GUI-CIDER: Mid-training GUI Agents via Causal Internalization and Density-aware Exemplar Reselection

Anonymous ACL submission

Abstract

Despite the rapid progress of multimodal large language models in building Graphical User Interface (GUI) agents, their real-world task completion is fundamentally bottlenecked by a lack of world knowledge about GUI operations. Existing solutions typically rely on expensive multi-agent scaffolding or conventional post-training paradigms, such as Supervised Fine-Tuning (SFT) and Reinforcement Learning (RL). However, post-training only allows agents to **implicitly** absorb world knowledge through action annotations or reward signals, leading to inefficient trajectory memorization rather than genuine comprehension. Therefore, an approach that enables **explicit** learning of this knowledge is imperative. To this end, we propose **GUI-CIDER**, a mid-training method that explicitly internalizes GUI world knowledge through **C**ausal **I**nternalization and **D**ensity-aware **E**xemplar **R**eselection. GUI-CIDER operates in three stages: (1) data synthesis, which distills static planning and dynamic causal knowledge from GUI trajectories into text; (2) exemplar reselection, which filters the corpus by rewarding causal structures and penalizing semantic redundancy; and (3) mid-training, where the refined data is used to embed the acquired knowledge. Extensive experiments on two GUI knowledge benchmarks and three task completion benchmarks demonstrate that **GUI-CIDER** consistently improves both the agent's understanding of GUI operations and its task success rates.

1 Introduction

With the rapid advances of multimodal large language models (MLLMs) in reasoning (Bai et al., 2025), planning (Wei et al., 2025; Chen et al., 2026b), perception (Yu et al., 2025), and decision-making (Sun et al., 2025), MLLM-based Graphical User Interface (GUI) agents (Tang et al., 2025) can now follow user instructions to autonomously control digital devices (e.g., computers (Sager et al., 2026) and smartphones (Wu et al., 2025a)) by simulating human actions (e.g., clicking and scrolling).

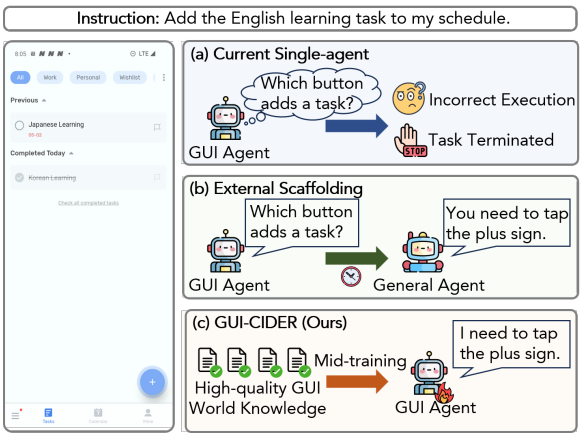


Figure 1: The motivation of GUI-CIDER. (a) Current single-agent methods lack GUI world knowledge and cannot understand that the plus sign represents adding a task. (b) External scaffolding can obtain low-level instructions by calling a general agent, but this is time-consuming. (c) GUI-CIDER enables a single agent to accomplish the task through mid-training on high-quality GUI world knowledge.

Existing work on GUI agents improves element grounding (Liu et al., 2026; Tang et al., 2026) and task completion (Bai et al., 2024; Xu et al., 2025) through post-training methods such as supervised fine-tuning (SFT) (Zhang and Zhang, 2024; Ma et al., 2024) and reinforcement learning (RL) (Lu et al., 2026; Luo et al., 2025). However, studies (Shi et al., 2025; Li et al., 2025) point out that as GUI agents continue to advance, the real capability bottleneck increasingly stems from **a lack of world knowledge related to GUI operations**.

Although plugging a capable general-purpose model into a multi-agent system (Yang et al., 2025; Wang et al., 2024) can compensate for GUI agents' deficiency in world knowledge, it introduces additional overhead and scaffolding.

In contrast, internalizing world knowledge

within the agent is more efficient, yet conventional post-training (SFT/RL) only **implicitly** encodes such knowledge through action labels or reward signals, encouraging trajectory memorization rather than genuine comprehension.

An approach that enables **explicit** learning is therefore imperative.

Consequently, as shown in Figure 1, we propose GUI-CIDER, a mid-training method for GUI agents that explicitly internalizes world knowledge into them through **C**ausal **I**nternalization and **D**ensity-aware **E**xemplar **R**eselection.

GUI-CIDER consists of three stages: (1) data synthesis stage, (2) exemplar reselection stage, and (3) mid-training stage. In the **data synthesis** stage, GUI-CIDER employs a dedicated synthesis pipeline to generate static planning knowledge and dynamic causal knowledge for the GUI agent domain from publicly available GUI agent datasets (Li et al., 2024; Lu et al., 2025; Zhang et al., 2024). In the **exemplar reselection** stage, GUI-CIDER filters the data produced in the previous stage through causal-informed retention and relative density estimation based on K-nearest neighbors, resulting in a high-quality corpus that exhibits strong reasoning structures and low redundancy. In the **mid-training** stage, GUI-CIDER uses this high-quality corpus to train the GUI agent via mid-training, thereby explicitly internalizing world knowledge into the GUI agent.

We conduct extensive experiments on three benchmarks (Li et al., 2024; Lu et al., 2025; Zhang et al., 2024) for GUI agent task completion and two benchmarks (Wang et al., 2025; Shi et al., 2025) for GUI agent knowledge. Experimental results show that GUI-CIDER achieves an average relative improvement of 9.70% in task success rate compared to post-training baselines. Meanwhile, on the GUI knowledge bench, it enables an 8B-scale agent to reach a level close to that of Claude-Sonnet-4.5.

Additionally, through model comparison analysis, we show that the target of mid-training should be general agents rather than one that has been excessively post-trained specifically in the GUI agent domain. Furthermore, we validate the rationality of the GUI-CIDER pipeline through ablation studies.

To summarize, our contributions are three-fold:

(i) We propose GUI-CIDER, a mid-training method for GUI agents that explicitly internalizes world knowledge relevant to GUI agents into them through **C**ausal **I**nternalization and **D**ensity-aware **E**xemplar **R**eselection.

(ii) We contribute a corpus of approximately 100M tokens generated from the data synthesis stage of GUI-CIDER, offering a valuable resource for related research in the community.

(iii) Through extensive experiments, we demonstrate that GUI-CIDER can not only improve GUI agents' world knowledge of GUI operations but also enhance their task completion performance.

2 Related Work

In this section, we first introduce recent improvements in GUI agents, and then we introduce related work on mid-training of (M)LLMs.

2.1 GUI Agents

GUI agents are a type of agent that operate intelligent terminals such as computers (Sager et al., 2026), web (He et al., 2024), and smartphones (Zhang and Zhang, 2024; Wu et al., 2025a) by simulating human actions like clicking and scrolling (Tang et al., 2025; Hu et al., 2025). Existing work can be broadly divided into two categories for constructing GUI agents: single-agent based and multi-agent system based. Single-agent based GUI agents are typically developed through pretraining and post-training. Pretraining enhances the agent's perception (Ma et al., 2024) and grounding capabilities (Wu et al., 2025b). Post-training methods, on the other hand, primarily improve the agent's task completion ability through techniques such as SFT (Wu et al., 2025c) and RL (Lu et al., 2026; Zhou et al., 2026; Tang et al., 2026). Multi-agent system based GUI agents distribute capabilities such as planning (Wang et al., 2024), reflection (Li et al., 2026), and execution (Yang et al., 2025; Agashe et al.) across different agents to adapt to different tasks. However, few existing works enhance the world knowledge of GUI agents through mid-training.

2.2 Mid-training for (M)LLM

Mid-training serves as a bridge (Tu et al., 2025; Mo et al., 2025) between pre-training and post-training, extending knowledge into specialized domains while preserving the general capabilities acquired during pre-training. Existing (M)LLMs (Team et al., 2025; Hu et al., 2024; Liu et al., 2024) conduct data collection, data synthesis, data selection, and data decontamination from high-quality mathematical (Paster et al., 2024; Han et al., 2024), QA (Wei et al.; Ding et al., 2023) and coding (Kocetkov et al.; Lozhkov et al., 2024; Luo et al., 2024)

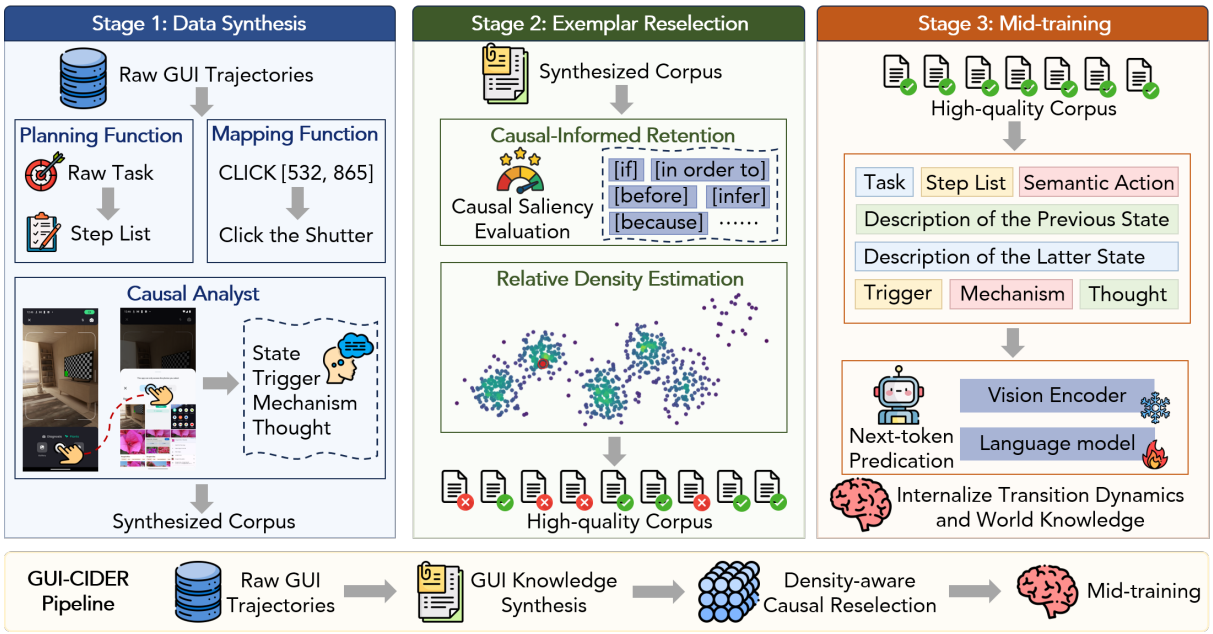


Figure 2: The pipeline of GUI-CIDER. In the data synthesis stage, GUI-CIDER synthesizes GUI world knowledge from raw GUI trajectories. In the exemplar reselection stage, GUI-CIDER selects a high-quality corpus with strong reasoning structures and low redundancy from the synthesized corpus. In the mid-training stage, GUI-CIDER internalizes the GUI world knowledge into the GUI agent through mid-training on the high-quality corpus.

domains. However, there is still very little work on internalizing domain knowledge for GUI agents through mid-training. UI-Venus-1.5 (Team et al., 2026) employed mid-training but did not open-source the data or provide specific details. Therefore, it is valuable to explore how GUI agents can internalize knowledge through mid-training.

3 GUI-CIDER

In this section, we introduce GUI-CIDER, a mid-training method for GUI agents, which stands for **C**ausal **I**nternalization and **D**ensity-aware **E**xemplar **R**eselection. As shown in Figure 2, GUI-CIDER consists of three stages: data synthesis, exemplar reselection, and mid-training. Next, we will introduce each of these stages in order.

3.1 Stage 1: Data Synthesis

Given a raw GUI agent domain dataset $\mathcal{D} = \{\tau_1, \tau_2, \ldots, \tau_N\}$, where each trajectory τ consists of a task instruction T and a sequence of screenshots and actions $\{(s_i, a_i)\}_{i=1}^{L}$, we synthesize an augmented, knowledge-rich sample x. Specifically, the synthesized sample x encompasses two primary dimensions: static planning knowledge and dynamic causal knowledge.

Static Planning Knowledge Extraction. To operationalize hierarchical task decomposition, we leverage a high-capacity LLM as a latent knowledge prior, formalizing the planning process as a structured reasoning task. Specifically, the planning function $\mathcal{P}(\cdot)$ is instantiated by an expert model that performs zero-shot reasoning to generate a hierarchical decomposition:

$$S = \mathcal{P}(T; \mathcal{M}_{exp}) = \{g_1, g_2, \ldots, g_n\}, \quad (1)$$

where $\mathcal{M}_{exp}$ denotes the expert reasoning engine and g_j represents a high-level sub-goal in natural language. This transformation converts abstract user intent into an actionable execution graph, providing dense supervisory signals for the agent's long-term planning.

Dynamic Causal Knowledge Synthesis. To explicitly model environment transition dynamics and decision-making logic while producing a purely textual knowledge sample, we reformulate knowledge extraction as a text-grounded semantic and causal induction process. This is achieved through two specialized reasoning modules:

(i) Semantic Behavioral Grounding: A mapping function $\mathcal{B}(a_t, v_t)$ that translates raw, low-level action primitives a_t and their corresponding UI metadata v_t (e.g., view hierarchy) into human-interpretable semantic descriptions a_t^{nl}. This stage bridges the gap between discrete pixel-level coordinates and high-level functional intent.

(ii) Textual State Abstraction and Causal Logic Induction: The visual screenshots s_{t-1} and s_t are first converted into natural language state descriptions d_{t-1} and d_t through a vision-language interface. For each transition under task T, we then employ a causal analyst $\mathcal{C}(\cdot)$ that operates solely over textual representations. By prompting the expert model to perform retrospective and counterfactual analysis on the described states, we extract the underlying transition logic in a self-contained textual rationale R_t:

$$\begin{aligned} R_t &= \mathcal{C}(T, d_{t-1}, a_t^{nl}, d_t \mid \mathcal{M}_{exp}) \\ &= \{d_{t-1}, d_t, Trig_t, Mech_t, CoT_t\}, \end{aligned} \tag{2}$$

where $Trig_t$, $Mech_t$, and CoT_t denote the action trigger, the underlying UI mechanism, and the chain-of-thought rationale, respectively. The state descriptions d_{t-1} and d_t are explicitly stored as part of R_t, making the rationale self-contained and eliminating the need for raw screenshots in the final sample.

The final synthesized sample x is thus defined as a purely textual, knowledge-rich tuple: $x = \langle T, S, a_t^{nl}, R_t \rangle$.

3.2 Stage 2: Exemplar Reselection

To refine the synthesized corpus $\mathcal{X} = \{x_1, \ldots, x_M\}$, we employ density-aware exemplar reselection. Let $\phi(x)$ be the embedding of sample x in a latent space $\mathcal{Z} \subseteq \mathbb{R}^d$.

Causal-Informed Retention. Following existing work (Chen et al., 2026a), we first define a causal saliency function $f(x)$ based on the count of causal-logic tokens:

$$f(x) = \tanh\left(\frac{\mathcal{K}(x)}{\gamma}\right), \tag{3}$$

where $\mathcal{K}(x)$ denotes the count of causal-logic tokens in R_t and γ controls the causal scaling. Here, causal-logic tokens broadly encompass words and phrases carrying causal or logical semantics (e.g., 'if', 'unless', 'because', 'due to'). Detailed causal-logic keywords can be found in Appendix E.

Relative Density Estimation. The local density $d(x)$ is defined based on the ratio of the K-nearest neighbor distance to the global mean distance. Let the raw ratio be

$$r(x) = \frac{\frac{1}{K}\sum_{z \in \text{KNN}(\phi(x))} \|\phi(x) - z\|^2}{\frac{1}{M}\sum_{z' \in \mathcal{X}} \|\phi(x) - z'\|^2}. \tag{4}$$

To obtain a density score in $[0, 1]$, we apply min-max normalization across all samples in the feature set $\mathcal{X}$:

$$d(x) = \frac{r(x) - \min_{x' \in \mathcal{X}} r(x')}{\max_{x' \in \mathcal{X}} r(x') - \min_{x' \in \mathcal{X}} r(x')}. \tag{5}$$

The retention probability $g(x)$ for each sample x is then given by a non-linear combination of its semantic density $d(x)$ and the causal saliency $f(x)$:

$$g(x) = \frac{1}{1 + \alpha d(x)} + \lambda \cdot f(x) \cdot \left(1 - \frac{1}{1 + \alpha d(x)}\right), \tag{6}$$

where α is a hyperparameter governing density sensitivity, and $\lambda \in [0, 1]$ is the weight for causal importance.

Finally, the high-quality corpus $\mathcal{X}_{\text{high}}$ is formed by retaining each sample x with probability $g(x)$:

$$\mathcal{X}_{\text{high}} = \{\, x \in \mathcal{X} \mid \xi_x \leq g(x) \,\}, \tag{7}$$

where $\xi_x \sim \text{Uniform}(0, 1)$ is sampled independently for every x.

3.3 Stage 3: Mid-training

In the mid-training stage, we directly perform next-token prediction on the high-quality corpus $\mathcal{X}_{\text{high}}$. For each synthesized sample x, we first format it into a single token sequence by concatenating its components in a fixed order. No distinction is made between input and output: the entire sequence is treated as a plain text stream for autoregressive language modeling. The training objective is the standard causal language modeling loss over all tokens in the sequence:

$$\mathcal{L}_{\text{mid}} = -\sum_{x \in \mathcal{X}_{\text{high}}} \sum_{i=1}^{L_x} \log P_\theta(y_i \mid y_{<i}), \tag{8}$$

where L_x is the total number of tokens in the serialized sequence of sample x, and y_i denotes the i-th token. By optimizing $\mathcal{L}_{\text{mid}}$, the model internalizes the transition dynamics $P(s_t \mid s_{t-1}, a_t)$ and the underlying world knowledge directly into its parametric memory, achieving causal internalization without necessitating external runtime scaffolds.

4 Is the Retention Function $g(x)$ a Good Function?

In this section, we first introduce the properties of a good retention function $g(x)$ under our task setting, and then provide theoretical support to prove that GUI-CIDER's retention function $g(x)$ satisfies all these properties.

4.1 Properties for the Retention Function $g(x)$

To effectively select high-value samples with strong reasoning structures and low redundancy, the retention function $g(x)$ should possess the following four properties:

Property 1 (Causal Monotonicity) *$g(x)$ should be monotonically increasing with respect to the causal saliency $f(x)$.*

Samples with more causal-logic tokens contain richer reasoning structures and therefore deserve higher retention probabilities.

Property 2 (Density Penalty) *$g(x)$ should be monotonically decreasing with respect to the density $d(x)$.*

Higher density indicates that many different samples share similar semantics, leading to redundancy; thus, the retention probability should be penalized accordingly.

Property 3 (Density Order Preservation) *$g(x)$ should satisfy $\frac{\partial}{\partial d}\big(d \cdot g(x)\big) > 0$.*

Although we filter the corpus, we must not invert the original density ordering of the semantic space, thereby preserving the relative density structure.

Property 4 (Density-Causal Synergy) *$g(x)$ should satisfy $\frac{\partial^2 g(x)}{\partial f \partial d} > 0$.*

In denser regions where semantic redundancy is high, the survival competition is fiercer. Therefore, an increase in causal saliency should provide a greater marginal benefit to the retention probability, enabling the most logically rigorous exemplars to stand out among highly redundant samples.

4.2 Theoretical Support

We now prove that the retention function defined in GUI-CIDER satisfies the three properties.

Proof of Property 1. For a fixed density d, the derivative of g with respect to f is

$$\frac{\partial g}{\partial f} = \lambda \left(1 - \frac{1}{1+\alpha d}\right) = \lambda \frac{\alpha d}{1+\alpha d}. \tag{9}$$

Since $\lambda \geq 0$, $\alpha > 0$, and $d \geq 0$, we have $\frac{\partial g}{\partial f} \geq 0$. Thus, $g(x)$ is monotonically non-decreasing in $f(x)$. For any sample with $d > 0$, the derivative is strictly positive, ensuring that higher causal saliency strictly increases the retention probability.

Proof of Property 2. The derivative with respect to density d is

$$\frac{\partial g}{\partial d} = -\frac{\alpha(1-\lambda f)}{(1+\alpha d)^2}. \tag{10}$$

Because $f \in [0, 1)$ and $\lambda \in [0, 1]$, we have $1 - \lambda f \geq 0$. With $\alpha > 0$, the derivative is non-positive, so $g(x)$ is monotonically non-increasing in $d(x)$. This directly imposes a redundancy penalty: denser samples receive lower retention probabilities.

Proof of Property 3. For the product $d \cdot g(x)$:

$$\begin{aligned}
\frac{\partial}{\partial d}(d \cdot g) &= g + d \cdot \frac{\partial g}{\partial d} \\
&= \lambda f + \frac{1-\lambda f}{1+\alpha d} - \frac{\alpha d(1-\lambda f)}{(1+\alpha d)^2} \\
&= \lambda f + (1-\lambda f)\frac{1+\alpha d - \alpha d}{(1+\alpha d)^2} \\
&= \lambda f + \frac{1-\lambda f}{(1+\alpha d)^2}. \qquad (11)
\end{aligned}$$

Since $\lambda f \geq 0$ and $1 - \lambda f \geq 0$ with a strictly positive denominator, we obtain $\frac{\partial}{\partial d}(d \cdot g) > 0$ for all valid parameter settings. This guarantees that if two samples have densities $d_1 < d_2$, then $d_1 \cdot g(x_1) < d_2 \cdot g(x_2)$ (all else being equal), faithfully preserving the original density ordering of the semantic space.

Proof of Property 4. The cross-partial derivative of $g(x)$ is:

$$\frac{\partial^2 g}{\partial f \partial d} = \frac{\partial}{\partial d}\left(\lambda \frac{\alpha d}{1+\alpha d}\right) = \frac{\lambda \alpha}{(1+\alpha d)^2}. \tag{12}$$

Given $\lambda > 0$ and $\alpha > 0$, we strictly have $\frac{\partial^2 g}{\partial f \partial d} > 0$. This guarantees that the marginal utility of causal saliency f increases monotonically with density d, effectively prioritizing high-quality reasoning structures within redundant clusters.

5 Experiment

In this section, we first introduce the implementation of our GUI-CIDER experiments, then present the main results and provide analysis.

Task Category	Dataset	Evaluation Format	Train Size	Test Size
GUI Agent Knowledge	MMBench-GUI L1	MCQ	-	3,561
	GUI Knowledge Bench	T/F, MCQ	-	3,483
GUI Agent Task Completion	AITZ	Action Generation	13,919	4,723
	AndroidControl		69,670	7836
	GUI-Odyssey		102,086	25,807

Table 1: Overview of datasets and evaluation format.

Method	AITZ			AndroidControl			GUI-Odyssey		
	Type	SR	TSR	Type	SR	TSR	Type	SR	TSR
Qwen3-VL-4B-Instruct									
Zero-shot	59.58	39.53	0.79	74.92	51.82	13.60	61.89	41.00	0.30
GUI-CIDER	60.46	41.22	0.99	76.24	53.58	14.43	64.83	43.45	0.42
Post-training	76.63	60.43	4.94	85.07	68.75	28.43	89.72	73.46	3.86
GUI-CIDER + Post-training	77.43	61.87	5.14	85.17	69.77	28.56	89.47	75.36	4.34
Qwen3-VL-8B-Instruct									
Zero-shot	56.91	40.82	0.99	73.92	52.49	13.96	67.86	44.16	0.36
GUI-CIDER	60.70	42.07	1.58	74.87	54.09	15.13	70.26	48.55	0.36
Post-training	72.98	58.16	4.15	83.50	65.34	23.46	88.82	71.74	3.32
GUI-CIDER + Post-training	73.70	60.33	5.14	83.41	66.82	25.35	89.65	73.36	3.63

Table 2: Results of AITZ, AndroidControl and GUI-Odyssey benchmarks. Whether compared with zero-shot or post-training methods, adding a mid-training process leads to improvements.

5.1 Implementation

Dataset. As shown in Table 1, we conduct extensive experiments on three benchmarks, AITZ (Zhang et al., 2024), AndroidControl (Li et al., 2024), and GUI-Odyssey (Lu et al., 2025), for GUI agent task completion, where the agent is required to output actions to accomplish tasks, and two benchmarks, MMbench-GUI L1 (Wang et al., 2025) and GUI knowledge bench (Shi et al., 2025), for GUI agent knowledge, both of which adopt the formats of multiple-choice questions (MCQs) and true-false (T/F) questions.

Evaluation Method. We used GUI-CIDER for data synthesis on the AITZ, AndroidControl, and GUI-Odyssey datasets. The base models were Qwen3-VL-4B-Instruct and Qwen3-VL-8B-Instruct (Bai et al., 2025). In the main results section, we refer to the models obtained by mid-training Qwen3-VL-4B-Instruct and Qwen3-VL-8B-Instruct with GUI-CIDER as GUI-CIDER-4B and GUI-CIDER-8B, respectively. For evaluation on MMBench-GUI L1 and GUI Knowledge Bench, we performed mid-training using a mixture of all synthesized data. For evaluation on AITZ, AndroidControl, and GUI-Odyssey, we conducted mid-training using the data synthesized from the corresponding dataset. Meanwhile, we adopted SFT as the baseline method for post-training.

Metrics. For the AITZ, AndroidControl, and GUI-Odyssey datasets, we report action type accuracy (type), step-wise success rate (SR), and task success rate (TSR). For MMbench-GUI L1 and GUI Knowledge Bench, we compute the accuracy of multiple-choice questions and true-false questions under different subsets.

5.2 Main Results

The results on the AITZ, AndroidControl, and GUI-Odyssey datasets are shown in Table 2, the results on MMbench-GUI L1 are shown in Table 3, and the results on the GUI knowledge benchmark are shown in Table 4.

Based on the above results, we find:

(i) As shown in Table 2, mid-training yields gains in task completion capability across models of different parameter scales. Furthermore, when post-training is applied after mid-training, the benefits of GUI-CIDER still manifest. In addition, a 4B-

Model	Windows	MacOS	Linux	iOS	Android	Web	Overall
Easy Level							
GPT-4o	62.47	67.89	62.38	58.52	56.41	58.51	60.16
Qwen-Max-VL	69.05	72.51	69.91	70.82	63.09	69.46	68.15
Qwen2.5-VL-72B	65.86	75.23	73.02	67.24	58.09	72.08	66.98
UI-TARS-72B-DPO	41.59	28.52	35.16	31.08	52.25	35.33	40.18
InternVL3-72B	74.67	78.72	79.16	83.57	80.10	81.18	79.15
GUI-CIDER-8B	95.19	97.62	96.91	90.43	93.44	94.98	94.69
Medium Level							
GPT-4o	56.33	63.13	59.70	54.06	57.69	54.98	57.24
Qwen-Max-VL	63.40	73.85	66.90	68.02	63.66	64.59	65.44
Qwen2.5-VL-72B	66.29	72.73	72.63	59.27	66.24	68.24	67.45
UI-TARS-72B-DPO	38.83	41.60	37.14	41.72	54.74	31.55	41.77
InternVL3-72B	71.46	78.58	79.88	78.43	81.36	78.67	77.89
GUI-CIDER-8B	95.56	91.67	96.39	89.57	89.84	88.13	92.00
Hard Level							
GPT-4o	60.69	60.38	52.42	45.27	50.93	50.83	53.49
Qwen-Max-VL	66.64	67.59	65.80	60.23	58.78	65.34	63.69
Qwen2.5-VL-72B	70.68	68.91	70.98	57.59	53.94	68.10	64.56
UI-TARS-72B-DPO	31.48	35.87	24.19	36.33	58.13	19.94	35.78
InternVL3-72B	75.08	77.44	76.19	70.37	75.73	78.11	75.70
GUI-CIDER-8B	93.33	90.48	94.33	92.17	88.20	93.15	91.83

Table 3: Results of MMBench-GUI L1 benchmark. GUI-CIDER enhances GUI understanding. Although the data sources do not cover every platform in MMBench-GUI, improvements are still observed across all of them.

scale GUI agent, after undergoing mid-training and post-training with GUI-CIDER, surpasses its 8B-scale counterpart, suggesting that for GUI agents, what matters may not be parameter scaling but rather knowledge scaling.

(ii) As shown in Table 3, GUI-CIDER-8B significantly outperforms the baselines, indicating that GUI-CIDER brings improvements to the GUI content understanding capability of GUI agents.

(iii) As shown in Table 4, overall, GUI-CIDER-8B clearly bridges the knowledge gap in GUI tasks, achieving performance close to that of Claude-Sonnet-4.5 at the 8B scale (66.51 vs. 66.53). Moreover, GUI-CIDER-8B surpasses all larger-scale models (e.g., o3, Gemini-2.5-Pro) on the objective subset (which assesses whether a task is truly completed), demonstrating that GUI-CIDER equips the GUI agent with a better understanding of tasks.

6 Further Analysis

In this section, we first compare the differences between models that have undergone post-training in the GUI agent domain and general models when used as the base model for GUI-CIDER, followed by an ablation study.

6.1 Model Comparison Analysis

We conduct an analysis to verify whether a GUI-specialized model that has already been post-trained in the GUI agent domain can acquire new world knowledge again through mid-training. We perform experiments on the AITZ dataset with OS-Atlas-pro-7B following the GUI-CIDER, and report results with the amount of GUI-CIDER-generated data increasing in 20% increments.

As shown in Figure 3, when using the general model Qwen3-VL-8B-Instruct as the base model, the GUI agent's SR consistently improves as more GUI-CIDER-generated data are incorporated. In contrast, when using OS-Atlas-pro-7B as the base model, the GUI agent's performance steadily declines. This is because OS-Atlas-pro-7B has undergone extensive post-training for GUI agents, which has already partially disrupted its original language representation capacity, making it difficult to learn

Model	Interface Knowledge			Interaction Knowledge			Procedure Knowledge		Overall
	state	widget	layout	effect	type	parameter	objective	workflow	
O3	83.03	84.12	88.39	74.83	75.98	45.75	69.45	95.47	73.30
Gemini-2.5-Pro	81.19	84.36	87.10	71.03	73.25	46.97	67.72	92.56	71.69
GPT-5-Chat	78.90	84.12	88.39	71.55	71.55	43.85	68.98	91.26	70.97
Claude-Sonnet-4.5	74.77	81.52	82.58	49.83	70.19	43.33	70.30	91.56	66.53
Qwen3-VL-8B-Instruct	66.97	76.30	79.35	60.86	63.54	45.06	71.02	78.96	65.23
Qwen2.5-VL-72B	69.27	77.49	80.00	61.72	64.91	38.99	62.20	85.44	63.88
Doubao-V-Pro	72.48	83.65	81.29	67.24	75.64	41.07	33.07	94.17	63.42
Claude-Sonnet-4	70.18	78.44	78.06	41.90	62.52	42.11	65.20	94.82	62.16
Qwen2.5-VL-7B	53.21	67.77	60.00	51.72	50.60	39.34	16.22	48.87	45.16
UI-TARS-1.5-7B	49.54	59.48	59.35	22.24	59.11	34.32	38.74	55.34	44.27
GUI-owl-7B	60.09	64.93	63.23	21.55	55.37	36.05	21.26	39.81	40.74
GLM-4.5	49.54	48.10	53.55	27.07	17.55	35.53	28.98	91.91	38.10
GUI-CIDER-8B	72.61	75.83	80.13	61.55	65.42	46.45	71.81	80.58	66.51

Table 4: Results of GUI knowledge benchmark. The performance of GUI-CIDER-8B is close to Claude-Sonnet-4.5.

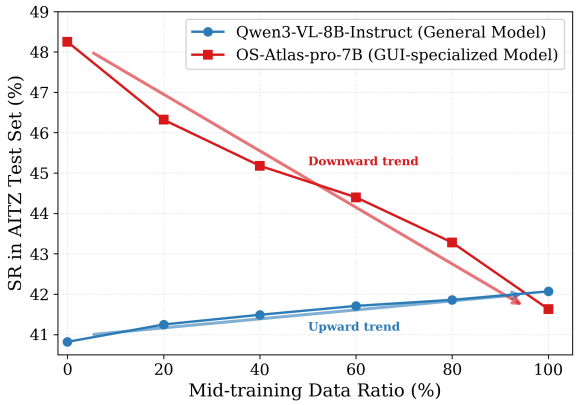


Figure 3: Comparison between a general model and a GUI-specialized model as the base model for mid-training, illustrating that models excessively post-trained in the GUI agent domain are no longer suitable for acquiring world knowledge through mid-training.

new world knowledge through mid-training. Therefore, performing mid-training on world knowledge before conducting post-training in the GUI agent domain would be a reasonable paradigm.

6.2 Ablation Study

We conduct an ablation study to investigate the necessity of the exemplar reselection stage in GUI-CIDER. Specifically, we compare SR after mid-training with the complete GUI-CIDER pipeline against a variant that removes the exemplar reselection stage on the GUI-Odyssey dataset. As shown in Table 5, removing the exemplar reselection stage leads to a substantial drop in SR. This is because directly incorporating large-scale unscreened data into mid-training introduces a considerable amount of low-quality and redundant samples. Such noisy supervision can mislead the GUI agent and encourage shortcut or hacking behaviors, ultimately harming generalization and decision-making capability.

Table 5: Ablation study of exemplar reselection stage.

Model	w/ Stage 2	w/o Stage 2
Qwen3-VL-4B-Instruct	43.45	41.06
Qwen3-VL-8B-Instruct	48.55	42.34

7 Conclusion

In this paper, we present GUI-CIDER, a mid-training framework that internalizes GUI world knowledge into GUI agents through causal internalization and density-aware exemplar reselection. Instead of relying on expensive external scaffolding or directly applying post-training to raw trajectories, GUI-CIDER synthesizes static planning knowledge and dynamic causal transition logic from GUI trajectories, then selects knowledge-rich and non-redundant exemplars with a retention function. Experiments on three task completion benchmarks and two GUI knowledge benchmarks show that GUI-CIDER improves both GUI operation understanding and downstream task success. These results suggest that knowledge scaling is a promising path toward more capable GUI agents.

Limitations

Due to computational resource constraints, GUI-CIDER employs LoRA instead of full parameter tuning during the mid-training stage. Additionally, the model parameters used for training range from 4B to 8B. Future work will further explore the effectiveness under full parameter tuning and scale the approach to larger models.

References

Saaket Agashe, Kyle Wong, Vincent Tu, Jiachen Yang, Ang Li, and Xin Eric Wang. Agent s2: A compositional generalist-specialist framework for computer use agents. In *Second Conference on Language Modeling*.

Hao Bai, Yifei Zhou, Mert Cemri, Jiayi Pan, Alane Suhr, Sergey Levine, and Aviral Kumar. 2024. Digirl: Training in-the-wild device-control agents with autonomous reinforcement learning. *Advances in Neural Information Processing Systems*, 37:12461–12495.

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

Qiguang Chen, Yantao Du, Ziniu Li, Jinhao Liu, Songyao Duan, Jiarui Guo, Minghao Liu, Jiaheng Liu, Tong Yang, Ge Zhang, and 1 others. 2026a. The molecular structure of thought: Mapping the topology of long chain-of-thought reasoning. *arXiv preprint arXiv:2601.06002*.

Yanyu Chen, Jiyue Jiang, Jiahong Liu, Yifei Zhang, Xiao Guo, and Irwin King. 2026b. Trace: Trajectory-aware comprehensive evaluation for deep research agents. In *Proceedings of the ACM Web Conference 2026*, pages 2524–2534.

Ning Ding, Yulin Chen, Bokai Xu, Yujia Qin, Shengding Hu, Zhiyuan Liu, Maosong Sun, and Bowen Zhou. 2023. Enhancing chat language models by scaling high-quality instructional conversations. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 3029–3051.

Xiaotian Han, Yiren Jian, Xuefeng Hu, Haogeng Liu, Yiqi Wang, Qihang Fan, Yuang Ai, Huaibo Huang, Ran He, Zhenheng Yang, and 1 others. 2024. Infimm-webmath-40b: Advancing multimodal pre-training for enhanced mathematical reasoning. *arXiv preprint arXiv:2409.12568*, 4.

Hongliang He, Wenlin Yao, Kaixin Ma, Wenhao Yu, Yong Dai, Hongming Zhang, Zhenzhong Lan, and Dong Yu. 2024. Webvoyager: Building an end-to-end web agent with large multimodal models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6864–6890.

Shengding Hu, Yuge Tu, Xu Han, Chaoqun He, Ganqu Cui, Xiang Long, Zhi Zheng, Yewei Fang, Yuxiang Huang, Weilin Zhao, and 1 others. 2024. Minicpm: Unveiling the potential of small language models with scalable training strategies. *arXiv preprint arXiv:2404.06395*.

Xueyu Hu, Tao Xiong, Biao Yi, Zishu Wei, Ruixuan Xiao, Yurun Chen, Jiasheng Ye, Meiling Tao, Xiangxin Zhou, Ziyu Zhao, and 1 others. 2025. Os agents: A survey on mllm-based agents for computer, phone and browser use. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7436–7465.

Denis Kocetkov, Raymond Li, Chenghao Mou, Yacine Jernite, Margaret Mitchell, Carlos Muñoz Ferrandis, Sean Hughes, Thomas Wolf, Dzmitry Bahdanau, Leandro Von Werra, and 1 others. The stack: 3 tb of permissively licensed source code. *Transactions on Machine Learning Research*.

Chunyi Li, Longfei Li, Zicheng Zhang, Xiaohong Liu, Min Tang, Weisi Lin, and Guangtao Zhai. 2025. Using gui agent for electronic design automation. *arXiv preprint arXiv:2512.11611*.

Ning Li, Xiangmou Qu, Jiamu Zhou, Muning Wen, Kounianhua Du, Xingyu Lou, Qiuying Peng, Jun Wang, and Weinan Zhang. 2026. Mobileuse: A hierarchical reflection-driven gui agent for autonomous mobile operation. *Advances in Neural Information Processing Systems*, 38:40361–40388.

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

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

Yuhang Liu, Zeyu Liu, Shuanghe Zhu, Pengxiang Li, Congkai Xie, Jiasheng Wang, Xueyu Hu, Xiaotian Han, Jianbo Yuan, Xinyao Wang, and 1 others. 2026. Infigui-g1: Advancing gui grounding with adaptive exploration policy optimization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 32267–32275.

Anton Lozhkov, Raymond Li, Loubna Ben Allal, Federico Cassano, Joel Lamy-Poirier, Nouamane Tazi, Ao Tang, Dmytro Pykhtar, Jiawei Liu, Yuxiang Wei, and 1 others. 2024. Starcoder 2 and the stack v2: The next generation. *arXiv preprint arXiv:2402.19173*.

Quanfeng Lu, Wenqi Shao, Zitao Liu, Lingxiao Du, Fanqing Meng, Boxuan Li, Botong Chen, Siyuan Huang, Kaipeng Zhang, and Ping Luo. 2025. Guiodyssey: A comprehensive dataset for cross-app gui navigation on mobile devices. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 22404–22414.

Zhengxi Lu, Yuxiang Chai, Yaxuan Guo, Xi Yin, Liang Liu, Hao Wang, Han Xiao, Shuai Ren, Pengxiang Zhao, Guangyi Liu, and 1 others. 2026. Ui-r1: Enhancing efficient action prediction of gui agents by reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 17608–17616.

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

Ziyang Luo, Can Xu, Pu Zhao, Qingfeng Sun, Xiubo Geng, Wenxiang Hu, Chongyang Tao, Jing Ma, Qingwei Lin, and Daxin Jiang. 2024. Wizardcoder: Empowering code large language models with evol-instruct. In *International Conference on Learning Representations*, volume 2024, pages 27168–27188.

Xinbei Ma, Zhuosheng Zhang, and Hai Zhao. 2024. Coco-agent: A comprehensive cognitive mllm agent for smartphone gui automation. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 9097–9110.

Kaixiang Mo, Yuxin Shi, Weiwei Weng, Zhiqiang Zhou, Shuman Liu, Haibo Zhang, and Anxiang Zeng. 2025. Mid-training of large language models: A survey. *arXiv preprint arXiv:2510.06826*.

Keiran Paster, Marco Dos Santos, Zhangir Azerbayev, and Jimmy Ba. 2024. Openwebmath: An open dataset of high-quality mathematical web text. In *International Conference on Learning Representations*, volume 2024, pages 20357–20379.

Pascal J Sager, Benjamin Meyer, Peng Yan, Rebekka von Wartburg-Kottler, Layan Etaiwi, Aref Enayati, Gabriel Nobel, Ahmed Abdulkadir, Benjamin F Grewe, and Thilo Stadelmann. 2026. A comprehensive survey of agents for computer use: Foundations, challenges, and future directions. *Journal of Artificial Intelligence Research*, 85.

Chenrui Shi, Zedong Yu, Zhi Gao, Ruining Feng, Enqi Liu, Yuwei Wu, Yunde Jia, Liuyu Xiang, Zhaofeng He, and Qing Li. 2025. Gui knowledge bench: Revealing the knowledge gap behind vlm failures in gui tasks. *arXiv preprint arXiv:2510.26098*.

Chuanneng Sun, Songjun Huang, and Dario Pompili. 2025. Llm-based multi-agent decision-making: Challenges and future directions. *IEEE Robotics and Automation Letters*.

Fei Tang, Zhangxuan Gu, Zhengxi Lu, Xuyang Liu, Shuheng Shen, Changhua Meng, Wen Wang, Wenqi Zhang, Yongliang Shen, Weiming Lu, and 1 others. 2026. Gui-g2: Gaussian reward modeling for gui grounding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 33214–33222.

Fei Tang, Haolei Xu, Hang Zhang, Siqi Chen, Xingyu Wu, Yongliang Shen, Wenqi Zhang, Guiyang Hou, Zeqi Tan, Yuchen Yan, and 1 others. 2025. A survey on (m) llm-based gui agents. *arXiv preprint arXiv:2504.13865*.

Meituan LongCat Team, Bei Li, Bingye Lei, Bo Wang, Bolin Rong, Chao Wang, Chao Zhang, Chen Gao, Chen Zhang, Cheng Sun, and 1 others. 2025. Longcat-flash technical report. *arXiv preprint arXiv:2509.01322*.

Venus Team, Changlong Gao, Zhangxuan Gu, Yulin Liu, Xinyu Qiu, Shuheng Shen, Yue Wen, Tianyu Xia, Zhenyu Xu, Zhengwen Zeng, and 1 others. 2026. Ui-venus-1.5 technical report. *arXiv preprint arXiv:2602.09082*.

Chengying Tu, Xuemiao Zhang, Rongxiang Weng, Rumei Li, Chen Zhang, Yang Bai, Hongfei Yan, Jingang Wang, and Xunliang Cai. 2025. A survey on llm mid-training. *arXiv preprint arXiv:2510.23081*.

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

Xuehui Wang, Zhenyu Wu, JingJing Xie, Zichen Ding, Bowen Yang, Zehao Li, Zhaoyang Liu, Qingyun Li, Xuan Dong, Zhe Chen, and 1 others. 2025. Mmbench-gui: Hierarchical multi-platform evaluation framework for gui agents. *arXiv preprint arXiv:2507.19478*.

Hui Wei, Zihao Zhang, Shenghua He, Tian Xia, Shijia Pan, and Fei Liu. 2025. Plangenllms: A modern survey of llm planning capabilities. *arXiv preprint arXiv:2502.11221*.

Yuxiang Wei, Zhe Wang, Jiawei Liu, Yifeng Ding, and LINGMING ZHANG. Magicoder: Empowering code generation with oss-instruct. In *Forty-first International Conference on Machine Learning*.

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

Zhiyong Wu, Zhenyu Wu, Fangzhi Xu, Yian Wang, Qiushi Sun, Chengyou Jia, Kanzhi Cheng, Zichen Ding, Liheng Chen, Paul Pu Liang, and 1 others. 2025b. Os-atlas: Foundation action model for generalist gui agents. In *International Conference on Learning Representations*, volume 2025, pages 5090–5108.

Zongru Wu, Rui Mao, Zhiyuan Tian, Pengzhou Cheng, Tianjie Ju, Zheng Wu, Lingzhong Dong, Haiyue Sheng, Zhuosheng Zhang, and Gongshen Liu. 2025c. See, think, act: Teaching multimodal agents to effectively interact with gui by identifying toggles. *arXiv preprint arXiv:2509.13615*.

Yifan Xu, Xiao Liu, Xinghan Liu, Jiaqi Fu, Hanchen Zhang, Bohao Jing, Shudan Zhang, Yuting Wang, Wenyi Zhao, and Yuxiao Dong. 2025. Mobilerl: Online agentic reinforcement learning for mobile gui agents. *arXiv preprint arXiv:2509.18119*.

Yan Yang, Dongxu Li, Yutong Dai, Yuhao Yang, Ziyang Luo, Zirui Zhao, Zhiyuan Hu, Junzhe Huang, Amrita Saha, Zeyuan Chen, and 1 others. 2025. Gta1: Gui test-time scaling agent. *arXiv preprint arXiv:2507.05791*.

Runpeng Yu, Xinyin Ma, and Xinchao Wang. 2025. Auto-controlled image perception in mllms via visual perception tokens. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 21822–21831.

Jiwen Zhang, Jihao Wu, Teng Yihua, Minghui Liao, Nuo Xu, Xiao Xiao, Zhongyu Wei, and Duyu Tang. 2024. Android in the zoo: Chain-of-action-thought for gui agents. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 12016–12031.

Zhuosheng Zhang and Aston Zhang. 2024. You only look at screens: Multimodal chain-of-action agents. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 3132–3149.

Yuqi Zhou, Sunhao Dai, Shuai Wang, Kaiwen Zhou, Qinglin Jia, and Jun Xu. 2026. Gui-g1: Understanding r1-zero-like training for visual grounding in gui agents. *Advances in Neural Information Processing Systems*, 38:95683–95705.

A End-to-End Algorithm

Algorithm 1 summarizes the full GUI-CIDER pipeline, including knowledge-rich sample synthesis, density-aware exemplar reselection, and mid-training on the retained corpus.

Algorithm 1 GUI-CIDER

Input: $\mathcal{D}$, $\mathcal{M}_{exp}$, ϕ, K, α, λ, γ
Output: $\mathcal{X}_{\text{high}}$, θ
Initialize synthesized corpus $\mathcal{X} \leftarrow \emptyset$
for all $\tau = (T, \{(s_t, a_t, v_t)\}_{t=1}^{L}) \in \mathcal{D}$ **do**
  $S \leftarrow \mathcal{P}(T; \mathcal{M}_{exp})$
  for $t = 1$ to L **do**
    $a_t^{nl} \leftarrow \mathcal{B}(a_t, v_t)$
    Convert s_{t-1}, s_t to text descriptions d_{t-1}, d_t
    $R_t \leftarrow \mathcal{C}(T, d_{t-1}, a_t^{nl}, d_t \mid \mathcal{M}_{exp})$
    $x_t \leftarrow \langle T, S, a_t^{nl}, R_t \rangle$
    $\mathcal{X} \leftarrow \mathcal{X} \cup \{x_t\}$
  end for
end for
for all $x \in \mathcal{X}$ **do**
  $f(x) \leftarrow \tanh(\mathcal{K}(x)/\gamma)$
  Compute $r(x)$ by the ratio of local K-NN distance to global mean distance
end for
Min-max normalize $\{r(x)\}_{x\in\mathcal{X}}$ into $\{d(x)\}_{x\in\mathcal{X}}$
for all $x \in \mathcal{X}$ **do**
  $b_x \leftarrow \frac{1}{1+\alpha d(x)}$
  $g(x) \leftarrow b_x + \lambda f(x)(1 - b_x)$
  Sample $\xi_x \sim \text{Uniform}(0, 1)$
  if $\xi_x \leq g(x)$ **then**
    $\mathcal{X}_{\text{high}} \leftarrow \mathcal{X}_{\text{high}} \cup \{x\}$
  end if
end for
Mid-train agent parameters θ on $\mathcal{X}_{\text{high}}$ with the causal LM loss $\mathcal{L}_{\text{mid}}$
return $\mathcal{X}_{\text{high}}, \theta$

B Additional Mathematical Proofs

In this appendix, we provide supplementary theoretical results for the density-aware exemplar reselection rule used in GUI-CIDER. Throughout, let

$$b(d) = \frac{1}{1 + \alpha d}, \tag{13}$$

so that the retention function in Section 4 can be rewritten as

$$\begin{aligned} g(x) &= b(d(x)) + \lambda f(x)\big(1 - b(d(x))\big) \\ &= \frac{1 + \alpha\lambda f(x) d(x)}{1 + \alpha d(x)}. \end{aligned} \tag{14}$$

This form makes explicit that $g(x)$ interpolates between a density-based baseline $b(d)$ and a causal-saliency correction controlled by λ.

B.1 Range, Boundary Cases, and Parameter Interpretation

Proposition A.1 (Range and Boundary Cases). For any sample x with $f(x) \in [0, 1]$, $d(x) \in [0, 1]$, $\alpha > 0$, and $\lambda \in [0, 1]$, the retention score satisfies

$$\frac{1}{1 + \alpha d(x)} \leq g(x) \leq 1 \tag{15}$$

and

$$\lambda f(x) \leq g(x) \leq 1. \tag{16}$$

Moreover, the following boundary cases hold:

$$g(x) = 1 \quad \text{if } d(x) = 0, \tag{17}$$

$$g(x) = \frac{1}{1 + \alpha d(x)} \quad \text{if } f(x) = 0, \tag{18}$$

and

$$g(x) = 1 \quad \text{if } \lambda = 1 \text{ and } f(x) = 1. \tag{19}$$

Proof. Since $f(x) \in [0,1]$ and $\lambda \in [0,1]$, we have

$$\begin{aligned} g(x) &= b(d(x)) + \lambda f(x)\big(1 - b(d(x))\big) \\ &\geq b(d(x)) = \frac{1}{1+\alpha d(x)}. \end{aligned} \tag{20}$$

Likewise,

$$g(x) \leq b(d(x)) + \big(1 - b(d(x))\big) = 1. \tag{21}$$

For the second lower bound,

$$\begin{aligned} g(x) - \lambda f(x) &= b(d(x))\big(1 - \lambda f(x)\big) \\ &\geq 0 \end{aligned} \tag{22}$$

which implies $g(x) \geq \lambda f(x)$. The boundary identities follow directly by substitution into the closed form of $g(x)$.

B.2 Proofs of the Four Desiderata

Proposition A.2 (Causal Monotonicity). For fixed $d(x)$, the retention function is monotonically non-decreasing in $f(x)$.

Proof. For a fixed density d, the derivative of g with respect to f is

$$\frac{\partial g}{\partial f} = \lambda\left(1 - \frac{1}{1+\alpha d}\right) = \lambda\frac{\alpha d}{1+\alpha d}. \tag{23}$$

Since $\lambda \geq 0$, $\alpha > 0$, and $d \geq 0$, we have $\frac{\partial g}{\partial f} \geq 0$. Therefore, $g(x)$ is monotonically non-decreasing in $f(x)$. It is strictly increasing whenever $\lambda > 0$ and $d > 0$.

Proposition A.3 (Density Penalty). For fixed $f(x)$, the retention function is monotonically non-increasing in $d(x)$.

Proof. Differentiating g with respect to d gives

$$\frac{\partial g}{\partial d} = -\frac{\alpha(1-\lambda f)}{(1+\alpha d)^2}. \tag{24}$$

Because $f \in [0,1]$ and $\lambda \in [0,1]$, we have $1 - \lambda f \geq 0$. Hence $\frac{\partial g}{\partial d} \leq 0$, so denser samples always receive no larger retention scores.

Proposition A.4 (Density Order Preservation). The reweighted density score $d \cdot g(x)$ is strictly increasing in d:

$$\frac{\partial}{\partial d}\big(d \cdot g(x)\big) > 0. \tag{25}$$

Proof. Using the closed form of g,

$$d \cdot g(x) = \frac{d + \alpha\lambda f d^2}{1+\alpha d}. \tag{26}$$

Applying the quotient rule yields

$$\begin{aligned} \frac{\partial}{\partial d}\big(d \cdot g(x)\big) &= \frac{(1+2\alpha\lambda f d)(1+\alpha d)}{(1+\alpha d)^2} \\ &\quad - \frac{\alpha(d+\alpha\lambda f d^2)}{(1+\alpha d)^2}. \end{aligned} \tag{27}$$

After simplification,

$$\frac{\partial}{\partial d}\big(d \cdot g(x)\big) = \frac{1 + 2\alpha\lambda f d + \alpha^2\lambda f d^2}{(1+\alpha d)^2}. \tag{28}$$

Every term in the numerator is non-negative, and the constant term is strictly positive. Therefore,

$$\frac{\partial}{\partial d}\big(d \cdot g(x)\big) > 0. \tag{29}$$

Thus, the reweighted density preserves the original ordering induced by d.

Proposition A.5 (Density-Causal Synergy). The retention rule exhibits positive interaction between causal saliency and density:

$$\frac{\partial^2 g(x)}{\partial f \partial d} > 0. \tag{30}$$

Proof. From Proposition A.2,

$$\frac{\partial g}{\partial f} = \lambda\frac{\alpha d}{1+\alpha d}. \tag{31}$$

Differentiating again with respect to d gives

$$\begin{aligned} \frac{\partial^2 g}{\partial f \partial d} &= \lambda\alpha\frac{(1+\alpha d) - \alpha d}{(1+\alpha d)^2} \\ &= \frac{\lambda\alpha}{(1+\alpha d)^2}. \end{aligned} \tag{32}$$

For $\lambda > 0$ and $\alpha > 0$, the above quantity is strictly positive. Hence the marginal value of causal saliency becomes larger in denser regions, which is exactly the desired synergy effect.

B.3 Expected Retained Corpus Size and Information Preservation

The next result formalizes the intuition that stochastic retention preserves a non-trivial fraction of the original reasoning signal.

Proposition A.6 (Expectation of the Retained Corpus). Let $\mathbf{1}_x$ be the Bernoulli indicator of whether sample x is retained, namely $\mathbf{1}_x = 1$ if $\xi_x \le g(x)$ and $\mathbf{1}_x = 0$ otherwise. Then

$$\mathrm{E}\big[|\mathcal{X}_{\text{high}}|\big] = \sum_{x\in\mathcal{X}} g(x). \tag{33}$$

In particular,

$$\sum_{x\in\mathcal{X}} \frac{1}{1+\alpha d(x)} \le \mathrm{E}\big[|\mathcal{X}_{\text{high}}|\big] \le |\mathcal{X}|. \tag{34}$$

Proof. By construction, $\mathbf{1}_x$ is a Bernoulli random variable with mean $g(x)$. Therefore,

$$\begin{aligned} |\mathcal{X}_{\text{high}}| &= \sum_{x\in\mathcal{X}} \mathbf{1}_x, \\ \mathrm{E}\big[|\mathcal{X}_{\text{high}}|\big] &= \sum_{x\in\mathcal{X}} \mathrm{E}[\mathbf{1}_x] = \sum_{x\in\mathcal{X}} g(x). \end{aligned} \tag{35}$$

The bounds follow immediately from Proposition A.1.

Proposition A.7 (Lower Bound on Preserved Causal Information). Define the causal information mass of a subset $\mathcal{A} \subseteq \mathcal{X}$ as

$$I_f(\mathcal{A}) = \sum_{x\in\mathcal{A}} f(x). \tag{36}$$

Then the expected preserved causal information ratio satisfies

$$\begin{aligned} \frac{\mathrm{E}[I_f(\mathcal{X}_{\text{high}})]}{I_f(\mathcal{X})} &= \frac{\sum_{x\in\mathcal{X}} f(x)g(x)}{\sum_{x\in\mathcal{X}} f(x)} \\ &\ge \lambda \frac{\sum_{x\in\mathcal{X}} f(x)^2}{\sum_{x\in\mathcal{X}} f(x)} \\ &\ge \lambda \bar{f} \end{aligned} \tag{37}$$

where $\bar{f} = \frac{1}{|\mathcal{X}|}\sum_{x\in\mathcal{X}} f(x)$ is the average causal saliency of the original corpus.

Proof. Since $I_f(\mathcal{X}_{\text{high}}) = \sum_x \mathbf{1}_x f(x)$, linearity of expectation gives

$$\begin{aligned} \mathrm{E}[I_f(\mathcal{X}_{\text{high}})] &= \sum_{x\in\mathcal{X}} f(x)\mathrm{E}[\mathbf{1}_x] \\ &= \sum_{x\in\mathcal{X}} f(x)g(x). \end{aligned} \tag{38}$$

By Proposition A.1, $g(x) \ge \lambda f(x)$ for every x, hence

$$\sum_{x\in\mathcal{X}} f(x)g(x) \ge \lambda \sum_{x\in\mathcal{X}} f(x)^2. \tag{39}$$

Dividing both sides by $\sum_x f(x)$ proves the first inequality. For the second inequality, Cauchy–Schwarz implies

$$\sum_{x\in\mathcal{X}} f(x)^2 \ge \frac{\big(\sum_{x\in\mathcal{X}} f(x)\big)^2}{|\mathcal{X}|}, \tag{40}$$

and therefore

$$\begin{aligned} \lambda \frac{\sum_x f(x)^2}{\sum_x f(x)} &\ge \lambda \frac{\sum_x f(x)}{|\mathcal{X}|} \\ &= \lambda \bar{f}. \end{aligned} \tag{41}$$

This shows that the thinning process retains a guaranteed fraction of the original causal signal in expectation.

B.4 Stability Under Score Estimation Errors

In practice, both $f(x)$ and $d(x)$ are estimated from synthesized text and embedding geometry. The next result shows that moderate perturbations in these quantities induce controlled perturbations in $g(x)$.

Proposition A.8 (Lipschitz Stability). Let $\hat{f}(x)$ and $\hat{d}(x)$ be perturbed estimates satisfying

$$|\hat{f}(x) - f(x)| \le \varepsilon_f, \qquad |\hat{d}(x) - d(x)| \le \varepsilon_d. \tag{42}$$

If $\hat{g}(x) = g(\hat{f}(x), \hat{d}(x))$, then

$$|\hat{g}(x) - g(x)| \le \alpha\varepsilon_d + \lambda \frac{\alpha}{1+\alpha}\varepsilon_f. \tag{43}$$

Proof. By the mean value theorem applied to the bivariate function $g(f, d)$, there exists a point on the line segment joining (f, d) and $(\hat{f}, \hat{d})$ such that

$$\begin{aligned} |\hat{g} - g| \le{}& \sup|\partial_d g| \cdot |\hat{d} - d| \\ &+ \sup|\partial_f g| \cdot |\hat{f} - f|. \end{aligned} \tag{44}$$

From the derivatives established above,

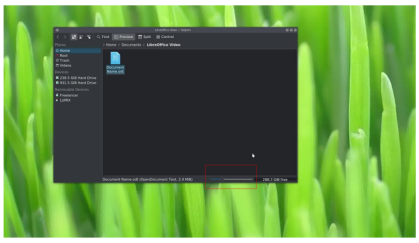
$$|\partial_d g| = \frac{\alpha(1-\lambda f)}{(1+\alpha d)^2} \le \alpha \tag{45}$$

and

$$\begin{aligned} |\partial_f g| &= \lambda \frac{\alpha d}{1+\alpha d} \\ &\le \lambda \frac{\alpha}{1+\alpha} \end{aligned} \tag{46}$$

because $d \in [0, 1]$. Substituting the error bounds yields

$$\begin{aligned} |\hat{g}(x) - g(x)| \le{}& \alpha\varepsilon_d \\ &+ \lambda \frac{\alpha}{1+\alpha}\varepsilon_f. \end{aligned} \tag{47}$$



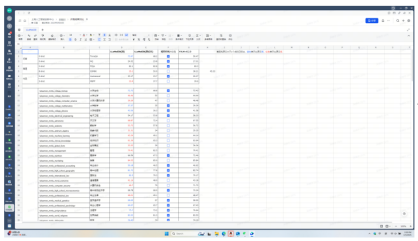
Question:
What happens if you drag the slider highlighted by the red bounding box at the bottom center of the window?

Options:
A. Adjust the system audio output volume for media played in this folder.
B. Increase the size of file icons and previews in the directory view.
C. Scrub through the selected document to preview its pages within the panel.
D. Zoom the desktop background wallpaper without affecting the file browser content.

Answer:
Correct Answer: B

Needed Knowledge:
In many Linux file managers (such as KDE Dolphin), the bottom status bar slider controls the folder view zoom, changing the size of file icons and thumbnails displayed in the directory.

Figure 4: Examples from GUI Knowledge Bench.



Question:
Which statement correctly describes the placement of the checkbox column in this spreadsheet?

Options:
A. It is immediately to the left of the first numeric column.
B. It is in the last visible column.
C. It is between two numeric score columns.
D. It is before the row headers.
E. It is adjacent to the row numbers

Answer:
Correct Answer: C

Needed Knowledge:
The checkbox column appears between two numeric score columns (between the score column for LLMa65B 校正 and the PJLm v0.2.0 score column).

Figure 5: Examples from MMBench-GUI L1.

Hence the retention rule is stable under bounded score estimation noise.

C Benchmark Details and Examples

We briefly summarize the five benchmarks used in our experiments and point readers to the corresponding examples in the appendix.

GUI Knowledge Bench. GUI Knowledge Bench is a diagnostic benchmark for GUI-specific knowledge rather than end-to-end execution. It evaluates whether a model can understand widget functions, interface states, interaction effects, and workflow progress across diverse GUI platforms. Representative examples are shown in Figure 4.

MMBench-GUI L1. MMBench-GUI is a hierarchical cross-platform benchmark for GUI agents, and the L1 split used here focuses on GUI content understanding. Its examples test whether a model can read interface content and reason about the semantics and relative placement of GUI elements. Representative examples are shown in Figure 5.

AITZ. AITZ (Android-In-The-Zoo) is an Android GUI navigation benchmark built from screen-action pairs with Chain-of-Action-Thought annotations. It evaluates whether an agent can infer the next GUI action from the current screen, prior context, and task instruction. A representative example is shown in Figure 6.

AndroidControl. AndroidControl studies real-world Android control at scale using human demonstrations paired with both high-level and low-level instructions. It is designed to evaluate how well agents follow everyday mobile tasks under realistic data diversity and task complexity.

GUI-Odyssey. GUI-Odyssey focuses on long-horizon cross-app mobile navigation, where successful completion requires carrying context across multiple apps and steps. Compared with single-app benchmarks, it places heavier demands on history tracking, planning, and cross-app reasoning.

D Examples of Synthetic Data

Figure 7 presents a representative synthetic training example produced by GUI-CIDER. Starting from a task and a concrete GUI transition, GUI-CIDER organizes the annotation into a step list, action intent, pre-state description, post-state description, trigger, mechanism, and reasoning. This format converts raw interaction traces into structured supervision

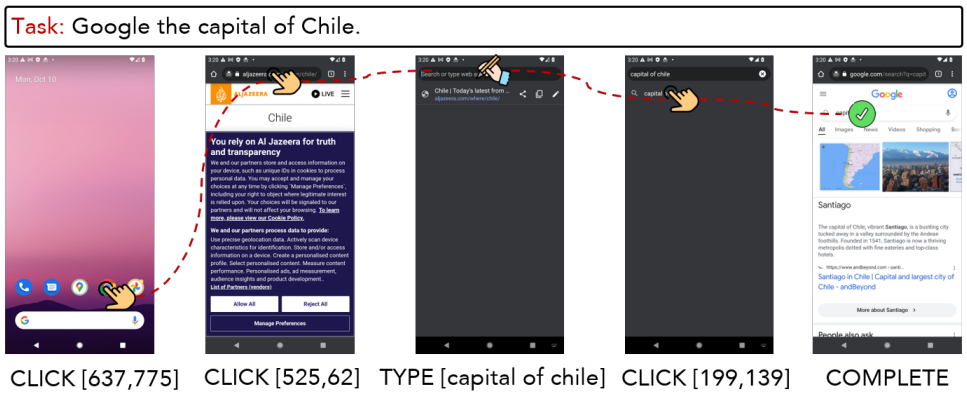


Figure 6: Examples from GUI agent task completion benchmarks.

Task: What's US dollar exchange rate against the British Pound?
Step list: ["Open a web browser","Navigate to a search engine (e.g., Google)", "Type 'USD to GBP exchange rate' in the search bar", "Press Enter to search", "Read the current exchange rate from the search results"]
Action Intent: Initiate a downward scroll gesture to reveal additional content, such as notifications, widgets, or the app drawer, by interacting with the upper portion of the screen.
Pre-state Description: The home screen displays a clean, minimal layout with a pink-to-purple gradient wallpaper, a date and weather widget at the top, and a dock containing five app icons (Phone, Messages, Gmail, Chrome, Files) at the bottom, with a Google search bar centered below the dock.
Post-state Description: The screen transitions to the app drawer, revealing a grid of app icons including Photos, Play Store, YouTube, Clock, Calendar, Camera, Chrome, Contacts, Drive, Files, Gmail, Google, Maps, Messages, Phone, Settings, and others, replacing the home screen widgets and dock with a comprehensive list of installed applications.
Trigger: A downward scroll gesture initiated from the upper portion of the screen (status bar area)
Mechanism: The gesture activates the app drawer navigation, which overlays the home screen with a full list of apps, enabling access to applications not present in the dock.
Reasoning: The downward scroll from the status bar is the standard Android gesture to open the app drawer, which is necessary to access the Google app or Chrome browser to search for the USD to GBP exchange rate. Since the home screen does not provide a direct search interface or relevant app in the dock (beyond Chrome, which is already accessible), opening the app drawer ensures access to the most efficient tools for completing the task. This action is optimal as it expands the available options to perform a web or app-based currency lookup.

Figure 7: Examples of GUI-CIDER Synthetic Data.

that captures not only what action should be taken, but also why the action is appropriate in the current GUI context.

E Keyword and Prompt Templates

Figures 8 and 9 summarize the lexical categories used in our data-synthesis pipeline to surface reasoning-rich textual patterns. These categories cover conditional and hypothetical statements, purpose and intent, explicit causal chains, temporal ordering, evidential language, verification cues, and comparison markers. In our pipeline, they are used as lightweight anchors for prompt design and quality inspection, helping the expert model generate rationales that more consistently expose triggers, mechanisms, and outcome-oriented reasoning.

Figure 10 presents the prompt templates used to instantiate the three key modules in Stage 1: the planning function $\mathcal{P}$, the semantic behavioral grounding function $\mathcal{B}$, and the causal analyst $\mathcal{C}$. Together, these templates standardize how raw task descriptions, low-level GUI actions, and paired pre-/post-state screenshots are converted into the textual tuple $\langle T, S, a_t^{nl}, R_t \rangle$ used in mid-training.

F Experimental Details

In our experiments, the Planning function is implemented using deepseek-v4-flash, while both the mapping function and the causal analyst are based on Qwen3-VL-32B-Instruct. The base models are Qwen3-VL-4B-Instruct and Qwen3-VL-8B-Instruct. We set the per-device training batch size to 4, gradient accumulation steps to 2, learning rate to 1.0e-5, and train for 2 epochs. Moreover, for the

Keywords for Conditions and Hypotheses

if / unless / whether / provided that / once / whenever / in case / if not / otherwise / given that / assuming that / suppose / on condition that / in the event that / as long as / only if / even if / if and only if / lest

Keywords for Purpose and Intent

in order to / aiming to / try to / intended to / purpose of / goal is / to achieve / for the sake of / so as to / in an effort to / with the aim of / for the purpose of / motivated by / designed to / meant to / serve to / aim to / seek to / strive to / in a bid to / with a view to / so that

Keywords for Causal Chains and Mechanisms

because / due to / leads to / result in / trigger / caused by / followed by / subsequently / initially / finally / meanwhile / afterward / as a result / for this reason / owing to / on account of / thanks to / in response to / as a consequence / thereby / in turn / bring about / give rise to / responsible for / play a role in / contribute to / lead to / led to / leading to / yield / produce / induced by / provoke / elicit / entail / imply / affect / influence / impact / determine / shape / drive / motivate / prompt / account for / stem from / originate from / be attributed to / be ascribed to

Figure 8: Keyword lexicon used in Stage 1 data synthesis (Part I). The listed markers target conditional and hypothetical reasoning, purpose and intent, and explicit causal chains or UI mechanisms. We use these lexical cues as lightweight anchors when eliciting and auditing reasoning-rich rationales from synthesized trajectories.

Action Type	Action Description
CLICK	Click at specified position.
TYPE	Enter text at designated location.
SCROLL	Scroll in direction.
PRESS_BACK	Go to previous screen.
PRESS_HOME	Go to home page.
ENTER	Press enter button.
OPEN_APP	Open specified app.
WAIT	Wait for screen to load.
LONG_PRESS	Long press at specified position.
COMPLETE	Indicate task finished.
IMPOSSIBLE	Indicate task impossible.

Table 6: Action space in our experiment.

post-training mentioned in the main text, we used a full-scale SFT approach, using the same training set as GUI-CIDER for training and the same test set for evaluation. For the same data, applying mid-training with GUI-CIDER followed by post-training yields improvements compared to direct post-training. This demonstrates that GUI-CIDER can further unlock the potential of the data. The total computational cost amounts to 1,400 hours on 80G GPUs. For GUI agent task completion benchmarks, we adhere to the assessment methods of existing works: for actions with coordinates such as CLICK and LONG_PRESS, a error of less than 14% is considered correct. For TYPE actions, an F1 score greater than 0.5 is required to be counted as correct. In all other cases, exact matching is necessary for correctness. And TSR for a task will be 1 only if SR for every single frame within that task is 1. Action space is shown as Table 6.

Keywords for Temporal Markers

before / after / during / prior to / preceding / following / subsequent / consequent / antecedent / precede / succeed

Keywords for Reasoning and Evidence

therefore / thus / hence / consequently / accordingly / indicate / suggest / demonstrate / reveal / infer / deduce / conclude / reason / argue / evidence / proof / support / corroborate / substantiate / consistent with / contradict / rule out / implicate / underscore / highlight / point to / signify

Keywords for Verification and Cross-checking

verify / check / confirm / detect / identify / compare / observe / recognize / validate / evaluate / assess / test / examine / scrutinize / inspect / monitor / trace

Keywords for Comparison and Analogy

whereas / while / compared with / in contrast / similarly / likewise / analogous

Figure 9: Keyword lexicon used in Stage 1 data synthesis (Part II). This part focuses on temporal markers, reasoning and evidence expressions, verification cues, and comparison or analogy terms, which help identify ordering constraints, supporting evidence, and contrastive reasoning in textualized GUI transitions.

Prompt for Planning Function

You are an expert in GUI task planning. Given a task description, break it down into a detailed step list. Each step should be a clear atomic action described in English. Output ONLY a JSON object in the following format: {{"step_list": ["step 1", "step 2", ...]}}

Task description: {task}

Prompt for Mapping Function

Role: You are an Expert UI Behavioral Analyst.
Task: Map a low-level execution token to a high-level semantic affordance.
Action: {action_raw}
the two coordinates of the click action are relative coordinates in thousandths: the first coordinate is the thousandths coordinate from left to right, and the second coordinate is the thousandths coordinate from top to bottom. For example, 500,500 means the exact center of the image.
Output: [Target Element]: (Name/Description) | [Action Intent]: (Detailed functional purpose).

Prompt for Causal Analyst

Role: Expert GUI Causal Analyst & Data Architect.
Task: Generate a high-fidelity state transition trajectory to train a GUI Agent's causal reasoning.

Inputs:
- Task Goal: {task}
- Action (A_t): {action_nl}
- Visuals: Image 1 (Pre-state $S_{{t}}$), Image 2 (Post-state $S_{{t+1}}$).

Output Structure (Strictly follow this template):

[Pre-state Description]:
(Brief one-sentence description of Image 1, focusing on the layout, the state of the target element, and relevant environmental context.)

[Post-state Description]:
(Brief one-sentence description of Image 2, explicitly highlighting visual deltas such as new overlays, changed icons, or list updates compared to Image 1.)

[Action Effect]:
- Trigger: (What interaction is performed, e.g., "Clicking the 'Cart' icon at [x,y]")
- Mechanism: (What UI function is invoked, e.g., "Triggers a navigation event to the checkout page via the app's routing engine")

[Reasoning]:
(A rigorous but brief explanation of why A_t was the optimal choice to fulfill the Task Goal given the state S_t, linking the affordance of the element to the objective.)

Figure 10: Prompt templates used to instantiate the planning function $\mathcal{P}$, the semantic mapping function $\mathcal{B}$, and the causal analyst $\mathcal{C}$.