Android in the Zoo: Chain-of-Action-Thought for GUI Agents

Jiwen Zhang[1,2] *, **Jihao Wu**[2], **Yihua Teng**[2], **Minghui Liao**[2],
Nuo Xu[2], **Xiao Xiao**[2], **Zhongyu Wei**[1†], **Duyu Tang**[2†]
[1]Fudan University [2]Huawei Inc.
jiwenzhang21@m.fudan.edu.cn
{wujihao,tengyihua,liaominghui1,xunuo4,xiaoxiao55}@huawei.com
zywei@fudan.edu.cn duyutang@huawei.com
https://github.com/IMNearth/CoAT

Abstract

Large language model (LLM) leads to a surge of autonomous GUI agents for smartphone, which completes a task triggered by natural language through predicting a sequence of actions of API. Even though the task highly relies on past actions and visual observations, existing studies typically consider little semantic information carried out by intermediate screenshots and screen operations. To address this, this work presents **Chain-of-Action-Thought** (dubbed **CoAT**), which takes the description of the previous actions, the current screen, and more importantly the action thinking of what actions should be performed and the outcomes led by the chosen action. We demonstrate that, in a zero-shot setting upon three off-the-shelf LMMs, CoAT significantly improves the action prediction compared to previous proposed context modeling. To further facilitate the research in this line, we construct a dataset **Android-In-The-Zoo** (**AITZ**), which contains 18,643 screen-action pairs together with chain-of-action-thought annotations. Experiments show that fine-tuning a 1B model (i.e. AUTO-UI-base) on our **AITZ** dataset achieves on-par performance with CogAgent-Chat-18B.

1 Introduction

Nowadays, smartphones have become an essential part of daily lives. AUTOnomous operation of Graphical User Interfaces (GUI) by human instructions can substantially simplify everyday routines. Such tasks, formalized as **GUI Navigation** (Li et al., 2020b; Sun et al., 2022b), therefore carry immense social importance, especially for people with physical disabilities (Nanavati et al., 2023).

Recent works have explored prompt engineering (Wen et al., 2023; Zhang and Zhang, 2023), finetuning (Hong et al., 2023) and memory augmentation (Lee et al., 2023) to utilize the capability of large language models (LLM) on interactive mobile environments. However, progress is held back due to the scarcity of attention paid on the underlying semantics of smartphone operations. GUI navigation usually entails initially observing the screen, considering the next action to take, and reflecting on the outcome of that action (Zhang et al., 2024a). Previous works (Zhang and Zhang, 2023; Cheng et al., 2024) ignore the logic behind diverse actions on the screen, concentrating solely on the coordinates of an operation, such as "click on (0.17, 0.89)", which is quite insufficient. As shown in Figure 1, we need explicit explanations for the intermediate results during GUI navigation:

- **Screen Context**: In which app or interface did the action occur? This helps to learn the background and possible effects of the action.
- **Action Think**: Why the specific action on the current screen is chosen? Does it facilitate the completion of user query? Such thinking process helps the agent to better capture the user intent.
- **Action Target**: Which UI element is the action operating on? A button, an icon, or a link?
- **Action Result**: What change will this action cause? Understanding this ensures the consistency of the agent decision-making process.

In order to equip existing GUI agents with such capability, we summarize the series of navigation steps as **Chain-of-Action-Thought** (**CoAT**), including the screen description, the thinking process about the next action, the textual next action description, and the possible action outcomes. Screen description, together with the screenshots, provides the agent with information basis for decision-making (Wang et al., 2021). Whereas action think, action description and action result demonstrate the rationale between operations. Equipped with CoAT, we achieve significant improvements in the

* This work was done during this author's internship at Shanghai Research Center of Huawei Inc.
† Corresponding Author.

Findings of the Association for Computational Linguistics: EACL 2024, pages 12016–12031
November 12-16, 2024

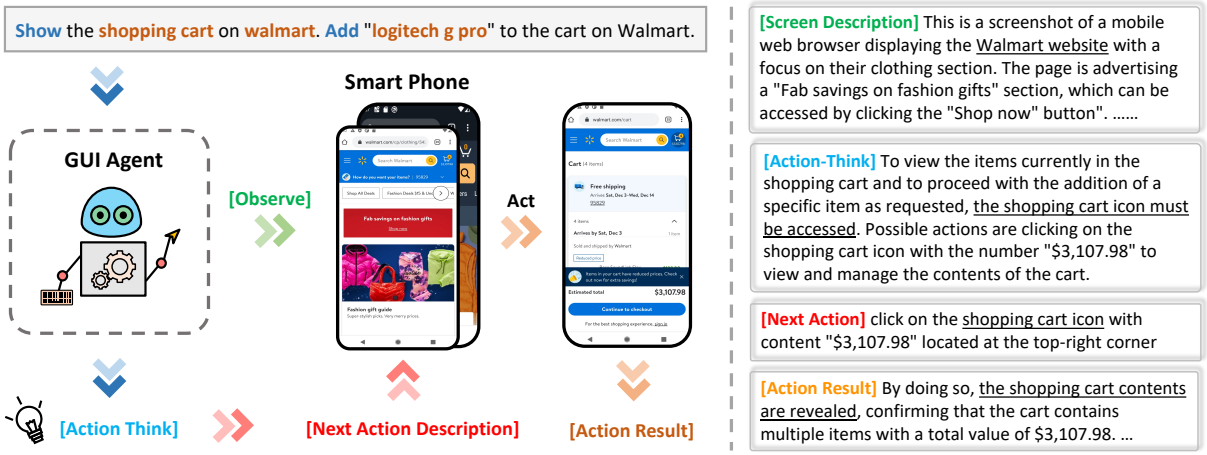


Figure 1: **The working process of Chain-of-Action-Thought**. The agent will observe the screen, think about actions on current screen to fulfill the user query, describe its next action, act and finally reflect on action results.

Dataset	#Episodes	#Unique Instructions	#Apps	#Steps	Annotation				
					screen desc	action coord	action desc	action thinking	episode feasibility
PixelHelp (Li et al., 2020b)	187	187	4	~4		✓			
MoTIF (Burns et al., 2021)	4707	270	125	4.5		✓			✓
UGIF (Venkatesh et al., 2022)	523	480	12	6.3		✓	✓		
Meta-GUI (Sun et al., 2022a)	4684	1125	11	5.3		✓			
AITW (Rawles et al., 2023)	715142	30378	357+	6.5		✓			
AITZ (Ours)	2504	2504	70+	7.5	✓	✓	✓	✓	✓

Table 1: **Comparison of AITZ to existing Android GUI datasets.** We consider the number of episodes, instructions, related apps, average steps and granularity of annotations. Specifically, action semantics includes action descriptions and action thinkings, while episode feasibility refers to the success verification of collected episodes.

action prediction across three off-the-shelf large multimodal models (LMM) compared to standard context prompting, including GPT-4V (OpenAI, 2023), Gemini-Pro-Vision (Team et al., 2023) and Qwen-VL-Max (Bai et al., 2023).

However, complex context modeling of language models emerges at a large model scale (Zhang et al., 2023). Without high quality CoAT-driven data, smaller models can not possess the desired ability through fine-tuning. To remedy this blank, we propose a new dataset **Android-In-The-Zoo** (**AITZ**). **AITZ** is the first dataset that connects the perception (of screen layouts and UI elements) and the cognition (of action decision-making process) together. Based on the screen episodes from (Rawles et al., 2023), we leverage the most-capable proprietary model, GPT-4V (OpenAI, 2023), and state-of-the-art icon detection model (Liu et al., 2018) to generate candidate answers for the screen descriptions, action thinkings and next action descriptions. These candidates are further validated and refined by human to guarantee alignment with the screenshots. Finally, **AITZ** contains about 19,000 screenshots spanning over 70 Android apps, coupled with 4× useful annotations compared with action coordinate labels only. We verify the effectiveness of CoAT by additionally finetuning a small multimodal agent from scratch on our **AITZ** dataset. Experiments show that our proposed chain-of-action-thought improves both the goal progress and the learning efficiency of GUI agents.

Our contributions are summarized as follows:

- We propose **Chain-of-Action-Thought** (**CoAT**), a novel prompting paradigm to explicitly capture the underlying semantics during navigation actions, allowing GUI agents to perceive, think and decide in an interleaved manner.
- We construct Android-In-The-Zoo (**AITZ**), the first and largest fine-grained dataset in the Android GUI navigation field. **AITZ** consisting of 2504 unique instructions and 18,643 screen-action pairs together with four types of semantic annotations, spanning over 70 Android apps.
- We conduct both zero-shot and fine-tuning evaluation on the **AITZ** dataset, validating the necessity and effectiveness of proposed chain-of-action-thought prompting.

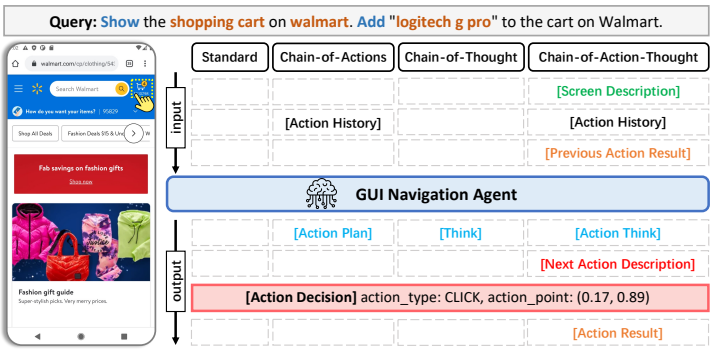


Figure 2: **Chain-of-Action-Thought compared with three typical prompting methods for GUI tasks,** including Standard (Rawles et al., 2023) prompting, Chain-of-Action (Zhang and Zhang, 2023) prompting and Chain-of-Thought (Wei et al., 2022) prompting.

Prompt	Metric	Model		
		QwenVL	Gemini-PV	GPT-4V
CoA	hit	94.5	**99.8**	99.3
	acc	44.4	47.7	**62.8**
CoT	hit	95.6	**97.5**	97.1
	acc	49.4	52.0	**64.1**
CoAT	hit	96.3	96.4	**98.2**
	acc	52.4	54.5	**73.5**

Table 2: **Quantitative comparison of three prompting methods** on Qwen-VL-Max, Gemini-1.0-Pro-Vision and GPT-4V. CoA and CoT are short for chain-of-action and chain-of-thought, respectively. "hit" means format hit rate, and "acc" means accuracy.

2 Chain-of-Action-Thought (CoAT)

2.1 Definition

Consider a general GUI navigation agent with a user query $u \in \mathcal{U}$ to solve. At time step t, an agent receives a screenshot observation $o_t \in \mathcal{O}$ from the environment and takes an action $a_t \in \mathcal{A}$ following some policy $\pi(a_t|o_t, h_{t-1}, u)$ where $h_{t-1} = (o_1, a_1, ..., o_{t-1}, a_{t-1})$ is the history for the agent. Directly learning the policy is challenging as the relations between history, current observations, and possible actions are highly implicit. For example, knowing the search bar is already active is necessary for an agent to make the next action decision to type text. Therefore, we define Chain-of-Action-Thought (CoAT) as a shortcut to comprehend the interaction dynamics during navigation.

The basic components of CoAT, marked as grey-bordered boxes on the right side of Figure 1, are:

- **Screen Description** (SD) describes the main content of the given screenshots, including the screen type and primary apps or widgets presented. Screen description provides the textual context for further decision-making.
- **Action Think** (AT) analyzes the user query and current screen, and combines the history information to infer the possible actions that help to fulfil the target. Mathematically, action think provides a conditional probability $p(AT|o_t, u, h_{t-1})$. If the action think summarizes the current state perfectly and contains reasonable action plans, the decision can be made by calculating $p(a_t|AT)$.
- **Next Action Description** (AD) illustrates the UI element or screen functions being operated, i.e. "click on the shopping cart icon" or "scroll up to open the app drawer". Action description helps to form a readable action history.
- **Action Result** (AR) connects the current screen o_t and next action a_t to the future observations o_{t+1}, by synthesizing the action outcomes after comparing the screenshot before and after the action. Usually, at time step t, we combine last action result AR_{t-1} with previous action descriptions to form a continuous and consistent history.

Since each CoAT component carries useful semantics, it is free to combine them according to language models used. Our further experiments will validate the effectiveness and flexibility of the application of proposed CoAT framework.

2.2 Comparison

Figure 2 compares proposed CoAT with Standard (Rawles et al., 2023), Chain-of-Action (CoA) (Zhang and Zhang, 2023) and Chain-of-Thought (CoT) (Wei et al., 2022) prompting methods. The proposed CoAT carries explicitly more semantic information about the screen and actions. To further validate the effectiveness of CoAT, we conduct a preliminary experiment on 50 episodes randomly sampled from AITW (Rawles et al., 2023) dataset. We select three most capable proprietary models, i.e. GPT-4V (OpenAI, 2023), Gemini-Pro-Vision (Team et al., 2023) and Qwen-VL-Max (Bai et al., 2023), to be the GUI agent and apply different prompting methods on them. To ensure an accurate measurement of action prediction accuracy, we use set-of-mark tagging method (Yan et al., 2023) to annotate UI elements on screen. As shown in Table 2, agents with CoAT surpass CoA and CoT by a large margin. Moreover, GPT-4V demonstrates optimal performance, making it a good collaborator for subsequent data collection.

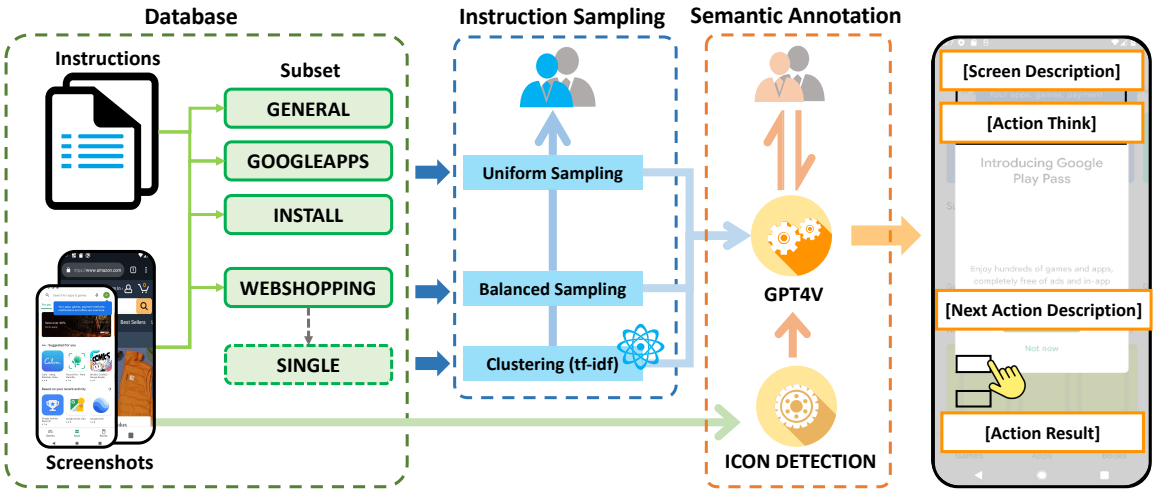


Figure 3: **AITZ data collection pipeline.** During sampling process, human annotators first verify the clustering results, and then check whether the sampled episode successfully complete the query. During annotation process, human annotators examine and correct the GPT generated semantic descriptions.

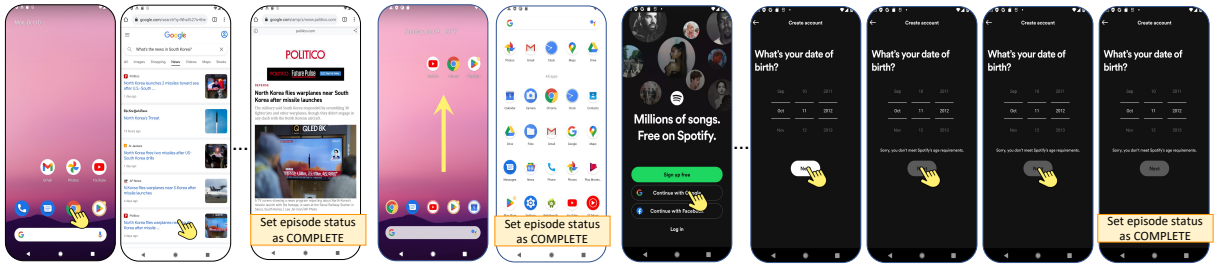


Figure 4: **Three typical cases of wrong episodes in AITW (Rawles et al., 2023) dataset.** We take the task to 'check the settings for the spotify app' as example. There exists 15 episodes corresponding to this instruction, and among them 13 do not actually open the spotify app. This highlights the reasonability to perform data validation.

3 Android in the Zoo (AITZ)

There is a lack of data that captures the underlying semantics of the CoAT paradigm, hindering small models from obtaining this ability. We therefore propose to construct a novel, high-quality and comprehensive dataset to remedy this blank.

3.1 Data Collection

Instruction Sampling We build our dataset upon the currently most scaled Android GUI navigation dataset, AITW (Rawles et al., 2023). AITW dataset has 715k episodes spanning 30k unique instructions. We observe that (1) the diversity of instructions mainly comes from the subset WEBSHOPPING, and these instructions have clear templates, as shown in Table 3; (2) the richness of episodes results from subset GOOGLEAPPS, where each instruction corresponds to more than 2000 episodes. However, within the AITW dataset, there exist numerous mismatch cases between the observed screenshots and the instructions (see Figure 4). Thus, we sample the instructions and episodes to reduce redundancy and filter the error cases by using a subset-specific sampling strategy:

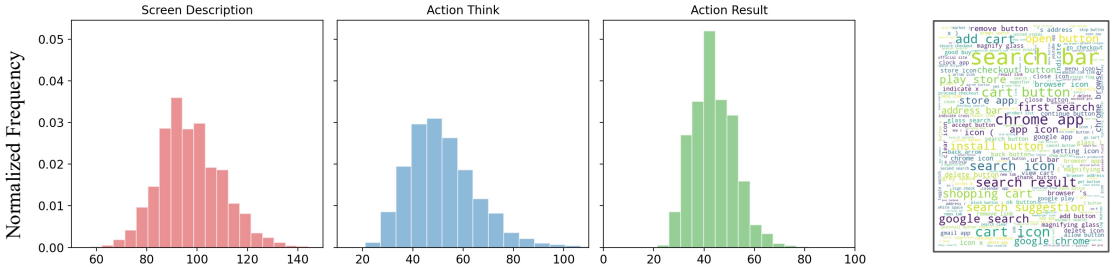
- For subset GENERAL, GOOGLEAPPS and INSTALL, as there are few unique instructions in each subset, we uniformly sample x samples for each instruction ($x = 3, 5, 3$ respectively).
- For subset WEBSHOPPING, we conduct balanced sampling on the categories of shopping websites/apps and the objects involved.
- For subset SINGLE, as the instructions are diverse and cluttered, we perform clustering and then conduct balanced sampling on the clustered data.

This results in a total number of 3461 unique instructions, corresponding to 7180 episodes. We recruit ten annotators to manually verify the correctness of the sampled episodes. Finally, for 5147 successful episodes, we randomly select one episode paired with each unique instruction.

Semantic Annotation It is crucial for GUI agents to understand the screen information and

Shopping web/app	Instruction Template	#Instructions	#Episodes
amazon	add something to the cart on amazon	80	180
	clear/empty cart, then add something to the cart on amazon	111	135
	clear/empty cart, search for something, select the first entry and add to cart on amazon	105	124
	clear cart, search for something, select the first entry, add to cart on amazon, and checkout	110	135
	show/view the shopping cart, search for something on amazon and add it to the cart	42	52
	show/view the shopping cart, add something to the cart on amazon, then checkout	59	75

Table 3: **An example of repeating instructions with the same template on WEBSHOPPING subset of AITW dataset.** We take instructions related to 'amazon' for demonstration. Similar templates can be found for samples related to other shopping websites/apps, including 'bestbuy', 'ebay', 'costco' and 'walmart'.



(a) Distribution of generated semantic annotations (b) Word Cloud of clicked elements

Figure 5: **Distributions of (a) the length of three different types of semantic annotations and (b) the phrase frequencies of clicked UI elements** on the **AITZ** dataset. The size of each word corresponds to its tf-idf score.

make decisions accordingly. To mitigate the lack of such detailed data, we leverage GPT-4V through Azure-API as the navigation expert and prompt it to do the screen description, action thinking, next action description and action result summarization tasks. Note that the amount of information used to generate semantic annotations varies. For example, the screen description is query-independent, whereas for next action description, both the query and the coordinate of golden actions are provided for reference (see Appendix A.2 for more details). Thanks to the correctness check at instruction sampling stage, the golden actions have all been verified. We then recruit three experts who have a good understanding of UI elements as annotators to examine whether the generated action description, action thinking and action result match the golden actions. Once inconsistency is found, annotators will manually revise the action descriptions, and enforce GPT-4V to regenerate the action thoughts and action results based on the correct descriptions.

3.2 Dataset Analysis

We compare our **AITZ** dataset with the most related Android GUI navigation datasets, including PixelHelp (Li et al., 2020b), MOTIF (Burns et al., 2021), UGIF (Venkatesh et al., 2022), MetaGUI (Sun et al., 2022b) and AITW (Rawles et al., 2023). Our dataset contains the same magnitude of human demonstration as these smaller datasets, but

Subset	Train		Test	
	#Episodes	#Screens	#Episodes	#Screens
GENERAL	323	2405	156	1202
INSTALL	286	2519	134	1108
GOOGLEAPPS	166	1268	76	621
SINGLE	844	2594	0	0
WEBSHOPPING	379	5133	140	1793
Total	1998	13919	506	4724

Table 4: **Detailed statistics of the training and test split of AITZ dataset.** Since SINGLE subset contains single-step tasks only, we place all SINGLE data and related episodes into the training set.

with a significantly greater richness of instructions. Table 1 demonstrates that our dataset is unique, converting rich semantic information.

In Figure 5, we provide statistics of the **AITZ** dataset, including the distribution of textual lengths and the word cloud of operated UI elements. Specifically, the majority of screen descriptions consist of 80∼120 words, while most action think have 30∼70 words. The action result exhibits a narrower range, from 20 to 80 words.

4 Experimental Setup

4.1 Baseline Models

CogAgent (Hong et al., 2023) is a LLM-based multimodal GUI agent built upon CogVLM (Wang et al., 2023b). It scales the image resolution up to 1120×1120 by fusing high-resolution features to every decoder layer with cross-attention. CogAgent

Mode	Model	Atomic									Episodic
		SCROLL	CLICK		TYPE		PRESS	STOP	Total		GP
			type	match	type	match			type	match	
ZS	CogAgent	56.41	79.90	51.50	**67.40**	**34.00**	**48.30**	4.76	65.86	44.52	13.82
	+CoAT	**70.22**	**88.23**	**66.15**	45.80	21.80	45.95	**24.60**	**72.59**	**53.28**	**17.13**
FT	AUTO-UI	**74.88**	44.37	12.72	73.00	67.80	49.09	60.12	73.79	34.46	6.59
	+CoAT	61.40	**74.56**	**32.20**	**87.80**	**81.40**	**57.70**	**74.40**	**82.98**	**47.69**	**14.51**

Table 5: **Main results of CogAgent and AUTO-UI on AITZ dataset.** ZS and FT are short for zero-shot and finetuning evaluation, respectively. For CLICK and TYPE actions, which is more complicated than the other three, we additionally report the action type prediction accuracy, marked as 'type' in this table. Total action-matching score is also included. 'GP' is short for goal progress. The best result of each model is marked in **bold**.

is pre-trained on a handful of tasks aimed to adapt it for GUI application scenarios, i.e. text recognition (Schuhmann et al., 2022), visual grounding (Li et al., 2023a), and GUI imagery (Hong et al., 2023). It is further finetuned with GUI tasks on web (Deng et al., 2023) and smartphones (Rawles et al., 2023). Since the training data for CogAgent is not publicly available, we conduct a zero-shot evaluation to assess to what extent CoAT supports the task.

AUTO-UI (Zhang and Zhang, 2023) is a specialized model for GUI navigation on AITW (Rawles et al., 2023) dataset. Screen features are extracted by the encoder from BLIP-2 (Li et al., 2023a) and fed into FLAN-Alpaca to decode actions. AUTO-UI is trained on a randomly split training set, covering 80% of AITW episodes, and evaluated on 10% randomly selected testing episodes. As AITW dataset has a large amount of repeating and problematic data, resulting in almost identical distributions between its training and test set. Therefore, we train this model from scratch on the training split of **AITZ** to validate the necessity and helpfulness of the fine-grained semantic annotations provided by **AITZ** dataset.

4.2 Evaluation Metrics

Atomic Metrics Following (Zhang and Zhang, 2023; Hong et al., 2023), we compute the screen-wise action-matching score ("match" for short). An action is correct if both the action type and the action details (i.e. scroll direction, typed text, clicked position and pressed button) match the gold ones.

Episodic Metrics As the GUI navigation is a sequential decision-making problem, it is crucial to evaluate the progress made by the agent towards the user query. Therefore, we propose to use goal progress, a metric indicating the relative position where the first error occurs in the sequence.

4.3 Implementation Details

We randomly split 70% episodes as training data, and 30% episodes as testing data (1998/506). It is notable that, as the episodes and instructions in **AITZ** are distinct, the training set and test set ensure no information leakage. The detailed statistics are in Table 4. For AUTO-UI, we adopt the same weight initialization strategies as (Zhang and Zhang, 2023) and fine-tune the models up to 10 epochs, with a learning rate of 1e-4. For CogAgent, we utilize the trained model weights from CogAgent-Chat and prompt it to use different semantic annotations. For both models, we keep the original output format unchanged but add extra information to the input or output of these models.

5 Experiments

5.1 Zero-Shot Evaluation

We perform a zero-shot evaluation to investigate the benefit of directly using these screen and action semantics as input. Here, we select CogAgent (Hong et al., 2023) for illustration as it is trained to perform GUI tasks and expected to possess generalization abilities since its foundation language model is CogVLM-7B. We verify the impact of the proposed chain-of-action thought by adding action think to the prompt input of CogAgent. As shown in Table 5, CoAT contributes significant improvements to the overall model performance. Moreover, the first and last line in Table 5 indicate the fact that fine-tuning a small agent with model size ~1B (i.e. AUTO-UI-base (Zhang and Zhang, 2023)) using CoAT can obtain comparable performance with a LLM-based agent, demonstrating the strong potential of CoAT on GUI navigation tasks.

A more detailed comparison between CogAgent and AUTO-UI on model architecture, training data and performance can be found in Appendix C.2.

	Semantic Annotations				Atomic								Episodic
	input		output		SCROLL	CLICK		TYPE		PRESS	STOP	Total		GP
	SD	PAR	AT	AD		type	match	type	match			type	match	
(1)					74.88	44.37	12.72	73.00	67.80	49.09	60.12	73.79	34.46	6.59
(2)	✓				**87.85**	49.52	20.21	81.40	64.20	53.52	49.80	80.55	39.33	10.71
(3)		✓			78.54	63.23	29.39	85.60	79.40	55.35	**79.17**	**83.91**	**48.35**	14.06
(4)	✓	✓			80.53	59.10	25.95	80.60	62.40	55.09	57.14	81.77	42.38	13.64
(5)			✓		80.87	43.09	13.16	**89.80**	78.60	46.74	25.00	73.45	32.68	9.08
(6)				✓	57.74	59.39	17.47	72.80	67.00	49.87	61.71	72.21	35.18	8.37
(7)			✓	✓	27.62	75.06	28.85	86.60	76.60	49.61	42.66	75.42	36.91	11.96
(8)	✓		✓	✓	31.28	81.29	**33.21**	79.40	61.40	51.70	35.12	77.54	37.66	13.34
(9)		✓	✓	✓	61.40	74.56	32.20	87.80	**81.40**	**57.70**	74.40	82.98	47.69	**14.51**
(10)	✓	✓	✓	✓	32.45	**82.46**	32.99	80.40	59.20	52.48	34.33	78.32	37.42	13.90

Table 6: **Ablation study of different semantic annotation components on AUTO-UI.** SD and PAR mean **s**creen **d**escription and **p**revious **a**ction **r**esult, whereas AT and AD represent **a**ction **t**hink and next **a**ction **d**escription, respectively. For CLICK and TYPE actions, which is more complicated than the other three, we additionally report the action type prediction accuracy, marked as 'type' in this table. Total action-matching score is also included. 'GP' is short for goal progress. The best result is marked in **bold** while the runner-up is underlined.

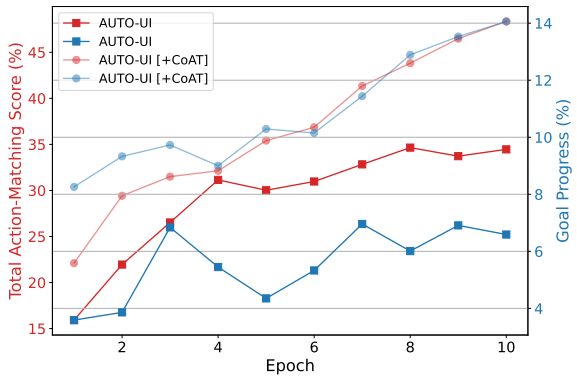


Figure 6: **Total action-matching score and goal progress over training epochs on AUTO-UI model.**

5.2 Fine-tuning Evaluation

To evaluate the influence of individual components of CoAT, we perform an ablation study by incorporating them alternately. We split the annotations into 'input' and 'output' groups, indicating where the extra information comes in during the model training. Specifically, we put screen description and previous action result as additional input information, as they do not provide direct help to the current action decision. Action think and next action description are added to the output so that the agent can learn such thinking process.

From Table 6, we observe that previous action result, especially combined with action think and action description, significantly improve the overall action prediction accuracy of AUTO-UI. As **the coherence of decision-making process is enhanced by previous action result**, there is a notable increase in the STOP action-matching score (from 60.12 to 79.17). Experiment (5)~(7) demonstrate that **learning to engage in action thinking without additional input is challenging**. However, when screen description and/or previous action result are added to the input, the performance of AUTO-UI improves immediately, especially in predicting CLICK actions. This validates the necessity and effectiveness of such semantic annotations. There is a minor decrease in both action-matching score and the goal progress when screen description is added, as seen in line (9) and (10). We attribute this to the low resolution of the visual encoder used by AUTO-UI, resulting in an inability to effectively utilize the information in screen descriptions. Figure 6 further illustrates the improvement in training efficiency when trained with our **AITZ** data.

5.3 Qualitative Analysis

We conduct the thorough analysis on wrong cases, as shown in Figure 7. AUTO-UI struggles with correctly judging the task execution progress, as the action history provided as a series of action types and coordinates is hard to understand. Previous action result mitigates this problem by explicitly describing the result of the previous actions in words. This highlights the importance of safeguarding the coherence of action decision by establishing connections between two time steps. For CogAgent, we carefully inspect its output, which is composed of three parts: action plan, next action and grounded operation. It seems that CogAgent does not take historical information into account,

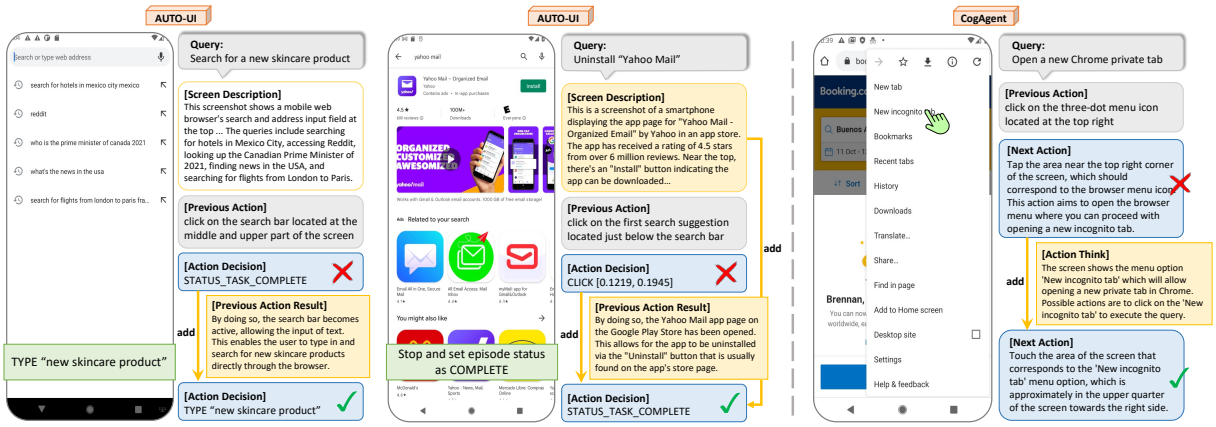


Figure 7: **Qualitative examples for AUTO-UI and CogAgent.** This figure presents qualitative results where different types of errors are corrected by applying additional semantic annotations (yellow shadowed boxes).

as its predictions at each step only consider the current information, leading to repetitive and ineffective actions. For example, as the corresponding action plan generated by CogAgent is to "1. Open Browser Menu, 2. Select 'New Incognito Tab' from the dropdown menu", it repeatedly attempts to open the menu icon (see the right-side case in Figure 7). Adding a short-cut action chain-of-thought, i.e. action think from **AITZ** dataset, into the model input helps to alleviate this issue.

5.4 Generalization Evaluation

Generalization abilities are crucial for GUI agents. Previous experiments in this paper are actually a reflection of the generalization ability over unseen instructions, as we put emphasis on separation based on instructions (see Section 3.1 and Section 4.3). The generalization over unseen apps is another important perspective. Hence, we re-partition the dataset based on the separation of apps, resulting in a train split (1519 episodes) spanning 63 apps and a test split (459 episodes) spanning 10 apps. We follow the implementation details in Section 4.3 and the results are shown in Table 7. By adding CoAT-driven data during training, the agent could generalize to unseen apps better (9.4% v.s. 5.1% on the episodic goal progress). This demonstrates that CoAT is generalizable and helpful for action decision-making on unseen apps.

6 Related Works

GUI Navigation AUTOmatic execution of user instructions on smartphones or websites is an advanced task, as it requires the agent to not only perceive but also deduce. Previous works concentrate on evaluating the ability of models to identify different UI elements (Shi et al., 2017; Zhang et al., 2021; Sunkara et al., 2022), and to fulfil a user-queried task by either statically operating on a series of pre-collected GUI screenshots (Li et al., 2020b; Venkatesh et al., 2022; Zhang and Zhang, 2023; Deng et al., 2023) or dynamically interacting with an alive Android device (Yang et al., 2023a). However, these works separate the ability of element recognition and action inference, causing a discrepancy between the user intent and the performed actions (Wei et al., 2022; Baechler et al., 2024). Our CoAT framework bridges this gap by allowing GUI agents to recall history actions, perceive the current screen, and decide on the future actions based on these useful semantics.

Large Multimodal Models (LMM) Recent years have witnessed the rise of numerous large multimodal models (Liu et al., 2023a,b; Zhu et al., 2023; Zeng et al., 2023). Usually, visual signals are encoded by a vision transformer (Dosovitskiy et al., 2020) and further incorporated in LLMs (Radford et al., 2021) through linear projection (Tsimpoukelli et al., 2021), Q-former (Li et al., 2023a) or cross-attention layers (Alayrac et al., 2022). For general purpose LMMs, the low resolution of visual encoders (224×224) captures only coarse visual information. CogAgent (Hong et al., 2023) deals with this problem by using the original ViT-L (Dosovitskiy et al., 2020) to encode high-resolution visual features up to 1120×1120, and fusing them with every decoder layers through cross-attention. Whereas Monkey (Li et al., 2023b) equips the visual encoder from QWen-VL (Bai et al., 2023) with individual LoRA adapter (Hu et al., 2021) for each patch to scale the image resolution up to 896×1344 pixels. Consequent

Model	Action-Matching Score						Goal Progress
	TOTAL	CLICK	TYPE	PRESS	STOP	SCROLL	
AUTO-UI	28.5	10.7	59.2	27.6	41.1	69.7	5.1
AUTO-UI + CoAT	31.8	19.7	61.2	49.1	55.2	74.9	9.4

Table 7: **Generalization results over the unseen apps under fine-tuning settings.**

works (Yu et al., 2024; Chen et al., 2024; Lu et al., 2024a) all incorporate high-resolution image encoders, indicating a popular trend for the future.

LMM as GUI Agents A number of works have utilized LMMs' domain knowledge and emergent zero-shot embodied abilities to perform complex task planning and reasoning (Yang et al., 2023b; Wang et al., 2023c; Ikeuchi et al., 2023). For GUI navigation, the introduction of LMMs surpasses previous works that transform the UI layouts and elements into the text-only HTML format (Li et al., 2020a; Zhang et al., 2021; Wang et al., 2023a). One line of work adopts GPT-4V directly as the GUI agent and prompts it to perform the task (Yan et al., 2023; Yang et al., 2023a; Zheng et al., 2024), while other methods focus on tuning a smaller LMM on GUI-related datasets to acquire the domain-specific knowledge (Zhang and Zhang, 2023), or train a LMM from scratch on GUI-specified pre-training tasks (Hong et al., 2023; Baechler et al., 2024; You et al., 2024; Cheng et al., 2024). We evaluate two agents on the proposed **AITZ** dataset, and prove that our proposed chain-of-action-thought helps agents adapt to GUI tasks better and more quickly.

7 Conclusion

In conclusion, our work aims to bolster the navigation ability of LMM-based GUI agents. We propose Chain-of-Action-Thought (**CoAT**) by analyzing human orienteering processes. We start by verifying that CoAT is superior to three typical context modeling methods. In order to inject CoAT-like thinking capabilities into existing GUI agents, we further generated a set of high-quality CoAT-driven data through cooperation between human experts and GPT-4V, namely Android-In-The-Zoo (**AITZ**) dataset. **AITZ** enriches this field with a robust dataset that bridges perception and cognition, facilitating effective training and reliable evaluation for GUI navigation agents. Experiments demonstrate the efficiency and usefulness of proposed chain-of-action-thought paradigm.

8 Limitations

We developed **CoAT** and **AITZ** with the goal of enabling LLM Agents to mimic the cognitive processes of humans. Although our experiments proved that it is possible to stimulate the reasoning ability of language models (i.e. GPT-4V (OpenAI, 2023), CogAgent (Hong et al., 2023) and AUTO-UI (Zhang and Zhang, 2023)) in GUI scenarios through zero-shot prompting or fine-tuning, the different model structure and training data used by current specified models for GUI tasks make the comparison less intuitive. To what extent the image resolution and GUI-related pretraining tasks (i.e. text recognition, GUI imagery (Hong et al., 2023), screen question-answering (Baechler et al., 2024; You et al., 2024) and GUI grounding (Cheng et al., 2024)) influence the navigation performance remains under-explored. We leave it for future work to precisely measure the impact of image resolution, text recognition ability, GUI grounding ability of LMMs on GUI navigation tasks.

9 Ethics

Android-In-The-Zoo (**AITZ**) dataset is sourced from open-source datasets AITW (Rawles et al., 2023), which is permitted for academic use. During our data collection, specifically, during the instruction-episode correctness checks, we ensured that privacy concerns were addressed, and the sampled data does not include any real personal information (fake or meaningless data are allowed). Since **AITZ** dataset contains only semantic annotations on smartphone operations, the use of this data poses neither ethical risks nor harmful guidance.

10 Acknowledgements

This work is supported by National Natural Science Foundation of China (No. 62176058) and National Key R&D Program of China (2023YFF1204800). The project's computational resources are supported by CFFF platform of Fudan University.

References

Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katherine Millican, Malcolm Reynolds, et al. 2022. Flamingo: a visual language model for few-shot learning. *Advances in Neural Information Processing Systems*, 35:23716–23736.

Gilles Baechler, Srinivas Sunkara, Maria Wang, Fedir Zubach, Hassan Mansoor, Vincent Etter, Victor Cărbune, Jason Lin, Jindong Chen, and Abhanshu Sharma. 2024. Screenai: A vision-language model for ui and infographics understanding. *arXiv preprint arXiv:2402.04615*.

Jinze Bai, Shuai Bai, Shusheng Yang, Shijie Wang, Sinan Tan, Peng Wang, Junyang Lin, Chang Zhou, and Jingren Zhou. 2023. Qwen-vl: A frontier large vision-language model with versatile abilities. *arXiv preprint arXiv:2308.12966*.

Andrea Burns, Deniz Arsan, Sanjna Agrawal, Ranjitha Kumar, Kate Saenko, and Bryan A Plummer. 2021. Mobile app tasks with iterative feedback (motif): Addressing task feasibility in interactive visual environments. *arXiv preprint arXiv:2104.08560*.

Zhe Chen, Weiyun Wang, Hao Tian, Shenglong Ye, Zhangwei Gao, Erfei Cui, Wenwen Tong, Kongzhi Hu, Jiapeng Luo, Zheng Ma, et al. 2024. How far are we to gpt-4v? closing the gap to commercial multimodal models with open-source suites. *arXiv preprint arXiv:2404.16821*.

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

Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Samuel Stevens, Boshi Wang, Huan Sun, and Yu Su. 2023. Mind2web: Towards a generalist agent for the web. *arXiv preprint arXiv:2306.06070*.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. 2020. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*.

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

Edward J Hu, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. 2021. Lora: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Katsushi Ikeuchi, Jun Takamatsu, Kazuhiro Sasabuchi, Naoki Wake, and Atsushi Kanehiro. 2023. Applying learning-from-observation to household service robots: three common-sense formulation. *arXiv preprint arXiv:2304.09966*.

Sunjae Lee, Junyoung Choi, Jungjae Lee, Hojun Choi, Steven Y Ko, Sangeun Oh, and Insik Shin. 2023. Explore, select, derive, and recall: Augmenting llm with human-like memory for mobile task automation. *arXiv preprint arXiv:2312.03003*.

Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. 2023a. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. *arXiv preprint arXiv:2301.12597*.

Toby Jia-Jun Li, Tom Mitchell, and Brad Myers. 2020a. Interactive task learning from gui-grounded natural language instructions and demonstrations. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 215–223.

Yang Li, Jiacong He, Xin Zhou, Yuan Zhang, and Jason Baldridge. 2020b. Mapping natural language instructions to mobile ui action sequences. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8198–8210.

Zhang Li, Biao Yang, Qiang Liu, Zhiyin Ma, Shuo Zhang, Jingxu Yang, Yabo Sun, Yuliang Liu, and Xiang Bai. 2023b. Monkey: Image resolution and text label are important things for large multi-modal models. *arXiv preprint arXiv:2311.06607*.

Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee. 2023a. Improved baselines with visual instruction tuning. *arXiv preprint arXiv:2310.03744*.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. 2023b. Visual instruction tuning. *arXiv preprint arXiv:2304.08485*.

Thomas F Liu, Mark Craft, Jason Situ, Ersin Yumer, Radomir Mech, and Ranjitha Kumar. 2018. Learning design semantics for mobile apps. In *Proceedings of the 31st Annual ACM Symposium on User Interface Software and Technology*, pages 569–579.

Haoyu Lu, Wen Liu, Bo Zhang, Bingxuan Wang, Kai Dong, Bo Liu, Jingxiang Sun, Tongzheng Ren, Zhuoshu Li, Yaofeng Sun, et al. 2024a. Deepseek-vl: towards real-world vision-language understanding. *arXiv preprint arXiv:2403.05525*.

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

Amal Nanavati, Vinitha Ranganeni, and Maya Cakmak. 2023. Physically assistive robots: A systematic review of mobile and manipulator robots that physically assist people with disabilities. *Annual Review of Control, Robotics, and Autonomous Systems*, 7.

OpenAI. 2023. Gpt-4 technical report. *arXiv:2303.08774*.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. 2021. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR.

Christopher Rawles, Alice Li, Daniel Rodriguez, Oriana Riva, and Timothy Lillicrap. 2023. Android in the wild: A large-scale dataset for android device control. *arXiv preprint arXiv:2307.10088*.

Christoph Schuhmann, Romain Beaumont, Richard Vencu, Cade Gordon, Ross Wightman, Mehdi Cherti, Theo Coombes, Aarush Katta, Clayton Mullis, Mitchell Wortsman, Patrick Schramowski, Srivatsa Kundurthy, Katherine Crowson, Ludwig Schmidt, Robert Kaczmarczyk, and Jenia Jitsev. 2022. Laion-5b: An open large-scale dataset for training next generation image-text models. *ArXiv*, abs/2210.08402.

Tianlin Shi, Andrej Karpathy, Linxi Fan, Jonathan Hernandez, and Percy Liang. 2017. World of bits: An open-domain platform for web-based agents. In *International Conference on Machine Learning*, pages 3135–3144. PMLR.

Liangtai Sun, Xingyu Chen, Lu Chen, Tianle Dai, Zichen Zhu, and Kai Yu. 2022a. Meta-gui: Towards multi-modal conversational agents on mobile gui. In *Conference on Empirical Methods in Natural Language Processing*.

Liangtai Sun, Xingyu Chen, Lu Chen, Tianle Dai, Zichen Zhu, and Kai Yu. 2022b. Meta-gui: Towards multi-modal conversational agents on mobile gui. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 6699–6712.

Srinivas Sunkara, Maria Wang, Lijuan Liu, Gilles Baechler, Yu-Chung Hsiao, Abhanshu Sharma, James Stout, et al. 2022. Towards better semantic understanding of mobile interfaces. *arXiv preprint arXiv:2210.02663*.

Gemini Team, Rohan Anil, Sebastian Borgeaud, Yonghui Wu, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, et al. 2023. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*.

Maria Tsimpoukelli, Jacob L Menick, Serkan Cabi, SM Eslami, Oriol Vinyals, and Felix Hill. 2021. Multimodal few-shot learning with frozen language models. *Advances in Neural Information Processing Systems*, 34:200–212.

Sagar Gubbi Venkatesh, Partha Talukdar, and Srini Narayanan. 2022. Ugif: Ui grounded instruction following. *arXiv preprint arXiv:2211.07615*.

Bryan Wang, Gang Li, and Yang Li. 2023a. Enabling conversational interaction with mobile ui using large language models. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, pages 1–17.

Bryan Wang, Gang Li, Xin Zhou, Zhourong Chen, Tovi Grossman, and Yang Li. 2021. Screen2words: Automatic mobile ui summarization with multimodal learning. In *The 34th Annual ACM Symposium on User Interface Software and Technology*, pages 498–510.

Weihan Wang, Qingsong Lv, Wenmeng Yu, Wenyi Hong, Ji Qi, Yan Wang, Junhui Ji, Zhuoyi Yang, Lei Zhao, Xixuan Song, Jiazheng Xu, Bin Xu, Juanzi Li, Yuxiao Dong, Ming Ding, and Jie Tang. 2023b. Cogvlm: Visual expert for pretrained language models. *ArXiv*, abs/2311.03079.

Zihao Wang, Shaofei Cai, Guanzhou Chen, Anji Liu, Xiaojian Ma, and Yitao Liang. 2023c. Describe, explain, plan and select: Interactive planning with large language models enables open-world multi-task agents. *arXiv preprint arXiv:2302.01560*.

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.

Hao Wen, Yuanchun Li, Guohong Liu, Shanhui Zhao, Tao Yu, Toby Jia-Jun Li, Shiqi Jiang, Yunhao Liu, Yaqin Zhang, and Yunxin Liu. 2023. Empowering llm to use smartphone for intelligent task automation. *arXiv preprint arXiv:2308.15272*.

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*.

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

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

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*.

Ya-Qi Yu, Minghui Liao, Jihao Wu, Yongxin Liao, Xiaoyu Zheng, and Wei Zeng. 2024. Texthawk: Exploring efficient fine-grained perception of multimodal large language models. *arXiv preprint arXiv:2404.09204*.

Yan Zeng, Hanbo Zhang, Jiani Zheng, Jiangnan Xia, Guoqiang Wei, Yang Wei, Yuchen Zhang, and Tao Kong. 2023. What matters in training a gpt4-style language model with multimodal inputs? *arXiv preprint arXiv:2307.02469*.

Chaoyun Zhang, Liqun Li, Shilin He, Xu Zhang, Bo Qiao, Si Qin, Minghua Ma, Yu Kang, Qingwei Lin, Saravan Rajmohan, et al. 2024a. Ufo: A ui-focused agent for windows os interaction. *arXiv preprint arXiv:2402.07939*.

Jiwen Zhang, Yaqi Yu, Minghui Liao, Wentao Li, Jihao Wu, and Zhongyu Wei. 2024b. Ui-hawk: Unleashing the screen stream understanding for gui agents. *Preprints*.

Xiaoyi Zhang, Lilian de Greef, Amanda Swearngin, Samuel White, Kyle Murray, Lisa Yu, Qi Shan, Jeffrey Nichols, Jason Wu, Chris Fleizach, et al. 2021. Screen recognition: Creating accessibility metadata for mobile applications from pixels. association for computing machinery, new york, ny, usa.

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, Hai Zhao, George Karypis, and Alex Smola. 2023. Multimodal chain-of-thought reasoning in language models. *arXiv preprint arXiv:2302.00923*.

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

Deyao Zhu, Jun Chen, Xiaoqian Shen, Xiang Li, and Mohamed Elhoseiny. 2023. Minigpt-4: Enhancing vision-language understanding with advanced large language models. *arXiv preprint arXiv:2304.10592*.

A Data Collection

Our data construction pipeline is shown in Figure 3. We leverage the strong world knowledge and generation ability of GPT-4V, combined with critical human verification, to ensure high-quality data. We ask our annotators to detect only factual errors, which could hardly introduce human bias.

A.1 Instruction Sampling Process

We first checked the instruction distribution in the original AITW dataset based on the split from (Zhang and Zhang, 2023). We find that the instruction distribution in the training, validation and test sets are almost the same, which means there is a serious problem of data leakage. To avoid such problem exists in our constructed datasets, we perform instruction sampling.

- SINGLE: Given the complexity and variety of instructions in this dataset, we first clustered them and then performed balanced sampling based on the categories. The clustering process is as follows: (1) Identify the main verb in each instruction, typically the first word, and group the instructions by this verb. (2) For each group of instructions, we manually classify those with fewer than 50 samples that show clear patterns. Then for groups with more than 50 samples, use tf-idf for clustering. Finally, we manually verify the clustering results.
- WEB_SHOPPING: We performed balanced sampling based on the types of shopping websites and the objects involved.
- GENERAL, INSTALL, GOOGLE_APPS: Since these three datasets have a limited number of instructions, we did not perform extensive filtering during sampling. Instead, we uniformly sampled x instructions per user. For INSTALL and GENERAL, x=3; for GOOGLE_APPS, x=5.

During the instruction sampling stage, we recruited 10 annotators to verify whether the episodes have successfully completed the tasks required by the instructions. Our data quality inspection team conducted a secondary validation of sampled results.

A.2 Semantic Annotation Process

We leverage GPT-4V through Amazon Azure-API as the navigation expert and prompt it to do the following generation tasks:

1. **Screen Description**: describe the main content of the given screenshots, including the screen type, and primary apps or widgets presented.
2. **Action Grounding**: given the coordinates of the correct next actions, generate action descriptions. Specifically, we simplify the action spaces into 5 action categories, including SCROLL(direction), TYPE(text), PRESS(button), CLICK(point) and STOP(task state). We ask GPT-4V to describe the UI element the click action is operating on, by drawing the bounding box of the clicked area through icon detection model from (Liu et al., 2018). The descriptions for other types of actions are generated using templates.
3. **Action Thinking**: think about what actions need to be performed on the current screen to complete the user query, and describe the results of the correct next action based on screenshots before and after the action.

PROMPT	
	Screen Description
	I will give you a screenshot of a mobile phone. **SCREEN**: __screen_img__ **TASK**: Your task is to summarize this screen about its main content and its functionality, i.e. the type of the screen, together with primary icons or apps on the screen. You should describe the screen with necessary details, but not too long. Your output must be less than five sentences.
	Action Think
	QUERY: The query you need to complete is to __query__ **ACTION HISTORY**: To proceed with the query, your past actions include: __action_history__ **SCREEN**: __screen_img__ **SCREEN DESCRIPTION**: __screen_desc__ **TASK**: Given the screen and above information, you have two tasks to do. Firstly, based on the history and the current screen, you should estimate the execution progress of query in one sentence. Answer with format: 'Reflection: ...' Secondly, you should analyze the screen for relevant details that might pertain to the given query. This includes checking for specific applications, icons, or buttons that are visible, and any information or results that are currently displayed on the screen. Then, describe possible actions you may conduct. You must answer by two sentences with the format: 'Think: ... Possible actions are ...'"
	Action Description
	To fulfill the following query, an expert have clicked on the screen. **QUERY**: __query__ **SCREEN**: __screen_img__ The screen labeled with expert action is also given to you. **SCREEN WITH ACTION**: __labelled_screen__ The expert action `__expert_action__` is labelled as a blue cross marker `+` on this screen. **EXPERT ACTION**: `__expert_action__ **SCREEN COORDINATE SYSTEM**: A coordinate (x, y) represents a point on the screen. The first value, labeled as `x`, horizontal, i.e. x ranges from 0 to 1, meaning the position of point ranges from the left to right, where x<0.4 means left, 0.4<=x<=0.6 means middle and x>0.6 means right. The second value, labeled as `y`, is vertical, i.e. y ranges from 0 to 1, meaning the position of point ranges from the bottom to top. where y<0.2 means bottom, 0.2<=y<0.4 means lower, + 0.4<=y<0.5 means lower middle, 0.5<=y<=0.6 means upper middle, 0.6<y<=0.8 means upper, and y>0.8 means top. **TASK**: Based on above information, your task is to answer: Which UI element (icon, app, search bar, results, etc) is this expert action clicking on and where is it located? You should think step by step as follows: The coordinate in expert action is ___ . As stated, the first value `x` is ___, which means the click point locates at ___. The second value `y` is ___, which means the click point locates at ___. Overall, the click point locates at the ___ and ___ part of the screen. Combined with the blue cross marker `+`, answer with format: the expert action is to click on the ____ located at ____ .
	Action Result
	To fulfill the following query, you have performed an action on screen. **QUERY**: __query__ **ACTION**: __correct_next_action__ The screenshots before and after the action are: **SCREEN BEFORE ACTION**: __screen_before__ **SCREEN AFTER ACTION**: __screen_after__ **TASK**: Your task is to explain why this action can facilitate the completion of query. Answer with format: `By doing so, ...` Your output must be within two sentences, one sentence about the consequences and one sentence about the reason. Keep your answer as concise and brief as possible.

Figure 8: Prompt to generate candidate answers for four types of semantic annotations.

Three experts who have a good understanding of UI elements are recruited as annotators to verify whether the generated action description matches the labelled golden actions and the generated action thinkings. Once inconsistency is found, annotators will manually revise the action descriptions, and enforce GPT-4V to regenerate the action thoughts based on the correct action descriptions. The prompt we use are shown in Figure 8.

A.3 Action Space

As stated before in Appendix A.2, we simplify the action spaces into 5 action categories. The reason behind this is, we observe that within the AITW dataset, 'DUAL_POINT' action type seamlessly covers both 'CLICK' and 'SCROLL' actions. In most cases, the action point of 'SCROLL' action conveys little information, but the scroll direction matters. There are also few operations that require dragging apps, such as editing the main screen. Therefore, we manually split the 'DUAL_POINT' action type into 'CLICK' and 'SCROLL', where 'CLICK' action involves coordinate prediction and 'SCROLL' action is purely textual. The action space is summarized as follows:

- **`CLICK(coord_y: float, coord_x: float)`**: This action clicks a specific point on the screen. It is necessary to combine the annotation of UI elements to identify the icon and/or area clicked. Note that we use the relative pixel coordinate system, where (0, 0) means the top-left and (1,1) means the bottom right corner of the screen. For example, `click (0.11, 0.92)` taps a point located at the top-right corner of the screen.
- **`SCROLL(direction: str)`**: This actions means the finger movements like a real human user. For example, `scroll up` means the action gesture is from bottom to top, leading either the app drawer to be opened, or the current screen to go down and reveal more contents. There are four options for direction: `up`, `down`, `left` and `right`.
- **`TYPE(text: str)`**: This action allow the agent to directly type texts into an input field, skipping the inefficient keyboard operations. For example, `type "what is CoAT"` inputs the string "what is CoAT" to the text input field at one time.

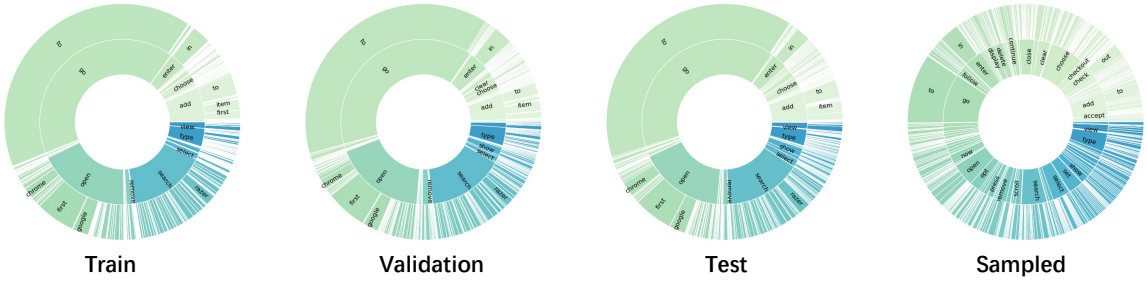


Figure 9: The instruction distribution (grouped by verbs and nouns) for SINGLE subset in original AITW dataset.

	Model Architecture			Training Data	
	Visual Encoder	Language Backbone	Image Resolution	Pre-training	Fine-tuning
AUTO-UI (1.2B)	Single Encoder (985M) BLIP2-opt-2.7b	FLAN-alphaca -base(200M)	224 x 224	/	AITW / **AITZ**
CogAgent (18B)	Dual Encoder (11B) Low-Res: EVA2-CLIP-E High-Res: EVA2-CLIP-L	CogVLM-7B	1120 x 1120	276M data spanning over text recognition, visual grounding and gui imagery tasks	1M data, including Mind2Web, AITW, public VQA data ...

Table 8: **Comparison between AUTO-UI (Zhang and Zhang, 2023) and CogAgent (Hong et al., 2023).** Note that, for low-resolution images, following CogVLM (Wang et al., 2023b), CogAgent adopts a visual encoder with 5B parameters and a visual expert module with 6B parameters.

- **`PRESS(button: str)`**: The Android system provides several system level shortcut buttons, such as `back button` that enables the user back to the previous interface, and `home button` that allows a direct return to the home screen. Moreover, `enter button` is another virtual button that submits the typed query. This action means to press on one of the system level virtual buttons.
- **`STOP(task_state: str)`**: This action allows the agent to stop and end the query execution in time, either when it considers the task is completed or the task is impossible. For example, `stop and set the query as completed` means the user query has been successfully completed.

We map the actions predicted by AUTO-UI and CogAgent to this space to ensure the reliability and consistency in comparison.

B Dataset Details

Since **AITZ** is built upon AITW, it inherits the dataset structure that contains five subsets, 4 of which are multi-step tasks (GENERAL, GOOGLEAPPs, INSTALL, WEBSHOPPING) and 1 is single-step tasks (SINGLE).

- GENERAL: Tasks including question-and-answering (i.e. "What is the capital of ...?") and interacting with 3rd party apps/websites (i.e. "Install/Open the xxx app"). Therefore, this subset has 24 apps in total, within which are google chrome(72%), google maps (9%), google play store (5%), clock (4%), settings (2%) and others (8%). The training split has 19 apps and the testing split has 17 apps, where 12 apps are shared across the training and testing split.
- INSTALL: High-level tasks related to installing, uninstalling and logging into apps. This subset invloves 79 apps in total via the entrance of google play store. The training split has 77 apps and the testing split has 71 apps, where 69 apps are shared across the training and testing split.
- WEBSHOPPING: Tasks related to shopping on e-commerce websites, including ebay (17%), amazon (17%), bestbuy (15%), walmart (13%), newegg (10%), target (10%), costco (9%), lowes (2%) and others (7%). As we have stated in Section 3.1, the apps involved in webshopping are relatively fixed, so different instructions are more crucial for distinguishing different scenarios. This is the reason why we have done instruction sampling to separate different instructions.
- GOOGLEAPPS: Tasks that involve the use of 14 Google applications, including settings (25%), google chrome (22%), google play store (15%), gmail (14%), google maps (6%), calendar (6%),

Model	Prompt	UI Reps.	Hit Rate	Total	CLICK	SCROLL	PRESS	TYPE	STOP
QWen-VL	CoA	txt	82.53	35.86	44.96	34.21	0	34.04	4.08
		tag	94.48	44.37	60.07	7.89	0	48.94	0
	CoT	txt	84.37	41.61	56.83	2.63	4.35	40.43	4.08
		tag	95.63	49.43	69.42	2.63	4.35	40.43	2.04
	CoAT	txt	94.02	52.41	72.3	7.89	13.04	34.04	10.2
		tag	96.32	51.95	70.5	2.63	8.7	46.81	10.2
Gemini-PV	CoA	txt	89.43	42.99	60.79	13.16	4.35	21.28	4.08
		tag	99.77	54.48	79.86	10.53	13.04	10.64	6.12
	CoT	txt	95.86	49.2	67.27	26.32	21.74	19.15	6.12
		tag	97.47	51.95	74.46	21.05	13.04	12.77	4.08
	CoAT	txt	97.01	52.41	69.42	23.68	30.43	34.04	6.12
		tag	95.4	53.33	72.66	23.68	21.74	29.79	4.08
GPT-4V	CoA	txt	92.41	55.17	74.1	42.11	39.13	8.51	10.2
		tag	99.31	62.76	86.69	44.74	26.09	14.89	4.08
	CoT	txt	98.16	66.21	89.57	39.47	39.13	12.77	18.37
		tag	97.01	64.14	86.33	39.47	39.13	21.28	10.2
	CoAT	txt	98.39	71.72	86.33	47.37	43.48	48.94	42.86
		tag	98.16	71.49	86.69	42.11	43.48	57.45	34.69

Table 9: **Complete comparison results of three prompting methods** on Qwen-VL-Max, Gemini-1.0-Pro-Vision and GPT-4V. "Prompt" means different prompting methods. "UI Reps." denotes the representation methods of screen elements, including set-of-mark tagging (tag) and textual representation (txt). "Hit Rate" means the format hit rate. The evaluation metric is the action prediction accuracy(%).

clock (5%), google photos (3%) and others (4%). The training split spans 14 apps and the testing split spans 10 apps.

- SINGLE: Single-step tasks that mainly come from WebShopping, spanning about 10 apps. Only used for training.

C Experiment Details

C.1 Comparison between Prompting Methods

In Section 2.2 we conducted a preliminary experiment to demonstrate that CoAT is more effective than previous context modeling methods. Specifically, for CoA prompting, the input to GUI agents includes system prompt, current screenshot, history actions and user request. For CoT prompting, the input to GUI agents includes system prompt, current screenshot and user request. For CoAT promprting, we firstly require the agent to observe current screenshot and generate screen descriptions. Then, the input contains system prompt, current screenshot, screen description, history actions, previous action results and user request.

For all threee prompting methods, the system prompt contains information about the valid action space and corresponding desired output format. If the representation of UI elements is set-of-mark tagging, another screenshot with annotated UI elements will be additionally added to the input, otherwise a textual representation of UI elements is appended. Figure 10 show a visualization example of these two UI representations.

The complete experiment results are shown in Table 9. From Table 9, GPT-4V prompted by CoAT takes the lead position in the overall performance and in the prediction of each type of actions. Compared with plain textual representations, agents equipped with set-of-mark tagging generally performs better. This encourages future work to put more emphasis on the visual perception of UI elements, improve the image resolution and multi-image processing ability of GUI agents.

C.2 Comparison between Baselines

As shown in Table 5, we conclude that "AUTO-UI + CoAT is on par with CogAgent-Chat-18B" based on the fact that the model architecture and training data of AUTO-UI is inferior to CogAgent, but after

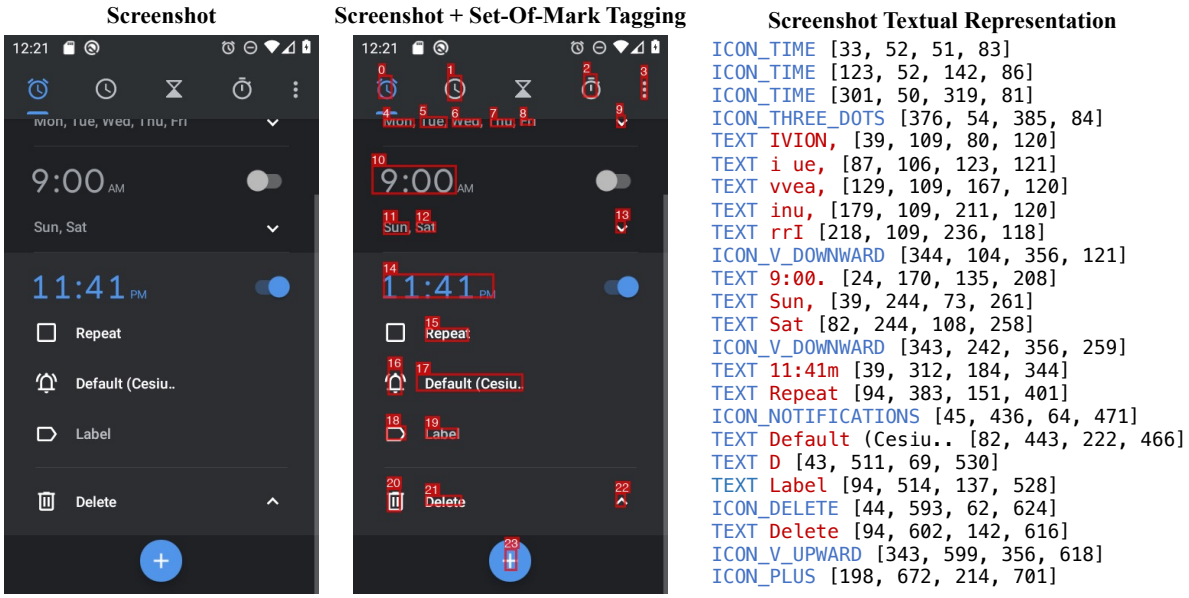


Figure 10: **Visualization of Set-Of-Mark tagging and corresponding textual representations.**

fine-tuning on **AITZ** dataset, they achieve similar performance on goal process (AUTO-UI is even slightly higher.) We summarize the differences between two models in Table 8 for a quick look. Following is the detailed explanation:

1. The lesser volume of training data used by AUTO-UI compared with CogAgent. Specifically, AUTO-UI underwent fine-tuning solely on the **AITZ** dataset, in contrast to CogAgent's extensive fine-tuning across the entire AITW dataset. Moreover, CogAgent introduced GUI imagery tasks during the pre-training phase. Hence, it is highly optimized for GUI scenarios.
2. The different resolution of visual encoders. Specifically, AUTO-UI employs the visual encoder from BLIP2 with a 224 x 224 resolution, whereas CogAgent combines ViT-L with the visual encoder from CogVLM to scale the resolution up to 1120 x 1120.
3. Despite AUTO-UI + CoAT being trained with significantly less data and without any additional pre-training efforts, it managed to outperform CogAgent in terms of action prediction accuracy and goal progress, underscoring the effectiveness and value of our proposed method and dataset, as shown in Table 5.

D Discussions About Screen Description

As we have stated in Section 5.2, the image resolution that LMMs can handle is crucial for understanding the screen description. Our statement is supported by two experiments:

(1) An exploration experiment on Monkey (Li et al., 2023b), with screen description as additional input: Monkey is a large multimodal model that could process images with resolutions up to 1344x896. We ablate the usage of screen description, and train the model to output the action think together with the action decision for 2 epochs. The total action matching score rises from 22.7% to 26.3%. (2) A validation experiment on UI-Hawk (Zhang et al., 2024b), with screen description as learning target. UI-Hawk is a specialized version of TextHawk (Yu et al., 2024) for UI understanding and we have observed similar improvements. For UI-Hawk, we integrate the learning of screen description by separating the training process into two stages. During stage one, the model learns to describe the screen. During stage two, the model learns to decide on its next-step action on a more complicated navigation dataset, GUI-Odyssey (Lu et al., 2024b). The total action matching score rises from 69% towards 72% by adding the stage one training process.

We leave it for future work to conduct a thorough analysis on the influencing factors of screen description, such as image resolution, model architecture, UI related pre-training, etc.