PAL-UI: Planning with Active Look-back for Vision-Based GUI Agents

Anonymous ACL submission

Abstract

Graphical User Interface (GUI) agents powered by Multimodal Large Language Models (MLLMs) promise human-like interaction with software applications, yet long-horizon tasks remain challenging due to memory limitations. Existing approaches either truncate history or rely on simple textual summaries, which risk losing critical information when past visual details become necessary for future decisions. In this paper, we propose **PAL-UI** (**P**lanning with **A**ctive **L**ook-back), a novel framework that enables GUI agents to adaptively retrieve past observations when required. PAL-UI combines a dual-level summary agent, capturing both observation-level cues and action-level outcomes, with a dedicated retrieval tool that allows the agent to recall specific historical screenshots during planning. We curate a step-level instruction dataset of 8.6K samples from mobile GUI navigation trajectories and train **PAL-UI-3B** and **PAL-UI-7B** models based on Qwen2.5-VL. Extensive experiments show that PAL-UI significantly outperforms baseline models and prior methods in mobile GUI tasks, even under data-efficient settings. Moreover, PAL-UI exhibits strong cross-domain generalization, achieving notable improvements in web navigation without additional training. Our work highlights the potential of active memory retrieval for long-horizon planning capabilities of vision-based GUI agents.

1 Introduction

Large Language Models (LLMs) have dramatically advanced the capabilities of AI systems in recent years (Brown et al., 2020; Zhao et al., 2023). This progress has spurred the development of GUI agents, i.e., autonomous agents that perform tasks via graphical user interfaces (GUI) (Wang et al., 2024; Hong et al., 2024). Early paradigms for LLM-driven GUI agents typically relied on converting visual interface information into textual form (e.g., reading an app's accessibility tree or metadata) so that LLMs could process it (Nakano et al., 2021; Zhang and Zhang, 2023). However, such text-based representations often require external modules and inject a large number of additional tokens into the context, limiting efficiency and fidelity. With the emergence of Multimodal LLMs (MLLMs) that can directly handle images as input (Liu et al., 2023), a new vision-based GUI agent paradigm has surfaced (Gou et al., 2024; Xu et al., 2024). These agents perceive raw screenshots of the interface and simulate human-like operations on the GUI, enabling end-to-end interaction without intermediate text conversions.

A central challenge for long-horizon GUI tasks is to incorporate memory of past observations and actions into the agent's planning process. In traditional text-based agents, a common strategy is to append the entire interaction history to the input context for planning (Yao et al., 2023). Unfortunately, directly extending this to vision-based agents is infeasible. Visual observations are much heavier than text, i.e., each image input to an MLLM is encoded into a large number of tokens (Bai et al., 2025), quickly exhausting the model's context length. Moreover, current MLLMs struggle to reason effectively over many images at once; recent studies show that their capability to handle multiple simultaneous images remains quite limited (Zhao et al., 2024). Simply providing all past screenshots as input can overwhelm the model with redundant information and obscure the relevant details. As a result, existing vision-enabled GUI agents resort to very restrictive memory usage: some only feed the most recent action or a brief textual summary of it into the context (Chen et al., 2024; Xu et al., 2024), while others retain only a few of the last screenshots as visual memory (Qin et al., 2025). These ad-hoc strategies risk losing critical information from earlier steps and ultimately constrain the agent's performance on complex, multi-step tasks.

To address the above limitations, we introduce

Planning with **A**ctive **L**ook-back for **GUI** task (**PAL-UI**), a paradigm that lets a GUI agent *selectively* consult its own history on demand. The key innovation is an **active look-back** mechanism: instead of carrying an ever-growing visual context, the agent invokes a tool interface to fetch *exact* historical evidence (e.g., a screenshot from a prior step) only when that evidence is relevant for the current decision. This yields the best of both worlds, broad access to rich past information without the context bloat and distraction that plague naïve long-context approaches. The agent runs in an iterative loop. At each step, it compacts the trajectory so far into two lightweight summaries: an observation-level outline capturing salient visual cues, and an action-level synopsis recording intents and executed operations. Based on these summaries, the agent decides whether to *look-back* for retrieval or *directly predict* the next action. In effect, PAL-UI mirrors how humans briefly "glance back" at a prior screen when needed using history as a searchable memory, not as a permanent burden on the model's context.

Training an agent to perform active look-back poses unique challenges, as typical demonstration datasets lack both tool-use annotations and explicit reasoning traces. To bridge this gap, we construct a synthetic instruction-tuning corpus that augments raw trajectories from AndroidControl (Li et al., 2024) with simulated tool-calling behavior. Concretely, we design a four-stage **deliberated look-back framework** that guides a stronger teacher model to decide when, what, and how to retrieve past observations. For these curated trajectories, we filter for correctness, rebalance samples to prevent recency bias in retrieval, enrich with non-retrieval cases to avoid overfitting, and finally standardize into a structured format. The resulting dataset comprises **8.6K** high-quality trajectories. We fine-tune a Qwen2.5-VL (Bai et al., 2025) backbone on this corpus to obtain two variants, **PAL-UI-3B** and **PAL-UI-7B**, both demonstrating the capacity to reflect, judge when to look back, and act effectively in long-horizon GUI planning tasks.

We evaluate our PAL-UI agent on a broad set of GUI navigation benchmarks. Experimental results demonstrate that PAL-UI significantly outperforms the base MLLM on long-horizon mobile UI tasks, achieving new state-of-the-art performance under comparable data settings. Notably, by effectively leveraging historical context, our method yields higher success rates than prior methods even with far fewer training examples. Moreover, although our training data focused on mobile app environments, the PAL-UI agent exhibits strong zero-shot transfer to other domains, such as web browser interfaces. It substantially improves task success on web-based GUI benchmarks compared to baselines, highlighting the generality of our approach.

2 Related Work

2.1 GUI Agents

With the rapid progress of large language models (LLMs) and multimodal LLMs (MLLMs), GUI agents (Gou et al., 2024; Qin et al., 2025) have made remarkable strides across diverse interface platforms. Early approaches (Zhang and Zhang, 2023; Zheng et al., 2024a) relied on structured representations such as HTML or accessibility trees for element localization, which struggled to generalize across environments. Recent studies shift toward vision-based GUI agents (Gou et al., 2024; Xu et al., 2024), which perceive screenshots directly and interact through human-like mouse and keyboard actions. Such paradigm paves the way toward more general and scalable GUI intelligence.

2.2 Memory Management for GUI Agents

Previous LLM-based agent systems (Yao et al., 2023) typically manage memory by appending all historical observations and actions to the input context. Yet, this poses unique challenges for vision-based GUI agents. Since their observations are screenshots (Cheng et al., 2024), storing full visual histories incurs prohibitive token costs (Bai et al., 2025) and introduces multi-image inputs that can hinder reasoning (Zhao et al., 2024). Therefore, most existing methods (Chen et al., 2024; Xu et al., 2024; Qin et al., 2025) keep only recent actions or observations, inevitably losing important information. To overcome this, we propose PAL-UI, a paradigm that enables agents to plan with active look-back, retrieving specific visual details from history on demand during inference. This design preserves rich historical context without suffering from context explosion or degraded reasoning.

2.3 MLLM-based Tool-use Agents

Enhancing MLLMs with tool calling has become a vital direction, as external tools help models overcome inherent capability limits. Early works (Wu et al., 2023; Yang et al., 2023c) used training-free prompting to invoke tools for improved visual per-

ception. Later works, e.g., LLaVA-Plus (Liu et al., 2024) and GPT4Tools (Yang et al., 2023b), advanced this idea by synthesizing tool-calling trajectories and fine-tuning models for explicit tool-use proficiency. With the rise of slow-thinking reasoning (Jaech et al., 2024; Guo et al., 2025), recent efforts have shifted toward "thinking with images" (Su et al., 2025; Zhou et al., 2025), where tools are invoked during reasoning to modify visual inputs, yielding substantial gains in reasoning tasks. For GUI agents, early approaches (Zheng et al., 2024a) relied on structured cues such as Set-of-Marks (SoM) (Yang et al., 2023a) or accessibility trees, while recent work like FOCUS (Tang et al., 2025) introduced a dual-system framework with fast and slow prediction to improve grounding. To our knowledge, PAL-UI is the first to leverage external tools for long-horizon planning in GUI agents, enabling active retrieval of past visual states through an active look-back mechanism.

3 Problem Formulation

We consider a GUI-based sequential decision process where an agent must achieve a specified goal by interacting with a user interface. Formally, at each time step i, the agent receives an observation o_i of the current state (e.g., a screenshot) and selects an action a_i from the set of possible actions (such as clicking a button, entering text, or scrolling). Executing action a_i changes the interface state, leading to a new observation o_{i+1}. The process continues until the agent finishes the global task G, which is typically given as a high-level instruction, or reaches a maximum number of steps.

The interaction process can be formalized as a trajectory $\tau = (o_0, a_1, o_1, \ldots, a_T, o_T)$, where o_0 is the initial observation and T is the total number of steps. The core challenge in this setting lies in *long-horizon planning and memory management*: as T grows, the agent accumulates an ever-growing history of observations and actions (often high-dimensional multimodal inputs such as images with text). A naive strategy of feeding the entire raw history into a vision-based agent quickly becomes infeasible due to context length limits and redundant information. More critically, without mechanisms for selectively compressing, retaining, and retrieving salient past information, LLM-based planners suffer degraded performance, as essential early-step details become buried within an overwhelming token stream. Our objective is to equip the agent with the ability to preserve key information throughout its interaction history and perform active look-back when required, all while operating within realistic context budgets. We follow previous studies (Bai et al., 2025; Qin et al., 2025) to adopt a unified action space for interacting with the GUI interface, with more details provided in Appendix A

4 Approach

We propose **PAL-UI** (**P**lanning with **A**ctive **L**ook-back), a framework to enhance the long-horizon planning capabilities of a GUI agent by combining dual-level summarization of the history with an active retrieval mechanism. Figure 1 gives an overview of our approach. In particular, PAL-UI compresses the interaction history into a succinct textual memory and equips the agent with a special tool to fetch detailed visual information from past steps on demand. We organize our approach as follows: first, we introduce the construction of the summarization agent and the retrieval tool (Section 4.1); second, we explain how the agent plans with active look-back at inference time (Section 4.2); finally, we describe the training data generation pipeline used to impart these capabilities to the agent (Section 4.3).

4.1 Tool Construction and Summary Agent

To prevent the agent's context from being overwhelmed by redundant visual tokens, we compress the history at two levels, i.e., observations and actions, using a **dual-level summary agent**. Meanwhile, we introduce a **memory retrieval tool** that the agent can invoke to recover observations from any past step when detailed information is needed.

Observation-level Captioner. At each step, we first summarize the newly observed visual input o_i to extract task-relevant information. Although a screenshot may contain cues essential for achieving the long-term goal, such cues are usually embedded within a large number of UI elements. Directly appending all raw observations $\{o_1, o_2, \ldots, o_i\}$ to the prompt would rapidly exhaust the context window as i grows, and the model would struggle to locate important details in the noise. Thus, we generate a concise observation caption at each step that distills the salient content of o_i, with particular emphasis on screen text and distinctive UI changes, relevant to the global goal G. We employ the large vision-language model Qwen-2.5-VL (Bai et al., 2025) as

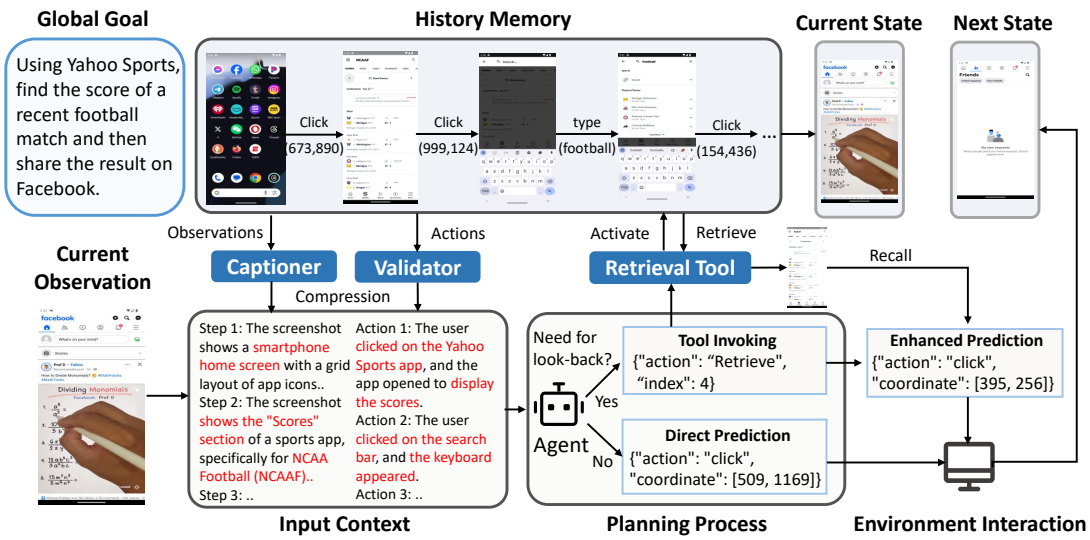

Figure 1: The framework of our proposed PAL-UI agent. We utilize an observation-level captioner and an action-level validator for memory compression, and then equip the agent with a retrieval tool to actively recall visual information from past memory.

our captioning module, due to its strong OCR capability and instruction-following performance. The captioner is prompted with the current screenshot o_i and the task goal G to produce a brief description focusing on the most relevant UI content (e.g., widget labels, displayed messages).

Action-level Validator. Beyond summarizing observations, we also summarize the outcome of each action to form an action memory. Prior works on GUI agents often record only the action sequence as history (Chen et al., 2024; Xu et al., 2024). However, without the accompanying state information, such action logs provide little insight into task progress. More importantly, these methods generally do not verify whether an action actually succeeded, leaving the agent susceptible to repeating failed attempts. To address this, we introduce an action-level validator that produces a compact natural-language summary of the last action a_i and evaluates its execution result by comparing the GUI state before and after the action. Using the same Qwen-2.5-VL model, we prompt it with a triple of the current action a_i, pre-action observation o_{i-1}, and the post-action observation o_i. The validator then generates a brief assessment, for example, "*The user typed 'football' into the search bar, and the search results for related content are displayed.*". This summary serves two purposes: it clarifies the intent of a_i in natural language (which can be easier for the LLM to reason than raw action code) and it confirms whether a_i achieved the expected effect on the GUI state.

Memory Retrieval Tool. While the dual-level summaries dramatically condense the history, converting rich visual observations into text can inevitably lose some details, e.g., the exact appearance of a screen or an unseen UI element that later becomes relevant. To allow the agent to recover such details when needed, we implement a Retrieve tool (`Retrieve`) for active look-back. The Retrieve tool can be invoked with a past step index j and returns the observation o_j (e.g., the screenshot image at step j $(j < i)$) to the agent's context. We add this tool to the agent's action space, so it can decide at any point to retrieve a past screen instead of executing a normal GUI action. The retrieved image o_j is then appended to the agent's current context, enabling the model to re-inspect that state before continuing the plan. In essence, the agent has the choice to temporarily step back and "refresh its memory" of a previous interface state.

We adopt index-based retrieval instead of semantic retrieval (Lewis et al., 2020) because GUI trajectories exhibit strong temporal dependencies: the screen the agent needs to revisit is typically determined by when it occurred, not by semantic similarity. In GUI environments, many screenshots are visually and semantically similar yet correspond to different functional states, making embedding-based retrieval unreliable and prone to retrieving the wrong step. Index-based retrieval therefore provides deterministic, noise-free access to the exact

past observation the agent intends to recall, which is essential for accurate action prediction.

4.2 Active Look-back Planning Process

With the ability to summarize history and retrieve past observations, the agent plans actions in a think-act loop that actively looks back when necessary. At the i-th step, the agent's context includes: (1) the task goal G; (2) the compressed memory m_i up to step i (consisting of relevant summaries of past observations and actions); and (3) the current observation o_i (the latest GUI screenshot). Using this context, the agent generates either a concrete GUI action a_i or a retrieval query. There are two possible outcomes for the planning at step i:

• **Direct Action Prediction**: In the typical case, the agent uses the context $\{G, m_i, o_i\}$ to directly predict the next GUI action a_i (discussed in Section 3). This prediction is informed by the distilled knowledge in the summaries such as what has been done so far and what key information is on the current screen, allowing the agent to decide the best next step toward the goal.

• **Active Retrieval Action**: If the agent is uncertain or needs more details from a previous state, it can choose to invoke the `Retrieve` tool for accessing previous screenshots. For example, the model might decide to "*look back at the login screen (step 2) to recall the exact error message*" before making a specified GUI action. When the retrieve tool is called, a past screenshot o_j is fetched and added to the context. The agent then continues with the augmented context $\{G, m_i, o_i, o_j\}$, and produces the next action a_i based on both the current state and the recalled observation from step j. By integrating this active look-back mechanism into the reasoning process, the agent can significantly improve its prediction accuracy, especially in situations where subtle details from earlier in the task are needed to choose the correct action.

Discussion and Comparison. Although PAL-UI includes both a memory module and retrieval tool, its design fundamentally differs from prior "memory + retrieval" paradigms used in text-based and multimodal agents. Prior works (Lewis et al., 2020; Qin et al., 2025) typically rely on passive caches or semantic stores where the model retrieves information opportunistically or through similarity search. In contrast, PAL-UI introduces an active look-back mechanism governed by a *deliberated, step-level reasoning* policy: the agent explicitly decides *when* to retrieve, *what* to retrieve (the exact historical step), and *why* retrieval is necessary for the current decision. This transforms retrieval from a peripheral capability into a structured planning primitive for long-horizon GUI tasks.

4.3 Training Data Construction

Equipping the agent with the above capabilities requires appropriate training data that demonstrates when and how to use the retrieval tool. However, conventional datasets do not contain examples of tool usage or reasoning traces. We therefore create a synthesized fine-tuning dataset of tool-augmented trajectories via a combination of distillation from a stronger model and careful data curation. The process consists of four main steps:

Seed Trajectory Collection. We begin with a large set of human demonstration trajectories from the training set of AndroidControl (Li et al., 2024), each consisting of a sequence of GUI screenshots $\{o_0, o_1, ..., o_T\}$, human actions $\{a_1, ..., a_T\}$, and a global task goal G. These serve as the ground-truth action sequences that our agent should ideally replicate. For each trajectory, we run the dual-level summary agent at each step to generate the compressed memory m_i.

Tool-Use Calling Curation. Given that seed trajectories consist only of observations and ground-truth actions, they lack explicit tool-use steps. To construct training trajectories that include retrieval actions, we require the teacher model to simulate when and how an agent would invoke the `Retrieve` tool. However, prompting the teacher model directly to produce tool calls proved ineffective, as the base model was not pretrained on such interactions. To address this gap, we design a four-stage *deliberated look-back framework* that gradually guides the teacher to integrate retrieval into its reasoning. At each step, the teacher model is prompted to perform the following steps:

• **History Revision**: The model reviews the compressed memory m_i to assess task progress toward the goal G, ensuring a clear understanding of what has been done so far.

• **Candidate Proposals**: The model proposes several plausible next actions based on the current observation o_i, encouraging it to enumerate possible strategies rather than committing prematurely.

• **Confidence Evaluation**: The model reflects on how confident it is in each proposed action and

whether it feels uncertain enough to warrant checking something in the past. If the model determines that a specific detail from an earlier step is needed, it should invoke the `Retrieve` tool.

• **Tool-Use Action Prediction**: Depending on the previous stage, the model will receive the retrieved observation o_j (from step j) if retrieval was requested and then outputs a_i with the additional context.

Data Filtering and Balancing. After synthesizing a large set of trajectories with tool calling, we then filter and balance this data to focus on high-quality examples. First, we discard any sample where the teacher's final predicted action a_i does not match the ground-truth human action for that step, ensuring our training data only contains correct predictions. Among the remaining samples, we identify those in which the teacher actually utilized the retrieval tool and succeeded in choosing the correct action afterward. We found about 4.3K such high-quality tool-use cases. Since the teacher model tended to prefer looking at very recent steps, which might bias the agent to only look back one step, we increased the sampling weight of samples where the retrieved step j was further back in history. Finally, to prevent the agent from overusing the tool when it's not necessary, we also include 4.3K high-quality samples where the teacher solved the step without any retrieval (direct action with correct outcome). This yields a balanced dataset of roughly 8.6K step-level samples, half with tool use and half without, all with correct decisions.

Compilation and Formatting. After obtaining data samples, we compile them into the Qwen2.5-VL format for supervised fine-tuning (SFT) (Bai et al., 2025). At each step, the input consists of the system prompt, compressed memory m_i, and the current observation o_i, while the output is composed of the reasoning process and actions. To make a coherent reasoning, we first employ Qwen3-32B (Yang et al., 2025) to synthesize all model responses from each stage into a logical natural language explanation. This explanation will serve as the thinking process, enclosed in `<think></think>` tokens. Finally, the action (`Retrieval` or other types of actions in Section 3) is encircled by `<tool_use></tool_use>` tokens and the retrieved screenshot will be injected into the next step. This standardized format teaches the agent how to reflect, when to look back, and how to act effectively.

5 Experiments

5.1 Experiment Setup

We conduct extensive experiments across both offline and online benchmarks to evaluate the efficacy of our approach. For offline evaluation, We utilize two high-level GUI planning tasks, **AndroidControl-High** (Li et al., 2024) and **GUI-Odyssey** (Lu et al., 2024). For online evaluation, we adopt **AndroidWorld** (Rawles et al., 2024) and **AndroidLab** (Xu et al., 2025). Furthermore, we evaluate our model on **Mind2Web** (Deng et al., 2023), a web dataset for cross-platform evaluation. More information regarding the evaluation benchmarks and implementation details are presented in Appendix B.

5.2 Results

Offline Performance. Table 1 presents results on two offline GUI benchmarks. Compared with the zero-shot Qwen2.5-VL base models, PAL-UI achieves significant improvements across all benchmarks. Specifically, PAL-UI-7B achieves a 58.9% overall score, marking an 8.3% absolute improvement over the base model. This gain is more pronounced in Step Success Rate (SR); for instance, PAL-UI-7B achieves 57.8% on AndroidControl-High, surpassing the base model by 10.7%. Notably, although trained exclusively on AndroidControl, our model exhibits strong cross-domain transferability, increasing out-of-domain GUI-Odyssey SR to 41.7%. Furthermore, PAL-UI consistently outperforms state-of-the-art SFT and RFT baselines. With a 58.9% overall score, PAL-UI-7B surpasses GUI-R1-7B (56.1%) and significantly outperforms the leading SFT method, NaviMaster-7B. It establishes new competitive performances in Grounding Accuracy (70.5% on AndroidControl-High; 46.8% on GUI-Odyssey) and SR. Finally, consistent improvement from 3B to 7B variants validates the scalability of our approach.

Online Performance. We present online benchmark results in Table 2. We employ the standard protocol from mobile-use (Li et al., 2025) framework for evaluation and limit the maximum steps to 25. As we can see, the baseline approaches show limited gains, highlighting their constraints in dynamic environments. Conversely, PAL-UI achieves substantial improvements: 11.1% on AndroidWorld and 9.8% on AndroidLab, significantly outperforming all baselines. These gains are more

Table 1: Results on two mobile GUI datasets. "Method" indicates the training and inference method. "ZS", "SFT" and "RFT" are short for zero-shot, supervised fine-tuning and reinforcement fine-tuning, respectively. **Bold** and underline denote the best and second best results.

Model	Method	AndroidControl-High			GUI-Odyssey			Overall
		Type	Grounding	Step Success	Type	Grounding	Step Success	
GPT-4o	ZS	63.1	30.9	21.2	37.5	14.2	5.4	28.7
Qwen2.5-VL-3B	ZS	47.8	46.5	38.9	37.4	26.5	26.7	37.3
Qwen2.5-VL-7B	ZS	68.7	59.7	47.1	55.6	37.8	34.4	50.6
OS-Atlas-4B	SFT	49.0	49.5	22.8	49.6	34.6	20.3	37.6
OS-Atlas-7B	SFT	57.4	54.9	29.8	60.4	39.7	27.0	44.8
NaviMaster-7B	SFT	**72.9**	-	54.0	64.4	-	36.9	-
UI-R1-3B	RFT	57.8	55.7	45.4	52.2	34.5	32.5	46.4
GUI-R1-3B	RFT	58.0	56.2	46.5	54.8	41.5	41.3	49.8
GUI-R1-7B	RFT	71.6	65.6	51.7	**65.5**	43.6	38.8	56.1
PAL-UI-3B	SFT	60.4	58.7	49.3	56.7	36.9	34.6	49.4
PAL-UI-7B	SFT	71.3	**70.5**	**57.8**	65.1	**46.8**	**41.7**	**58.9**

Table 2: Performance of PAL-UI on online GUI tasks. AW_{SR} and AL_{SR} indicates the task-level success rate for AndroidWorld and AndroidLab, respectively.

Setting	Avg. Step	AW_{SR}	AL_{SR}
OS-Atlas-7B	15.8	20.6	11.9
UI-R1-7B	15.3	23.4	15.7
GUI-R1-7B	13.2	26.8	18.3
Qwen2.5-VL-7B	16.4	19.8	12.3
PAL-UI-7B	11.6	**30.9**	**22.1**

pronounced in online tasks than offline settings, underscoring our method's potential in such scenarios. Furthermore, PAL-UI achieves higher accuracy with fewer average steps, indicating that the Active-look mechanism facilitates superior decision-making and prevents behavioral drift from erroneous actions that typically lead to redundant reasoning.

Cross-platform Performance. Since PAL-UI is trained solely on mobile data and evaluated on mobile benchmarks, we further examine its ability to generalize across platforms. We present the results of PAL-UI on Multimodal-Mind2Web, an web navigation benchmark in Table 3. Despite being trained solely on mobile trajectories, PAL-UI achieves consistent improvements on web tasks, with average gains of 3.8 and 2.4 points for the 7B and 3B models, respectively. While these gains are smaller than those observed in mobile domain, they still demonstrate that PAL-UI effectively transfers its planning capability to a different platform. This ability to maintain performance across different environments highlights the robustness and generalization potential of our approach.

6 Further Analysis

Ablation Study. We evaluate the components of the PAL-UI framework, with results summarized in Table 4. We present our findings below: First, the dual-level summary agent (SA) consistently improves performance in zero-shot and SFT settings. For instance, on the out-of-domain GUI-Odyssey benchmark, SA boosts SR from 34.4 to 38.9, confirming it mitigates information loss inherent in action-only memory. Second, while SFT alone yields in-domain gains (SR on AC-High improves from 47.1 to 53.2), its effect on generalization is limited, with only a minor increase on GUI-Odyssey (34.4 to 36.4). This highlights the difficulty of transferring knowledge without a richer memory mechanism. Third, omitting planning with active look-back (PAL) hampers effectiveness, as SR drops on AC-High from 53.2 to 52.3. This indicates that without rich historical summaries, the agent struggles to judge the relevance of retrieved information for correct actions. Finally, full PAL-UI, integrating SFT, SA, and PAL, achieves the strongest results on both benchmarks (SR 57.8 on AC-High and 41.7 on GUI-Odyssey). These results show that the synergy of summarization and active retrieval is essential for robust long-horizon planning and cross-domain generalization.

Context Length Comparison. We further analyze the trade-off between context length and performance across three paradigms: PAL-based reasoning, action-only memory, and full screenshot memory. Using 50 trajectories from the AndroidControl test set, we measure input token lengths for each setting (Table 5). Results indicate that

Table 3: Cross-platform results of PAL-UI on Multimodal-Mind2web. We report the model performance on three benchmark splits: cross-task, cross-website, and cross-domain.

Model	Cross-Task			Cross-Website			Cross-Domain		
	Operation	Element	SR	Operation	Element	SR	Operation	Element	SR
Qwen2.5-VL-3B	53.8	26.5	25.9	50.3	25.3	23.4	53.5	30.4	28.3
Qwen2.5-VL-7B	61.3	33.1	32.3	57.1	31.7	29.8	61.0	36.8	34.4
PAL-UI-3B	54.5	27.9	27.6	54.6	25.9	25.1	57.1	33.3	31.9
PAL-UI-7B	68.7	36.0	35.0	69.2	36.4	35.2	69.6	39.6	37.9

Table 4: Ablation results. "SA" indicates summary agents. "PAL" indicates planning with active look-back mechanism. "Full" indicates our full approach, where we train the model on our constructed dataset and leverage SA and PAL during inference.

Setting	AC-High		GUI-Odessey	
	Type	SR	Type	SR
Zero-Shot	68.7	47.1	55.6	34.4
+ SA	70.9	54.6	62.8	38.9
+ SFT	**73.4**	53.2	56.3	36.4
+ SFT & SA	72.9	56.3	59.4	38.4
+ SFT & PAL	67.4	52.3	56.6	37.1
+ Full	71.3	**57.8**	**65.1**	**41.7**

Table 5: Comparison of context length and performance for different paradigm. *None*: not using memory; *+A*: using only actions as memory; *+ 5O* and *+ AO*: using recent 5 screenshot or all historical screenshots as memory; *+SA*: using summary agent for memory processing; *+ PAL*: using the full approach.

Setting	Len.	AC-High		GUI-Odessey	
		Type	SR	Type	SR
None	4.31K	65.4	45.8	52.6	34.6
+ A	4.37K	68.7	47.1	55.6	34.4
+ 5O	12.63K	69.1	48.3	56.8	36.4
+ AO	16.38K	67.8	45.5	54.6	34.9
+ SA	4.97K	70.9	54.6	62.8	38.9
+ PAL	7.33K	**71.3**	**57.8**	**65.1**	**41.7**

screenshot-based methods consume the most tokens with minimal benefit, as redundant visual data increases computational costs and prediction errors. Conversely, the action-only approach uses the fewest tokens but performs poorly due to losing critical historical details. Our PAL-UI balances these extremes, adding modest tokens while delivering significant gains. Notably, PAL-UI achieves a 6% average performance improvement using only 44% of the tokens required by the screenshot-based approach.

Effect of Summarization Models. We conduct experiments to evaluate how various summarization models affect final performance, specifically leveraging the Qwen2.5-VL (Bai et al., 2025) and InternVL-3 (Zhu et al., 2025) series. As shown in Table 6, model choice does not significantly impact overall results. This likely stems from current MLLMs' robust OCR and visual comprehension, which allow accurate extraction of key GUI information and identification of action intentions across screenshots. Consequently, a compact model can be employed for memory compression to improve performance with minimal overhead.

Table 6: Results of PAL-UI with different summarization models.

Summarization Model	AC-High		GUI-Odessey	
	Type	SR	Type	SR
Qwen2.5-VL-3B	70.9	57.4	63.8	41.0
Qwen2.5-VL-7B	71.3	57.8	65.1	41.7
InternVL3-2B	69.4	56.2	64.2	39.8
InternVL3-8B	72.5	58.4	64.0	41.3

7 Conclusion

We presented PAL-UI (Planning with Active Look-back), a framework empowering GUI agents to selectively retrieve past observations rather than carrying full history. By combining dual-level summarization with an active retrieval tool, PAL-UI balances efficiency and completeness, handling long-horizon tasks with improved accuracy and reduced context overhead. We introduced a deliberated look-back framework for constructing tool-augmented trajectories, yielding 8.6K high-quality instruction-tuning samples. Experiments show that PAL-UI significantly outperforms base MLLMs and baselines on mobile navigation benchmarks and generalizes to out-of-domain web environments. These results underscore the importance of active memory retrieval for robust GUI planning. Future work will explore more complex tasks, reinforcement learning integration, and broader applicability to real-world interactive systems.

8 Limitations

In this paper, we present PAL-UI, a framework that enables GUI agents to proactively retrieve historical memories to enhance reasoning. Although PAL-UI achieves promising performance on downstream benchmarks, certain limitations remain. First, while we construct valuable tool-use data, the current trajectories primarily focus on mobile tasks, leaving the potential of our approach on web or desktop platforms not fully explored. Second, the proposed method can be integrated with reinforcement learning to further enhance the flexibility and effectiveness of proactive retrieval. Third, additional memory editing tools can be incorporated for more efficient context management within GUI scenarios. We will further investigate these directions in future studies.

References

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

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, and 1 others. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Wentong Chen, Junbo Cui, Jinyi Hu, Yujia Qin, Junjie Fang, Yue Zhao, Chongyi Wang, Jun Liu, Guirong Chen, Yupeng Huo, and 1 others. 2024. Guicourse: From general vision language models to versatile gui agents. *arXiv preprint arXiv:2406.11317*.

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

Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Sam Stevens, Boshi Wang, Huan Sun, and Yu Su. 2023. Mind2web: Towards a generalist agent for the web. *Advances in Neural Information Processing Systems*, 36:28091–28114.

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

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

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.

Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, and 1 others. 2024. Openai o1 system card. *arXiv preprint arXiv:2412.16720*.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Ning Li, Xiangmou Qu, Jiamu Zhou, Jun Wang, Muning Wen, Kounianhua Du, Xingyu Lou, Qiuying Peng, and Weinan Zhang. 2025. Mobileuse: A gui agent with hierarchical reflection for autonomous mobile operation. *arXiv preprint arXiv:2507.16853*.

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

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. 2023. Visual instruction tuning. *Advances in neural information processing systems*, 36:34892–34916.

Shilong Liu, Hao Cheng, Haotian Liu, Hao Zhang, Feng Li, Tianhe Ren, Xueyan Zou, Jianwei Yang, Hang Su, Jun Zhu, and 1 others. 2024. Llava-plus: Learning to use tools for creating multimodal agents. In *European conference on computer vision*, pages 126–142. Springer.

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

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

Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, and 1 others. 2021. Webgpt: Browser-assisted question-answering with human feedback. *arXiv preprint arXiv:2112.09332*.

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

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

Zhaochen Su, Linjie Li, Mingyang Song, Yunzhuo Hao, Zhengyuan Yang, Jun Zhang, Guanjie Chen, Jiawei Gu, Juntao Li, Xiaoye Qu, and 1 others. 2025. Openthinkimg: Learning to think with images via visual tool reinforcement learning. *arXiv preprint arXiv:2505.08617*.

Fei Tang, Yongliang Shen, Hang Zhang, Siqi Chen, Guiyang Hou, Wenqi Zhang, Wenqiao Zhang, Kaitao Song, Weiming Lu, and Yueting Zhuang. 2025. Think twice, click once: Enhancing gui grounding via fast and slow systems. *arXiv preprint arXiv:2503.06470*.

Shuai Wang, Weiwen Liu, Jingxuan Chen, Yuqi Zhou, Weinan Gan, Xingshan Zeng, Yuhan Che, Shuai Yu, Xinlong Hao, Kun Shao, and 1 others. 2024. Gui agents with foundation models: A comprehensive survey. *arXiv preprint arXiv:2411.04890*.

Chenfei Wu, Shengming Yin, Weizhen Qi, Xiaodong Wang, Zecheng Tang, and Nan Duan. 2023. Visual chatgpt: Talking, drawing and editing with visual foundation models. *arXiv preprint arXiv:2303.04671*.

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

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

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Jianwei Yang, Hao Zhang, Feng Li, Xueyan Zou, Chunyuan Li, and Jianfeng Gao. 2023a. Set-of-mark prompting unleashes extraordinary visual grounding in gpt-4v. *arXiv preprint arXiv:2310.11441*.

Rui Yang, Lin Song, Yanwei Li, Sijie Zhao, Yixiao Ge, Xiu Li, and Ying Shan. 2023b. Gpt4tools: Teaching large language model to use tools via self-instruction. *Advances in Neural Information Processing Systems*, 36:71995–72007.

Zhengyuan Yang, Linjie Li, Jianfeng Wang, Kevin Lin, Ehsan Azarnasab, Faisal Ahmed, Zicheng Liu, Ce Liu, Michael Zeng, and Lijuan Wang. 2023c. Mm-react: Prompting chatgpt for multimodal reasoning and action. *arXiv preprint arXiv:2303.11381*.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. React: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*.

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

Bingchen Zhao, Yongshuo Zong, Letian Zhang, and Timothy Hospedales. 2024. Benchmarking multi-image understanding in vision and language models: Perception, knowledge, reasoning, and multi-hop reasoning. *arXiv preprint arXiv:2406.12742*.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, and 1 others. 2023. A survey of large language models. *arXiv preprint arXiv:2303.18223*, 1(2).

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

Yaowei Zheng, Richong Zhang, Junhao Zhang, Yanhan Ye, Zheyan Luo, Zhangchi Feng, and Yongqiang Ma. 2024b. Llamafactory: Unified efficient fine-tuning of 100+ language models. *arXiv preprint arXiv:2403.13372*.

Yiyang Zhou, Yangfan He, Yaofeng Su, Siwei Han, Joel Jang, Gedas Bertasius, Mohit Bansal, and Huaxiu Yao. 2025. Reagent-v: A reward-driven multi-agent framework for video understanding. *arXiv preprint arXiv:2506.01300*.

Jinguo Zhu, Weiyun Wang, Zhe Chen, Zhaoyang Liu, Shenglong Ye, Lixin Gu, Hao Tian, Yuchen Duan, Weijie Su, Jie Shao, and 1 others. 2025. Internvl3: Exploring advanced training and test-time recipes for open-source multimodal models. *arXiv preprint arXiv:2504.10479*.

A Action Spaces

We follow previous studies (Bai et al., 2025; Qin et al., 2025) to adopt a unified action space for interacting with the GUI interface. Specifically, we introduce the following three types of actions:

- General actions across all platforms: `Click`, `Type`, `Scroll`, `Drag`, `Wait`, and `Finished`;
- Actions for mobile platform: `LongPress`,`OpenApp`, `PressHome`, and `PressBack`;
- Actions for web platform: `Hotkey`, `LeftDouble`, and `RightSingle`.

Note that some actions may require action-related content (e.g., coordinates for `Click` and text string for `Type`). During inference, we prompt the GUI agent to generate a reasoning text and an action, encircled in `<think></think>` and `<tool_use></tool_use>`.

B Experiments Setup

B.1 Datasets and Metrics

We conduct extensive experiments across both offline and online benchmarks to evaluate the efficacy of our approach. Initially, we utilize two high-level GUI planning tasks: **AndroidControl-High** (Li et al., 2024) and **GUI-Odyssey** (Lu et al., 2024). To further examine the model's performance in dynamic environments, we incorporate two more challenging online benchmarks, **AndroidWorld** (Rawles et al., 2024) and **AndroidLab** (Xu et al., 2025). Furthermore, despite being trained exclusively on mobile data, we evaluate our model on the **Mind2Web** (Deng et al., 2023) dataset for cross-platform evaluation. For offline mobile task, we follow prior work (Gou et al., 2024; Xu et al., 2024) and adopt a subtask-level evaluation paradigm where we examine the model's prediction on each step and report the *Type Match Score*, *Grounding Accuracy (GA)*, and *Step Success Rate (SR)* for offline mobile task. For offline web task, we following the official evaluation protocol and reporting Operation F1 (Operation), Element Accuracy (Element), and Step Success Rate (SR). For online tasks, we report the overall task successful rate.

B.2 Implementation Details

Following prior work (Luo et al., 2025), we employ 3B and 7B variants of Qwen2.5-VL (Bai et al., 2025) as the backbone model and use the 7B version as the summary agent for its potential in basic GUI scene understanding. All training and evaluation processes are conducted on 8 NVIDIA A100-80G GPUs. For SFT training, we follow previous studies and utilize LLaMA-Factory framework (Zheng et al., 2024b). We set the global batch size to 8 and learning rate to $1e-5$. During inference, we utilize the same prompt with unified tool-calling method across all experiments to ensure a fair comparison.

Table 7: Ablation results with different training approaches.

Setting	Time	AC-High		GUI-Odessey	
		Type	SR	Type	SR
Zero-Shot	-	68.7	47.1	55.6	34.4
+ SFT	1.3h	**73.4**	53.2	56.3	36.4
+ RL	9.4h	71.9	56.7	61.8	39.1
+ PAL-UI	2.4h	71.3	**57.8**	**65.1**	**41.7**

C Supplementary Results

Ablation of Different Training Strategies We evaluate the impact of different training strategies on model training. Specifically, we evaluate three training approaches: SFT, RL, and our proposed PAL-UI. To ensure a fair comparison, we use the same previously collected 8.6K instruction dataset for training. For SFT, we directly used the original seed dataset without incorporating the PAL module. For RL, we employ the same seed dataset and adhere to the reward design outlined in GUI-R1 (Luo et al., 2025) for training. We set the temperature to 1.0 and sample 8 responses for rollout. All models are trained for one epoch. The results are presented in Table 7. As can be observed, RL significantly outperforms SFT, demonstrating its superior generalization capability, which aligns with findings from prior works. PAL-UI surpasses both SFT and RL, indicating that PAL-UI achieves the best performance using same data. In terms of training time, RL demands substantially more time than SFT. In comparison, although PAL-UI introduces a slight increase in context length due to the summary agent and active look-back mechanism, resulting in marginally longer training time than SFT, it achieves markedly better performance, thereby substantiating its superiority.

Analysis of Retrieval Behavior. We further analyze how the agent employs the retrieve tool during planning. Concretely, we sample 100 agent responses from the AndroidControl test set in which

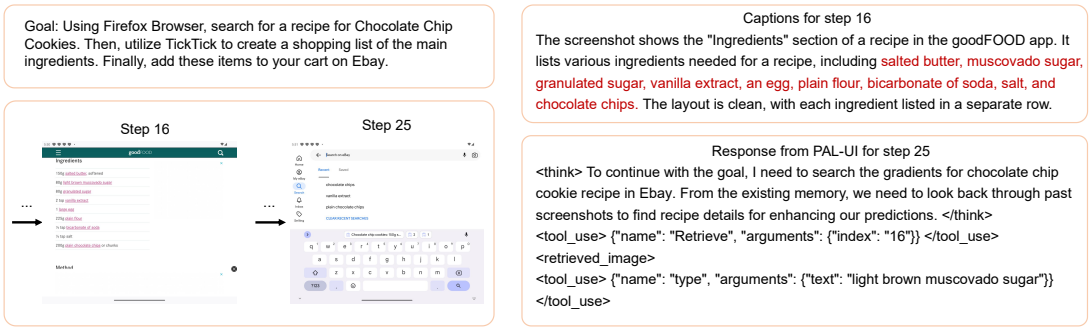

Figure 2: An illustrative example of active look-back mechanism.

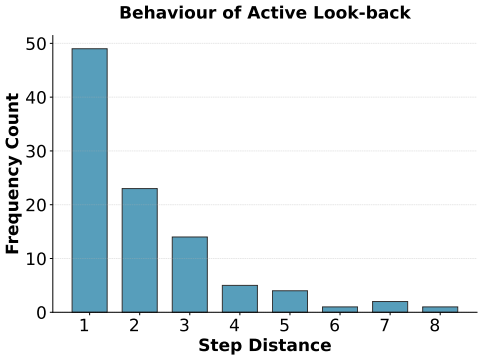

Figure 3: The distribution of distance for retrieval steps.

the retrieval tool is invoked, and record the distance between the retrieved step and the current step (Figure 3). The results show that most retrievals occur within five steps of the current step, with the maximum distance extending to eight steps. This tendency likely reflects the relative simplicity of current navigation benchmarks: tasks usually involve short, localized interactions (e.g., simple GUI manipulations) that demand only limited long-range memory. We believe that in more complex or semantically demanding GUI environments, where long-distance dependencies play a larger role, the benefits of active look-back may become even more pronounced.

Case Study. To quantitatively analyze our proposed PAL-UI agent, We present a case study about the active look-back mechanism in Figure 2. As we can see, although the proposed observation-level captioner can preserve effective information via screenshot captions, it's sometimes unable to fully capture all details, which might be important in subsequent steps. In this case, the agent is required to determine the input for the type action on eBay app (to purchase the corresponding ingredient) based on the observation at step 16. However, although the captions successfully record the type of ingredient, certain detailed information is lost (*e.g.* the quantity of ingredients and the specific subtype of ingredients). Through our proposed PAL module, the agent is able to revisit the observation screenshot from this step and make an accurate decision (typing "*light brown muscovado sugar*" instead of merely "*muscovado sugar*").