability of our framework.

6 Conclusion

In this paper, we introduce ExpeLAgent, a novel experiential learning-based mobile GUI agent for personalized task execution. By integrating a user profiling module that models user-specific preferences and a universal skill module that extracts reusable and transferable action patterns, ExpeLAgent effectively addresses the challenges of personalization and task success. To mitigate cold-start issues, we introduce a Shared Memory mechanism, leveraging historical interactions from similar users. Experimental results show that ExpeLAgent significantly outperforms baselines in both task success and personalization metrics, demonstrating the effectiveness of experiential learning for mobile GUI agents. In future work, we aim to explore privacy-preserving methods to enable secure experience sharing across users.

Limitations

Although ExpeLAgent significantly improves task execution and personalization of mobile GUI agents through experiential learning, several limitations remain. First, existing benchmarks lack support for long-term personalized modeling, as most datasets do not contain continuous user interaction histories. As a result, our experiments are mainly conducted on the FingerTip 20K dataset, which limits the scope of evaluation. Second, since experiments are performed on a de-identified dataset, systematic privacy mechanisms for real-world deployment, such as sanitization protocols for shared memory and user profiles, are beyond the scope of this work and remain important directions for future research. In the future, we plan to construct larger-scale long-term personalized interaction datasets to further improve generalization and robustness, and to explore privacy-preserving mechanisms for deployment in open real-world environments.

References

Alibaba Tongyi Lab. 2025. Mobile-agent-v3.5. https://github.com/X-PLUG/MobileAgent. MobileAgent series release.

Anthropic. 2026. Claude sonnet 4.6. https://www.anthropic.com/claude. Anthropic model release.

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, Wenbin Ge, Zhifang Guo, Qidong Huang, Jie Huang, Fei Huang, Binyuan Hui, Shutong Jiang, Zhaohai Li, Mingsheng Li, and 45 others. 2025. Qwen3-vl technical report. *Preprint*, arXiv:2511.21631.

Gongwei Chen, Xurui Zhou, Rui Shao, Yibo Lyu, Kaiwen Zhou, Shuai Wang, Wentao Li, Yinchuan Li, Zhongang Qi, and Liqiang Nie. 2025. Less is more: Empowering gui agent with context-aware simplification. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5901–5911.

Jingxuan Chen, Derek Yuen, Bin Xie, Yuhao Yang, Gongwei Chen, Zhihao Wu, Li Yixing, Xurui Zhou, Weiwen Liu, Shuai Wang, Rui Shao, Liqiang Nie, Yasheng Wang, Jianye Hao, Jun Wang, and Kun Shao. 2024. SPA-Bench: A comprehensive benchmark for SmartPhone agent evaluation. *arXiv preprint arXiv:2410.15164*.

Kanzhi Cheng, Qiushi Sun, Yougang Chu, Fangzhi Xu, Li YanTao, Jianbing Zhang, and Zhiyong Wu. 2024. Seeclick: Harnessing gui grounding for advanced visual gui agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9313–9332.

DeepSeek-AI, Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, and 1 others. 2024. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*.

Zhiqi Ge, Juncheng Li, Xinglei Pang, Minghe Gao, Kaihang Pan, Wang Lin, Hao Fei, Wenqiao Zhang, Siliang Tang, and Yueting Zhuang. 2025. Iris: Breaking gui complexity with adaptive focus and self-refining. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 24559–24568.

Team GLM, :, Aohan Zeng, Bin Xu, Bowen Wang, Chenhui Zhang, Da Yin, Dan Zhang, Diego Rojas, Guanyu Feng, Hanlin Zhao, Hanyu Lai, Hao Yu, Hongning Wang, Jiadai Sun, Jiajie Zhang, Jiale Cheng, Jiayi Gui, Jie Tang, and 40 others. 2024. Chatglm: A family of large language models from glm-130b to glm-4 all tools. *Preprint*, arXiv:2406.12793.

Taicheng Guo, Xiuying Chen, Yaqi Wang, Ruidi Chang, Shichao Pei, Nitesh V Chawla, Olaf Wiest, and Xiangliang Zhang. 2024. Large language model based multi-agents: A survey of progress and challenges. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence (IJCAI)*, pages 8048–8057.

Tiantian He, Yihang Chen, Keyue Jiang, Ka Yiu Lee, Kaiwen Zhou, Kun Shao, and Shuai Wang. 2026. EE-MCP: Self-evolving MCP-GUI agents via automated environment generation and experience learning. *arXiv preprint arXiv:2604.09815*.

Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, and 1 others. 2023. Metagpt: Meta programming for a multi-agent collaborative framework. In *The twelfth international conference on learning representations*.

Wenyi Hong, Weihan Wang, Qingsong Lv, Jiazheng Xu, Wenmeng Yu, Junhui Ji, Yan Wang, Zihan Wang, Yuxiao Dong, Ming Ding, and 1 others. 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.

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*.

Jiayi Kuang, Ying Shen, Jingyou Xie, Haohao Luo, Zhe Xu, Ronghao Li, Yinghui Li, Xianfeng Cheng, Xika Lin, and Yu Han. 2025. Natural language understanding and inference with mllm in visual question answering: A survey. *ACM Computing Surveys*, 57(8):1–36.

Sunjae Lee, Junyoung Choi, Jungjae Lee, Munim Hasan Wasi, Hojun Choi, Steve Ko, Sangeun Oh, and Insik Shin. 2024. Mobilegpt: Augmenting llm with human-like app memory for mobile task automation. In *Proceedings of the 30th Annual International Conference on Mobile Computing and Networking*, pages 1119–1133.

Wei Li, Renshan Zhang, Rui Shao, Jie He, and Liqiang Nie. 2025. Cogvla: Cognition-aligned vision-language-action model via instruction-driven routing & sparsification. *arXiv preprint arXiv:2508.21046*.

Yanda Li, Chi Zhang, Wenjia Jiang, Wanqi Yang, Bin Fu, Pei Cheng, Xin Chen, Ling Chen, and Yunchao Wei. 2024a. Appagent v2: Advanced agent for flexible mobile interactions. *arXiv preprint arXiv:2408.11824*.

Zaijing Li, Yuquan Xie, Rui Shao, Gongwei Chen, Dongmei Jiang, and Liqiang Nie. 2024b. Optimus-1: Hybrid multimodal memory empowered agents excel in long-horizon tasks. *Advances in neural information processing systems*, 37:49881–49913.

Zijun Liu, Yanzhe Zhang, Peng Li, Yang Liu, and Diyi Yang. 2024. A dynamic llm-powered agent network for task-oriented agent collaboration. In *First Conference on Language Modeling*.

Yibo Lyu, Gongwei Chen, Rui Shao, Weili Guan, and Liqiang Nie. 2026. Personalalign: Hierarchical implicit intent alignment for personalized gui agent with long-term user-centric records. *arXiv preprint arXiv:2601.09636*.

Microsoft AI. 2025. Mai-ui: A mobile-centric ui understanding model. https://huggingface.co/microsoft. Technical report.

OpenAI. 2025. Gpt-4.5 system card. https://openai.com/index/introducing-gpt-4-5/. OpenAI technical report.

Rui Shao, Tianxing Wu, Jianlong Wu, Liqiang Nie, and Ziwei Liu. 2024. Detecting and grounding multimodal media manipulation and beyond. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(8):5556–5574.

Libo Sun, Jiwen Zhang, Siyuan Wang, and Zhongyu Wei. 2026. MAGNET: Towards adaptive GUI agents with memory-driven knowledge evolution. *arXiv preprint arXiv:2601.19199*.

V Team, Wenyi Hong, Wenmeng Yu, Xiaotao Gu, Guo Wang, Guobing Gan, Haomiao Tang, Jiale Cheng, Ji Qi, Junhui Ji, Lihang Pan, Shuaiqi Duan, Weihan Wang, Yan Wang, Yean Cheng, Zehai He, Zhe Su, Zhen Yang, Ziyang Pan, and 74 others. 2026. Glm-4.5v and glm-4.1v-thinking: Towards versatile multimodal reasoning with scalable reinforcement learning. *Preprint*, arXiv:2507.01006.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. 2024a. Voyager: An open-ended embodied agent with large language models. ArXiv:2305.16291.

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

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

Zheng Wu, Heyuan Huang, Yanjia Yang, Yuanyi Song, Xingyu Lou, Weiwen Liu, Weinan Zhang, Jun Wang, and Zhuosheng Zhang. 2025. Quick on the uptake: Eliciting implicit intents from human demonstrations for personalized mobile-use agents. *arXiv preprint arXiv:2508.08645*.

Yuquan Xie, Zaijing Li, Rui Shao, Gongwei Chen, Kaiwen Zhou, Yinchuan Li, Dongmei Jiang, and Liqiang Nie. 2025. Mirage-1: Augmenting and updating gui agent with hierarchical multimodal skills. *arXiv preprint arXiv:2506.10387*.

Bufang Yang, Lilin Xu, Liekang Zeng, Kaiwei Liu, Siyang Jiang, Wenrui Lu, Hongkai Chen, Xiaofan Jiang, Guoliang Xing, and Zhenyu Yan. 2025a. Contextagent: Context-aware proactive llm agents with open-world sensory perceptions. *arXiv preprint arXiv:2505.14668*.

Qinglong Yang, Haoming Li, Haotian Zhao, Xiaokai Yan, Jingtao Ding, Fengli Xu, and Yong Li. 2025b. Fingertip 20k: A benchmark for proactive and personalized mobile llm agents. *arXiv preprint arXiv:2507.21071*.

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

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

Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, Fei Huang, and Jingren Zhou. 2025b. Qwen3 embedding: Advancing text embedding and reranking through foundation models. *arXiv preprint arXiv:2506.05176*.

Hongbo Zhao, Fei Zhu, Haiyang Guo, Meng Wang, Rundong Wang, Gaofeng Meng, and Zhaoxiang

Zhang. 2025. Mllm-cl: Continual learning for multimodal large language models. *arXiv preprint arXiv:2506.05453*.

Boyuan Zheng, Michael Y. Fatemi, Xiaolong Jin, Zora Zhiruo Wang, Apurva Gandhi, Yueqi Song, Yu Gu, Jayanth Srinivasa, Gaowen Liu, Graham Neubig, and Yu Su. 2025. SkillWeaver: Web agents can self-improve by discovering and honing skills. *arXiv preprint arXiv:2504.07079*.

A Action Space Definition

Action	Parameter
`click`	coordinates=(x,y), content=""
`long_click`	coordinates=(x,y), content=""
`type`	text=""
`scroll`	coordinates=(x,y), direction=""
`press_back`	-
`press_home`	-
`press_recent`	-
`wait`	-
`finish`	-

Table 6: Definition of the mobile GUI action space.

B Statistical Behavior Analysis

To provide auxiliary behavioral signals for profile construction, a set of structured statistical features is extracted from recent interaction histories.

`app_pref_by_domain`. App preferences grouped by task domains, e.g., *Food Delivery* → {Meituan, Eleme}, and *Shopping* → {Taobao, JD}.

`top_apps_overall`. The most frequently used applications across all interaction records.

`top_intent_classes`. Frequently occurring task categories derived from `intentClass` statistics.

`top_scenarios`. Distribution statistics of commonly observed scenario labels.

`op_ratio`. Relative frequency distribution of operation types, such as `click`, `scroll`, `type`, and `back`.

`top_content_tokens`. High-frequency UI texts extracted from click and typing actions, such as "search", "confirm", and "checkout".

`top_content_ngrams`. Frequently occurring action-path n-grams that reflect sequential navigation behaviors across interfaces.

`top_queries`. Frequently entered user queries or search keywords.

These statistical features are jointly incorporated with recent interaction histories during profile updating, enabling the profile constructor to capture both semantic preferences and long-term behavioral regularities.

C Prompts

C.1 User Profile Constructor Prompt

A core limitation of prior mobile agents is their inability to model long-term individual preferences: raw interaction histories are noisy and unstructured, making direct retrieval unreliable and preference inference shallow. To overcome this, the User Profile Constructor distills each user's behavioral patterns into a compact, structured representation organized across three dimensions, namely App Preference, Content Interest, and Operation Preference. The prompt is designed around an *iterative update* principle: rather than rebuilding the profile from scratch on every call, it merges newly observed interactions with the existing profile, removing stale entries and surfacing emerging preferences. This design ensures that the profile stays concise and current, enabling the downstream Instruction Planner to transform ambiguous user requests into precise, preference-aligned instructions. The full prompt is shown below.

User Profile Constructor

```
You are a User Profile Constructor responsible
for maintaining a concise and dynamically
updated user profile based on interaction
history.
Input: Previous User Profile:
{previous_user_profile} Recent User History:
{recent_user_history} Statistical Behavior
Summary: {statistical_behavior_summary}
Task: Update the existing user profile
incrementally using the latest interaction
history and statistical behavior data.
Organize the updated profile into the
following three categories:
1. App Preference: Frequently preferred
applications, platforms, or tool types for
specific tasks or domains.
2. Content Interest: Recurring topics,
information domains, or semantic themes the
user consistently focuses on.
3. Operation Preference: Stable interaction
habits, operation styles, or decision-making
patterns demonstrated during task execution.
Update Rules: 1. Perform iterative
profile refinement rather than full profile
reconstruction. 2. Jointly consider Recent
User History and Statistical Behavior Summary,
where statistical signals are used to identify
stable behavioral patterns. 3. Remove
outdated preferences that no longer appear
or are contradicted by recent interaction
patterns. 4. When conflicts occur,
prioritize more recent and repeatedly observed
behavior patterns. 5. Keep all entries
```

short, abstract, and generalized instead of instance-specific. 6. Avoid redundant or semantically overlapping entries.
Output Requirements: - Output ONLY the updated JSON object. - Do not include explanations, comments, or markdown formatting.
The updated user profile:

C.2 Universal Skill Distiller Prompt

A key bottleneck for mobile GUI agents is low cross-task generalization: prior work retrieves raw historical trajectories as references, but concrete interaction details (specific coordinates, app-version UI layouts, hard-coded values) do not transfer across contexts. Motivated by the cognitive observation that humans abstract successful experiences into reusable *procedural schemas* rather than replaying them verbatim, the Universal Skill Distiller converts each canonical action sequence into a parameterized skill template. The prompt instructs the LLM to separate the invariant procedural logic from instance-specific parameters, replacing concrete values with symbolic placeholders (e.g., <query>, <date>) so the resulting skill generalizes to analogous tasks across different applications. The full prompt is shown below.

Universal Skill Distiller

You are a Universal Skill Distiller for a GUI agent.
Input: TASK INSTRUCTION: {task_instruction}
ACTION SEQUENCE: {action_sequence}
Output STRICT JSON with the following schema:

```
{
  "name": " ",
  "description": " ",
  "parameters": [{
    "name": " ",
    "type": " ",
    "semantic": " "
  }],
  "action_template": [{
    "verb": " ",
    "target": " ",
    "value": " ",
    "direction": " "
  }],
  "applicable_domains": [" "],
  "applicable_apps": [" "],
  "applicable_intent_classes": [" "],
  "preconditions": " ",
  "postconditions": " "
}
```

HARD RULES
A. PARAMETERIZATION
A1. Only parameterize values explicitly mentioned in the instruction or values that clearly vary across similar tasks, such as queries, dates, amounts, contact names, store names, song titles, and durations.
A2. NEVER parameterize stable UI labels.
A3. EVERY declared parameter MUST be referenced at least once in action_template, either in the target field or value field, using the format <param_name>.
B. CONSISTENCY
B1. action_template MUST follow the same execution order as the original action sequence. DO NOT reorder steps.
B2. description and action_template MUST be semantically consistent.
C. META VERBS
C1. NEVER include: finished, unknown in action_template.
Output ONLY the JSON object.

C.3 Instruction Personalization Prompt

Mobile users typically express high-level, target-oriented instructions (e.g., "order takeout") rather than step-by-step procedural commands. A generic agent interpreting such an instruction must guess the intended app, content, and operational choices, frequently producing executions that diverge from the user's actual habits. The Refiner closes this gap by grounding the raw instruction in the user's structured preference profile before any retrieval or planning begins. Specifically, it injects App Preference to resolve the target application, Content Interest to fill in implicit content choices, and Operation Preference to anticipate decision patterns, transforming an ambiguous intent into a complete, personalized instruction that subsequent agents can act on directly. This enrichment step mirrors how a human assistant, having learned a user's habits over time, would automatically fill in unspoken details without requiring explicit clarification. The full prompt is shown below.

Instruction Personalization

You are an AI assistant that specializes in interpreting and completing user instructions based on the user's profile and historical interaction records. Your goal is to transform short, underspecified, or ambiguous user inputs into complete, precise, and executable commands that align with the user's habitual preferences and prior behaviors.
Input: Task Instruction: {instruction} User Profile: {user_profile} Historical Instructions: {historical_instructions}
Task: Expand the user's current task instruction into a complete, explicit, and executable instruction by leveraging the user's profile and historical interaction behavior.
Requirements: 1. Use only information from the User Profile and Historical Instructions to personalize the instruction. 2. Keep the output in the same language as the user's original instruction unless the profile

```
indicates otherwise. 3. If insufficient information is available, produce a general but actionable expanded instruction. Output only the final expanded instruction without additional explanations.
Output:
```

C.4 Action Planning Prompt

Even with a fully personalized instruction and a retrieved skill template, directly commanding the Executor to execute without a high-level plan leaves it vulnerable to compounding errors across long action sequences. The Planner addresses this by synthesizing three information sources: the personalized instruction (capturing what to accomplish and how the user prefers it), the matched Universal Skill (providing the procedural template), and similar historical demonstrations. From these, it generates a plan that decomposes the task into logically ordered steps, giving the Executor a reliable roadmap to follow rather than making all decisions greedily from the current screen state. This hierarchical separation between planning and execution is the key design that allows ExpeLAgent to maintain coherent long-horizon behavior on complex GUI tasks. The full prompt is shown below.

Action Plannning

```
You are a GUI Agent Planner. Your role is to generate execution plans for GUI agents based on user tasks and available skills.
Input: Task Instruction: {instruction} Related Skills Reference: {skill_info} Historical Reference: {history_info}
Task: Analyze the user intent and available skills, and generate a detailed execution plan for the GUI agent.
Requirements: 1. Analyze the user intent and the available skills. 2. Generate a detailed execution plan in Chinese. 3. The plan should include: - High-level goal decomposition. - Specific operation steps (referencing the skills). - Consideration of user type and personal preferences. 4. The plan format should be natural language description, divided into steps.
Output only the execution plan, without any additional explanations.
Output:
```

C.5 Action Execution Prompt

The Executor is the lowest-level agent in ExpeLAgent's execution hierarchy: it observes the current screen state and selects the next atomic action to advance toward the task goal. Unlike a standalone GUI agent that must reason about intent, planning, and low-level control simultaneously, the Executor operates under a well-specified context assembled from four components: the personalized instruction (ensuring preference alignment), the global action plan from the Action Sequence Planner (providing structural guidance), the already-executed action history (preventing repetition and enabling recovery from failures), and the list of available UI elements (grounding actions in the actual interface). This rich context allows the Executor to focus exclusively on *local decision-making*, adapting its strategy step by step in response to execution feedback without losing sight of the overall goal. The full prompt is shown below.

Action Execution

```
You are an intelligent Android GUI agent.
Input Task Instruction: {instruction} Action Sequence Plan: {action_sequence_plan} Executed_actions_so_far: {executed_actions_so_far} Execution_Feedback: {feedback} UI Elements List: {ui_list_text}
Task: Given the current task instruction, the planned action sequence, the actions that have already been executed, execution feedback, and the list of available UI elements on the screen, your task is to decide the next appropriate GUI action to progress toward completing the task. You must reason about the current UI state, evaluate whether the previous actions were successful, adapt your strategy if necessary, and generate the next valid action from the defined action space. If all task objectives have been achieved, you should explicitly indicate task completion.
Requirements: 1. You must strictly operate within the provided Android GUI environment and interact only with the UI elements listed in the UI Elements List. 2. The generated action must be consistent with the current execution state and must not repeat any actions that have already been executed. 3. You should actively utilize execution feedback to adjust your decision-making process. If the feedback indicates failure or unexpected behavior, you must select an alternative strategy or a different UI element. 4. The output must follow the predefined action format and comply with all critical rules specified below.
Output:
```

D Details of Shared Memory Mechanism

This section provides additional details on the shared memory mechanism, including the embedding model used for user profile similarity, the user profile embedding process, and the selection of the neighbor set size.

To ensure efficient on-device deployment, the embedding model is instantiated with a lightweight architecture, i.e., Qwen3-Embedding-0.6B. This model is used to encode user profiles into dense

vector representations for similarity computation.

Each user profile is represented as a concatenation of four components:

$$\mathbf{p}_u = [\mathbf{e}_u^{\text{app}}; \mathbf{e}_u^{\text{content}}; \mathbf{e}_u^{\text{op}}; \mathbf{e}_u^{\text{basic}}], \tag{8}$$

where $\mathbf{e}_u^{\text{app}}$, $\mathbf{e}_u^{\text{content}}$, $\mathbf{e}_u^{\text{op}}$, and $\mathbf{e}_u^{\text{basic}}$ denote the embeddings of app preferences, content interests, operation habits, and basic information, respectively.

The top-k most similar users are retrieved based on cosine similarity over profile embeddings, where the neighbor set size is set to $k = 2$ in all experiments.

E User Clustering Details

Field	**Encoding Rule**
Occupation	student $\rightarrow$ 0 other $\rightarrow$ 1 retired $\rightarrow$ 2
Age	$< 25 \rightarrow 0$ 25–39 $\rightarrow$ 1 $\geq 40 \rightarrow 2$
Family status	single $\rightarrow$ 0 married, no children $\rightarrow$ 1 other $\rightarrow$ 2

Table 7: Feature encoding for user clustering.

Since most baselines do not explicitly learn user profiles, a more general user grouping strategy is adopted for evaluation. Demographic information of each user, including occupation, age, and family status, is collected during the construction of the FingerTip 20K dataset. Each user is represented by a three-dimensional vector encoding these three attributes. The encoding rules are listed in Table 7.

Users are partitioned into three clusters using KMeans with `n_clusters=3`, `random_state=42`, and `n_init=10`. The resulting clusters include: (1) a student group consisting of 63 users aged 18–24 and mostly unmarried, (2) a middle-aged group with children consisting of 13 users, and (3) a young working-professional group consisting of 7 unmarried users aged 23–27.