SeeClick: Harnessing GUI Grounding for Advanced Visual GUI Agents

Anonymous ACL submission

Abstract

Graphical User Interface (GUI) agents are designed to automate complex tasks on digital devices, such as smartphones and desktops. Most existing GUI agents interact with the environment through extracted structured data, which can be notably lengthy (e.g., HTML) and occasionally inaccessible (e.g., on desktops). To alleviate this issue, we propose a novel visual GUI agent – *SeeClick*, which only relies on screenshots for task automation. In our preliminary study, we have discovered a key challenge in developing visual GUI agents: GUI grounding – the capacity to accurately locate screen elements based on instructions. To tackle this challenge, we propose to enhance *SeeClick* with GUI grounding pre-training and devise a method to automate the curation of GUI grounding data. Along with the efforts above, we have also created *ScreenSpot*, the first realistic GUI grounding benchmark that encompasses mobile, desktop, and web environments. After pre-training, *SeeClick* demonstrates significant improvement in *ScreenSpot* over various baselines. Moreover, comprehensive evaluations on three widely used benchmarks consistently support our finding that advancements in GUI grounding directly correlate with enhanced performance in downstream GUI agent tasks. The model, data and code will be open-sourced.

1 Introduction

A perennial topic in machine intelligence is the development of Graphical User Interface (GUI) agent systems, like Siri and Copilot, to automate complex tasks on computing devices, thereby reducing human workload (Shi et al., 2017; Li et al., 2020a). Recent advances in Large Language Models (LLMs) such as GPT-4 (OpenAI, 2023) have significantly propelled the evolution of GUI agents (Gur et al., 2023; Zhou et al., 2023). These agents interact with the environment by interpreting structured texts, e.g., HTML from webpages,

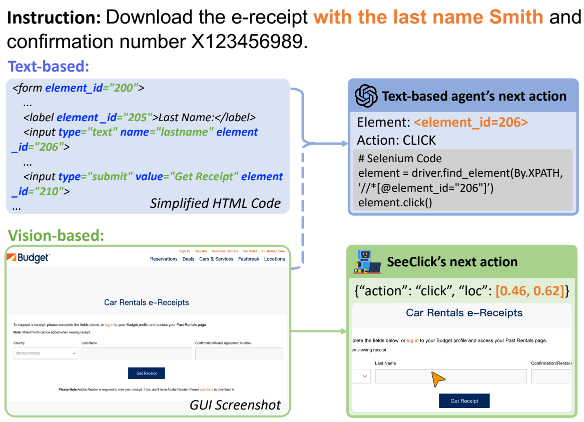


Figure 1: Text-based agents select target elements from structured texts, occasionally augmented with screenshots. *SeeClick* employs a vision-based methodology to predict action locations solely relying on screenshots.

then elicit LLM for planning, reasoning, and execution (Kim et al., 2023; Zheng et al., 2023).

However, GUI agents depend on structured text face three inherent limitations: (1) Structured text is not always accessible, especially for iOS or desktop applications where acquiring such information is challenging (Shaw et al., 2023); (2) The verbose nature of structured text constitutes an inefficient context for LLMs, while also omitting crucial information such as layout, images, and icons (Deng et al., 2023); (3) The variety of structured text - including HTML, DOM, and Android VH - necessitates the curation of task-specific observation and action spaces (Kim et al., 2023; Zhou et al., 2023). These entrenched deficiencies in text-based approaches call for an alternative solution.

In this paper, we introduce *SeeClick*, a visual GUI agent built on Large Vision-Language Models (LVLMs). Inspired by human interaction with GUIs, as illustrated in Figure 1, *SeeClick* is designed to perform low-level actions like clicking or typing directly by observing interface screenshots. This innovative approach bypasses the interaction with cumbersome structured text, empowering *SeeClick* to universally adapt to various GUI platforms. Building such visual agents presents a foundational challenge: GUI grounding - the capacity to accurately locate screen elements based on instructions, which is absent in current LVLMs.To tackle this challenge, *SeeClick* enhances LVLM with a GUI grounding pre-training strategy. We devise a method to automate the curation of web grounding data and adapt public mobile UI datasets to obtain mobile grounding data. *SeeClick* employs the above-curated dataset for continual pre-training of the LVLM, enabling it to accurately locate elements such as text, widgets, and icons in various GUI environments.

Given GUI grounding is a fundamental yet underexplored capacity for GUI agents, we establish *ScreenSpot*, the first realistic GUI grounding evaluation benchmark across various GUI platforms. *ScreenSpot* contains over 600 screenshots and 1200 instructions from iOS, Android, macOS, Windows, and webpages, and specifically includes both text-based elements and a variety of widgets and icons. Evaluation results confirm *SeeClick*'s superiority over current LVLMs, validating the effectiveness of GUI grounding pre-training.

Finally, we adapt *SeeClick* to mobile and web agent tasks, including MiniWob (Shi et al., 2017), AITW (Rawles et al., 2023), and Mind2Web (Deng et al., 2023). As a purely vision-based agent, *SeeClick* achieves impressive performance. It surpasses the strong visual baseline Pix2Act while utilizing merely 0.3% training data. Moreover, experimental results on these three benchmarks consistently support our findings that improvement in GUI grounding directly correlates with enhanced agent task performance.

Our main contributions are as follows:

- We develop a unified visual GUI agent *SeeClick*, which solely relies on interface screenshots to perform clicking and typing actions across diverse GUI platforms.
- We prospectively explore GUI grounding for visual GUI agents, and enhanced *SeeClick* with proposed GUI grounding pre-training strategy.
- We create a realistic GUI grounding benchmark *ScreenSpot*, encompassing more than 1200 instructions from various GUI platforms.
- Experimental results on *ScreenSpot* and three agent tasks demonstrate that enhancing agents' grounding capacity is key to improving performance in downstream agent tasks.

2 Related work

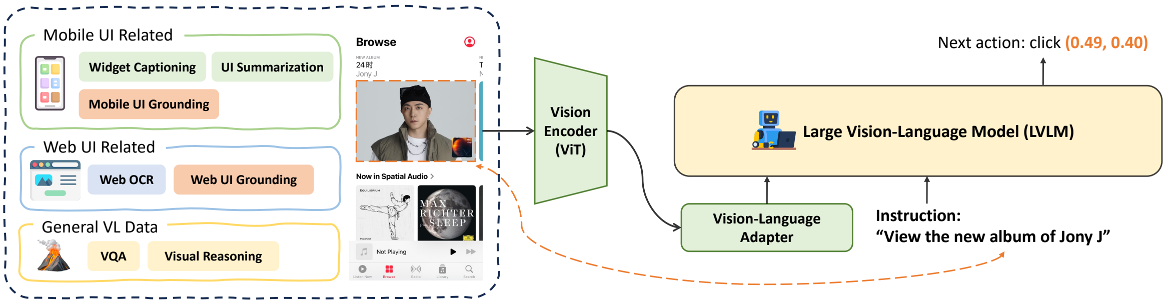
Autonomous GUI Navigation Early research explored task automation in simplified web (Shi et al., 2017; Liu et al., 2018; Gur et al., 2018) and mobile UI (Li et al., 2020a; Burns et al., 2022; Li and Li, 2022). With LLM advancements (OpenAI, 2023; Touvron et al., 2023; Xu et al., 2023; Sun et al., 2023; Wu et al., 2024, *inter alia*), LLM-centric agents have become the dominant paradigm. A line of works focused on prompting ChatGPT and GPT-4 for web tasks, via in-context learning (Zheng et al., 2023) and self-refine (Kim et al., 2023). Other research explored training LLMs as specialized agents. Deng et al. (2023) devised a two-stage method for identifying target elements within intricate HTML. Gur et al. (2023) proposed to interact with websites via programming.

Given the constraints of LLM to only process text, recent efforts have attempted vision-based GUI navigation (Shaw et al., 2023; Zhan and Zhang, 2023; Hong et al., 2023). These methods primarily utilize GPT-4V (Yan et al., 2023; Gao et al., 2023) and also require GUI metadata as input (Yang et al., 2023a; Zheng et al., 2024). In this paper, we construct a universal visual GUI agent *SeeClick* by customizing open-sourced LVLM, capable of operating across various GUI platforms without needing any GUI metadata.

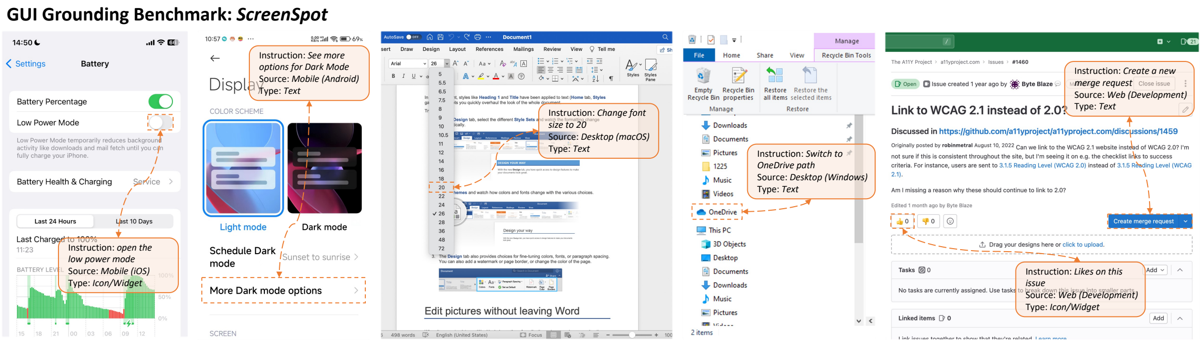
Large Vision-Language Models Recent research has invested tremendous effort in constructing LVLMs capable of jointly processing image and text (Liu et al., 2023a; Zhu et al., 2023; Ye et al., 2023; Li et al., 2023), integrating vision encoders with LLMs through connecting layers, inheriting LLMs' linguistic and reasoning skills to perform vision-language tasks. A series of studies focused on grounding with LVLMs (Wang et al., 2023; Bai et al., 2023; Chen et al., 2023a), such as providing bounding boxes for objects when generating responses (Chen et al., 2023b; Peng et al., 2023). Nonetheless, these efforts primarily addressed natural images and did not explore GUI contexts. This paper focuses on grounding in GUIs and explores the potential of LVLMs as visual agents.

3 Approach

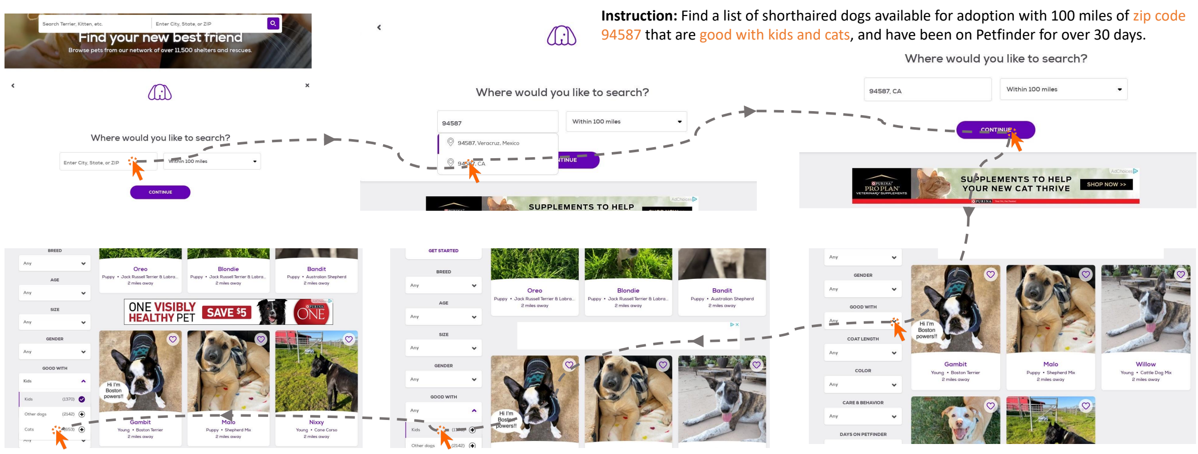
Our preliminary study highlights a major challenge in developing visual GUI agents: GUI grounding, the capacity to locate screen elements based on instructions. Although recent LVLMs have claimed grounding capability on natural images (Bai et al.,



(a) Overview of SeeClick's framework and GUI grounding pre-training.



(b) Examples of the proposed GUI grounding benchmark *ScreenSpot*.



(c) SeeClick as a visual GUI agent in downstream task.

Figure 2: Overview of our universal visual GUI agent *SeeClick*. (a) depicts the framework of *SeeClick* and GUI grounding pre-training. (b) provides examples of *ScreenSpot* across various GUIs and types of instructions. (c) displays the real-world application of *SeeClick* when adapted to downstream web agent tasks.

2023; Wang et al., 2023), GUI screenshots differ significantly with dense text and numerous icons and widgets. These differences impair existing LVLMs' grounding performance in GUI contexts and limit their potential as visual GUI agents.

This paper seeks to harness LVLMs with GUI grounding skills, paving the way for a visual GUI agent that executes instructions only relying on screenshots. As presented in Figure 2, *SeeClick* is a foundational model for GUIs, and tailored for adaption to agent tasks. Next, we introduce the birth of *SeeClick*, including the formalization of GUI grounding task, the construction of continual pre-training data, and training details.

3.1 GUI grounding for LVLMs

As GUI grounding is the core capability of *SeeClick*, we first elucidate how to train LVLM for language generation to perform grounding tasks. Given an interface screenshot s and a collection of elements $\{(x_i, y_i)|_i\}$ on it, where x_i denotes the textual description of the i-th element and y_i indicates the element's location (represented as a bounding box or point). As depicted in Figure 2(a), LVLM predicts the location of the element y based on the interface screenshot s and its textual description x, i.e. calculating $p(y|s, x)$.

A potential challenge is how LVLMs predict numerical coordinates in a language generation format. Previous studies (Chen et al., 2021; Wang et al., 2023; Shaw et al., 2023) divide the image into 1000 bins, and creating a new 1,000-token vocabulary $\{< p0 >, < p1 >, ..., < p999 >\}$ to represent the x and y coordinates. In this work, we adopt a more intuitive manner used in LVLMs (Chen et al., 2023b; Bai et al., 2023), treating numerical values as natural languages without any additional tokenization or pre-/post-processing. For instance, in Figure 2(a), for a smartphone screenshot and the instruction "View the new album of Jony J", we craft a query prompt: "In the UI, where should I click if I want to <instruction>?". Subsequently, we normally compute the cross-entropy loss between the model output and the ground truth "click (0.49, 0.40)" to optimize the LVLM.

3.2 Data Construction

We train *SeeClick* using three collections of data: web UI data crawled from the internet, mobile UI data reorganized from public datasets and general vision-language instruction-following data.

Web Data. Web UIs, featuring a variety of layouts and design styles across websites, are ideal for training LVLMs' general recognition and grounding capabilities across different GUI contexts. We collect approximately 300k web pages from the latest Common Crawl repository to serve as our training data for web UI. For each webpage s, we collect two types of elements from the HTML code as exemplified in Figure 3: (1) elements that display visible text content; and (2) elements with a special "title" attribute that display descriptive text when hovering. This method ensures that we gather a series of interactable elements y and their corresponding instructions x, while encompassing a wide range of text and icon elements. In addition to the grounding task $p(y|s, x)$, we also include web OCR task $p(x|s, y)$, predicting text description based on coordinates.

Figure 3: Example of two types of elements automatically collected from the webpage.

Mobile Data. For mobile UI, we include three types of data: widget captioning, mobile UI grounding, and mobile UI summarization. The widget captioning dataset provides language descriptions for mobile UI elements; for example, the description "play music" for the play button on a music player interface. We utilize the training split of the dataset provided by (Li et al., 2020b), containing nearly 20k screenshots, 40k widgets, and 100k descriptions. We derive mobile UI grounding data by reversing the process of widget captioning, treating language descriptions as instructions and corresponding widgets as target elements. To improve diversity, we also incorporate the automatically collected elements and instructions from RICO (Li et al., 2020a). The mobile data involves diverse elements and instructions, facilitating the generalization of *SeeClick*'s grounding proficiency to diverse GUI contexts. We finally include mobile UI summarization data (Wang et al., 2021) to enhance overall interface comprehension.

General Data. To maintain LVLM's general capacities on natural images, we incorporate the general vision-language instruction-following data from LLaVA (Liu et al., 2023a), covering conversation, detailed description, and complex reasoning.

Finally, we mix the data above and craft 30 task-specific prompts for each added GUI task, resulting in a 1M dataset to train *SeeClick*.

3.3 Training Details

We build *SeeClick* through continual pre-training on a recent advanced LVLM, Qwen-VL (Bai et al., 2023), which possesses grounding capabilities and a higher resolution of 448*448. We train Qwen-VL on the dataset we constructed (as described in Section 3.2) for about 10k steps (around 1 epoch) to obtain our GUI base model *SeeClick*. During training, we employ LoRA (Hu et al., 2021) to fine-tune both the visual encoder and LLM. Further details and task examples are provided in Appendix A.

4 *ScreenSpot*: A Grounding Benchmark

We recognize GUI grounding proficiency as essential for constructing visual GUI agents. However, the constrained capabilities of earlier vision-language models resulted in limited attention, with

LVLMs	Model Size	GUI Specific	Mobile		Desktop		Web		Average
			Text	Icon/Widget	Text	Icon/Widget	Text	Icon/Widget	
MiniGPT-v2	7B	✗	8.4%	6.6%	6.2%	2.9%	6.5%	3.4%	5.7%
Qwen-VL	9.6B	✗	9.5%	4.8%	5.7%	5.0%	3.5%	2.4%	5.2%
GPT-4V	-	✗	22.6%	24.5%	20.2%	11.8%	9.2%	8.8%	16.2%
Fuyu	8B	✓	41.0%	1.3%	33.0%	3.6%	33.9%	4.4%	19.5%
CogAgent	18B	✓	67.0%	24.0%	**74.2%**	20.0%	**70.4%**	28.6%	47.4%
SeeClick	9.6B	✓	**78.0%**	**52.0%**	72.2%	**30.0%**	55.7%	**32.5%**	**53.4%**

Table 1: Results of different LVLMs on *ScreenSpot*. The best results in each column are highlighted in **bold**. Benefiting from efficient GUI grounding pre-training, *SeeClick* significantly enhanced LVLMs' ability to locate GUI elements following instructions, and surpassed the strong baseline CogAgent with a smaller model size.

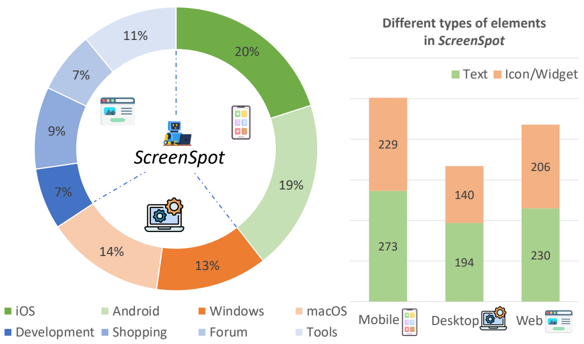


Figure 4: Statistic of our proposed GUI grounding benchmark *ScreenSpot*. The left illustrates the diverse GUI environments included. The right displays the types of elements within each GUI category.

scant research (Li et al., 2021; Li and Li, 2022; Zhang et al., 2023) largely confined to an Android dataset (Deka et al., 2017) collected in 2017.

To address this research gap, we introduce *ScreenSpot*, an up-to-date, realistic grounding evaluation benchmark encompassing various GUI platforms. It is designed to assess vision-language models' ability to locate screen elements based on instructions (Figure 2(b) provides some examples). *ScreenSpot* has two distinctive features: (1) Various GUI platforms. It includes over 600 interface screenshots from mobile (iOS, Android), desktop (macOS, Windows), and web platforms, along with 1200+ instructions and corresponding actionable elements; (2) Icons/Widgets. *ScreenSpot* includes a substantial number of icons and widgets in each GUI, which is more challenging to locate than texts (statistics are in Figure 4). See Appendix B for annotation details and examples.

To measure models' effectiveness in real-world scenarios, *ScreenSpot* is carefully curated to ensure the samples are novel and not included in existing training resources. We recruited experienced annotators to collect GUI interfaces and label instructions along with the bounding boxes for actionable elements. For mobile and desktop, annotators were asked to select commonly used apps and operations; for web, we chose several types of websites (development, shopping, forum, and tools) from the web environment WebArena (Zhou et al., 2023).

5 Experiments

In this section, we first evaluate the GUI grounding capabilities of representative LVLMs and our proposed *SeeClick*. Subsequently, we adapt *SeeClick* to mobile and web agent tasks, analyzing the correlation between the advanced grounding capacity and downstream task performance, while exploring the potential of purely vision-based GUI agents.

5.1 GUI Grounding on *ScreenSpot*

As the foundation of visual GUI agents, GUI grounding has not received adequate attention in current LVLMs evaluations (Liu et al., 2023b; Yu et al., 2023). Therefore, we evaluate LVLMs on our GUI-specific benchmark *ScreenSpot*.

Compared LVLMs & Evaluation. We primarily evaluated two types of LVLMs: (1) Generalist LVLMs capable of tasks such as dialogue, recognition and grounding, including MiniGPT-v2 (Chen et al., 2023a), Qwen-VL (Bai et al., 2023) and GPT-4V; (2) Recently released LVLMs specifically designed for GUI tasks, including Fuyu (Bavishi et al., 2023) and CogAgent (Hong et al., 2023).

Considering that GUI agents require clicking on the correct position, we calculate the click accuracy as the metric, defined as the proportion of test samples where the model predicted location falls in the ground truth element bounding box (Li et al., 2022; Zhang et al., 2023). More details about evaluation on *ScreenSpot* is in Appendix B.

Results. As shown in Table 1, while generalist LVLMs have excelled in natural image grounding, their GUI grounding performance on *ScreenSpot* is poor due to the significant differences between

GUIs and natural images. Even GPT-4V struggles with accurately locating screen elements.

In comparison, GUI-specific LVLMs have significant improvements. *SeeClick* achieved the best average performances across GUI platforms and two types of elements, even with fewer parameters than CogAgent. This demonstrates the efficiency of our GUI grounding pre-training; with the rich UI elements and diverse instructions collected from the web and mobile, *SeeClick* quickly learns to understand human instructions for element localization, even in completely unseen scenarios like iOS and desktop. *SeeClick* exhibits slightly inferior performance in locating text within desktop and web compared to CogAgent, possibly due to lower resolution and much smaller training data. Notably, all models struggle with locating icons/widgets, highlighting the difficulty of identifying and grounding non-text elements on GUIs, which is the unique challenge posed by *ScreenSpot*.

5.2 Visual GUI Agent Tasks

This section explores *SeeClick*'s application to mobile and computer agent tasks: MiniWob, AITW, and Mind2Web. We trained *SeeClick* on the respective training splits and tested it on the test sets. Across these tasks, with provided instructions and memory of previous actions, *SeeClick* determines the next action solely by observing interface screenshots. The detailed task settings, action formats and interaction examples are in Appendix C.

5.2.1 MiniWob

MiniWob (Shi et al., 2017) comprises about 100 types of web automation tasks, where the agent is asked to interact with a simplified web environment to accomplish human instructions. Existing open-source training data often lacks corresponding interface screenshots (Furuta et al., 2023). Therefore, we rollout 50 successful episodes using an LLM strategy for each task in (Zheng et al., 2023), resulting in a 2.8K episodes dataset to train *SeeClick*.

Compared Methods & Evaluation. We compared *SeeClick* with a range of offline training methods. Among these, the state-of-the-art method WebGUM (Furuta et al., 2023) uses screenshots as auxiliary input but still selects HTML elements as actions. Pix2Act (Shaw et al., 2023) is the only prior vision-based approach, trained with extensive demonstration data to perform actions. To verify the effectiveness of GUI grounding pre-training, we also report the results of the LVLM baseline Qwen-VL when trained with the same 2.8K dataset.

Methods	Modality	Dataset	Score
Compared with text-based models over 45 tasks			
CC-Net (SL)	DOM+Image	2.4M	35.6%
WebN-T5	HTML	12K	55.2%
MM-WebN-T5	HTML+Image	347K	63.4%
WebGUM	HTML+Image	2.8K	65.5%
WebGUM	HTML+Image	347K	**86.1%**
SeeClick	Image	2.8K	<u>73.6%</u>
Compared with vision-based models over 35 tasks			
CC-Net (SL)	Image	2.4M	23.4%
Pix2Act	Image	1.3M	64.6%
Qwen-VL	Image	2.8K	48.4%
SeeClick	Image	2.8K	<u>**67.0%**</u>

Table 2: Average scores of different methods on MiniWob. The best results in each setting are **bold**. Methods achieving the highest performance with limited data are <u>underlined</u>. *SeeClick* outperforms a range of offline training methods as a purely vision-based model.

Due to the variance in evaluation task sets among different methods (Liu et al., 2018; Furuta et al., 2023; Shaw et al., 2023), for fairness, we report performance in two groups based on the overlapping MiniWob tasks. We compute the success rate over 50 random seeds for each task and then compute the mean over all tasks as the final score. We provided task-wise scores in Appendix C.2.

Results. As depicted in Table 2, purely vision-based *SeeClick* surpassed strong baselines with substantially less training data. Notably, with an equivalent amount of 2.8K training data, it outperformed the offline sota WebGUM, which uses both HTML and screenshots as input. Moreover, thanks to LVLM's powerful reasoning and planning abilities and our GUI grounding pre-training, *SeeClick* exceeded the sota visual method Pix2Act, using less than 0.3% training data.

Furthermore, *SeeClick* significantly surpassed the LVLM baseline Qwen-VL by nearly 20 percentage points, underscoring the importance of GUI grounding in boosting LVLM's performance. To analyze in detail, we provide task-level comparisons in Figure 5. *SeeClick* notably excelled in tasks with dynamic interface layouts and element positions, confirming our hypothesis that general LVLMs struggle with accurately clicking, and *SeeClick* markedly improves this aspect.

5.2.2 AITW

We evaluate *SeeClick* in smartphone environments with Android automation dataset Android In The

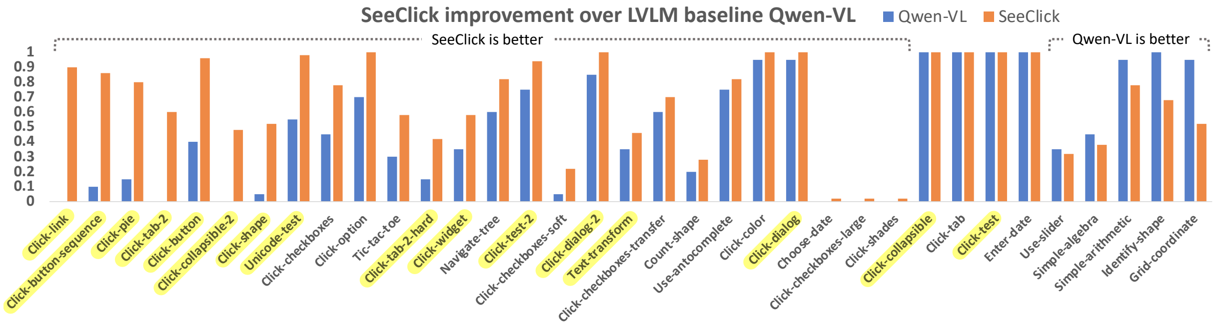


Figure 5: Comparison of *SeeClick* and Qwen-VL on MiniWob. Tasks marked with yellow shadows feature dynamic webpage layouts, simulating real-world GUI agent applications (details in appendix Figure 11). *SeeClick* outperformed Qwen-VL in most tasks, highlighting the effectiveness of GUI grounding pre-training.

Methods	Modality	General	Install	GoogleApps	Single	WebShopping	Overall	ClickAcc
ChatGPT-CoT	Text	5.9	4.4	10.5	9.4	8.4	7.7	-
PaLM2-CoT	Text	-	-	-	-	-	39.6	-
GPT-4V	Image	41.7	42.6	49.8	**72.8**	45.7	50.5	-
Qwen-VL	Image	49.5	59.9	46.9	64.7	50.7	54.3	57.4
SeeClick	Image	**54.0**	**66.4**	**54.9**	63.5	**57.6**	**59.3**	**66.4**

Table 3: Average scores of different methods on AITW. ClickAcc calculates the accuracy of click operation. The best results in each column are **bold**. *SeeClick* exhibits the best performance among competing baselines.

Wild (AITW) (Rawles et al., 2023), which encompasses 30k instructions and corresponding 715k operation trajectories. Previous approaches split train/val/test episode-wise, which poses a clear risk of overfitting due to: (1) instructions in the test set have appeared in training, and (2) an average of 20 similar trajectories per instruction. In this work, we opt for an instruction-wise split, with 545/688/306/700/700 instructions from General/Install/GoogleApps/Single/WebShopping respectively, and retain one trajectory per instruction. We selected 80% for training and the remaining for testing in each subset. This split avoids overfitting and reflects the performance of agents on unseen instructions. Further details are in Appendix C.3.

Compared Methods & Evaluation. We compare *SeeClick* with two types of baselines: (1) API-based LLMs such as ChatGPT-CoT (Zhan and Zhang, 2023), PaLM2-CoT (Rawles et al., 2023) and the latest GPT-4V (Yan et al., 2023); (2) Our trained LVLM baseline Qwen-VL.

We follow Rawles et al. (2023) to adopt the screen-wise action matching score as the main metric and additionally compute the click accuracy (ClickAcc), which calculates the accuracy when both reference and prediction are click operations.

Results. As illustrated in Table 3, *SeeClick* achieved the best average performance among both API-based LLMs and trained LVLMs. Specifically, *SeeClick* exhibited a 9% increase in click accuracy over Qwen-VL, supporting the idea that GUI grounding enhances agent task performance through precise clicking.

5.2.3 Mind2Web

To assess *SeeClick*'s capabilities in web navigation, we utilize the recently introduced Mini2Web dataset (Deng et al., 2023), which comprises over 2000 open-ended tasks collected from 137 real websites, each with high-level instruction and corresponding human action trajectory. Mind2Web was originally designed for text-based agents, which select actionable elements from simplified HTML in each step. This work explores visual web agents that predict click positions directly from screenshots. For this purpose, we parsed screenshots and target element bounding boxes from the raw dump of Mind2Web. To the best of our knowledge, this is the first attempt of web agents relying solely on screenshots as inputs for navigating real websites.

Compared Methods & Evaluation. We compare with html-based web agents Mind2Act (Deng et al., 2023) and our visual baseline Qwen-VL. Mind2Act employs a two-stage method, where a small LM first generates candidate elements from raw HTML, then a large LM selects the target via multi-choice QA; Mind2Act (gen) directly generates the target element instead. GPT-3.5 and GPT-4 adopt the same multiple-choice QA formulation and include three demonstrations for in-context learning.

Methods	w/o HTML	Cross-Task			Cross-Website			Cross-Domain		
		Ele.Acc	Op.F1	Step SR	Ele.Acc	Op.F1	Step SR	Ele.Acc	Op.F1	Step SR
MindAct (gen)	✗	20.2	52.0	17.5	13.9	44.7	11.0	14.2	44.7	11.9
MindAct	✗	**55.1**	75.7	**52.0**	**42.0**	65.2	**38.9**	**42.1**	66.5	**39.6**
GPT-3.5-Turbo	✗	20.3	56.6	17.4	19.3	48.8	16.2	21.6	52.8	18.6
GPT-4	✗	41.6	60.6	36.2	35.8	51.1	30.1	37.1	46.5	26.4
Qwen-VL	✓	15.9	86.7	13.3	13.2	**83.5**	9.2	14.1	84.3	12.0
SeeClick	✓	28.3	**87.0**	25.5	21.4	80.6	16.4	23.2	**84.8**	20.8

Table 4: Comparsion of methods on Mind2Web. The best results in each column are **bold**. Improvements of *SeeClick* over LVLM baseline are underline, with GUI grounding pre-training nearly doubling the step success rate.

We calculate element accuracy (Ele.Acc), Operation F1 (Op.F1) and step success rate (Step SR). For vision-based methods, a prediction is considered correct if the predicted coordinate falls in the target element's bounding box. All other settings are following (Deng et al., 2023).

Results. As displayed in Table 4, *SeeClick* nearly doubled the Ele.Acc and Step SR compared to Qwen-VL. This indicates that *SeeClick*'s improvement in GUI grounding correlates with enhanced performance in web agent tasks. HTML-based methods yield lower Op.F1 as around 20% of groundturth elements are filtered out during candidate generation. Although *SeeClick* can operate without extra HTML information, its performance trails sota HTML-based methods, since predicting click coordinates is much more difficult than choosing from HTML candidates. This highlights the difficulty of grounding in intricate interfaces, suggesting substantial room for improvement in visual agents for real-world application.

5.2.4 Grounding and Agent Performance

To investigate the correlation between grounding and agent performance, we analyze the average score improvements of several *SeeClick*'s checkpoints on *ScreenSpot* and three downstream tasks. As depicted in Figure 6, enhanced GUI grounding capacity consistently boosts agent task performance, highlighting its crucial role in developing advanced visual GUI agents.

5.2.5 *SeeClick* as Unified GUI Agent

To access the potential of vision-based solutions in unifying GUI agent tasks, we evaluated jointly training *SeeClick* on three downstream tasks. As shown in Table 5, the unified model exhibited a slight performance decline, possibly due to the significant distinct interface of different GUIs.

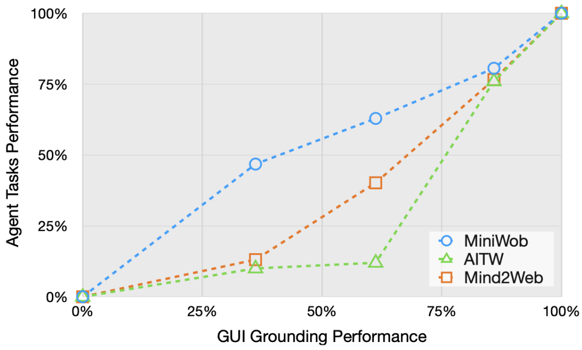


Figure 6: The correlation between agent tasks performance improvement and enhanced grounding ability.

	MiniWob	AITW	Mind2web
Qwen-VL$_{separate}$	48.4	54.3	11.5
SeeClick$_{separate}$	67.0	59.3	20.9
SeeClick$_{unified}$	64.1	57.1	19.5

Table 5: Separate v.s. unified training performance.

6 Conclusion

In this paper, we introduce a visual GUI agent - *SeeClick*, which only relies on screenshots for GUI task automation. We found a key challenge in developing such visual GUI agents: GUI grounding - the capacity to accurately locate screen elements based on human instructions. To address this challenge, we propose to enhance *SeeClick* via GUI grounding pre-training, and devise methods to automate the curation of GUI grounding data from web and mobile. For benchmarking the progress in GUI grounding, we created *ScreenSpot*, the first realistic evaluation dataset encompassing mobile, desktop, and web platforms. Results on *ScreenSpot* demonstrate a significant improvement of *SeeClick* over LVLM baselines. Moreover, comprehensive evaluations across three GUI automation tasks consistently support our finding that advancements in GUI grounding directly correlated with improved performance in downstream agent tasks.

Limitations

SeeClick currently simplifies the GUI action space to mainly focus on clicking and typing, excluding complex actions like dragging and double-clicking. Additionally, limited by the performance of open-source LVLMs, training on agent-specific data is necessary for *SeeClick* to execute multi-step tasks on interfaces like mobile and computer.

Ethical considerations

GUI agents are developed to automate tasks and enhance efficiency on digital devices. These technologies are especially significant for individuals with visual impairments. Here are some ethical considerations:

Privacy Issues. The operation of GUI agents involves accessing and interacting with user interfaces that may contain personal or sensitive information. Ensuring data protection and user consent are paramount to maintaining privacy integrity.

Safety in Read-World Interactions. When GUI agents interact with the real world, there's a risk of unintended harmful actions. Ensuring these agents operate within safe parameters is crucial to prevent negative outcomes.

Bias. The development of GUI agents must address potential biases in their algorithms, which could result in unequal performance across different user groups or interface designs. Mitigating bias is essential for equitable access and effectiveness.

Addressing these concerns requires ongoing research and development efforts, ensuring that the benefits of GUI agents are realized without compromising ethical standards.

References

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*.

Rohan Bavishi, Erich Elsen, Curtis Hawthorne, Maxwell Nye, Augustus Odena, Arushi Somani, and Sağnak Taşırlar. 2023. Introducing our multimodal models.

Andrea Burns, Deniz Arsan, Sanjna Agrawal, Ranjitha Kumar, Kate Saenko, and Bryan A Plummer. 2022. A dataset for interactive vision-language navigation with unknown command feasibility. In *European Conference on Computer Vision*, pages 312–328. Springer.

Jun Chen, Deyao Zhu, Xiaoqian Shen, Xiang Li, Zechun Liu, Pengchuan Zhang, Raghuraman Krishnamoorthi, Vikas Chandra, Yunyang Xiong, and Mohamed Elhoseiny. 2023a. Minigpt-v2: large language model as a unified interface for vision-language multi-task learning. *arXiv preprint arXiv:2310.09478*.

Keqin Chen, Zhao Zhang, Weili Zeng, Richong Zhang, Feng Zhu, and Rui Zhao. 2023b. Shikra: Unleashing multimodal llm's referential dialogue magic. *arXiv preprint arXiv:2306.15195*.

Ting Chen, Saurabh Saxena, Lala Li, David J Fleet, and Geoffrey Hinton. 2021. Pix2seq: A language modeling framework for object detection. In *International Conference on Learning Representations*.

Biplab Deka, Zifeng Huang, Chad Franzen, Joshua Hibschman, Daniel Afergan, Yang Li, Jeffrey Nichols, and Ranjitha Kumar. 2017. Rico: A mobile app dataset for building data-driven design applications. In *Proceedings of the 30th annual ACM symposium on user interface software and technology*, pages 845–854.

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*.

Hiroki Furuta, Ofir Nachum, Kuang-Huei Lee, Yutaka Matsuo, Shixiang Shane Gu, and Izzeddin Gur. 2023. Multimodal web navigation with instruction-finetuned foundation models. *arXiv preprint arXiv:2305.11854*.

Difei Gao, Lei Ji, Zechen Bai, Mingyu Ouyang, Peiran Li, Dongxing Mao, Qinchen Wu, Weichen Zhang, Peiyi Wang, Xiangwu Guo, et al. 2023. Assistgui: Task-oriented desktop graphical user interface automation. *arXiv preprint arXiv:2312.13108*.

Izzeddin Gur, Hiroki Furuta, Austin Huang, Mustafa Safdari, Yutaka Matsuo, Douglas Eck, and Aleksandra Faust. 2023. A real-world webagent with planning, long context understanding, and program synthesis. *arXiv preprint arXiv:2307.12856*.

Izzeddin Gur, Ulrich Rueckert, Aleksandra Faust, and Dilek Hakkani-Tur. 2018. Learning to navigate the web. 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*.

Geunwoo Kim, Pierre Baldi, and Stephen McAleer. 2023. Language models can solve computer tasks. *arXiv preprint arXiv:2303.17491*.

Bo Li, Yuanhan Zhang, Liangyu Chen, Jinghao Wang, Jingkang Yang, and Ziwei Liu. 2023. Otter: A multi-modal model with in-context instruction tuning. *arXiv preprint arXiv:2305.03726*.

Gang Li and Yang Li. 2022. Spotlight: Mobile ui understanding using vision-language models with a focus. In *The Eleventh International Conference on Learning Representations*.

Liunian Harold Li, Pengchuan Zhang, Haotian Zhang, Jianwei Yang, Chunyuan Li, Yiwu Zhong, Lijuan Wang, Lu Yuan, Lei Zhang, Jenq-Neng Hwang, et al. 2022. Grounded language-image pre-training. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10965–10975.

Yang Li, Jiacong He, Xin Zhou, Yuan Zhang, and Jason Baldridge. 2020a. Mapping natural language instructions to mobile ui action sequences. *arXiv preprint arXiv:2005.03776*.

Yang Li, Gang Li, Luheng He, Jingjie Zheng, Hong Li, and Zhiwei Guan. 2020b. Widget captioning: Generating natural language description for mobile user interface elements. *arXiv preprint arXiv:2010.04295*.

Yang Li, Gang Li, Xin Zhou, Mostafa Dehghani, and Alexey Gritsenko. 2021. Vut: Versatile ui transformer for multi-modal multi-task user interface modeling. *arXiv preprint arXiv:2112.05692*.

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

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. 2023a. Visual instruction tuning. In *Neural Information Processing Systems*.

Yuan Liu, Haodong Duan, Yuanhan Zhang, Bo Li, Songyang Zhang, Wangbo Zhao, Yike Yuan, Jiaqi Wang, Conghui He, Ziwei Liu, et al. 2023b. Mmbench: Is your multi-modal model an all-around player? *arXiv preprint arXiv:2307.06281*.

OpenAI. 2023. GPT-4 technical report.

Zhiliang Peng, Wenhui Wang, Li Dong, Yaru Hao, Shaohan Huang, Shuming Ma, and Furu Wei. 2023. Kosmos-2: Grounding multimodal large language models to the world. *arXiv preprint arXiv:2306.14824*.

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*.

Peter Shaw, Mandar Joshi, James Cohan, Jonathan Berant, Panupong Pasupat, Hexiang Hu, Urvashi Khandelwal, Kenton Lee, and Kristina Toutanova. 2023. From pixels to ui actions: Learning to follow instructions via graphical user interfaces. In *Advances in Neural Information Processing Systems*.

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.

Qiushi Sun, Zhangyue Yin, Xiang Li, Zhiyong Wu, Xipeng Qiu, and Lingpeng Kong. 2023. Corex: Pushing the boundaries of complex reasoning through multi-model collaboration. *arXiv preprint arXiv:2310.00280*.

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

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.

Wenhai Wang, Zhe Chen, Xiaokang Chen, Jiannan Wu, Xizhou Zhu, Gang Zeng, Ping Luo, Tong Lu, Jie Zhou, Yu Qiao, et al. 2023. Visionllm: Large language model is also an open-ended decoder for vision-centric tasks. *arXiv preprint arXiv:2305.11175*.

Zhiyong Wu, Chengcheng Han, Zichen Ding, Zhenmin Weng, Zhoumianze Liu, Shunyu Yao, Tao Yu, and Lingpeng Kong. 2024. Os-copilot: Towards generalist computer agents with self-improvement.

Fangzhi Xu, Zhiyong Wu, Qiushi Sun, Siyu Ren, Fei Yuan, Shuai Yuan, Qika Lin, Yu Qiao, and Jun Liu. 2023. Symbol-llm: Towards foundational symbol-centric interface for large language models. *arXiv preprint arXiv:2311.09278*.

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

Qinghao Ye, Haiyang Xu, Guohai Xu, Jiabo Ye, Ming Yan, Yiyang Zhou, Junyang Wang, Anwen Hu, Pengcheng Shi, Yaya Shi, et al. 2023. mplug-owl: Modularization empowers large language models with multimodality. *arXiv preprint arXiv:2304.14178*.

Weihao Yu, Zhengyuan Yang, Linjie Li, Jianfeng Wang, Kevin Lin, Zicheng Liu, Xinchao Wang, and Lijuan Wang. 2023. Mm-vet: Evaluating large multimodal models for integrated capabilities. *arXiv preprint arXiv:2308.02490*.

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

Zhizheng Zhang, Wenxuan Xie, Xiaoyi Zhang, and Yan Lu. 2023. Reinforced ui instruction grounding: Towards a generic ui task automation api. *arXiv preprint arXiv:2310.04716*.

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*.

Longtao Zheng, Rundong Wang, Xinrun Wang, and Bo An. 2023. Synapse: Trajectory-as-exemplar prompting with memory for computer control. In *NeurIPS 2023 Foundation Models for Decision Making Workshop*.

Shuyan Zhou, Frank F Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Yonatan Bisk, Daniel Fried, Uri Alon, et al. 2023. Webarena: A realistic web environment for building autonomous agents. *arXiv preprint arXiv:2307.13854*.

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 Details of *SeeClick* Pre-training

A.1 Pre-training Tasks

SeeClick employs pre-training tasks as outlined in Table 6. For the grounding task, we incorporate two forms: predicting center point coordinates (text_2_point) and predicting bounding box (text_2_bbox). For the task of generating text for elements (similar to OCR), we also include two categories: predicting text based on center point (point_2_text, widget captioning) coordinates and based on bounding boxes (bbox_2_text). Our preliminary experiments indicated that predicting points was slightly better than bounding boxes, likely due to the variable sizes of UI elements. Consequently, we increased the proportion of data with point localization. Finally, about 1 million samples are used for the continual pre-training of *SeeClick*.

For tasks involving coordinates, positions are represented as either the point (x,y) or the bounding box of (left, top, right, down), where each value is a two-decimal place number in the range [0,1] indicating the ratio of the corresponding position to the width or height of the image. Figure 7 provides some examples of the pre-training data.

Domain	Task	Sample Num
Web	text_2_point	271K
	text_2_bbox	54K
	point_2_text	54K
	bbox_2_text	54K
Mobile	text_2_point	274K
	text_2_bbox	56K
	UI summarization	48K
	widget captioning	42K
General	LLaVA	145K
Total		1M

Table 6: All training data used by *SeeClick*.

A.2 Training Configurations

We employed the aforementioned data for continual pre-training of Qwen-VL-Chat to develop *SeeClick*. To enhance LVLM's understanding of GUI images, we unlocked the gradients of its visual encoder and applied LoRA for fine-tuning. We adopt AdamW as the optimizer and use a cosine annealing scheduler with an init learning rate of 3e-5 and a global batch size of 64. All training takes around 24 hours on 8 NVIDIA A100 GPUs.

Figure 7: Examples of *SeeClick* pre-training tasks.

B *ScreenSpot* Annotation & Evaluation

B.1 Human Annotation

We convened four experienced annotators, all either Ph.D. or master students in computer science, proficient in using mobile phones and computers and familiar with GUI operations. Initially, we assigned different GUI types to the annotators, such as iOS, Windows, and Web. Then, annotators were required to capture screenshots during their routine use (e.g., various apps) and subsequently annotate the clickable regions of frequently interacted elements using bounding boxes with annotation tool [1]. Finally, these annotators were instructed to write corresponding English text commands for the annotated screen elements. All annotated interfaces and operational elements were in English and post-processed to remove personal information.

[1]http://makesense.bimant.com

B.2 Sample Showcase

Figure 10 provides more examples of *ScreenSpot*, which contains a variety of common GUI scenarios for mobile, desktop, and web platforms.

B.3 Evaluation Detail

For comparing baselines, we tested the models' grounding capabilities using their officially recommended approach. For instance, with CogAgent, we randomly selected prompts from the official set provided, such as "What steps do I need to take to <instruction>? (with grounding)", then the output coordinates (or the centers of bounding boxes) were taken as predicted points. For GPT-4V, we follow Yang et al. (2023b) to enable it to locate screen elements based on instructions. *SeeClick*'s predictions with points were marginally better than bounding boxes, thus we selected point prediction for final evaluation.

B.4 *SeeClick* Case Study & Error Analysis

Figure 8 presents some examples of *SeeClick* on *ScreenSpot*. *SeeClick* can comprehend human instructions and accurately locate screen elements. To conduct a detailed analysis of localization performance, we quantified the distances between predicted points and ground truth (the center of target elements) in Figure 9. It's noteworthy that even incorrect predictions mostly occur near the target bounding box, suggesting the model recognizes the target but needs improvement in fine-grained localization.

C Downstream Agent Tasks

In this section, we first detail the formulation of *SeeClick* as a visual GUI agent, then separately introduce the settings for three downstream tasks, and finally show *SeeClick*'s interaction cases with the GUI across these tasks.

C.1 Formulation of *SeeClick* as Visual GUI Agent

Action Space *SeeClick* involves common human-UI interaction operations. Following AITW, we assigned an `action_type id` to each action type for model prediction.

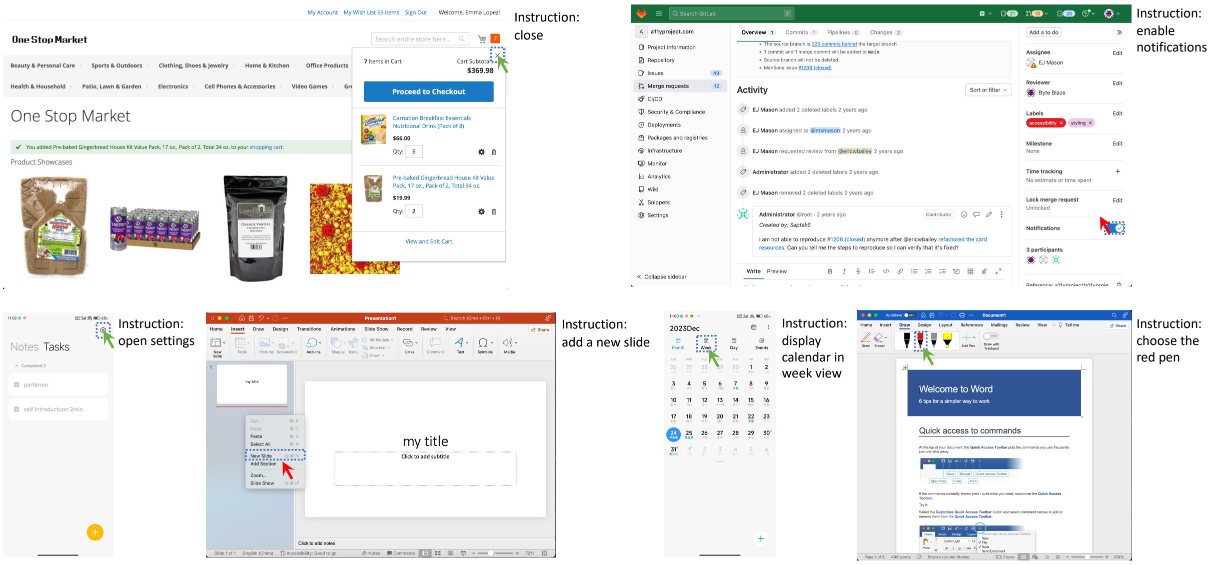


Figure 8: *SeeClick* on *ScreenSpot*. Blue dashed boxes represent the ground truth bounding boxes, while green and red pointers indicate correct and incorrect predictions.

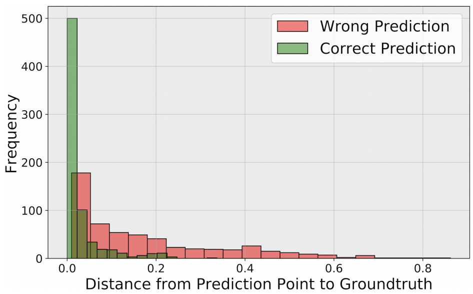


Figure 9: Distance distribution of prediction point to ground truth. Most incorrect predictions are also close to the answer, suggesting the model recognizes the target but needs improvement in fine-grained localization.

- `click(x,y)`: `4`. A click action at (x,y), where each value is a [0,1] number indicating the ratio of the corresponding position to the width or height of the image.
- `type("typed_text")`: `3`. An action of typing a piece of text.
- `select("value")`: `2`. An action for selecting an option from a dropdown menu on a webpage.
- `swipe(direction)`: Swipe actions for the screen, swipe up/down/left/right are assigned the ids `1`, `0`, `8`, and `9` respectively.
- `PRESS BACK`: `5`. The action for returning to the previous step.
- `PRESS HOME`: `6`. The action for returning to the homepage.
- `PRESS ENTER`: `7`. The action of pressing the ENTER key to submit input content.

The first two actions, clicking and typing, are universally applicable across various GUIs. The third action, select, is defined according to the specifications in Mind2Web. The latter four actions, along with two additional states, `TASK COMPLETE` and `TASK IMPOSSIBLE`, are defined following the AITW framework for Android environments.

Agent Formulation *SeeClick* is an autonomous agent capable of executing human instructions on GUIs. It takes as input the instruction, a screenshot of the current interface and a series of (k=4 in our setting) previous actions, to predict the next action to be taken. Specifically, *SeeClick* uses the following prompt to execute each step of the agent:

<img>Image</img>
User: Please generate the next move according to the UI screenshot, instruction and previous actions.
Instruction:
<instruction>
Previous actions:
Step1: <step1>
Step2: <step2>
Step3: <step3>
Step4: <step4>
SeeClick: <next action>

During training and testing, we organize the data by step into the format described above.

C.2 MiniWob

MiniWob is a classic simplified web agent environment, built on Chrome, allowing low-level operations such as clicking and typing. It comprises around 100 tasks, where each task can templatize

	Gen.	Inst.	GApps.	Sing.	WShop.	Ovr.
Auto-UI	**68.2**	76.9	71.4	84.6	70.3	74.3
CogAgent	65.4	78.9	75.0	**93.5**	71.1	**76.9**
SeeClick	67.6	**79.6**	**75.9**	84.6	**73.1**	76.2

Table 7: Comparison on the origin split of AITW.

random variants and corresponding instructions controlled by a random seed, creating up to billions of possible task instances. We use 50 successful trajectories for each task provided in (Zheng et al., 2023) for training and test each task with 50 random seeds, following standard practices.

We report the average success rate across random seeds and tasks, automatically provided by the MiniWob environment. A task is considered successfully completed if executed correctly, while incorrect executions or exceeding the maximum number of actions (set as 30 here) are counted as failures. For the baselines in Table 2, we use the task-wise scores provided in their papers to calculate the average score for tasks overlapping with *SeeClick*. We also provided a task-wise comparison in Table 8.

C.3 AITW

AITW is a recently collected dataset for Android smartphone automation, where each sample comprises an instruction and an action trajectory with screenshots. AITW is divided into five subsets: General, Install, GoogleApps, Single, and WebShopping, totally including over 30K instructions and 700K episodes.

Despite AITW's large scale, as stated in Section 5.2.2, the current train-test split poses a significant risk of overfitting, leading to experimental results that do not accurately reflect an agent's generalization ability in the real world. We also conducted experiments on *SeeClick* using the origin split, as shown in Table 7, *SeeClick* is comparable to CogAgent's performance. We believe that due to the severe overfitting, designing new agent frameworks or enlarging model size is unlikely to yield much improvements on this split.

To address the aforementioned issue, we propose to divide the train/val/test in an instruction-wise manner. Specifically, we selected 545/688/306/700/700 instructions from the General/Install/GoogleApps/Single/WebShopping subsets, and retained only one annotated episode for each instruction. To avoid imbalance in joint training, we randomly chose 700 instructions from Single and WebShopping. Given the similarity among instructions within Single and WebShopping, these 700 instructions are representative of performance on these two subsets. Next, we allocate 80% for training and the remaining 20% for testing, and select additional 5*100 episodes to form the validation set from the origin data. The data used for training, validation, and testing will be open-sourced to serve as an effective evaluation.

The other settings are consistent with previous work, calculating a screen-wise matching score that considers both the correctness of the action type and its value (e.g., the click point or typed text). The screen-wise matching score is correlates with the task completion score judged by humans (Rawles et al., 2023).

C.4 Mind2web

Mind2Web is a recently proposed dataset for developing generalist web agents for real-world websites, originally designed for text-based agents. Therefore, the origin observation in each step only includes the HTML code of the current webpage. To train and evaluate visual-based agents, we extracted web screenshots and the bounding boxes of target operational elements for each step from Mind2Web's raw dump. One issue with Mind2Web's original HTML observation is that it captures the entire page, including scrolling, with its screenshots being long captures (e.g., 1920*12000). Predicting operational positions from such high-resolution long screenshots is impractical for current LVLMs and does not align with human operations. To address this, for target elements not at the top, we randomly crop around their location, maintaining a consistent screenshot resolution of 1920*1080 for all observed interfaces.

Mind2Web first calculates Element Accuracy (Ele.Acc) which compares the predicted element with groundtruth, and Operation F1 (Op.F1) which calculates the token-level F1 score for the predicted operation. Operation F1 is equivalent to the accuracy of click operations but takes into account the correctness of input values for type and select operations. For our vision-based approach, Element Accuracy is computed as the accuracy of predicted click points falling in the groundtruth elements' bounding box. Then, a step is considered successful (Step SR) if both the predicted element and operation are correct.

C.5 Case Study

MiniWob Figure 11(a) illustrates the difference between static and dynamic layout tasks. Static layout tasks have fixed element positions during training and testing, while dynamic layout tasks display varying interfaces and element positions with instructions, further challenging the agent's ability to accurately locate the target. Figure 11(b) provides examples of *SeeClick*'s interaction with MiniWob. *SeeClick* relies solely on the interface screenshot for arithmetic, reasoning, etc.

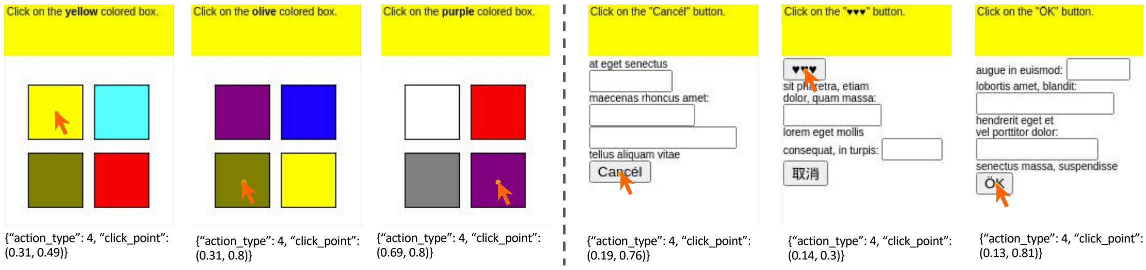
AITW Figure 12 provides *SeeClick*'s operations on AITW. Predictions marked in red below indicate that they were computed as incorrect in AITW. Some errors occur because the current step's answer is not unique. For example in step 5, the model's predicted input "DuckDuckGo Privacy Browser" is also a potentially correct action.

Mind2Web Figure 13 shows several examples of *SeeClick* on the real-world website benchmark Mind2Web. *SeeClick* can comprehend instructions and click on the correct elements within complex interfaces.

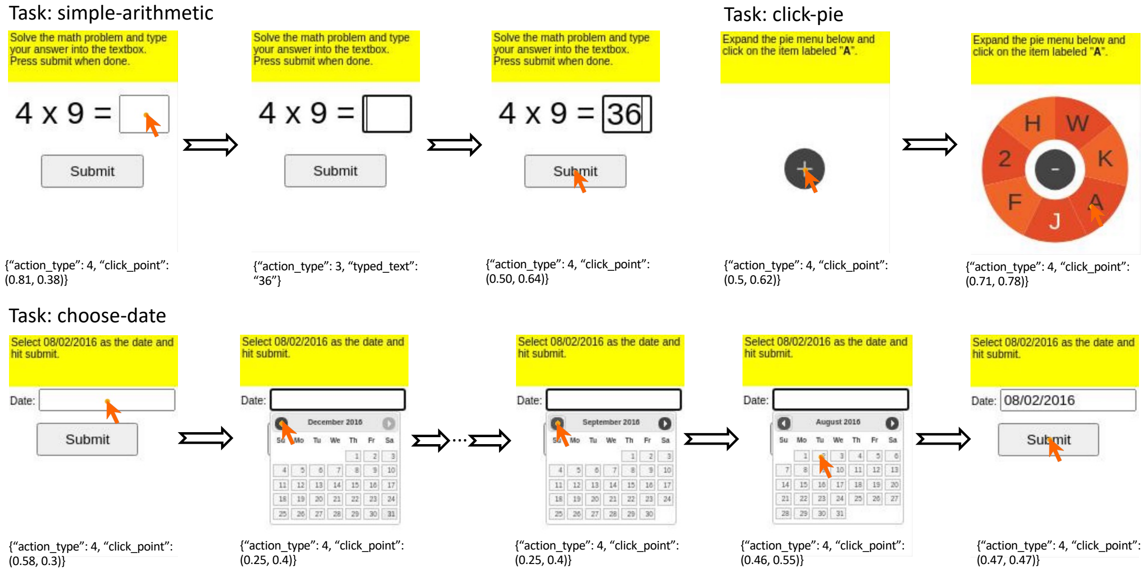
	CC-Net (SL)	WebN-T5	WebGUM	Pix2Act	Qwen-VL	*SeeClick*
Choose-date	0.12	0.00	0.13	0.06	0.0	0.02
Click-button	0.78	1.0	1.0	0.32	0.42	0.96
Click-button-sequence	0.47	1.0	1.0	1.0	0.08	0.86
Click-checkboxes	0.32	0.96	1.0	0.99	0.44	0.78
Click-checkboxes-large	0.0	0.22	0.99	1.0	0.0	0.02
Click-checkboxes-soft	0.04	0.54	0.98	0.91	0.06	0.22
Click-checkboxes-transfer	0.36	0.63	0.99	0.76	0.60	0.70
Click-collapsible-2	0.17	0.00	0.95	0.31	0.0	0.48
Click-collapsible	0.81	0.00	0.98	0.80	1.0	1.0
Click-color	0.82	0.27	0.34	0.88	0.96	1.0
Click-dialog	0.95	1.0	1.0	0.12	0.96	1.0
Click-dialog-2	0.88	0.24	0.43	0.73	0.84	1.0
Click-link	0.59	1.0	1.0	0.86	0.0	0.90
Click-option	0.21	0.37	1.0	0.0	0.70	1.0
Click-pie	0.15	0.51	0.99	0.81	0.16	0.80
Click-shades	0.04	0.0	0.0	0.76	0.0	0.02
Click-shape	0.11	0.53	0.72	0.19	0.04	0.52
Click-tab	0.95	0.74	1.0	0.54	1.0	1.0
Click-tab-2	0.27	0.18	0.95	0.52	0.0	0.60
Click-tab-2-hard	0.19	0.12	0.95	0.0	0.16	0.42
Click-test	1.0	1.0	1.0	1.0	1.0	1.0
Click-test-2	0.95	1.0	1.0	1.0	0.72	0.94
Click-widget	0.56	1.0	1.0	0.87	0.38	0.58
Count-shape	0.21	0.41	0.68	0.0	0.20	0.28
Copy-paste	0.04	-	-	-	0.96	0.80
Copy-paste-2	0.01	-	-	-	0.96	0.80
Email-inbox	0.09	0.38	0.99	-	0.08	0.80
Email-inbox-forward-nl	0.0	0.6	1.0	-	0.24	0.74
Email-inbox-forward-nl-turk	0.0	0.33	1.0	-	0.16	0.56
Email-inbox-nl-turk	0.05	0.23	0.98	-	0.40	0.68
Enter-date	0.02	0.0	1.0	0.59	1.0	1.0
Enter-password	0.02	0.97	1.0	-	1.0	1.0
Enter-text	0.35	0.89	1.0	-	1.0	1.0
Enter-text-dynamic	0.39	0.98	1.0	-	0.96	1.0
Focus-text	0.99	1.0	1.0	-	1.0	1.0
Focus-text-2	0.96	1.0	1.0	-	0.84	0.96
Find-word	0.05	-	-	-	1.0	0.10
Grid-coordinate	0.66	0.49	1.0	0.97	0.96	0.52
Guess-number	0.21	0.0	0.11	-	1.0	1.0
Login-user	0.0	0.82	1.0	-	1.0	1.0
Login-user-popup	0.02	0.72	0.99	-	0.86	0.98
Multi-layouts	0.00	0.83	1.0	-	0.44	0.72
Multi-orderings	0.0	0.88	1.0	-	0.42	0.86
Identify-shape	0.68	-	-	0.94	1.0	0.68
Navigate-tree	0.32	0.91	1.0	0.07	0.60	0.82
Search-engine	0.15	0.34	0.96	-	0.56	0.84
Simple-algebra	0.03	-	-	0.99	0.48	0.38
Simple-arithmetic	0.38	-	-	0.67	0.92	0.78
Text-transform	0.19	-	-	0.91	0.36	0.46
Tic-tac-toe	0.32	0.48	0.56	0.76	0.30	0.58
Unicode-test	0.86			0.64	0.54	0.98
Use-autocomplete	0.07	0.22	0.98	0.95	0.72	0.82
Use-slider	0.18	-	-	0.69	0.38	0.32
Use-spinner	0.47	0.07	0.11	-	0.24	0.16
Read-table	0.01	-	-	-	0.90	0.72
Average	0.336 (55)	0.552 (45)	0.861 (45)	0.646 (35)	0.564 (55)	0.712 (55)

Table 8: Mean scores across 55 MiniWob tasks.

Figure 10: More examples of GUI grounding benchmark *ScreenSpot*.



(a) Comparison between static layout (left, click-color) and dynamic layout (right, unicode-test).



(b) Example episodes of SeeClick on MiniWob tasks.

Figure 11: Example episodes of *SeeClick* on MiniWob. The model's prediction output is below the screenshot, with action_type 4 indicating a click and action_type 3 indicating typing.

Instruction: open app "DuckDuckGo Privacy Browser" (install if not already installed) and enter user name: "cleaving@outlook.com" and password: "freighters"

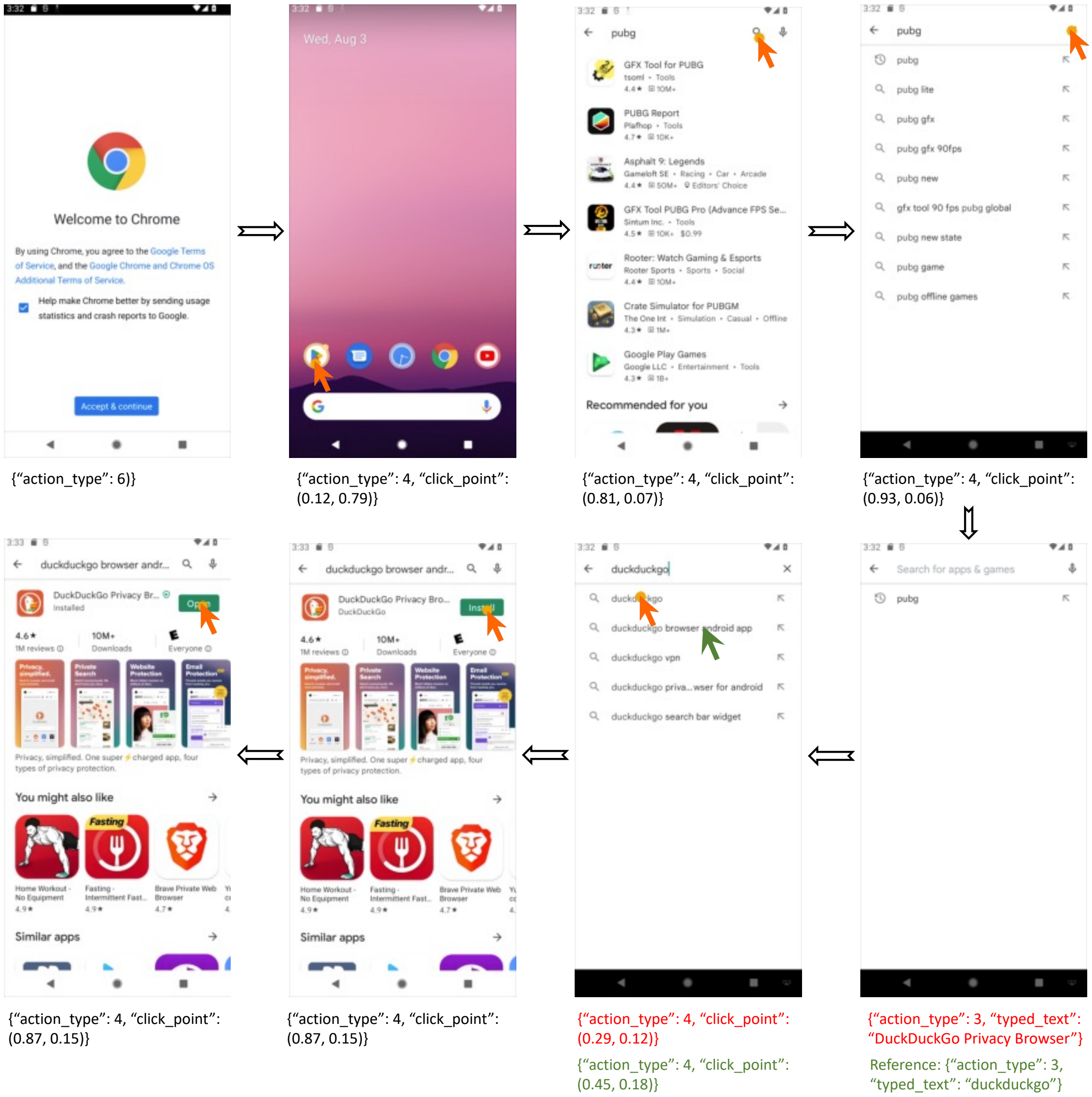


Figure 12: Example episodes of *SeeClick* on AITW. The model's prediction output is below the screenshot, with action_type 4 indicating a click, action_type 3 indicating typing and action_type 6 indicating PRESS HOME. Steps with the red prediction and green reference indicate a failed step.

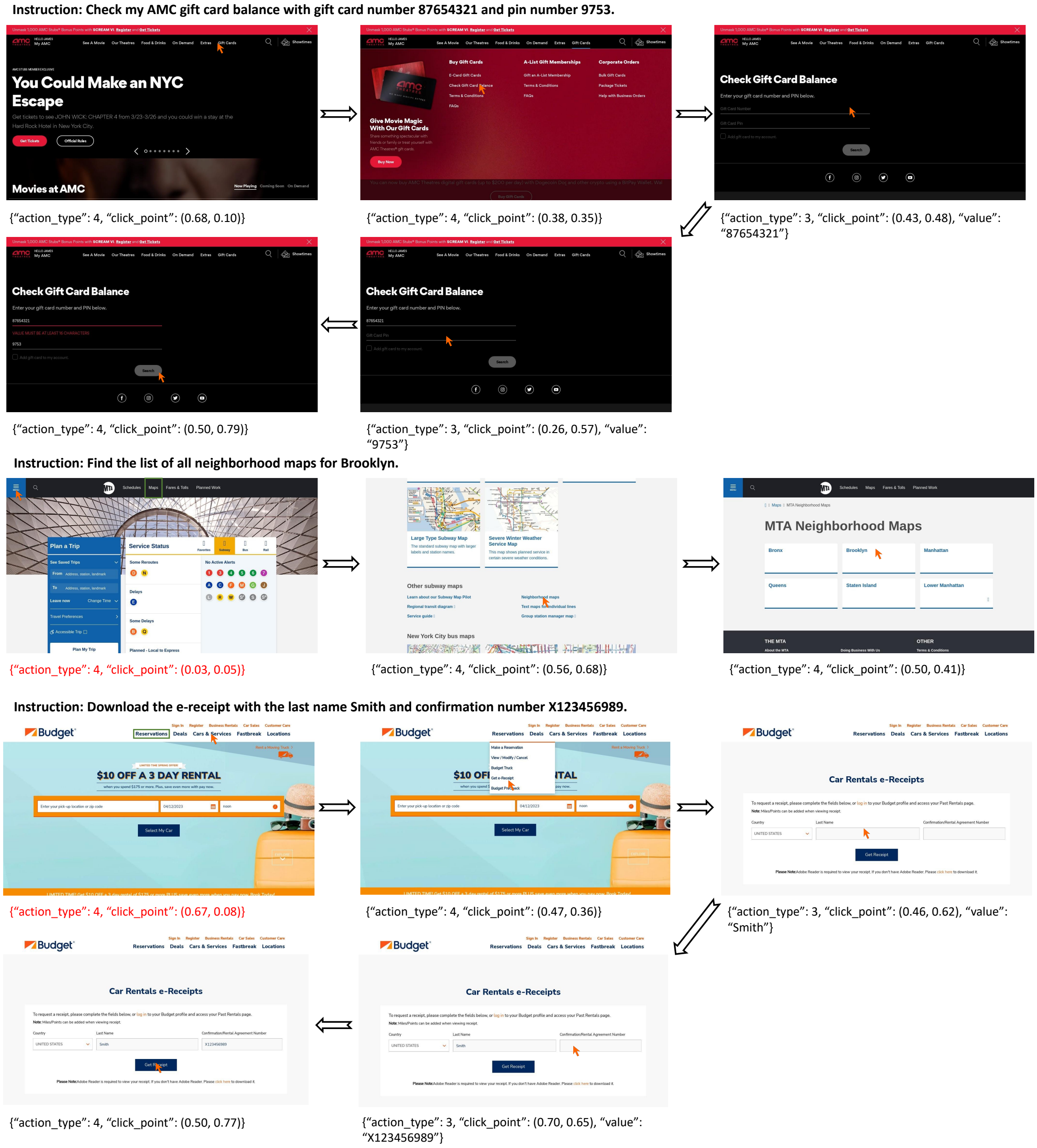


Figure 13: Example episodes of *SeeClick* on Mind2Web. The model's prediction output is below the screenshot, with action_type 4 indicating a click and action_type 3 indicating typing. Steps with the red prediction and green reference bounding box indicate a failed step.