Dynamic Planning for LLM-based Graphical User Interface Automation

Anonymous ACL submission

Abstract

The advent of large language models (LLMs) has spurred considerable interest in advancing autonomous LLMs-based agents, particularly in intriguing applications within smartphone graphical user interfaces (GUIs). When presented with a task goal, these agents typically emulate human actions within a GUI environment until the task is completed. However, a key challenge lies in devising effective plans to guide action prediction in GUI tasks, though planning have been widely recognized as effective for decomposing complex tasks into a series of steps. Specifically, given the dynamic nature of environmental GUIs following action execution, it is crucial to dynamically adapt plans based on environmental feedback and action history. We show that the widely-used ReAct approach fails due to the excessively long historical dialogues. To address this challenge, we propose a novel approach called Dynamic Planning of Thoughts (D-PoT) for LLM-based GUI agents. D-PoT involves the dynamic adjustment of planning based on the environmental feedback and execution history. Experimental results reveal that the proposed D-PoT significantly surpassed the strong GPT-4V baseline by +12.7% (34.66% → 47.36%) in accuracy. The analysis highlights the generality of dynamic planning in different backbone LLMs, as well as the benefits in mitigating hallucinations and adapting to unseen tasks. Code is available at `Anonymous`.

1 Introduction

Building autonomous agents capable of assisting humans in addressing real-world challenges has long been a central pursuit of artificial intelligence research (Searle, 1972; Wooldridge and Jennings, 1995; Maes, 1994). Recently, there has been a surge in exploration within the realm of autonomous agents, fueled largely by the emergence of large language models (LLMs) (Chowdhery et al., 2023; Wei et al., 2022). One prevalent application scenario involves automating graphical user interfaces (GUIs) on smartphones, where LLMs are tasked with perceiving smartphone GUIs and sequentially predicting action commands until the task is completed (Rawles et al., 2023; Yang et al., 2023b).

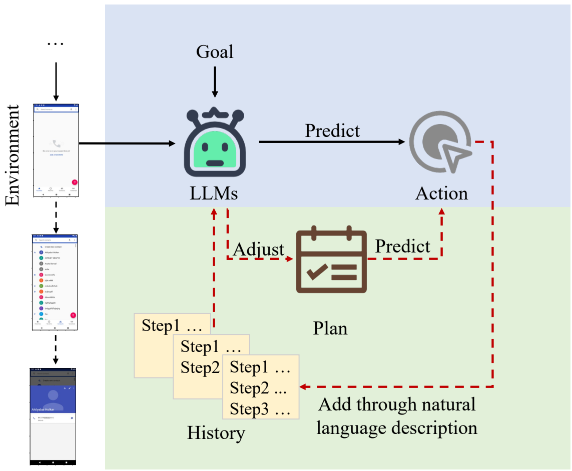


Figure 1: The proposed dynamic planning method incorporates the execution history to adjust the plan to predict the action and subsequently supplements the execution history with the predicted action.

While previous studies have made significant strides by enhancing environment perception through fine-grained GUI grounding (Zhang and Zhang, 2023; Hong et al., 2023; Yan et al., 2023; Yang et al., 2023b; Cheng et al., 2024; You et al., 2024), there has been limited focus on the planning capabilities of GUI agents. Evidence suggests that decomposing a complex task into a series of plans is effective in eliciting the ability of LLMs within agent systems (Zhu et al., 2024; Huang et al., 2024; Song et al., 2023). Additionally, given that the environment evolves based on action predictions, it is crucial to dynamically adapt plans based on environmental feedback and execution history.

Existing LLMs-based GUI agents typically takes actions directly prior planning or adjustment of plans based on environmental feedback (e.g., new GUI screenshot) and execution history (e.g., previous steps described in natural language). For instance, as depicted in Figure 1, the static[1] methods (black data stream) directly predicts actions directly based on the screenshot and goal. Those approaches struggles to handle complex real-world scenarios, where users often adjust subsequent actions based on past steps. We will show that the widely-used ReAct approach (Yao et al., 2022) fails due to excessively long historical dialogues, revealing its inadequacy in handling complex real-world scenarios (Section 3).

To address the challenge above, we propose a novel method called Dynamic Planning of Thoughts (D-PoT) method to enable the LLM-based agent to formulate effective plans based on environment feedback and execution history during task execution (with dashed lines in Figure 1). Concretely, D-PoT dynamically adjusts its plans by incorporating new screenshots and execution history throughout the goal attainment process. Moreover, our proposed method allows for continuous refinement of the current plan, ensuring persistent optimization until the desired goal is achieved. Experimental results demonstrate that our planning mechanism substantially improves the task performance. Additionally, analysis highlights the efficacy of dynamic planning in mitigating hallucinations and adapting to unseen tasks.

Our key contributions are as follows:

(i) D-PoT dynamically formulates plans and selects steps for action prediction based on the new screenshots and execution history, thereby advancing the LLMs-based agent.

(ii) D-PoT achieves a notable improvement in accuracy scores of +12.7% (34.66% → 47.36%) compared with the strong GPT-4V baseline.

(iii) Analysis highlights the efficacy of dynamic planning in not only enhancing action prediction accuracy but also in in mitigating hallucinations and adapting to previously unfamiliar tasks.

2 Related Work

Our work is related to LLMs-based GUI agents. This section will first review the recent progress of the work on building the GUI agents and then discuss the planning mechanism of the agents.

2.1 LLMs-based GUI Agent

LLMs have spurred considerable interest in the realm of language agents. Notable examples include AutoGPT (Yang et al., 2023a), HuggingGPT (Shen et al., 2023), and MetaGPT (Xi et al., 2023), all of which explored the integration of LLMs as the core of agents.

This work focuses on ultilizing LLMs as intelligent assistants for smartphones. These assistants are crafted to assist people in accomplishing their daily tasks and meeting life's requirements, especially enhancing accessibility for individuals with disabilities. Notably, the advent of multi-modal LLMs such as GPT-4V, showcasing robust image understanding capabilities (Yang et al., 2023c), has prompted previous research to predominantly concentrate on comprehending GUI interactions. For instance, MM-Navigator delved into leveraging optical character recognition (OCR) parsing to enhance GPT-4V's GUI comprehension (Yan et al., 2023), while AppAgent reinforced the understanding of Application GUI elements by introducing the roles of distinct GUI (Yang et al., 2023b). In addition to these, CogAgent fine-tuned the agent's understanding of GUI to enhance performance (Hong et al., 2023).

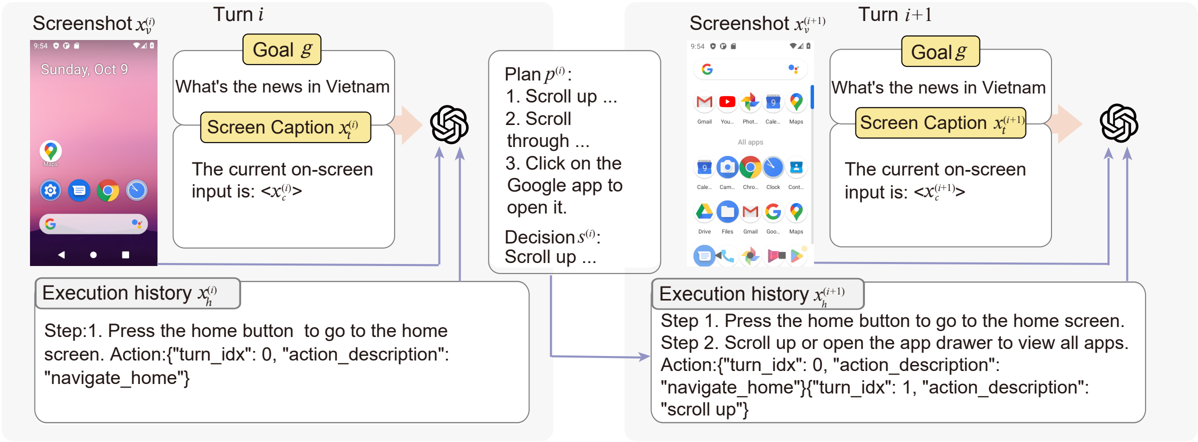
In contrast to the prior research that concentrates on multimodal perception, our work focuses on the planning mechanism to enhance the LLMs proficiency in planning and effectively tackle multi-step tasks on smartphones.

2.2 Planning Mechanisms for LLMs

LLMs have shown considerable potential in constructing agents with strong capabilities in following instructions and maintaining coherent chains of thought (CoT) via solving complex problems (Wei et al., 2022; Kojima et al., 2022; Zhang et al., 2022). Notably, the CoT prompting technique has enabled LLMs to engage in effective step-by-step problem-solving process (Huang and Chang, 2023; Yao et al., 2024; Wang et al., 2022; Chen et al., 2022). To address more complex problems, divide-and-conquer prompting strategies have been proposed, e.g., dividing problems into manageable steps (Zhou et al., 2022; Lee and Kim, 2023) or sequential solutions (Wang et al., 2023).

The research above mainly focuses on enhancing the reasoning abilities of LLMs. However, the ReAct (Yao et al., 2022) has inspired researchers to explore more suitable ways for LLMs to complete

[1] The methods are static due to be not aware of historical information during task execution.


Turn i
Screenshot $x_v^{(i)}$
Sunday, Oct 9
Goal g
What's the news in Vietnam
Screen Caption $x_t^{(i)}$
Execution history $x_h^{(i)}$
Turn i+1
Screenshot $x_v^{(i+1)}$
Goal g
What's the news in Vietnam
Screen Caption $x_t^{(i+1)}$
Execution history $x_h^{(i+1)}$

The current on-screen input is: <$x_c^{(i)}$>

Plan $p^{(i)}$:
1. Scroll up ...
2. Scroll through ...
3. Click on the Google app to open it.

Decision $s^{(i)}$: Scroll up ...

Step:1. Press the home button to go to the home screen. Action:{"turn_idx": 0, "action_description": "navigate_home"}

The current on-screen input is: <$x_c^{(i+1)}$>

Step 1. Press the home button to go to the home screen.
Step 2. Scroll up or open the app drawer to view all apps.
Action:{"turn_idx": 0, "action_description": "navigate_home"}{"turn_idx": 1, "action_description": "scroll up"}

Figure 2: Overview of D-PoT. In turn, i, the D-PoT makes a plan based on visual input and textual input, predicts the action to be performed, and then updates the execution history, and then proceeds to the next turn $i + 1$.

agentic tasks by leveraging their reasoning abilities. This approach involves LLMs first observing and reasoning before taking action, such as utilizing external tools to identify and rectify errors (Gou et al., 2023; Shinn et al., 2023), or planning before executing (Wang et al., 2023; Hao et al., 2023).

3 Investigating the Necessity of Dynamic Planning in GUI Agent

3.1 Challenge of GUI Automation

GUI automation is a long-episode task where the LLM first receives a goal and an initial screen. To achieve this goal, it must navigate through multiple screens continuously until the task is complete. This presents a challenge for the LLM, requiring it to understand the current progress of the task and the execution history to avoid performing redundant actions in similar environments.

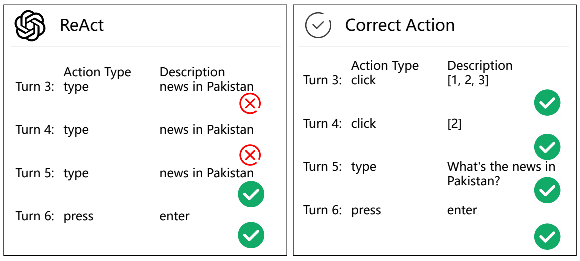
3.2 ReAct Fails Due to the Excessively Long Historical Dialogues

ReAct is a widely used method in the LLM-based agent (Yao et al., 2022). It encourages the LLM to think before taking action when encountering a new environment. Each round of input includes all previous thoughts and actions. We experiment with using ReAct for the GUI Automation task. We sampled 20 tasks from a general dataset and conducted experiments using GPT-4V. The experimental setup is detailed in Section 5.3.

Based on the Table 1, we observe that accuracy does not always improve with the increased length of input historical dialogues. The best performance is achieved when the history length is 2. This is likely because, in the GUI Automation task, the input length is substantial, with each round containing at least 2000 tokens. Consequently, the performance does not significantly improve with an increase in the length of historical dialogues.

History Length	Accuracy	Inference Cost (tokens/ep)	Inference Speed(s/ep)
0	32.58	22265.3	55.1
1	42.42	40961.1	63.3
2	45.45	57176.6	77.4
4	44.70	80300.2	91.1
∞	43.94	101151.1	119.7
D-PoT	45.45	25418.7	50.3

Table 1: History Length refers to the number of dialogue rounds inputted into ReAct, where "∞" indicates that all historical dialogues are included. Accuracy calculation details can be found in Section 5.1. Inference cost represents the average tokens used per episode for API call, and inference speed indicates the number of seconds required to complete each task.



ReAct

	Action Type	Description	
Turn 3:	type	news in Pakistan	✗
Turn 4:	type	news in Pakistan	✗
Turn 5:	type	news in Pakistan	✓
Turn 6:	press	enter	✓

Correct Action

	Action Type	Description	
Turn 3:	click	[1, 2, 3]	✓
Turn 4:	click	[2]	✓
Turn 5:	type	What's the news in Pakistan?	✓
Turn 6:	press	enter	✓

Figure 3: ReAct is misled by incorrect decisions.

Additionally, ReAct can be misled by incorrect decisions made in the last turn. We speculate that the reason is because ReAct focuses more on the most recent interaction rather than the overall task completion progress. For instance, as depicted in Figure 3. In the third turn, a "type" action is performed, but it is incorrect. This error persist until the fifth turn when it is finally resolved.

When facing frequent interactions or complex task inputs, reducing inference cost and speeding up inference become critical challenges. The key issue lies in how to effectively input execution history to make dynamic plans and guide LLM-based agents in understanding task progress.

4 Method

In light of the above experimental results and analysis, we propose Dynamic Planning of Thoughts (D-PoT) to mitigate the challenge. On a high level, D-PoT consists of three stages:

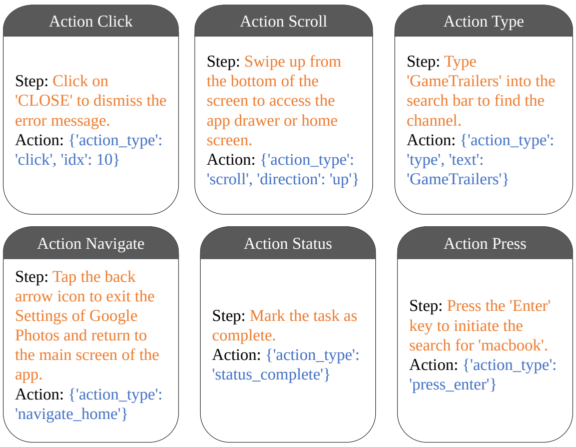
(1) planning initialization: the LLMs initiate the planning process by generating an overall plan, considering the ultimate goal, current visual input, and prior execution history. The plan helps the LLM grasp the progress of the current task. Once the plan is formulated, the LLMs will select the most plausible step for execution. (2) dynamic planning adjustment: the executed step is appended to the execution history. This updated history list then carefully shapes subsequent planning cycles. In this way, the agent can access all historical information rather than just focusing on the most recent turns. Moreover, these historical details occupy only a small number of tokens, this reduces inference cost, speeds up inference, and improves decision efficiency in subsequent turns. The framework of D-PoT is shown in Figure 2.

4.1 Planning Initialization

In pursuit of the task goal g, the LLMs engage in k turns of interactions until task completion. Specifically, at each turn i ($i = 1, \dots, k$), the LLMs f processes the visual input $x_v^{(i)}$ (i.e., the current screenshot) and the textual input $x_t^{(i)}$. It then generates the plan p_i and identifies the optimal step $s^i \in p^i$ to execute:

$$(p^{(i)}, s^{(i)}) = f(x_v^{(i)}, x_t^{(i)}), \tag{1}$$

where the textual input $x_t^{(i)}$ consists of the task goal g, screen caption $x_c^{(i)}$, and execution history $x_h^{(i)}$.



Action Click

Step: Click on 'CLOSE' to dismiss the error message.
Action: {'action_type': 'click', 'idx': 10}

Action Scroll

Step: Swipe up from the bottom of the screen to access the app drawer or home screen.
Action: {'action_type': 'scroll', 'direction': 'up'}

Action Type

Step: Type 'GameTrailers' into the search bar to find the channel.
Action: {'action_type': 'type', 'text': 'GameTrailers'}

Action Navigate

Step: Tap the back arrow icon to exit the Settings of Google Photos and return to the main screen of the app.
Action: {'action_type': 'navigate_home'}

Action Status

Step: Mark the task as complete.
Action: {'action_type': 'status_complete'}

Action Press

Step: Press the 'Enter' key to initiate the search for 'macbook'.
Action: {'action_type': 'press_enter'}

Figure 4: Examples of six types of available actions.

The textual input is further wrapped with prompts (Appendix A.1) before feeding the LLMs along with the visual input. Concretely, we articulate our task goal at the text's outset by prompting "*Your ultimate goal is: <g>*". Subsequently, we append the screen caption results under the heading "*The current on-screen input is: <$x_c^{(i)}$>*". Then, we include execution history, structured as "*Here are previous actions: <$x_h^{(i)}$>*".

After feeding the inputs, we request the LLMs to generate a plan $p^{(i)} = [p_1^{(i)}, p_2^{(i)}, \dots]$, which consists of a sequence of steps to achieve the ultimate goal. Within those steps, the LLMs is also required to identify the optimal step $s^{(i)} \in p^{(i)}$.

Action Type	Action Description
Click	Idx
Scroll	Direction (up, down, left and right)
Typ	Text
Navigate	Home / Back
Status	Complete
Press	Enter

Table 2: Six types of available actions.

In practice, $s^{(i)}$ is confined to a finite set of available actions in the GUI automation task and will be transformed into the JSON format for execution. Following Rawles et al. (2023), we utilize six distinct types of actions as presented in Table 2. There is no overlap between the different actions. Examples are provided in Figure 4.

4.2 Dynamic Planning Adjustment

After the execution of $s^{(i)}$, the LLMs becomes anchored in the subsequent interaction turn with an updated visual input $x_v^{(i+1)}$ (e.g., a new screenshot). Simultaneously, we refine the execution history $x_h^{(i+1)}$ by concatenating $x_h^{(i)}$ and $s^{(i)}$:

$$x_h^{(i+1)} = \textsc{Concat}(x_h^{(i)}, s^{(i)}), \tag{2}$$

where CONCAT denotes the concatenation operation between strings.

Model	Overall	General	GoogleApps	Install	Single	WebShopping
PaLM-2 ZS (Rawles et al., 2023)	30.9	-	-	-	-	-
ChatGPT 5-shot (Zhang and Zhang, 2023)	7.72	5.93	10.47	4.38	9.39	8.42
GPT-4V ZS	34.66	29.69	35.75	43.50	32.95	31.42
GPT-4V 4FS	39.71	34.90	34.97	**50.10**	41.62	36.96
GPT-4V ReAct	42.73	36.20	42.49	46.60	49.13	**39.22**
D-PoT	46.47	40.10	**49.74**	47.18	**58.96**	36.34
D-PoT w/ reference	**47.36**	**42.19**	49.48	49.61	**58.96**	36.55

Table 3: Main results (%). Segment 1: fine-tuned Llama 2 baseline; Segment 2: in-context learning LLM baselines, "ZS" stands for "zero-shot" and "5-shot" stands for using 5-shot in-content learning (Section 5.2); Segment 3: GPT-4V as agent model, "D-PoT" represents our proposed framework. "D-PoT w/ reference" represents seeking similar task goals during the planning initialization stage as a reference (Detailed discussion provided in Section 5.5). The best result is reported in boldface.

Consequently, the execution history is organized with consecutive elements in the format of "*step <turn id>: <action>*". This updated execution history $x_h^{(i+1)}$ is subsequently employed according to the planning initialization process outlined in Section 4.1 for turn $(i+1)$ until the task reaches completion. The task is considered complete when $i = k$ or the LLMs predicts the "Status" action type with the "Complete" action description.

5 Experiments

5.1 Dataset and Setup

We utilize the popular AITW dataset (Rawles et al., 2023) for evaluating our D-PoT. More details about the AITW dataset are in Appendix A.2. We sampled 60 episodes from each subset for analysis to get more convincing results, and incorporated screen caption results into textual input, detecting GUI icons using OCR and IconNet (Sunkara et al., 2022). Each GUI icon is associated with a bounding box and OCR-detected text. In line with prior research (Zhang and Zhang, 2023; Yan et al., 2023), our primary evaluation metric is the screen-wise action matching score, computed as the ratio of correct actions to the episode length. More details are shown in Appendix A.3.

5.2 Baseline

To verify the proposed D-PoT, we used several recent agent methods as our comparison systems:

- **PaLM-2 ZS** (Rawles et al., 2023): This setting evaluates the zero-shot performance of PaLM-2 by providing a textual description of the screen and prompting it to predict an action from the supported actions in AITW.
- **ChatGPT 5-shot** (Zhang and Zhang, 2023): ChatGPT's performance is assessed with a 5-shot prompt format similar to PaLM-2. The experiments are conducted using the ChatGPT API.
- **GPT-4V ZS**: Zero-shot prompting with GPT-4V. The model is presented with a screenshot image and a textual description of the screen, tasked with predicting an action from the available actions.
- **GPT-4V 4FS**: Few-shot prompting with 4 examples. The model is presented with a screenshot image and a textual description of the screen, tasked with predicting an action from the available actions.
- **GPT-4V ReAct**: It represents that the interaction method of LLM is ReAct, which includes a history input of 4 turns. The inputs of every turns are screenshots, goals, and screen captions.

5.3 Implementation Details

We use the GPT-4V (Achiam et al., 2023) interface provided by OpenAI as the backbone of our agent. The GPT-4V model we use is "gpt-4-vision-preview". We set the "max_tokens" as 300 and the "temperature" as 0. We also fine-tune public large models, i.e., Llama2-7B (Touvron et al., 2023) and LLaVa-7B (Liu et al., 2024), to verify the general effectiveness of our approach. For the finetuning experimental setup, training epochs are set as 3, without eval set between epochs. The maximum length of the input sequence is 2560 tokens. Text input includes the goal, screen descriptions in HTML syntax, and execution history. For inputs with a "Plan" experimental group, the step is spliced at the end of the input. The fine-tuning results of these open source LLMs we put in Section 5.8.

5.4 Main Results

Table 3 presents the main results of the test sets for AITW. Based on the results, we have the following findings:

(i) The proposed D-PoT achieves substantial performance gains on all comparison methods in terms of Overall scores. Particularly, D-PoT exhibits +12.7% (34.66% → 47.36%) improvement on the strong baseline GPT-4V ZS. This presents the effectiveness of our D-PoT, that is, both environmental feedback and action history are beneficial for the GUI task.

(ii) We observe that our D-PoT gains improvement on the comparison methods (PaLM-2 ZS, ChatGPT 5-shot, Fine-tuned Llama-2, GPT-4V ZS, GPT-4V 4FS and GPT-4V ReAct) in almost all five categories (General, GoogleApps, Install, Single, and WebShopping). This indicates that our D-PoT is generalized to different GUI tasks.

Accuracy	**w/ GPT-4V**	**w/ Human**
Click	17.83	27.39
Scroll	0.00	1.27
Typ	2.55	9.55
Navigate Home	0.64	3.82
Navigate Back	0.00	0.00
Press	0.00	2.55
Complete	2.55	7.64
Total	23.57	**52.23**

Table 4: Comparison of GPT-4V generated planning and human-annotated planning in the Install dataset (%). The best average result is reported in boldface.

(iii) We observed that improvement of D-PoT on certain tasks, such as the Install and WebShopping datasets, is not significant. We think that this slight improvement may be attributed to the generated low-quality plans. To verify it, we select 20 episodes from the Install dataset and label them with corresponding plans (e.g., Click, Scroll, Typ, Navigate Home, Navigate Back, Press, and Complete, see Table 2). These human-annotated plans are input into LLMs instead of plans generated by GPT-4V and are prompted to select steps and predict actions. Table 4 shows a significant improvement for these 20 episodes, with the Total accuracy scores increasing from 23.57%→52.23%. The high-quality plans are beneficial for the GUI task, which means that one of the slight decrease reasons is attributable to low-quality planning generated by GPT-4V, likely failing to stimulate this ability to generate high-quality plans during supervision fine-tuning.

5.5 Alleviating Planning Hallucinations and Errors

To mitigate planning hallucinations and errors, we additionally seek similar task goals during the planning initialization stage as a reference. Initially, we encode the goal of each episode using sentence-transformer and identify the goals of the two most similar episodes from the remaining testsets (Reimers and Gurevych, 2019). We then combine the predicted actions of these two episodes as a reference for the plan. Additionally, we utilize InstructBlip to extract captions from the initial screen of each episode task, indicating starting point of the task (Li et al., 2023). These inputs are incorporated into the prompt for planning initialization, as outlined in the Appendix A.1.

The experimental results are shown in Table 3. We observe that when all predicted actions from similar tasks are as a reference, the proposed D-PoT with reference gains the improvement of 0.89% accuracies on the D-PoT in terms of Overall scores. Specifically, on the General and Install datasets, incorporating references result in accuracy improvements of 2.09% and 2.43%, respectively. This indicates that **D-PoT is effective at alleviating planning hallucinations and errors.**

5.6 Ablation Study of Varied Planning

To study the impact of dynamic planning, we randomly sampled 20 episodes from each data subset, with a total of 100 episodes as the dataset for the ablation experiment, and built several baselines.

- **No Planning (NP)**: Its inputs are screenshots, goals, and screen captions. We ask the LLMs to predict actions directly based on these inputs without specifying a plan.
- **ReAct**: It represents that the interaction method of LLM is ReAct, which includes a history input of 4 turns. The inputs of every turns are screenshots, goals, and screen captions.
- **Static Planning (SP)**: It represents the utilization of planning statically. We will ask LLMs to generate a plan at the beginning of the episode and add the plan to the prompt during the whole episode.
- **Dynamic Planning (DP)**: It represents the utilization of planning, excluding selecting steps and updating execution history. The inputs of DP are screenshots, goals, and screen captions.

Methods	**Static Planning**	**Dynamic Planning**	**Updating History**	**Overall**	**General**	**GoogleApps**	**Install**	**Single**	**WebShopping**
NP	✗	✗	✗	32.45	28.03	40.32	33.79	26.98	33.13
ReAct	✗	✗	✓	38.97	35.61	46.77	37.24	39.68	**35.54**
SP	✓	✗	✗	28.58	21.97	38.71	20.69	42.86	18.67
DP	✗	✓	✗	42.01	31.06	50.81	40.69	**57.14**	30.72
D-PoT	✗	✓	✓	**45.75**	**45.45**	**52.42**	**44.14**	52.38	34.34

Table 5: The ablation studies on planning mechanisms. Static Planning: Create a plan based on the provided screenshot and goal at the outset of the episode. And utilize this plan consistently throughout the episode to direct LLMs in predicting actions; Dynamic Planning: Continuously adapt and formulate plans during task execution, considering all available input information; Updating History: Incorporate the steps into the execution history and utilize them in the planning process. Each experiment's execution or omission of a particular process is denoted by ✓ (if performed) or ✗ (if not performed). The best average result is in boldface.

When receiving a new screenshot, we ask LLMs to generate a plan and then take action. Table 5 presents the detailed results of the test set for the AITW dataset.

First, the accuracy scores of DP and D-PoT are higher than those of NP, SP and ReAct. This means that dynamic planning is significantly superior to static planning in the graphical user interface automation task. We think that this superiority contributes to two potential or possible factors: 1) This planning greatly stimulates the understanding ability of the LLMs-based agent for the graphical user interface automation task; 2) Throughout task execution, the historical information extracted by steps helps the LLMs-based agent flexibly update its plan for the environment changes and unseen scenarios, especially compared to ReAct, reduces inference cost and greatly improves the performance.

Second, the comparison among NP, ReAct, DP, and D-PoT reveals that integrating planning leads to substantial enhancements preceding the predicted action. We think that this effect arises as the generative planning may prompt LLMs to engage in GUI automation, thereby enhancing their comprehension of the intended goal. This demonstrates that the proposed D-PoT obtains notable enhancement via plan integration before action prediction.

Third, we observe that D-PoT outperformed DP in terms of Overall scores. This indicates that incorporating execution history into LLMs enhances GUI automation through dynamic planning. In other words, historical information is beneficial for LLMs in GUI automation, especially dynamic planning based on historical steps. Moreover, the accuracy scores of D-PoT are inferior to those of DP on the Single datasets. In addition to the generated low-quality plans in Table 4, part of the reasons may be that the short episode length reduces the reliance on historical information for the Single dataset.

5.7 Exploring the Proportion and Correct Rate of Predicted Actions

To conduct a detailed analysis of the impact of dynamic planning, we dive into the correct rate and the proportion of predicted actions. Specifically, we combine five categories for an overarching analysis, more details of the correct rate of predicted actions are in Appendix A.4. Table 6 presents the overall predicted ratio and the predicted accuracy ratio for different actions. Due to the potential occurrence of unpredictable actions in LLMs, it's possible that the sum of predicted probabilities may not equal 1.

Our observations based on these statistics reveal the following two findings:

(i) **Dynamic planning empowers LLMs to enhance their task management capabilities.** Within the DP and D-PoT experimental groups, we observed a noteworthy increase in both the prediction proportion and accuracy rate of "Complete" actions. This suggests that dynamic planning enhances the comprehension of LLMs-based agent in the current task.

(ii) **Dynamic planning reduces the invalid predicted click action.** We observed a significant decrease in the prediction ratio for "Click" with the introduction of dynamic planning, but the prediction accuracy rate is not affected. Existing work indicates that GPT-4V is more likely to predict the "Click" action (Yan et al., 2023). However, the proposed D-PoT minimizes invalid and erroneous click actions, showcasing a better comprehension of the implementation progress of

Model	Click	Scroll	Typ	Home	Back	Press	Complete
NP	78.57 / 26.67	4.29 / 1.27	4.29 / 2.38	4.13 / 1.11	1.59 / 0	0.48 / 0.16	1.43 / 1.43
ReAct	70.79 / 27.14	4.92 / 1.59	**7.94 / 4.92**	3.02 / 1.43	**1.90 / 0.16**	2.54 / 0.32	3.33 / 3.02
SP	71.11 / 19.84	8.41 / 1.43	1.43 / 0.63	8.73 / 0.48	2.06 / 0	**2.7 / 0.48**	3.81 / 3.33
DP	71.90 / 25.08	3.49 / 1.11	3.17 / 2.54	**6.98 / 2.06**	0.63 / 0	1.59 / 0.32	11.75 / 8.57
D-PoT	68.45 / 26.83	**7.01 / 2.7**	3.51 / 3.02	1.03 / 0.79	0.41 / 0	0.62 / 0	**18.14 / 9.52**
D-PoT w/ reference	**64.29 / 29.37**	**10.48 / 2.7**	5.87 / 3.33	2.38 / 0.95	0.95 / 0	3.49 / 0.16	11.9 / 8.10

Table 6: The predicted ratio and the predicted accuracy ratio for different actions(%). the number on the left of "/" is the predicted ratio, and the number on the right of "/" is the predicted accuracy ratio. The best result is in boldface.

the current plan.

5.8 Adaptation to Unfamiliar Tasks

As new applications continually emerge, their interfaces often pose unfamiliarity to agents. Despite the diversity of GUI tasks, there exists a semblance of similarity in screen navigation logic. Even when the interface is unknown, certain screen transition patterns remain consistent. Consequently, the proposed D-PoT utilizes dynamic planning to capture environmental changes and historical steps, which will be beneficial for adaptation to unfamiliar tasks. To validate this, we select two base models, Llama2-7B and LLaVa-7B, for fine-tuning. Llama2-7B serves to verify the effectiveness of the D-PoT method on plain text, while LLaVa-7B serves to verify its effectiveness on multimodal data. We randomly choose the GoogleApps dataset as the training set and the remaining datasets as the test set. The five datasets contain various task categories. We utilize both the D-PoT instruction from our method and the action instruction from AITW for fine-tuning.

Methods	General	Install	Single	WebShopping
Llama2-7B*(in-domain)*				
w/ all data	28.56	35.18	27.35	19.92
Llama2-7B*(out-domain)*				
NP Baseline	13.08	17.12	3.87	8.71
Plan by GPT-4V	**24.67**	**23.46**	**39.48**	**19.48**
Plan by Itself	17.81	17.58	15.87	12.46
LLaVa-7B*(out-domain)*				
NP Baseline	17.81	17.98	1.66	10.91
Plan by GPT-4V	27.19	26.77	44.46	20.61
Plan by Itself	**30.73**	**29.39**	**45.94**	**21.67**

Table 7: Finetuning results of Llama2-7B and LLaVa-7B. Segment 1: "w/ all data" stands for the model is fine-tuned with 1% randomly sampled training data to help adapt to this task (Zhang and Zhang, 2023). Segments 2 & 3: The training set is 180 episodes in the GoogleApps, and the test set is 180 episodes in other datasets. "*GPT-4V*" stands for planning is made by GPT-4V. "*itself*" stands for planning made by the finetuned model itself. The best average result is in boldface.

The results in Table 7 indicate that LLMs fine-tuned with D-PoT data exhibit significant improvements in other tasks and demonstrate robust adaptability to unknown tasks compared to direct fine-tuning with action instructions. Even on the Llama2-7B model, the experimental results of fine-tuning using only a small amount of D-PoT data are comparable to those of fine-tuning using the full AITW dataset. This verifies the effectiveness of D-PoT for out-of-domain tasks.

Additionally, in the experiment with LLaVa-7B, we observed that allowing LLaVa-7B to learn dynamic planning rather than following the planned prediction actions formulated by GPT-4V, yielded higher accuracy scores. This indicates that our fine-tuned LLaVa-7B model learned the plan from the GoogleApp dataset and is capable of planning effectively for tasks in other domains. This further supports the notion that **D-PoT can adapt LLMs to unfamiliar tasks.**

5.9 Error Analysis.

To dive into the mistakes of GPT-4V in dynamic planning and facilitate future studies, we categorize three common errors that lead to discrepancies between the predictions of GPT-4V and human-annotated predictions. More details are presented in Appendix B.

6 Conclusion

This study introduces a prompting approach called D-PoT, designed to facilitate interactions in a multimodal environment. D-PoT encourages LLMs to dynamically update planning based on feedback from the environment and execution history. Through the application of D-PoT, we demonstrate that the D-PoT surpasses the widely adopted GPT-4V baseline on the AITW benchmark dataset. Meanwhile, our findings indicate that the D-PoT excels in adapting to unfamiliar tasks, and can predict different actions more correctly.

7 Limitation

This study utilizes the powerful zero-shot capability of LLMs to forecast smartphone actions by incorporating prompt constraints. Our focus lies predominantly on exploring the efficacy of dynamic planning in enhancing action prediction within a given scenario during an episode. In terms of social impact, employing LLM-based agents on mobile phones holds promise for assisting individuals with disabilities. It's worth noting that applying LLMs-based agents on smartphones presents certain constraints. While we find promise in the observed improvement in predicted action accuracy over longer episodes through dynamic planning, practical implementation remains a distant goal. Many challenges stem from the limited knowledge of the mobile phone domain within LLMs, highlighting inherent imperfections. These issues warrant further investigation in future research endeavors.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W Cohen. 2022. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *arXiv preprint arXiv:2211.12588*.

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*.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. 2023. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113.

Zhibin Gou, Zhihong Shao, Yeyun Gong, Yelong Shen, Yujiu Yang, Nan Duan, and Weizhu Chen. 2023. Critic: Large language models can self-correct with tool-interactive critiquing. *arXiv preprint arXiv:2305.11738*.

Shibo Hao, Yi Gu, Haodi Ma, Joshua Hong, Zhen Wang, Daisy Wang, and Zhiting Hu. 2023. Reasoning with language model is planning with world model. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 8154–8173, Singapore. Association for Computational Linguistics.

Wenyi Hong, Weihan Wang, Qingsong Lv, Jiazheng Xu, Wenmeng Yu, Junhui Ji, Yan Wang, Zihan Wang, Yuxiao Dong, Ming Ding, et al. 2023. Cogagent: A visual language model for gui agents. *arXiv preprint arXiv:2312.08914*.

Jie Huang and Kevin Chen-Chuan Chang. 2023. Towards reasoning in large language models: A survey. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 1049–1065, Toronto, Canada. Association for Computational Linguistics.

Xu Huang, Weiwen Liu, Xiaolong Chen, Xingmei Wang, Hao Wang, Defu Lian, Yasheng Wang, Ruiming Tang, and Enhong Chen. 2024. Understanding the planning of llm agents: A survey. *arXiv preprint arXiv:2402.02716*.

Takeshi Kojima, Shixiang (Shane) Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. In *Advances in Neural Information Processing Systems*, volume 35, pages 22199–22213.

Soochan Lee and Gunhee Kim. 2023. Recursion of thought: A divide-and-conquer approach to multi-context reasoning with language models. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 623–658, Toronto, Canada. Association for Computational Linguistics.

Dongxu Li, Junnan Li, Hung Le, Guangsen Wang, Silvio Savarese, and Steven C.H. Hoi. 2023. LAVIS: A one-stop library for language-vision intelligence. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, pages 31–41, Toronto, Canada. Association for Computational Linguistics.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. 2024. Visual instruction tuning. *Advances in neural information processing systems*, 36.

Pattie Maes. 1994. Agents that reduce work and information overload. *Communications of the ACM*, 37(7):30–40.

Christopher Rawles, Alice Li, Daniel Rodriguez, Oriana Riva, and Timothy P Lillicrap. 2023. Androidinthewild: A large-scale dataset for android device control. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

JohnR. Searle. 1972. Speech acts: An essay in the philosophy of language. *Mind,Mind*.

Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. 2023. Hugginggpt: Solving ai tasks with chatgpt and its friends in hugging face. In *Advances in Neural Information Processing Systems*, volume 36, pages 38154–38180. Curran Associates, Inc.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 36, pages 8634–8652. Curran Associates, Inc.

Chan Hee Song, Jiaman Wu, Clayton Washington, Brian M Sadler, Wei-Lun Chao, and Yu Su. 2023. Llm-planner: Few-shot grounded planning for embodied agents with large language models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2998–3009.

Srinivas Sunkara, Maria Wang, Lijuan Liu, Gilles Baechler, Yu-Chung Hsiao, Jindong Chen, Abhanshu Sharma, and James WW Stout. 2022. Towards better semantic understanding of mobile interfaces. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 5636–5650.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Lei Wang, Wanyu Xu, Yihuai Lan, Zhiqiang Hu, Yunshi Lan, Roy Ka-Wei Lee, and Ee-Peng Lim. 2023. Plan-and-solve prompting: Improving zero-shot chain-of-thought reasoning by large language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2609–2634.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2022. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*.

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

Michael Wooldridge and Nicholas R. Jennings. 1995. Intelligent agents: Theory and practice. *The Knowledge Engineering Review*, page 115–152.

Zhiheng Xi, Wenxiang Chen, Xin Guo, Wei He, Yiwen Ding, Boyang Hong, Ming Zhang, Junzhe Wang, Senjie Jin, Enyu Zhou, et al. 2023. The rise and potential of large language model based agents: A survey. *arXiv preprint arXiv:2309.07864*.

An Yan, Zhengyuan Yang, Wanrong Zhu, Kevin Lin, Linjie Li, Jianfeng Wang, Jianwei Yang, Yiwu Zhong, Julian McAuley, Jianfeng Gao, et al. 2023. Gpt-4v in wonderland: Large multimodal models for zero-shot smartphone gui navigation. *arXiv preprint arXiv:2311.07562*.

Hui Yang, Sifu Yue, and Yunzhong He. 2023a. Auto-gpt for online decision making: Benchmarks and additional opinions.

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

Zhengyuan Yang, Linjie Li, Kevin Lin, Jianfeng Wang, Chung-Ching Lin, Zicheng Liu, and Lijuan Wang. 2023c. The dawn of lmms: Preliminary explorations with gpt-4v (ision). *arXiv preprint arXiv:2309.17421*, 9(1):1.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2024. Tree of thoughts: Deliberate problem solving with large language models. *Advances in Neural Information Processing Systems*, 36.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2022. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*.

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

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

Zhuosheng Zhang, Aston Zhang, Mu Li, and Alex Smola. 2022. Automatic chain of thought prompting in large language models. *arXiv preprint arXiv:2210.03493*.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc Le, et al. 2022. Least-to-most prompting enables complex reasoning in large language models. *arXiv preprint arXiv:2205.10625*.

Yuqi Zhu, Shuofei Qiao, Yixin Ou, Shumin Deng, Ningyu Zhang, Shiwei Lyu, Yue Shen, Lei Liang, Jinjie Gu, and Huajun Chen. 2024. Knowagent: Knowledge-augmented planning for llm-based agents. *arXiv preprint arXiv:2403.03101*.

A Example Appendix

A.1 Dynamic planning prompting

We use the following prompt for Planning Initialization.

```
Imagine that you are a robot operating a mobile. Like how humans operate the mobile, you can click on the screen, type some text, go home, go back to the last screen, scroll up, down, left and right, or mark the status as complete. Given a goal and a mobiel screen, you need to make a plan to achieve your goals based on the current screen, and choose the steps that should be achieved on the current screen from the plan you have made. Since achieving this goal is a **continuous process**, you will be given the **previous steps and actions** that have been performed, so please pay attention to this information. There may be multiple ways to achieve your goals, but what you need to do is create the plan that best suits your current situation based on the current screen input.

**Your ultimate goal is: check out phone information.**
The current on-screen input is:
Screen:
<p id=0 class=‘‘text’’ alt=‘‘vvaiipaper,’’>vvaiipaper,</p>
<p id=1 class=‘‘text’’ alt=‘‘sieep,’’>sieep,</p>
<p id=2 class=‘‘text’’ alt=‘‘iolL’’>iolL</p>
<p id=3 class=‘‘text’’ alt=‘‘SIZE’’>SIZE</p>
<p id=4 class=‘‘text’’ alt=‘‘Sound’’>Sound</p>
<img id=5 class=ICON\_VOLUME\_STATE alt=‘‘’’></p>\n <p id=6 class=‘‘text’’ alt=‘‘Volume,’’>Volume,</p>
<p id=7 class=‘‘text’’ alt=‘‘vibration,’’>vibration,</p>
<p id=8 class=‘‘text’’ alt=‘‘Do’’>Do</p>
<p id=9 class=‘‘text’’ alt=‘‘Not’’>Not</p>
<p id=10 class=‘‘text’’ alt=‘‘Disturb’’>Disturb</p>\newline <p id=11 class=‘‘text’’ alt=‘‘Storage’’>Storage</p>
<p id=12 class=‘‘text’’ alt=‘‘used’’>used</p>
<p id=13 class=‘‘text’’ alt=‘‘GB free’’>GB free</p>
<p id=14 class=‘‘text’’ alt=‘‘49\%’’>49\%</p>
<p id=15 class=‘‘text’’ alt=‘‘-32.63’’>-32.63</p>
<p id=16 class=‘‘text’’ alt=‘‘Privacy’’>Privacy</p>
<p id=17 class=‘‘text’’ alt=‘‘Permissions,’’>Permissions,</p>
<p id=18 class=‘‘text’’ alt=‘‘account’’>account</p>
<p id=19 class=‘‘text’’ alt=‘‘personal’’>personal</p>
<p id=20 class=‘‘text’’ alt=‘‘data’’>data</p>
<p id=21 class=‘‘text’’ alt=‘‘activity,’’>activity,</p>
<p id=22 class=‘‘text’’ alt=‘‘Location’’>Location</p>
<img id=23 class=ICON\_LOCATION alt=‘‘’’></p>
<p id=24 class=‘‘text’’ alt=‘‘On’’>On</p>
<p id=25 class=‘‘text’’ alt=‘‘have access’’>have access</p>
<p id=26 class=‘‘text’’ alt=‘‘- 4 apps’’>- 4 apps</p>
<p id=27 class=‘‘text’’ alt=‘‘location’’>location</p>
<p id=28 class=‘‘text’’ alt=‘‘to’’>to</p>
<p id=29 class=‘‘text’’ alt=‘‘Security’’>Security</p>
<p id=30 class=‘‘text’’ alt=‘‘lock, fingerprint’’>lock, fingerprint</p>
<p id=31 class=‘‘text’’ alt=‘‘Screen’’>Screen</p>
Here are previous actions: (format: action \u2192 action description)
Previous Actions:
{’’step\_idx’’: 0, ’’action\_description’’: ’’scroll up’’}
{’’step\_idx’’: 1, ’’action\_description’’: ’’click []’’}
{’’step\_idx’’: 2, ’’action\_description’’: ’’scroll up’’}
And the previous steps:
Previous Steps:
Step 1. Swipe up from the bottom of the screen to access the app drawer.
Step 2. Tap on the ’Settings’ icon to open the settings menu.
Step 3. Scroll up to reveal more settings options.

Please formulate an operational guide for future operations for solving the goal. The guide includes:
1. Plan: A **multi-step future** plan **(start from current screen, DON’T include previous steps)**; steps indexed by numbers.
2. Step: Based on the current screen and Previous Steps, provide the **immediate** step that needs to be taken from the Plan.
"**Output Format:** A JSON dictionary strictly following the format: "{’plan’: ’...<Your Plan Here>’, ’step’: ’...<Your Step Here>’} "If the goal has already been implemented, no more planning is required, Provide {’plan’: ’1. Mark the task as complete’, ’step’: ’Mark the task as complet’}.
**Please do not output any content other than the JSON format.**
```

We use the following prompt for Planning Initialization with references.

```
Imagine that you are a robot operating a mobile. Like how humans operate the mobile, you can click on
     the screen, type some text, go home, go back to the last screen, scroll up, down, left and
    right, or mark the status as complete. Given a goal and a mobiel screen, you need to make a plan
     to achieve your goals based on the current screen, and choose the steps that should be achieved
     on the current screen from the plan you have made. Since achieving this goal is a **continuous
    process**, you will be given the **previous steps and actions** that have been performed, so
    please pay attention to this information. There may be multiple ways to achieve your goals, but
    what you need to do is create the plan that best suits your current situation based on the
    current screen input.
**Your ultimate goal is: What is the price of a 12’ ladder at Home Depot?.**
I also give you two similar examples as a reference, here are their goal, the initial caption of
    mobile screen, and all the execution actions to complete goal:
Goal: What’s the price of the 1000-Watt EGO Power+ Snow Blower?
Caption: The information on the phone screen is a screenshot of the Google Play Store, displaying
    various apps available for download. The screenshot provides a visual representation of the apps
     that can be found on the Google Play Store, allowing users to easily browse and choose from a
    variety of options.
Execution history: {\"step_idx\": 0, \"action_description\": \"click [9]\"}

{\"step_idx\": 1, \"action_description\": \"click [9]\"}

{\"step_idx\": 2, \"action_description\": \"click []\"}

{\"step_idx\": 3, \"action_description\": \"type\"}

{\"step_idx\": 4, \"action_description\": \"press_enter\"}

{\"step_idx\": 5, \"action_description\": \"click [Shopping]\"}

{\"step_idx\": 6, \"action_description\": \"scroll up\"}

{\"step_idx\": 7, \"action_description\": \"click [Official Site - Shop Ego Lb5300]\"}

\\{\"step_idx\": 8, \"action_description\": \"status_complete\"\\}

Goal: What’s the price of the new iPhone on eBay?
Caption: The information displayed on the phone screen is a screenshot of the Google Calendar app.
    The screenshot shows the current date and time, as well as a list of upcoming events for the
    next few days. It also highlights some of the features of the Google Calendar app, such as the
    ability to add events, set reminders, and manage multiple calendars. The screenshot provides an
    overview of the user’s schedule and helps them stay organized and on top of their upcoming
    events.
Execution history: {\"step_idx\": 0, \"action_description\": \"click [9]\"}

{\"step_idx\": 1, \"action_description\": \"click [weather like in]\"}

{\"step_idx\": 2, \"action_description\": \"click [google.com/search?q=wea]\"}

{\"step_idx\": 3, \"action_description\": \"type\"}

{\"step_idx\": 4, \"action_description\": \"click [iPhone on]\"}

{\"step_idx\": 5, \"action_description\": \"scroll up\"}

{\"step_idx\": 6, \"action_description\": \"click [iPhones for Sale - New & Used]\"}

{\"step_idx\": 7, \"action_description\": \"scroll up\"}

{\"step_idx\": 8, \"action_description\": \"click [H]\"}

\\{\"step_idx\": 9, \"action_description\": \"status_complete\"\\}

The current on-screen input is:
Screen: <p id=0 class=\"text\" alt=\"Mon, Oct 10\">Mon, Oct 10</p>
<img id=1 class=ICON_CLOUD alt=\"\"></p>
<p id=2 class=\"text\" alt=\"56\u00b0F\">56\u00b0F</p>
<img id=3 class=ICON_CALL alt=\"\"></p>
```

```
<img id=4 class=ICON_CHAT alt=\"\"></p>
<img id=5 class=ICON_PLAY alt=\"\"></p>
<img id=6 class=ICON_GOOGLE alt=\"\"></p>
<img id=7 class=ICON_MIC alt=\"\"></p>
<img id=8 class=ICON_NAV_BAR_RECT alt=\"\"></p>
<img id=9 class=ICON_NAV_BAR_CIRCLE alt=\"\"></p>
<img id=10 class=ICON_V_BACKWARD alt=\"\"></p>

Here are previous actions: (format: action \u2192 action description)
Previous Actions:
{'action_type': 'click', 'idx': 15}
And the previous steps:
Previous Steps:
Step 1. Press the home button to exit the email setup screen.

Please formulate an operational guide for future operations for solving the goal. The guide includes:
1. Plan: A **multi-step future** plan **(start from current screen, DON'T include previous steps)**;
    steps indexed by numbers.
2. Step: Based on the current screen and Previous Steps, provide the **immediate** step that needs to
     be taken from the Plan.
"**Output Format:** A JSON dictionary strictly following the format: "{'plan': '...<Your Plan Here>',
     'step': '...<Your Step Here>'} "If the goal has already been implemented, no more planning is
    required, Provide {'plan': '1. Mark the task as complete', 'step': 'Mark the task as complet'}.
**Please do not output any content other than the JSON format.**
```

A.2 The statistics for AITW dataset

AITW is a comprehensive benchmark tailored for GUI control, comprising natural language instructions, screenshots, and associated actions. Agent predicts execution actions based on screenshots and task goals across five categories shown in Table 8. The dataset spans over 350 applications and websites, totaling 715,000 episodes with 30,000 unique instructions. Subsequently, each filtered subset is partitioned episode-wise into training, validation, and test sets following 80/10/10 splits.

Dataset	**Episodes**	**Screens**	**Instructions**
General	9,476	85,413	545
Install	25,760	250,058	688
GoogleApps	625,542	4,903,601	306
Single	26,303	85,668	15,366
WebShopping	28,061	365,253	13,473

Table 8: Statistics for AITW dataset.

A.3 Evaluation metrics

Specifically, for click actions, correctness is determined if the selected element is within a 14% screen distance from the gold gestures or falls within the same detected bounding box as the user's gestures. Given the error in OCR identification, we select the top left, top right, bottom left, bottom right, and center of the box as sample points for calculating coordinate distances. Regarding scroll actions, correctness is assessed if the selected direction aligns with the scroll direction of the user's gestures. For other actions, correctness is established only if the types of actions match (Rawles et al., 2023).

A.4 The correct rate of predicted actions in ablation studies

We compute the proportion of actions within the dataset in Table 9.

Category	Proportion (%)
Click	52.54
Scroll	13.97
Typ	10.67
Navigate Home	4.44
Navigate Back	0.79
Press	1.75
Complete	15.87

Table 9: The proportion of actions on AITW.

We provide the predicted action accuracy for all datasets of ablation experiments in Table 10.

Model	Action	General	GoogleApps	Install	Single	Webshopping
NP	Click	23.48	33.06	25.52	23.81	26.51
	Scroll	0.67	2.42	2.07	-	0.60
	Typ	3.03	-	2.76	-	4.22
	Navigate Home	-	1.61	2.07	-	1.20
	Navigate Back	-	-	-	-	-
	Press Enter	-	-	-	1.59	-
	Complete	0.76	3.23	1.38	1.59	0.60
SP	Click	16.67	30.65	14.48	25.40	16.87
	Scroll	3.03	0.81	2.76	-	-
	Typ	-	-	0.69	3.17	0.60
	Navigate Home	-	0.81	0.69	-	0.60
	Navigate Back	-	-	-	-	-
	Press Enter	-	-	-	-	-
	Complete	2.27	6.45	2.07	**26.98**	0.60
DP	Click	16.67	36.29	24.14	**30.16**	22.29
	Scroll	0.76	1.61	2.07	-	0.60
	Typ	2.27	0.81	**4.14**	**1.59**	3.01
	Navigate Home	**4.52**	**2.42**	**3.45**	-	**1.81**
	Navigate Back	-	-	-	-	-
	Press Enter	-	-	-	-	-
	Complete	9.85	9.68	**6.90**	22.22	**3.01**
D-PoT	Click	**27.27**	35.48	21.38	23.81	25.90
	Scroll	3.03	**3.23**	**5.52**	-	0.60
	Typ	**3.79**	0.81	3.45	**1.59**	**4.22**
	Navigate Home	-	1.61	1.38	-	0.60
	Navigate Back	-	-	-	-	-
	Press Enter	-	-	-	-	-
	Complete	**11.36**	**11.29**	6.21	**26.98**	**3.01**
D-PoT w/ reference	Click	21.21	**37.90**	**28.28**	28.57	**30.72**
	Scroll	**4.55**	**3.23**	4.83	-	-
	Typ	2.77	-	**4.14**	**6.35**	**4.82**
	Navigate Home	0.76	1.61	1.38	-	0.60
	Navigate Back	-	-	-	-	-
	Press Enter	-	-	-	-	-
	Complete	9.85	**11.29**	5.52	22.22	1.2

Table 10: The correct rate of predicted actions of GPT-4V and D-PoT in ablation studies. We mainly collected the correct rate of “Click”, “Scroll”, “Typ’, “Navigate” and “Complete” actions. To make it look nice, we’ll replace 0 with “-”. The best average result is in boldface.

B Errors Examples

The three errors are shown here.

The first common error we identify is a bias of GPT-4V on mobile tasks. GPT-4V often exhibits “preferences” in its planning. As illustrated in Figure 5(a), when tasked with searching for specific information, GPT-4V tends to click on Google, while the human-annotated prediction suggests clicking on Chrome. Similarly, in Figure 5(b), when required to input text in the search bar, GPT-4V may plan to clear the search bar first, whereas the human prediction is to directly input the text.

The second common error we recognize is instruction overlap in the AITW dataset. The same operation on one mobile screen can correspond to two different actions. For instance, in Figure 5(a), when searching for an item, GPT-4V may click on ’search’ or the search entry, whereas the human prediction is to press. In Figure 5(b), when returning to the home page, GPT-4V often clicks on the “home” button below, while the human instruction is to “navigate home”. The third common error we classify as confusion in gesture operations. For example, in Figure 7, when swiping down to view more apps, the corresponding gesture should be from bottom to top, indicating “scroll up”. However, GPT-4V also suggests swiping down, but its predicted instruction is “scroll down”.

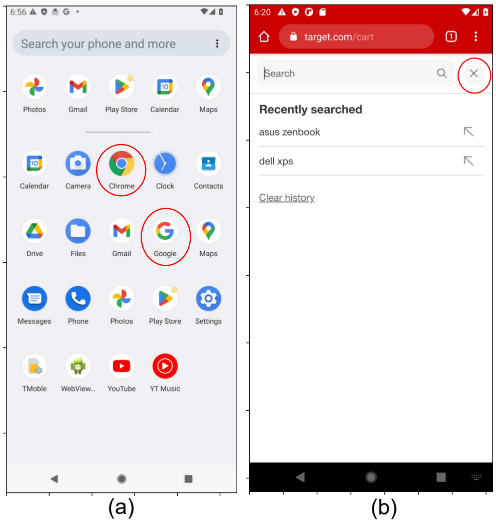


Figure 5: The first common error is a bias of GPT-4V on mobile tasks. The red circles are the steps that GPT-4V performs in a dynamic schedule.

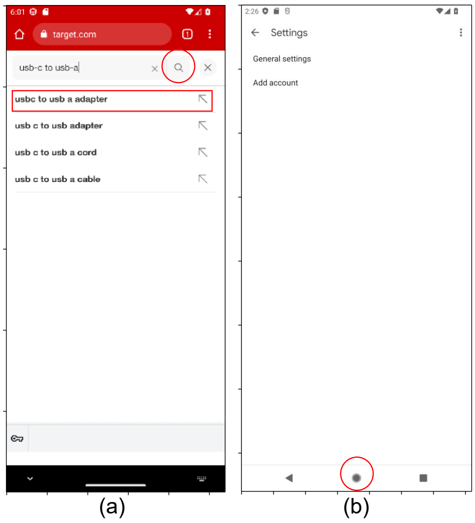


Figure 6: The second common error we recognize is instruction overlap in the AITW dataset. The red circles are the steps that GPT-4V performs in a dynamic schedule

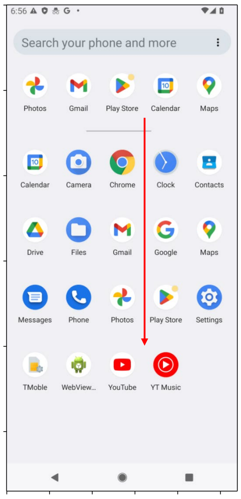


Figure 7: The third common error we classify as confusion in gesture operations. The red arrow indicates that the GPT-4V wants to slide under in dynamic planning