ScreenAgent: A Vision Language Model-driven Computer Control Agent

Runliang Niu[1] , **Jindong Li**[1] , **Shiqi Wang**[1] , **Yali Fu**[1] , **Xiyu Hu**[1] ,
Xueyuan Leng[1] , **He Kong**[1] , **Yi Chang**[1,2] , **Qi Wang**[1,2*]
[1] School of Artificial Intelligence, Jilin University
[2] Engineering Research Center of Knowledge-Driven Human-Machine Intelligence,
Ministry of Education, China
niurl19@mails.jlu.edu.cn, qiwang@jlu.edu.cn

Abstract

Large Language Models (LLM) can invoke a variety of tools and APIs to complete complex tasks. The computer, as the most powerful and universal tool, could potentially be controlled by a trained LLM agent. Powered by the computer, we can hopefully build a more generalized agent to assist humans in various daily digital works. In this paper, we construct an environment for a Vision Language Model (VLM) agent to interact with a real computer screen. Within this environment, the agent can observe screenshots and manipulate the Graphical User Interface (GUI) by outputting mouse and keyboard actions. We also design an automated control pipeline that includes planning, acting, and reflecting phases, guiding the agent to continuously interact with the environment and complete multi-step tasks. Additionally, we construct the ScreenAgent Dataset, which collects screenshots and action sequences when completing daily computer tasks. Finally, we train a model, ScreenAgent, which achieves comparable computer control capabilities to GPT-4V and demonstrated more precise UI positioning capabilities. Our attempts could inspire further research on building a generalist LLM agent. The code and more detailed information are at https://github.com/niuzaisheng/ScreenAgent.

1 Introduction

Large Language Models (LLM), such as ChatGPT and GPT-4, have recently demonstrated exceptional performance in natural language processing tasks like generation, understanding, and dialogue. They have also significantly revolutionized research in other artificial intelligence fields. In particular, the development of these technologies paves the way for the study of intelligent LLM agents [Wang *et al.*, 2023b]. An LLM agent is an AI entity with a large language model as its core computational engine. It possesses capabilities like Perception, Cognition, Memory, and Action, enabling the agent to perform highly proactive autonomous behaviors [Wang *et al.*, 2023c]. In LLM agent-related research,

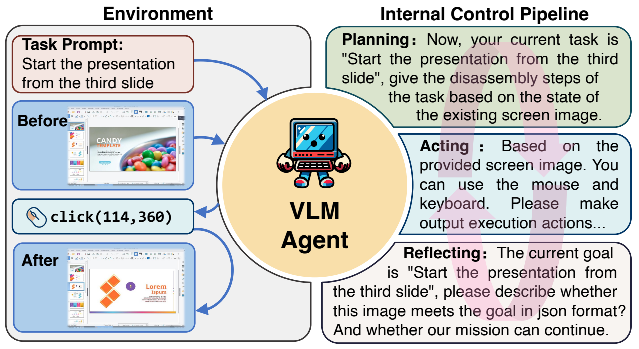

Environment
Internal Control Pipeline
Task Prompt:
Start the presentation from the third slide
Before
click(114,360)
After
VLM Agent
Planning: Now, your current task is "Start the presentation from the third slide", give the disassembly steps of the task based on the state of the existing screen image.
Acting : Based on the provided screen image. You can use the mouse and keyboard. Please make output execution actions...
Reflecting: The current goal is "Start the presentation from the third slide", please describe whether this image meets the goal in json format? And whether our mission can continue.

Figure 1: We construct a realistic computer-controlling environment and design a control pipeline for the agent. The VLM agent retrieves instruction prompts and real computer states from the environment, then runs its internal control pipeline, going through the planning, acting, and reflecting phases. It outputs the next action operation, utilizes function calls to perform actions, induces changes in the computer environment, and achieves genuine real-time interaction between the agent and the environment.

how to enable agents to learn to effectively use tools for expanding their action space has drawn extensive attention.

With the growing prevalence of electronic devices, such as personal computers, smartphones, and smart electronic instruments, our lives are becoming more intertwined with the digital world. Daily activities often require frequent interaction with the screens of electronic devices. If an agent can seamlessly navigate these devices by controlling screens according to user needs, it would mark a significant step towards more general intelligence [Yin *et al.*, 2023]. Indeed, a screen interaction agent must possess powerful visual information processing capabilities, and the ability to execute computer control instructions as shown in Fig. 1. Therefore, to achieve such a goal, it is necessary to create a real interactive environment for the VLM agent, then guide the model and environment to form a continuous interactive pipeline and train the agent to improve its performance. However, it's highly challenging to implement these functions within a unified framework and achieve satisfactory results from both the project engineering and theoretical research perspectives.

Despite recent progress, some aspects still need to be further explored. For instance, CogAgent [Hong *et al.*, 2023]

*corresponding author

specializes in GUI understanding and planning, showcasing remarkable proficiency in addressing diverse vision-modal challenges. However, CogAgent lacks the ability to invoke tools in visual scenarios. Subsequently, AppAgent [Yang *et al.*, 2023a] concentrates on smartphone operations, enabling navigation and acquiring new application usage skills through autonomous exploration or by observing human demonstrations. While AppAgent uses auxiliary positioning tags to help the model select UI elements, it lacks an end-to-end interaction method. As a result, current Vision Language Models (VLM) agents are typically unable to interact with real computer or mobile environments to generate and execute continuous manipulative commands.

To address these issues, we propose ScreenAgent, an automated agent to handle continuous screen operations. We drive the agent to continuously interact with the environment through three phases: planning, acting, and reflecting phases. In particular, the reflecting phase enables the agent to perform reflective behaviors, making the entire pipeline more comprehensive and aligned with human action and thought processes. It autonomously assesses the execution status of the current action, providing feedback based on the ongoing state. This capability enhances its performance for subsequent actions, enabling our agent to possess the capability of a continuous thinking chain. Consequently, our agent can understand the next steps and engage in complete tool invocation to execute a series of continuous manipulative commands. The major contributions are summarized as follows:

- We present a Reinforcement Learning (RL) environment that enables the VLM agent to directly interact with a real computer screen via VNC protocol. By observing the screenshots, our agent can interact with the GUI through basic mouse and keyboard operations.
- We develop an automated pipeline that encompasses the Planning phase, Acting phase, and Reflecting phase. This integrated pipeline facilitates the agent's continuous interaction with the environment, distinguishing our agent from others.
- We propose the ScreenAgent dataset, which includes action sequences for completing generic tasks on Linux and Windows desktops. Moreover, we provide a fine-grained scoring metric to comprehensively evaluate the various capabilities that are necessary for a VLM agent in computer-controlling tasks.
- We test GPT-4V and two state-of-the-art open-source VLMs on our test-set. The results demonstrate that GPT-4V is capable of controlling computers but lacks precise positioning capabilities. Thus, we train ScreenAgent to enable precise positioning and achieve comparable results to GPT-4V in all aspects. Our work can facilitate further research into developing a generalist agent.

2 Related Work

2.1 Multimodal Large Language Models

The LLMs have demonstrated powerful contextual understanding and text generation capabilities, enabling the implementation of complex multi-turn question-answering systems. LLaMA [Touvron *et al.*, 2023] is a series of foundational language models spanning from 7 billion to 65 billion parameters, with Vicuna-13B [Chiang *et al.*, 2023], an open-source chatbot, being refined through fine-tuning the LLaMA architecture. GPT-4 is an advancement by OpenAI following the success of GPT-3 which introduces several noteworthy improvements. GPT-4V(ision) [Yang *et al.*, 2023b], building upon GPT-4, has added multimodal capabilities. LLaVA [Liu *et al.*, 2023b] and LLaVA-1.5 [Liu *et al.*, 2023a] connect the pre-trained CLIP [Radford *et al.*, 2021] visual encoder with Vicuna, achieving multimodal capabilities. Fuyu-8B[1] does not use an image encoder but opts for a pure decoder Transformer architecture. CogVLM [Wang *et al.*, 2023e] is a powerful open-source Visual Language Model that supports image understanding and multi-turn dialogues. In addition, Monkey [Li *et al.*, 2023] introduces an efficient training method that enhances input resolution capability.

2.2 Computer Control Environment & Dataset

In simulated environments, agents can be trained to emulate clicking and typing. WebNav [Nogueira and Cho, 2016] creates a navigation environment with links, testing the agent's sequential decision-making ability. MiniWoB++ [Liu *et al.*, 2018] provides a lot of simplified ATARI-like web-browser-based tasks as a RL environment. WebShop [Yao *et al.*, 2023] offers tasks for controlling the browser to complete the purchase process. SWDE [Hao *et al.*, 2011] preserves webpage HTML files to train information extraction models. WebSRC [Chen *et al.*, 2021] is a QA-style dataset that contains a large number of questions and answers about webpage screenshots. Mind2Web [Deng *et al.*, 2023] introduces a dataset for developing generalist web agents. Seq2act [Li *et al.*, 2020a] integrates three datasets for Android, AndroidHowTo, Rico-SCA, and PixelHelp. Screen2Words [Wang *et al.*, 2021] is a large-scale screen summarization dataset for Android UI screens. META-GUI [Sun *et al.*, 2022] is a dataset for training a multi-modal conversational agent on mobile GUI. [Burns *et al.*, 2022] provides a dataset of unknown command feasibility on Android.

2.3 Large Language Model-driven Agents

With the advancement of LLMs, the capabilities of intelligent agents have also been enhanced. WebGPT [Nakano *et al.*, 2021] conducts fine-tuning on GPT-3 to address extended questions within a text-based web-browsing environment, enabling the model to explore and navigate the web for answers. ToolFormer [Schick *et al.*, 2023] integrates an assortment of utilities, featuring a calculator, Q&A system, and search engine, among others. Voyager [Wang *et al.*, 2023a] stands as the inaugural embodiment of a Large Language Model-powered, lifelong learning agent within the Minecraft environment. RecAgent [Wang *et al.*, 2023d] proposes that agents can generate high-level thoughts through the operation of memory reflection. ProAgent [Ye *et al.*, 2023] introduces a novel paradigm in process automation that seamlessly integrates agents powered by LLM. CogAgent [Hong *et al.*, 2023], an 18-billion-parameter visual language model,

[1]https://www.adept.ai/blog/fuyu-8b

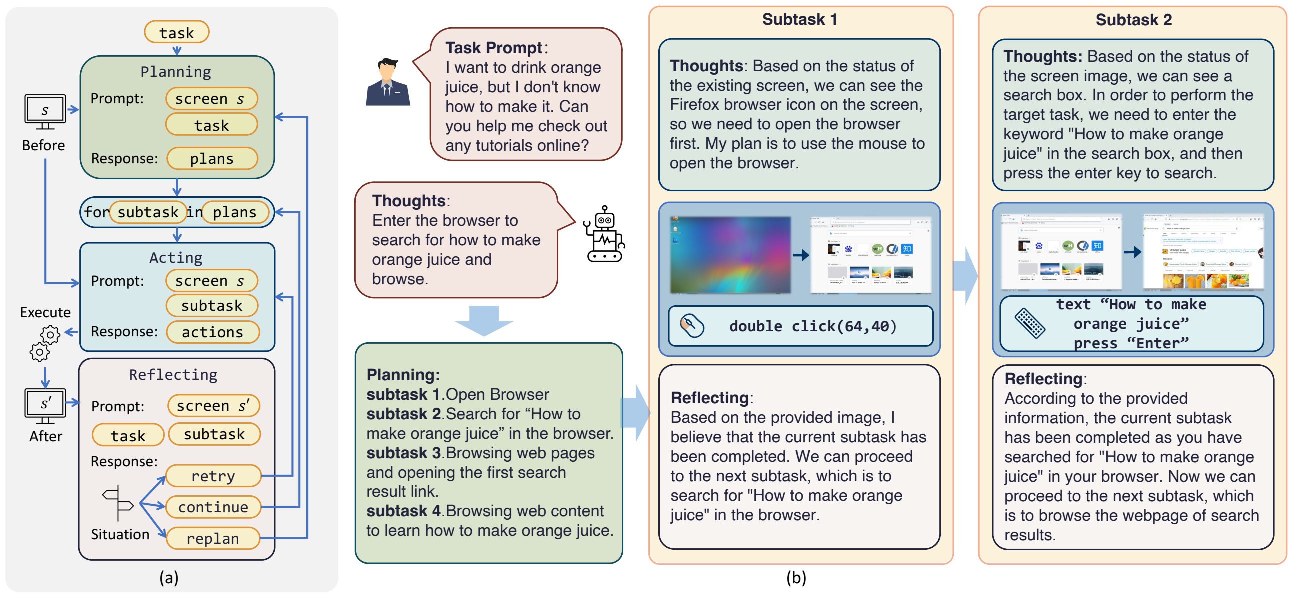

task
Planning
Prompt: screen s task
Response: plans
Before
for subtask in plans
Acting
Prompt: screen s subtask
Response: actions
Execute
Reflecting
Prompt: screen s' task subtask
Response:
After
retry
continue
replan
Situation
(a)
Task Prompt: I want to drink orange juice, but I don't know how to make it. Can you help me check out any tutorials online?
Thoughts: Enter the browser to search for how to make orange juice and browse.
Planning: subtask 1.Open Browser subtask 2.Search for "How to make orange juice" in the browser. subtask 3.Browsing web pages and opening the first search result link. subtask 4.Browsing web content to learn how to make orange juice.
Subtask 1
Thoughts: Based on the status of the existing screen, we can see the Firefox browser icon on the screen, so we need to open the browser first. My plan is to use the mouse to open the browser.
double click(64,40)
Reflecting: Based on the provided image, I believe that the current subtask has been completed. We can proceed to the next subtask, which is to search for "How to make orange juice" in the browser.
Subtask 2
Thoughts: Based on the status of the screen image, we can see a search box. In order to perform the target task, we need to enter the keyword "How to make orange juice" in the search box, and then press the enter key to search.
text "How to make orange juice" press "Enter"
Reflecting: According to the provided information, the current subtask has been completed as you have searched for "How to make orange juice" in your browser. Now we can proceed to the next subtask, which is to browse the webpage of search results.
...
(b)

Figure 2: The overview of our computer control pipeline, which includes planning, acting, and reflecting phases. Sub-figure (a) presents the flowchart of our pipeline, while sub-figure (b) provides an illustrative example. Based on the user's task prompt, the agent initially decomposes the task into sub-tasks. In each sub-task, the agent first describes the screen and generates mouse and keyboard operations in a function-call style. In the reflecting phase, the agent decides whether to proceed to the next sub-task, retry the current sub-task, or reformulate the entire plan.

is meticulously designed for GUI comprehension and navigation. AppAgent [Yang *et al.*, 2023a] proposes a framework that allows a multimodal agent to interact with smartphone applications.

3 Framework

In this section, we introduce our Reinforcement Learning (RL) environment and the autonomous control pipeline within the agent. Through this environment, a VLM agent can interact with a real computer screen, observe screen images, select actions, and autonomously complete specific tasks.

3.1 Computer Control Environment

We created a computer control environment to evaluate the capabilities of VLM agents. This environment connects to a desktop operating system using an implementation of the remote desktop protocol (VNC) and allows the sending of mouse and keyboard events to the controlled desktop. The formal definitions of this environment are outlined as follows:

- $\mathcal{A}$**-Action Space.** We define an action as a form of a function call. If the agent outputs an action in the required JSON-style format, the action will be parsed and executed by the environment. All action types and corresponding action attributes are defined in Table 1.
- $\mathcal{S}$**-State Space.** The screenshot image is utilized as the state space of the environment. The environment will collect screenshots s and s', denoting the state before and after each action, respectively.
- $\mathcal{R}$**-Reward Function.** Due to the highly open-ended nature of the task, the reward function is flexibly opened to different interfaces, which can integrate different existing or future reward models.

Action Type		Attributes
Mouse	Move	Mouse Position(width:int, height:int)
	Click	Mouse Button(left/middle/right), Mouse Position(width:int, height:int)
	Double Click	Mouse Button(left/middle/right), Mouse Position(width:int, height:int)
	Scroll Up	Scroll Repeat(int)
	Scroll Down	Scroll Repeat(int)
	Drag	Drag End Position(width:int, height:int)
Keyboard	Press	Keyboard Key or Combined-keys (string)
	Text	Keyboard Text(string)
Wait Action		Wait Time(float)
Plan Action		Element(string)
Evaluate Action		Situation(success/retry/reformulate) Advice(string)

Table 1: All supported action types and action attributes.

Through remote control, the agent can perform arbitrary tasks on the screen, which creates a highly challenging open environment having a large state and action space.

3.2 Control Pipeline

To guide the agent to continually interact with the environment and complete multi-step complex tasks, we designed a control pipeline including the planning, acting, and reflecting phases. The whole pipeline is depicted in Fig. 2. This pipeline will ask the agent to disassemble the complex task,

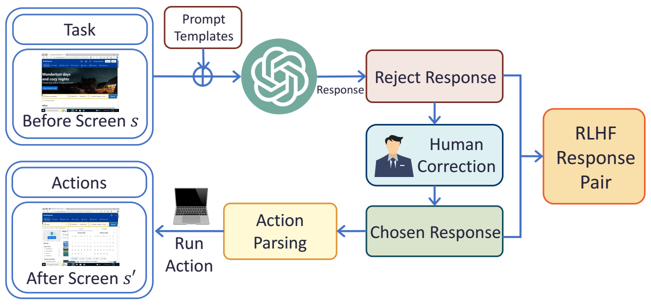


Figure 3: Data annotation process. We invoke GPT-4V to generate an original response, and annotators correct this response as the golden labeled response. The environment parses executable actions from the text and sends them for real computer execution. The original response and the golden labeled response form a pair, which can be utilized for training in future Reinforcement Learning from Human Feedback (RLHF) processes.

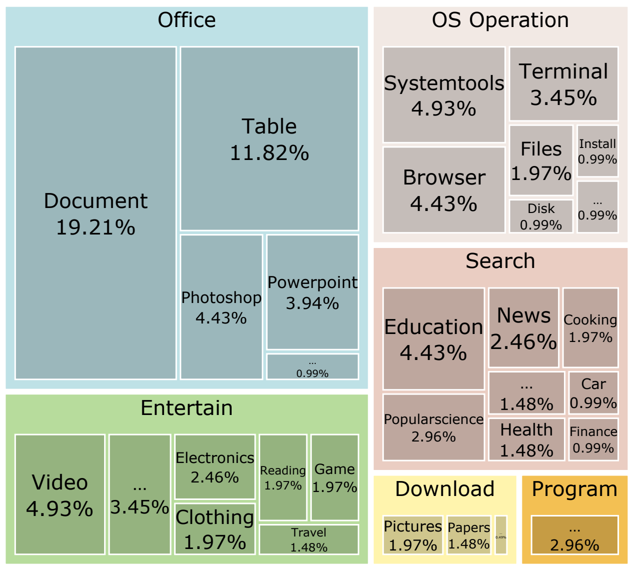
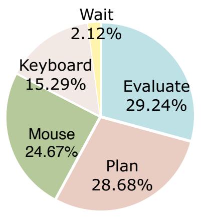
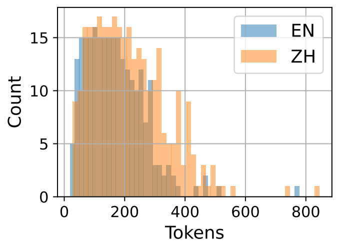


Figure 4: Task type statistics in ScreenAgent training-set.





(a) (b)

Figure 5: The statistical information of ScreenAgent training-set: (a) Distribution of action types; (b) Chosen response token number distribution.

execute sub-tasks, and evaluate execution results. The agent will have the opportunity to retry some sub-tasks or adjust previously established plans to accommodate the current occurrences. The details are depicted as follows:

Planning Phase. In the planning phase, based on the current screenshot, the agent needs to decompose the complex task relying on its own common-sense knowledge and computer knowledge.

Acting Phase. In the acting phase, based on the current screenshot, the agent generates low-level mouse or keyboard actions in JSON-style function calls. The environment will attempt to parse the function calls from the agent's response, and convert them to device actions defined in the VNC protocol. Then our environment will send actions to the controlled computer. The environment will capture the after-action screen as input for the next phase.

Reflecting Phase. The reflecting stage requires the agent to assess the current situation based on the after-action screen. The agent determines whether needs to retry the current sub-task, go on to the next sub-task, or make some adjustments to the plan list. This phase is crucial within the control pipeline, providing some flexibility to handle a variety of unpredictable circumstances.

4 ScreenAgent Dataset & CC-Score

ScreenAgent Dataset. Existing computer-controlling datasets typically have a narrow range of applicability scenarios. For instance, building upon the foundational premise that it is easy to obtain UI element metadata through HTML or developer modes, WebNav [Nogueira and Cho, 2016], Mind2Web [Deng *et al.*, 2023], and SWDE [Hao *et al.*, 2011] mainly focused on web browsing, while Seq2act [Li *et al.*, 2020a] and Screen2Words [Wang *et al.*, 2021] are tailored for Android. However, the mouse and keyboard are also common and universal interfaces to control a computer. To fill the gap in this type of control method, we build an interactive annotation process (shown in Fig. 3) to construct the ScreenAgent Dataset which is collected from Linux and Windows operating systems for completing specific tasks. This dataset endeavors to cover a wide range of daily computer usage scenarios, including daily office, booking, information retrieval, card games, entertainment, programming, system operations, and so on. As illustrated in Fig. 4, the ScreenAgent Dataset encompasses 39 sub-task categories across 6 themes. The dataset has 273 complete task sessions, with 203 sessions (3005 screenshots) for training and 70 sessions (898 screenshots) for testing. Fig. 5 shows important statistical information about the dataset.

CC-Score. To assess an agent's capability in the computer control task, we design a fine-grained evaluation metric VLM Agent Computer Control Score (CC-Score) for assessing the similarity of control action sequences. This metric takes into account both the sequential order and actions' attribution. We develop specific similarity metrics for every action type. For mouse actions, the metrics include four aspects of consistency: action type, mouse operation type, mouse button, and whether the click coordinates are within the annotated feasible bounding box. For text and keyboard actions, the metrics

Model	Plan total 284	Action Type total 650	Mouse Action Type total 232	Mouse Button total 209	Mouse Position total 218	Keyboard Keys or Text total 134	Reflecting Situation Assessment total 546
LLaVA-1.5	<u>0.78</u>	0.75	0.71	0.74	0.72	0.45	0.98
CogAgent-VQA	0.00	0.03	0.06	0.06	0.05	0.01	0.39
CogAgent-Chat (original output)	0.00	0.00	0.00	0.00	0.00	0.00	0.30
CogAgent-Chat (helped by GPT-3.5)	0.29	0.38	0.44	0.45	0.42	0.17	0.76
GPT-4V(ision)	**0.87**	**0.86**	<u>0.85</u>	<u>0.85</u>	<u>0.83</u>	<u>0.77</u>	**1.00**
ScreenAgent	0.72	<u>0.83</u>	**0.91**	**0.92**	**0.91**	**0.82**	**1.00**

Table 2: Proportion of successful function calls on ScreenAgent test-set.

Model	CC-Score	Plan (BLEU)	Action Type (F1)	Mouse Action Type (F1)	Mouse Button (F1)	Mouse Position (Accuracy)	Keyboard Keys or Text (BLEU)	Reflecting Situation Assessment (F1)
LLaVA-1.5	0.51	0.29	0.91	0.90	0.96	0.03	0.70	<u>0.52</u>
CogAgent-Chat (helped by GPT-3.5)	0.33	<u>0.32</u>	0.83	0.86	0.02	0.07	0.74	0.51
GPT-4V(ision)	**0.63**	**0.47**	**0.98**	**0.96**	**0.99**	<u>0.12</u>	**0.92**	**0.60**
ScreenAgent	<u>0.61</u>	0.31	**0.98**	<u>0.94</u>	<u>0.97</u>	**0.51**	<u>0.87</u>	<u>0.52</u>

Table 3: Comparison of VLM fine-grained score in all successful matched action on ScreenAgent test-set.

involve two aspects: action type consistency and the BLEU score of the input text, single key, or keyboard shortcut combination. For the entire action sequence, we employ an alignment algorithm that identifies the maximum matching pairs of predicted action and labeled action, while maintaining the sequence order. This approach maximizes the overall score, which is used as the measure of sequence similarity. Ultimately, the CC-Score encompasses the normalized scores of predicted and labeled sequences, the F1 values for each action attribute classification, and the unigram similarity values for text types.

Consider two action sequences: a label action sequence $L = \{l_1, l_2, ..., l_n\}$ and a predict action sequence $P = \{p_1, p_2, ..., p_m\}$. Define a score matrix S as an $n \times m$ matrix, where S_{ij} represents the similarity score between action l_i and p_j. A possible alignment is a sequence $C = \{(c_1, d_1), (c_2, d_2), ..., (c_k, d_k)\}$, where each element (c_i, d_i) is a pair of action such that $c_i \in L \cup \{\varnothing\}$ and $d_i \in P \cup \{\varnothing\}$. Each (c_i, d_i) pair consists either of corresponding actions from L and P, or a combination of an action from either L or P with an empty ($\varnothing$) value. Importantly, this sequence must adhere to the constraint that the order of non-null actions in L and P within the alignment is preserved as in their original sequences. This means that if (c_i, d_i) and (c_j, d_j) are two pairs in C where $c_i, c_j \neq \varnothing$ and $d_i, d_j \neq \varnothing$, and if $i < j$, then c_i must precede c_j in L and d_i must precede d_j in P. This constraint ensures that the alignment respects the sequential nature and integrity of the original action sequences.

For a given alignment C, its score is the sum of the similarity scores for all matched pairs of actions in the alignment, that is, $\sum_{(i,j)\in C} S_{ij}$. We choose the alignment with the highest score as the best matching alignment C^*. Finally, the CC-Score for the prediction and label action sequence is calculated as:

$$\text{CC-Score}(L, P) = \frac{1}{|L|} \sum_{(i,j)\in C^*} S_{ij}.$$

5 Experiment

5.1 Evaluation Results on ScreenAgent Test-set

Apart from GPT-4V, we select several recently released SoTA VLMs for testing, including LLaVA-1.5 [Liu *et al.*, 2023a] and CogAgent [Hong *et al.*, 2023]. LLaVA-1.5 is a VLM with 13 billion parameters; however, it only supports image inputs up to 336 × 336 pixels. CogAgent is an 18-billion-parameter visual language model designed for GUI comprehension and navigation. It demonstrates proficiency at a resolution of 1120 × 1120 pixels, enabling it to detect small elements and text.

We test these models' capabilities from two aspects: The ability to follow instructions to output the correct function call format, shown in Table 2. And the ability to complete specific tasks assigned by the user, shown in Table 3.

Following instructions and executing correct function calls is the most fundamental skill for an LLM agent when learning to use external tools. Table 2 presents the success rate of these function calls for each attribute key. This assessment focuses on whether the model can accurately execute various functions encompassing the attribute items expected by manual action annotations. Note that, this evaluation does not consider the consistency of the attribute values with the golden labeling; it only examines whether the model's output includes the necessary attribute keys. From the table,

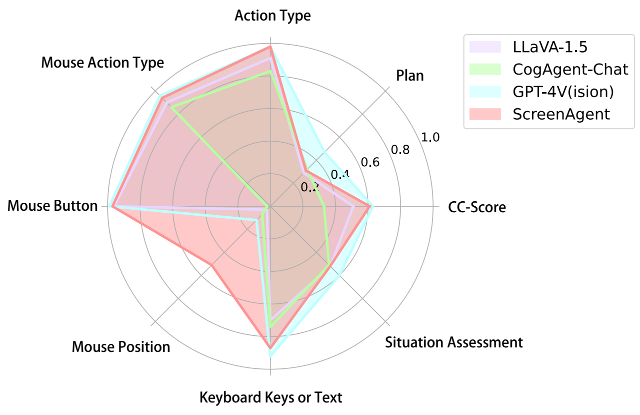


Figure 6: ScreenAgent can complete computer control tasks most excellently compared with other VLMs/Agents.

GPT-4V and LLaVA-1.5 achieve higher scores, while CogAgent and its upstream model CogAgent-VQA underperformed. CogAgent-VQA and CogAgent-Chat almost entirely disregarded the JSON format action definitions in prompts, resulting in a very low score on successful function calls. Therefore, rendering them completely incapable of interacting with our environment. To ensure fairness in comparison, we utilize OpenAI GPT-3.5 to extract action into JSON-style function calls from the original CogAgent-Chat responses, indicated as "CogAgent-Chat (helped by GPT-3.5)". Even so, its scores are significantly lower than those of LLaVA-1.5 and GPT-4V, although CogAgent has been trained on Mind2Web web browsing simulation datasets.

Table 3 displays the fine-grained scores of predicted attribute values for each action within the successfully parsed function calls. As can be seen, GPT-4V remains the best performer, with an action-type prediction F1 score of 0.98. This implies that it can accurately select appropriate mouse or keyboard actions. Additionally, it can precisely choose the mouse action type, typing text, or pressing keys consistent with the golden label actions.

The ability for precise positioning is crucial in computer-controlling tasks. As indicated by the "Mouse Position" column in Table 3, current VLMs have yet to develop the precise positioning capabilities necessary for computer manipulation. GPT-4V refuses to give precise coordinate results in its answers, and two open-source models also fail to output the correct coordinates with our prompts.

Another significant challenge for all models is the reflection phase. In this phase, the agent is required to determine whether the sub-task has been completed in the current state, and decide whether to go further or make some adjustments. This is crucial for constructing a continuous interactive process. Regrettably, all models show insufficient accuracy in this determination, with GPT-4V achieving only a 0.60 F1 score. This implies that human intervention is still necessary during task execution.

5.2 Fine-tuning Training

To demonstrate the potential for ongoing research on the task, we continue to fine-tune the CogAgent-Chat model on our ScreenAgent training-set to enhance its function call ability. Similar to the approach adopted in recent VLM works [Chen *et al.*, 2023], we mix data from multiple datasets and construct four distinct training phases, which is illustrated in Table 4. We reformulate two objective detection datasets, COCO [Lin *et al.*, 2015] and Widget Captions [Li *et al.*, 2020b], into mouse-click tasks to enhance the model's positioning ability. For Mind2Web, we implement a series of complex data augmentations to align with our task objectives.

Dataset	Samples	1	2	3	4
COCO	42404	20%	10%	-	-
Widget Captions	41221	20%	10%	-	-
Mind2Web	12846	30%	40%	50%	30%
ScreenAgent Dataset	3005	30%	40%	50%	70%

Table 4: Training data proportions and division of four training phases. Percentages indicate the proportion of samples from this data set at each phase.

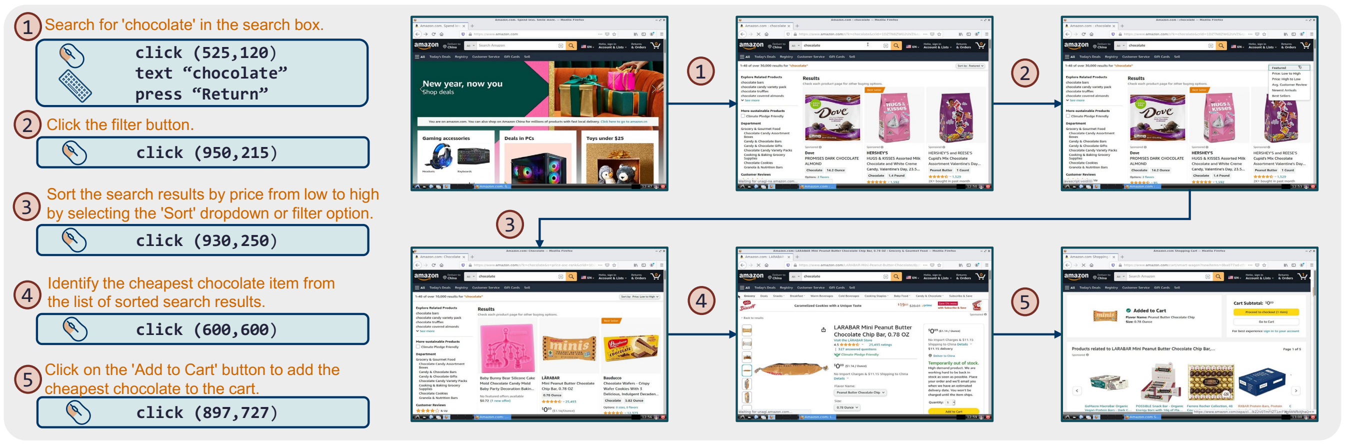
After fine-tuning, ScreenAgent achieves the same level of following instructions and making function calls as GPT-4V, as shown in Table 2. In Table 3, our ScreenAgent model achieves a performance level comparable to GPT-4V, and far exceeds existing models in the precision of mouse clicks. This indicates that the process of fine-tuning effectively enhances the model's precise positioning capabilities. Additionally, we observe that ScreenAgent has a significant gap compared to GPT-4V in terms of task planning, highlighting GPT-4V's common-sense knowledge and task-planning abilities.

5.3 Case Study

To evaluate our ScreenAgent on computer control tasks, we provide two cases. In Fig. 7, we present the workflow of ScreenAgent that executes a chain of actions. In Fig. 8, we compare different agents in executing the details of the three phases in the pipeline. Fig. 8 (a) shows the planning process of all the agents, where we find that our ScreenAgent produces the most concise and effective plan. Fig. 8 (b) presents four different click action tasks, each representing a step in a specific task. Results show that LLaVA clicks on the bottom-left corner on all screens, CogAgent fails to generate click positions, and in the fourth task, only our agent can correctly click on the position. Fig. 8 (c) shows that our agent can recognize whether an action needs to be re-tried after reflection and successfully execute the action following a failure.

6 Conclusion

In this paper, we build a new environment for the screen control task. VLM agents can manipulate a real computer by generating mouse and keyboard control commands. To encourage the agent to continuously interact with the environment and accomplish complex multi-step tasks, we design a control pipeline that includes planning, acting, and reflecting phases. Furthermore, we unveil a new dataset that covers a wide range of everyday digital works. We propose a fine-grained metric to assess the agent's computer-controlling capabilities with



1 Search for 'chocolate' in the search box.

```
click (525,120)
text “chocolate”
press “Return”
```

2 Click the filter button.

```
click (950,215)
```

3 Sort the search results by price from low to high by selecting the 'Sort' dropdown or filter option.

```
click (930,250)
```

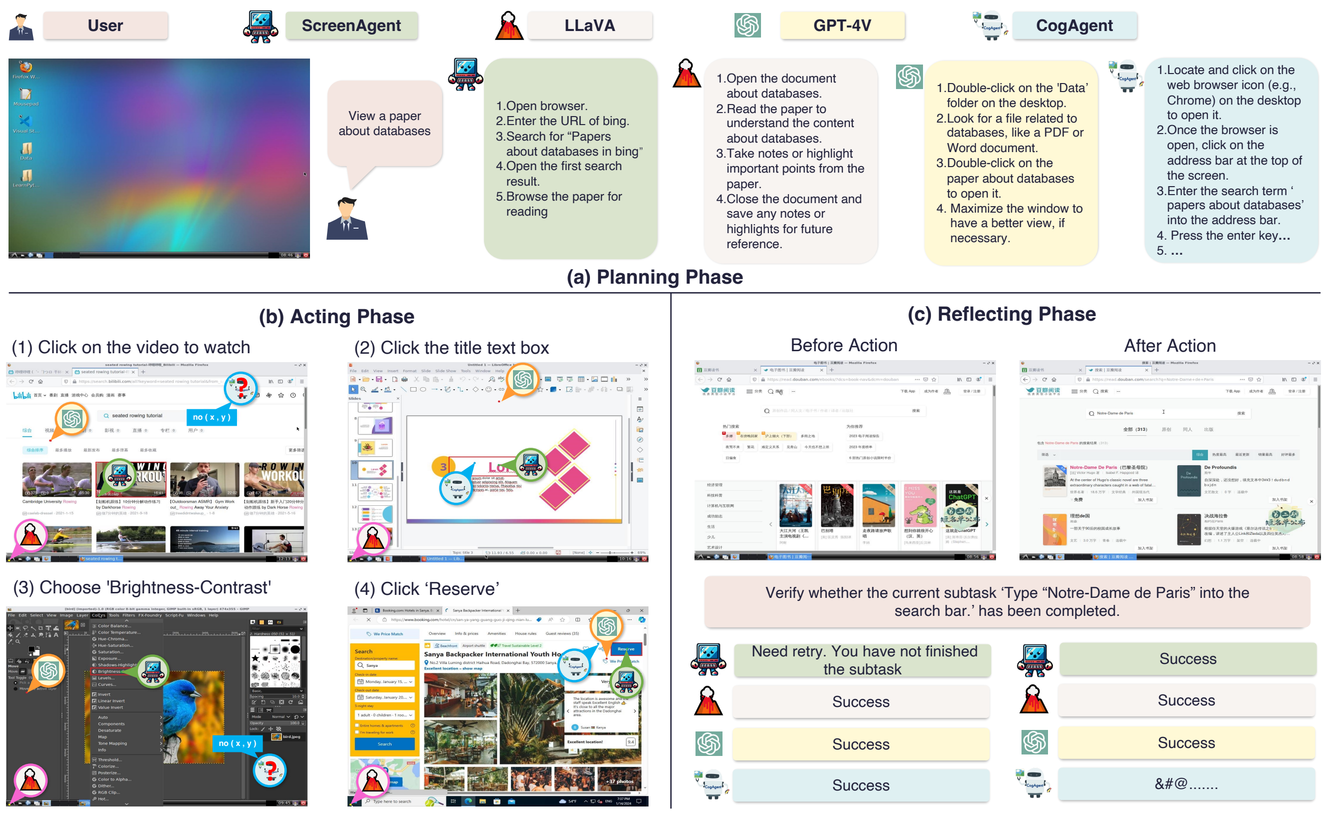
4 Identify the cheapest chocolate item from the list of sorted search results.

```
click (600,600)
```

5 Click on the 'Add to Cart' button to add the cheapest chocolate to the cart.

```
click (897,727)
```

Figure 7: An example of the action task, where we assign ScreenAgent the following task: "Find and add the cheapest chocolate to the cart on Amazon". The diagram delineates the actions that the agent needs to perform, along with the alterations in the computer environment before and after the execution of these actions.



View a paper about databases

ScreenAgent:
1.Open browser.
2.Enter the URL of bing.
3.Search for “Papers about databases in bing”
4.Open the first search result.
5.Browse the paper for reading

LLaVA:
1.Open the document about databases.
2.Read the paper to understand the content about databases.
3.Take notes or highlight important points from the paper.
4.Close the document and save any notes or highlights for future reference.

GPT-4V:
1.Double-click on the 'Data' folder on the desktop.
2.Look for a file related to databases, like a PDF or Word document.
3.Double-click on the paper about databases to open it.
4. Maximize the window to have a better view, if necessary.

CogAgent:
1.Locate and click on the web browser icon (e.g., Chrome) on the desktop to open it.
2.Once the browser is open, click on the address bar at the top of the screen.
3.Enter the search term ‘ papers about databases’ into the address bar.
4. Press the enter key...
5. ...

(1) Click on the video to watch

(2) Click the title text box

(3) Choose 'Brightness-Contrast'

(4) Click ‘Reserve’

Verify whether the current subtask ‘Type “Notre-Dame de Paris” into the search bar.’ has been completed.

	Before Action	After Action
ScreenAgent	Need retry. You have not finished the subtask	Success
LLaVA	Success	Success
GPT-4V	Success	Success
CogAgent	Success	&#@.......

Figure 8: An example to show the execution results of multiple VLM agents among the planning, acting, and reflecting phases.

both action-level and task-level evaluation. We test OpenAI GPT-4V and two state-of-the-art VLM models on the test-set. The results indicate that GPT-4V has the potential to act as a computer-controlling agent, but it lacks precise positioning capabilities. Finally, we train the ScreenAgent model, inherited from CogAgent, to achieve scores comparable to GPT-4V but with greater accuracy in UI element positioning. We hope that our work could inspire further research in the construction of more powerful and generalized agents. In terms of technical limitations, due to the input restrictions of VLM, our model can only process single-frame images, not videos or multi-frame images. In addition, the VLM’s language capability is limited by the foundation language model.

Ethical Statement

The rational use of automated agents with autonomous decision-making capabilities brings significant societal benefits, including improving the accessibility of computers, reducing duplication of human effort on digital work, and aiding in computer education. However, the potential negative impacts of these agents, such as employment impact, fraud and abuse, privacy issues, and the risk of misoperation, cannot be overlooked. The method could potentially be used to bypass the CAPTCHA test for automatic account registration, spreading misinformation, or conducting illegal activities. We are focused on and committed to the responsible use of AI technology.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (No. 62206107), National Science and Technology Major Project under Grant (No. 2023YFF0905400) and New H3C Technologies Co., Ltd.

References

[Burns *et al.*, 2022] Andrea Burns, Deniz Arsan, Sanjna Agrawal, Ranjitha Kumar, Kate Saenko, and Bryan A. Plummer. A dataset for interactive vision language navigation with unknown command feasibility. In *European Conference on Computer Vision (ECCV)*, 2022.

[Chen *et al.*, 2021] Xingyu Chen, Zihan Zhao, Lu Chen, Danyang Zhang, Jiabao Ji, Ao Luo, Yuxuan Xiong, and Kai Yu. Websrc: A dataset for web-based structural reading comprehension, 2021.

[Chen *et al.*, 2023] Jun Chen, Deyao Zhu, Xiaoqian Shen, Xiang Li, Zechun Liu, Pengchuan Zhang, Raghuraman Krishnamoorthi, Vikas Chandra, Yunyang Xiong, and Mohamed Elhoseiny. Minigpt-v2: large language model as a unified interface for vision-language multi-task learning, 2023.

[Chiang *et al.*, 2023] Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E Gonzalez, et al. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality. *See https://vicuna. lmsys. org (accessed 14 April 2023)*, 2023.

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

[Hao *et al.*, 2011] Qiang Hao, Rui Cai, Yanwei Pang, and Lei Zhang. From one tree to a forest: a unified solution for structured web data extraction. In *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '11, page 775–784, New York, NY, USA, 2011. Association for Computing Machinery.

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

[Li *et al.*, 2020a] Yang Li, Jiacong He, Xin Zhou, Yuan Zhang, and Jason Baldridge. Mapping natural language instructions to mobile ui action sequences. In *Annual Conference of the Association for Computational Linguistics (ACL 2020)*, 2020.

[Li *et al.*, 2020b] Yang Li, Gang Li, Luheng He, Jingjie Zheng, Hong Li, and Zhiwei Guan. Widget captioning: Generating natural language description for mobile user interface elements, 2020.

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

[Lin *et al.*, 2015] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. Microsoft coco: Common objects in context, 2015.

[Liu *et al.*, 2018] Evan Zheran Liu, Kelvin Guu, Panupong Pasupat, Tianlin Shi, and Percy Liang. Reinforcement learning on web interfaces using workflow-guided exploration. In *International Conference on Learning Representations (ICLR)*, 2018.

[Liu *et al.*, 2023a] Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee. Improved baselines with visual instruction tuning, 2023.

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

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

[Nogueira and Cho, 2016] Rodrigo Nogueira and Kyunghyun Cho. End-to-end goal-driven web navigation. In *Advances In Neural Information Processing Systems*, pages 1903–1911, 2016.

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

[Schick *et al.*, 2023] Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. *arXiv preprint arXiv:2302.04761*, 2023.

[Sun *et al.*, 2022] Liangtai Sun, Xingyu Chen, Lu Chen, Tianle Dai, Zichen Zhu, and Kai Yu. Meta-gui: Towards multi-modal conversational agents on mobile gui. *arXiv preprint arXiv:2205.11029*, 2022.

[Touvron *et al.*, 2023] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

[Wang *et al.*, 2021] Bryan Wang, Gang Li, Xin Zhou, Zhourong Chen, Tovi Grossman, and Yang Li. Screen2words: Automatic mobile ui summarization with multimodal learning. In *The 34th Annual ACM Symposium on User Interface Software and Technology*, UIST '21, page 498–510, New York, NY, USA, 2021. Association for Computing Machinery.

[Wang *et al.*, 2023a] Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*, 2023.

[Wang *et al.*, 2023b] Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, et al. A survey on large language model based autonomous agents. *arXiv preprint arXiv:2308.11432*, 2023.

[Wang *et al.*, 2023c] Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, Wayne Xin Zhao, Zhewei Wei, and Ji-Rong Wen. A survey on large language model based autonomous agents, 2023.

[Wang *et al.*, 2023d] Lei Wang, Jingsen Zhang, Hao Yang, Zhiyuan Chen, Jiakai Tang, Zeyu Zhang, Xu Chen, Yankai Lin, Ruihua Song, Wayne Xin Zhao, Jun Xu, Zhicheng Dou, Jun Wang, and Ji-Rong Wen. When large language model based agent meets user behavior analysis: A novel user simulation paradigm. *arXiv preprint arXiv:2306.02552*, 2023.

[Wang *et al.*, 2023e] Weihan Wang, Qingsong Lv, Wenmeng Yu, Wenyi Hong, Ji Qi, Yan Wang, Junhui Ji, Zhuoyi Yang, Lei Zhao, Xixuan Song, et al. Cogvlm: Visual expert for pretrained language models. *arXiv preprint arXiv:2311.03079*, 2023.

[Yang *et al.*, 2023a] Zhao Yang, Jiaxuan Liu, Yucheng Han, Xin Chen, Zebiao Huang, Bin Fu, and Gang Yu. Appagent: Multimodal agents as smartphone users. 2023.

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

[Yao *et al.*, 2023] Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. Webshop: Towards scalable real-world web interaction with grounded language agents, 2023.

[Ye *et al.*, 2023] Yining Ye, Xin Cong, Shizuo Tian, Jiannan Cao, Hao Wang, Yujia Qin, Yaxi Lu, Heyang Yu, Huadong Wang, Yankai Lin, et al. Proagent: From robotic process automation to agentic process automation. *arXiv preprint arXiv:2311.10751*, 2023.

[Yin *et al.*, 2023] Shukang Yin, Chaoyou Fu, Sirui Zhao, Ke Li, Xing Sun, Tong Xu, and Enhong Chen. A survey on multimodal large language models. *arXiv preprint arXiv:2306.13549*, 2023.