
Navigating the Digital World as Humans Do: UNIVERSAL VISUAL GROUNDING FOR GUI AGENTS

Boyu Gou¹ Ruohan Wang¹ Boyuan Zheng¹ Yanan Xie² Cheng Chang² Yiheng Shu¹
Huan Sun¹ Yu Su¹

¹The Ohio State University ²Orby AI

{gou.43, sun.397, su.809}@osu.edu, yan'an@orby.ai

ABSTRACT

Multimodal large language models (MLLMs) are transforming the capabilities of graphical user interface (GUI) agents, facilitating their transition from controlled simulations to complex, real-world applications across various platforms. However, the effectiveness of these agents hinges significantly on the robustness of their grounding mechanisms. Prevalent GUI agents predominantly utilize text-based inputs such as HTML or accessibility trees, which, despite their utility, often introduce noise, incompleteness, and increased computational overhead. In this paper, we propose SeeAct-V, a generic vision-only framework for building GUI agents. It involves an MLLM as a planner to determine the next action, as well as a grounding model to retrieve the coordinates of target elements from screenshots. We introduce UGround, a universal pixel-level visual grounding model developed specifically for GUIs. This model, trained on 1.3 million diverse samples, is designed to ground open-ended element descriptions directly via pixel coordinates, and to function across different operating systems. Our comprehensive evaluation across six benchmarks, including desktop, mobile, and web platforms, demonstrates that UGround not only outperforms existing visual grounding models, but also matches or exceeds the performance of state-of-the-art methods that rely on HTML or accessibility trees. These results underscore UGround’s practicability in significantly advancing the field of vision-based GUI agents, illustrating its ability to navigate digital environments with human-like perception and precision.



Figure 1: Examples of agent tasks across benchmarks, and performance on the **GUI grounding benchmark** (♣: ScreenSpot), **offline agent benchmarks** (♠: Multimodal-Mind2Web, AndroidControl, and OmniAct), and **online agent benchmarks** (♥: Mind2Web-Live and AndroidWorld).

1 INTRODUCTION

GUI (graphical user interface) agents, autonomous agents acting in the digital world via operating on GUIs, have been rapidly co-evolving with large language models (LLMs). On the one hand, the general multimedia understanding and production capability of (multimodal) LLMs empower GUI agents to generalize beyond simple simulated settings (Shi et al., 2017; Humphreys et al., 2022) to diverse and complex real-world environments, including the web (Deng et al., 2023; Zhou et al., 2023;

Yao et al., 2022), desktop (Xie et al., 2024; Wu et al., 2024) and mobile operating systems (Rawles et al., 2023; 2024; Yan et al., 2023). On the other hand, GUI agents have become one of the most important testbeds for LLMs, providing both the necessary breadth and depth for driving continued development as well as a pathway to many commercially viable automation applications.

Most humans perceive the digital world visually and act via keyboards, mice, or touchscreens. In principle, the embodiment of a GUI agent should already be *complete* if it can 1) visually perceive the GUI renderings, and 2) have effectors equivalent to a keyboard for typing and equivalent to a mouse or touchscreen for pixel-level operations like clicking and hovering.¹ However, current GUI agents assume more than that. For perception, most current agents rely on reading the underlying text-based representations such as HTML or accessibility (a11y) trees (Deng et al., 2023; Gur et al., 2024; Zhou et al., 2023).² Only with the recent advances in multimodal LLMs (MLLMs) does visual perception become broadly viable, but text-based representations are still used jointly (Zheng et al., 2024; Koh et al., 2024; Zhang et al., 2024a). For effectors, most current agents act via selecting from a list of options, e.g., HTML elements (Deng et al., 2023; Zheng et al., 2024) or labeled bounding boxes (He et al., 2024; Zhang et al., 2024a) instead of pixel-level operations on the GUI. Obtaining those options in turn requires access to text-based representations and/or separate models for detecting objects and text (Wang et al., 2024; Kapoor et al., 2024).

However, there is no free lunch, and these additional requirements come with their limitations. On the one hand, *text-based representations are noisy and incomplete*. The full HTML contains a considerable amount of irrelevant information. The a11y tree is more compact and mainly contains semantic information, but similar to other voluntary meta annotations, it widely suffers from incomplete and incorrect annotations.³ In contrast, visual renderings, by design, are information-complete and only contain information relevant to users. On the other hand, *the additional input increases latency and inference costs*. Zheng et al. (2024) found that HTML can consume up to 10 times more tokens to encode than the corresponding visual. Meanwhile, obtaining the a11y tree can be time-consuming in itself, especially in OS environments. The added latency and cost at every step are further compounded in the long-horizon agent tasks, significantly compromising user experience and practicality.

In this work, we are interested in *how far a human-like vision-only embodiment with visual input and pixel-level operations can go*. There have been a few attempts (Shaw et al., 2023; Hong et al., 2024; Cheng et al., 2024), but they are rarely adopted in state-of-the-art solutions. We find that a major bottleneck is *grounding*, i.e., mapping textual plans generated by an LLM to the precise locations on the GUI. There are three desiderata for GUI agent grounding: 1) *High accuracy*. A single grounding error can get an agent stuck and fail the whole task. 2) *Strong generalization*. It should work on different GUIs: desktop (Windows, Linux, macOS), mobile (Android, iOS), different websites, etc. 3) *Flexibility*. It should plug and play in different MLLMs instead of being tightly coupled with a certain model. Existing visual grounding methods for GUI agents (Shaw et al., 2023; Hong et al., 2024; Cheng et al., 2024) fall short in meeting these desiderata, hindering the advances towards vision-only GUI agents.

The main contributions of this work are three-fold:

1. We make careful arguments and a strong case for human-like vision-only embodiment for GUI agents that perceive the digital world entirely visually and take pixel-level operations on the GUI.
2. Through a series of data and modeling innovations, we develop UGround, a universal pixel-level visual grounding model for GUI agents that is highly accurate and works universally across different GUIs and with different MLLMs.
3. We present the most comprehensive evaluation for GUI agents to date, covering six benchmarks spanning three categories: grounding (desktop/mobile/web), offline agent evaluation (desktop/mobile/web), and online agent evaluation (mobile/web). The results demonstrate: 1) UGround substantially outperforms existing visual grounding models for GUI agents across the board, by up to 28% absolute. 2) With a strong visual grounding model like UGround, GUI agents can achieve on

¹ Except for auditory perception, which is out of scope of this study. ² The a11y tree is a compact yet informative representation intended for assistive technologies to facilitate people with disabilities, e.g., visual impairment. ³ A 2024 survey over the top one million websites found that 95.9% of the home pages had accessibility conformance errors such as missing alternative text for images or missing form input labels, with an average of 56.8 errors per page (WebAIM, 2024).

par or better end-to-end performance than state-of-the-art agents that use additional text-based input, clearly demonstrating the promise of GUI agents that navigate the digital world as humans do.

2 METHOD

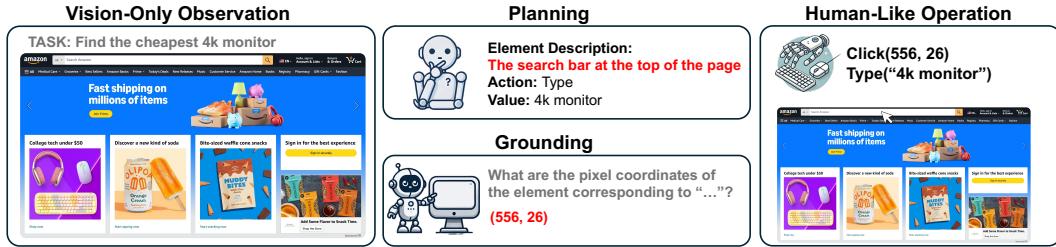


Figure 2: **SeeAct-V Framework.** SeeAct-V is designed to operate using screenshots as the sole input source, without relying on HTML or accessibility trees, and to leverage pretrained MLLMs. It involves an MLLM as a planning model, and a visual grounding model to retrieve coordinates from screenshots. The screenshot and the task are first analyzed by the planner, such as GPT-4, to determine the next action. It is prompted to generate an element description if the action involves an element or a specific area. Based on the description, a grounding model identifies the element coordinates from the screenshot. Finally, we perform human-like interactions at the coordinates.

2.1 OVERVIEW

Visual grounding is a vital capability of GUI agents to execute textual plans. Previous research has been largely around SoM method (He et al., 2024; Koh et al., 2024; Xie et al., 2024) or training unified models that are capable of pixel-level visual grounding (Cheng et al., 2024; Hong et al., 2024). In contrast, we introduce a two-module vision only framework, SeeAct-V, with minimalist designs. Similar to previous work (Li et al., 2020a; Zheng et al., 2024), it has 1) A “planner”, which generates a textual description about the target element or area involved in the next step.¹ 2) A grounding model, which takes the textual plan generated by the planner and outputs the precise target element on the GUI to interact with. This two-module framework is universal across all GUI platforms. As mentioned earlier, existing GUI agents leverage text-based representations such as HTML or accessibility trees for either the planner or the grounding module or both and suffer from a number of limitations. In contrast, we advocate for vision-only GUI agents here, which do not require accessibility trees, OCR, or icon detection. With various pretrained MLLMs such as GPT-4V and LLaVA series applicable as planners, our main mission in this work is to develop a universal visual grounding model to empower the second module of SeeAct-V.

Towards this mission, we develop UGround, which addresses critical challenges of visual grounding in GUI agents: handling large variable image sizes and small icons, understanding intricate and noisy UIs, and comprehending diverse referring expressions. These challenges have not been effectively addressed by previous lower-resolution models, OCR or icon detection models, or traditional visual grounding models. We aspire to elevate the grounding performance to a new level of accuracy with UGround and make it a robust universal model for GUI visual grounding.

2.2 DATA CONSTRUCTION

To train a good visual grounding model that can generalize across applications, websites, and diverse referring expressions, we need a large, high-quality, and diverse set of <screenshot, element description, element coordinate> triplets.

Webpages offer a rich source of elements with accessible coordinates, many of which are labeled by text or ARIA labels, making them ideal for data synthesis. In addition, given the similarities among

¹ Here we use “planner” to refer to a *textual* plan/description about the next step, rather than a directly executable plan as in traditional planning (Kambhampati et al., 2024).

GUIs, we hypothesize that web data can effectively generalize to other platforms. Consequently, our approach begins with low-cost synthesis of GUI visual grounding data using webpage metadata.

Common Attributes in GUI Referring Expressions. Previous work often train with metadata of webpage elements (Hong et al., 2024; Cheng et al., 2024). However, their performance still falls short of a desired level for SeeAct-V. GUI referring expressions can be very diverse, which places high demands on the training data. For example, an element can be described in great detail, or simply as “*the button at the top right corner*”. Simply training on HTML attributes of the element does not cover the latter description, not to mention even more complex cases. Nevertheless, on the other hand, referring expressions of GUI elements actually have distinct patterns, which can help design better data to fit and generalize to the highly diverse expressions used in practice.

Based on our analysis, three types of attributes are commonly used to refer to GUI elements (illustrative examples are shown in Figure 3): 1) **Visual Features**, i.e., salient visual features displayed in screenshots, such as text content, image content, element types (e.g., buttons, input fields, select menu), shapes, colors, etc. 2) **Positions**, including both absolute and relative positions to other elements. Besides straightforward positional information, contextual references (e.g., “*the ‘Add to Bag’ button for ‘item A’*”, “*the link ‘text’ under the section ‘Title’*”) often pose more challenges during grounding. To understand this type of referring expression, the grounding model needs to understand the positional relationship of elements and their context in the screenshots. 3) **Functionality**, i.e., main functions of elements. They are particularly important for non-textual elements, as they are rarely described solely through visual features in real-world use.

Template-Based Web Data Synthesis. We create a generation template to simulate the above-listed attributes, which is a random combination of the following fields: 1) **Primary Descriptor**. We use either the visual or functional information of an element as its primary descriptor. HTML attributes like `inner-text`, `alt` are used as primary visual clues, and accessibility labels like `alt`, `aria-label` are used as roles of elements. However, HTML elements often do not contain high-quality accessibility labels. To address this challenge, we use an open-source MLLM LLaVA-NeXT-13B (Liu et al., 2024) to uncover visual and functional clues about elements through element screenshots paired with key HTML attributes. Similar to (Lai et al., 2023), we employ an LM Llama-3-8B-Instruct (AI@Meta, 2024) to shorten these interpretations into concise element descriptions. 2) **Absolute Position**. We generate absolute positional descriptions according to the absolute coordinates of elements in screenshots, by a simple rule to cover most positions. 3) **Relative Position**. We first find the neighborhood of elements, and generate corresponding relative position descriptions (for example, the element below or above). Contextual references are more difficult to synthesize, and are always difficult cases in practice. On the other hand, they are very crucial to helping the model understand the context of elements, not simply detecting text and icons. We heuristically create textual labels of common and salient elements, including radio buttons, checkboxes, input fields, and select boxes, by their HTML attributes and nearby elements. (for example, a radio button labeled “on”). Some structures of HTML DOM trees (for instance, an element inside of a title element) are also leveraged to create contextual references.

Another challenge we encounter is the mismatch of the correctness and clarity between training and inference. Due to limitations in grounded understanding, even advanced MLLMs like GPT-4o tend to hallucinate on positional descriptions. Hence, it is crucial to teach the model to deal with suboptimal or ambiguous but acceptable descriptions. Otherwise, we experimentally find that the model often takes shortcuts by relying heavily on the positional information while neglecting other descriptions, which makes it difficult for the model to generalize to slightly ambiguous or incorrect descriptions provided by MLLMs. We randomly add small portions of suboptimal or wrong positional descriptions

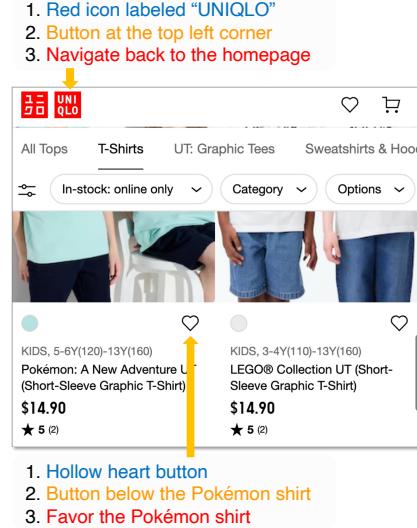


Figure 3: Examples of referring expressions of the three types: **visual features**, **positions**, and **functionality**

| Training Dataset | Type | Annotation | # Element | # Screenshot | Platform |
|-----------------------------------|--------------------------------|-------------|-----------|--------------|----------|
| Web-Synthetic | Referring Expression | Template | 18.1M | 773K | Web |
| Web-Refer | Referring Expression | GPT | 150K | 150K | Web |
| Web-Instruct | Low-Level Instruction | GPT | 258K | 258K | Web |
| GUIAct (Chen et al., 2024) | Low-Level Instruction & Action | GPT | 140K*2 | 13K*2 | Web |
| AndroidControl (Li et al., 2024) | Low-Level Action | Human | 47K | 47K | Android |
| Widget Caption (Li et al., 2020b) | Functionality | Human | 41.2K*2 | 15K*2 | Android |
| UIBERT (Bai et al., 2021) | Low-Level Action | Human | 15.6K | 4.6K | Android |
| AITZ (Zhang et al., 2024b) | Low-Level Action | GPT + Human | 7.8K | 7.8K | Android |
| Total | | | 19M | 1.3M | |

Table 1: Overview of the training data used for UGround.

to simulate this phenomenon (for example, use a faraway element as a neighbor, or randomly change the positional description).

We collect metadata of elements (salient HTML attributes, bounding boxes, screenshots) from Common Crawl¹, and then apply our curated template and rules to the raw data to get the grounding data. Elements on the same screenshots are merged into multi-turn conversations of single queries to accelerate training speed by tens of time. We finally collect a large set of 18.1M element descriptions, spread over 773K screenshots (i.e. 773K training instances).

Supplementary Data. Despite the rich visual features and functionalities extracted from the web raw data and element captions, capturing high-quality visual clues and functional roles remains challenging, particularly for the latter. Also, while the synthetic web data effectively teaches precision and positional descriptions, it still lacks diversity, functionality coverage of other platforms, and nuance of more free-form descriptions encountered in real-world applications. To address these gaps, we incorporate high-quality opensource Android data, and generate more free-form data by GPT-4o, to cover more diverse and nuanced descriptions not fully represented in our synthetic dataset. Additionally, by including more data from Android, we enhance the model’s performance on mobile UI.

As detailed in Table 1, we gather open-source datasets. We use referring expression data from UIBert and GUIAct-Thoughts, and captions of functionality of Android elements from Widget Caption. We also collect command-style data from AITZ-Thoughts, Android Control-Instructions, and GUIAct-Questions, which provide more free-form descriptions and often contain element roles within the commands.

To further expand our dataset, we use GPT-4 to generate descriptions and instructions for arbitrary elements from webpages. This includes generating actions or commands corresponding to the elements and detailed descriptions of the elements, adding substantial diversity and complexity to our training data.

We intentionally do not let humans revise the GPT-4 generated data, since in practice, our grounding model needs to ground textual plans generated by MLLMs including GPT-4; this helps simulate the less accurate and suboptimal descriptions in real deployment, ensuring that the model learns to handle descriptions of varied qualities, enhancing its robustness and practical usefulness.

Similar to web synthetic data, elements on the same screenshots are batched to single examples, totaling 0.5M training instances.

2.3 MODEL DESIGN

We adopt a widely used open-source model architecture, LLaVA-NeXT (Liu et al., 2024), as our backbone model. We make a few adaptations to make it better work for our goals.

Input-Output Formulation. We always let the model answer “*What are the pixel element coordinates corresponding to {Description}*” with the screenshot. Following previous work, we represent output coordinates in natural language. Most previous work uses normalized coordinates (percentiles or per thousand). However, while normalization is effective for models with fixed square input sizes, it is

¹ <https://commoncrawl.org/>

less suited for variable-sized image inputs or coordinates requiring higher accuracy. Thus, we opt for natural pixel coordinates (e.g., “(1344, 1344)”).

Image Resolution. GUI screenshots are significantly larger than natural images, often requiring a resolution of about 1000px for clear screen readability. LLaVA is initially built for 336px images, and later scaled up to at most 772px by AnyRes (Cheng et al., 2023; Gao et al., 2024; Liu et al., 2024; Xu et al., 2024; Dong et al., 2024), which resizes and splits a large image into small slices, encoding them independently with vision encoders, and adds a special token symbolizing the end of each row to help the LM understand image shapes. To balance the training cost and resolution, we enlarge the allowed input sizes to 36 ViT (Dosovitskiy et al., 2020) slices, and use CLIP@224px (Radford et al., 2021) as the image encoder, pushing the upper bound to 1344*1344 (landscape) and 896*2016 (portrait) during training and inference. We only resize the image by width to preserve aspect ratios and pad white strips to the bottom as needed. Additionally, we substitute the language backbone with Vicuna-1.5-7b-16k (Zheng et al., 2023) to handle long visual contexts. We empirically find the low-resolution image fusion commonly used in AnyRes to be ineffective in GUI grounding, as 224px is too small to provide informative global context, and hence remove it from the model design.

3 EXPERIMENTS

3.1 EXPERIMENTAL SETUP

To comprehensively evaluate the GUI grounding performance of UGround, and the effectiveness of SeeAct-V, we evaluate our model on both grounding and agent benchmarks.

We begin with ScreenSpot (Cheng et al., 2024), which evaluates the single-step GUI grounding capability across platforms. We further incorporate UGround into agent frameworks in the manner of SeeAct-V on five agent benchmarks, namely Multimodal-Mind2Web (Deng et al., 2023; Zheng et al., 2024), AndroidControl (Li et al., 2024), OmniACT (Kapoor et al., 2024), Mind2Web-Live(Pan et al., 2024), and AndroidWorld (Rawles et al., 2024), to demonstrate its practicability in real use.

We mainly investigate GPT-4¹ and GPT-4o as planners, as they are demonstrated to be SOTA models on most of these benchmarks. For grounding, we compare with SeeClick (Cheng et al., 2024), the prior SOTA model on ScreenSpot, as another potential flexible grounding model. We also compare SeeAct-V to prior prompt-only SOTA methods, either variations of textual choices or SoM, with either text or images as inputs. We follow most of the prompts of baseline methods. We replace inputs to pure screenshots and make small changes to incorporate SeeAct-V to the baseline frameworks.

3.2 GUI GROUNDING: SCREENSPOT

ScreenSpot is a benchmark designed specifically for GUI visual grounding, consisting of 1.2K single-step instructions and bounding boxes of the target elements. It spans many mobile, desktop and web environments on mainstream platforms, and categorizes element types into text and icons.

We evaluate on two settings, with different MLLMs as planners:

Original Instructions: Most of the original labeled queries are of instruction style (for example, “view iPhone storage”), not exactly element descriptions. They are often close to functional referring expressions in our training dataset.

Referring Expression: During realistic running, we do not have human-labeled ground-truth action instructions. Therefore, we perceive the original instructions as simple one-step tasks, and use different MLLMs as planners to generate element referring expressions for the elements to interact with, and then evaluate grounding models on the referring expressions.

We omit the grounding performance of general grounding models trained on natural images except GPT-4 and GPT-4o, as they are well-studied and perform poorly on ScreenSpot (Cheng et al., 2024).

¹ Due to the deprecation of the name GPT-4V and the frequent abbreviation of GPT-4-Turbo as GPT-4, we refer to GPT-4V, GPT-4-Turbo, and GPT-4 collectively as GPT-4 in this paper. Detailed endpoint names in baselines are provided in the appendix.

Table 2: Results on ScreenSpot. We compare with performance on both the original instructions of ScreenSpot, and the referring expressions generated by different planners given the original instructions as single-step tasks.

| Planner | Grounding | Mobile | | Desktop | | Web | | Average |
|-----------------|-----------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|
| | | Text | Icon/Widget | Text | Icon/Widget | Text | Icon/Widget | |
| Original Instr. | GPT-4 | 22.6 | 24.5 | 20.2 | 11.8 | 9.2 | 8.8 | 16.2 |
| | GPT-4o | 20.2 | 24.9 | 21.1 | 23.6 | 12.2 | 7.8 | 18.3 |
| | CogAgent | 67.0 | 24.0 | 74.2 | 20.0 | 70.4 | 28.6 | 47.4 |
| | SeeClick | 78.0 | 52.0 | 72.2 | 30.0 | 55.7 | 32.5 | 53.4 |
| | UGround | 82.8 | 60.3 | 82.5 | 63.6 | 80.4 | 70.4 | 73.3 |
| LLaVA-NeXT-72b | SeeClick | 74.7 | 41.1 | 69.1 | 27.1 | 40.9 | 24.3 | 46.2 |
| | UGround | 84.6 | 47.2 | 83.5 | 50.0 | 78.3 | 57.3 | 66.8 |
| GPT-4 | SeeClick | 76.6 | 55.5 | 68.0 | 28.6 | 40.9 | 23.3 | 48.8 |
| | UGround | 90.1 | 70.3 | 87.1 | 55.7 | 85.7 | 64.6 | 75.6 |
| GPT-4o | SeeClick | 81.0 | 59.8 | 69.6 | 33.6 | 43.9 | 26.2 | 52.3 |
| | UGround | 93.4 | 76.9 | 92.8 | 67.9 | 88.7 | 68.9 | 81.4 |

3.3 OFFLINE AGENT EVALUATION

Web Agents: We use Multimodal-Mind2Web (Zheng et al., 2024), the multimodal version of the large-scale web agent dataset Mind2Web (Deng et al., 2023), for our evaluation on web. The SOTA approach SeeAct in (Zheng et al., 2024) also splits each step into two steps. MLLMs like GPT-4 firstly generate an action by a screenshot, then the target element is selected from either SoM choices or textual HTML elements from top-50 element filtered by another small LM trained on Mind2Web.

Table 3: Element Accuracy on Multimodal-Mind2Web. ‘I’, ‘T’, ‘R’, ‘S’ stand for images, textual elements, raw images, and SoM images respectively. \dagger denotes model checkpoints finetuned on Mind2Web. The SoM results from (Zheng et al., 2024) is tested on subsets of 30 tasks for each split.

| Input | Planner | Grounding | Cross-Task | Cross-Website | Cross-Domain | Avg. |
|------------|--------------------|-----------|-------------|---------------|--------------|-------------|
| Image | CogAgent \dagger | - | 54.2 | 50.0 | 54.7 | 52.3 |
| Image | SeeClick \dagger | - | 23.8 | 15.3 | 16.2 | 18.4 |
| I + T | GPT-4 | Choice | 46.4 | 38.0 | 42.4 | 42.3 |
| I(R&S) + T | GPT-4 | SoM | 29.6 | 20.1 | 27.0 | 25.6 |
| Image | GPT-4 | SeeClick | 29.6 | 28.5 | 30.7 | 29.6 |
| | GPT-4 | UGround | 45.1 | 44.7 | 44.6 | 44.8 |
| | GPT-4o | SeeClick | 32.1 | 33.1 | 33.5 | 32.9 |
| | GPT-4o | UGround | 47.7 | 46.0 | 46.6 | 46.8 |

Similar to the prior evaluation in (Zheng et al., 2024), we divide the full webpage screenshots into viewport-sized blocks and simulate scrolling down by heuristically turning empty actions into scrolling. We also add CogAgent and SeeClick-Mind2Web which are trained or finetuned on Mind2Web, to compare with unified visual GUI agents in the same way.¹ We report element accuracy here, which measures which is the most related metric to our grounding comparison.

Table 4: Step Accuracy on AndroidControl.

| Input | Planner | Grounding | Step Accuracy | |
|-----------|---------|-----------|---------------|-------------|
| | | | High-Level | Low-Level |
| Acc. Tree | GPT-4 | Choice | 42.1 | 55.0 |
| | Image | GPT-4 | 39.4 | 47.2 |
| | | GPT-4 | 46.2 | 58.0 |
| | GPT-4o | SeeClick | 41.8 | 52.8 |
| | | UGround | 48.4 | 62.4 |

¹ Empirically, the finetuned models do not scroll down, but always generate an action at the top of the webpages.

Mobile Agents: We utilize AndroidControl (Li et al., 2024), a large-scale Android interaction dataset comprising 15K unique tasks over 833 apps. Following experiments in Li et al. (2024), we use 500 random steps from the test set. We compare with the SOTA zero-shot method, the text-only version of M3A (Rawles et al., 2024). We also follow the two settings of High-Level tasks and Low-Level tasks, which are to evaluate with the high-level instructions or both high-level and human-labeled low-level commands at each step.

OS Agents: We use OmniACT (Kapoor et al., 2024) to evaluate our model’s performance on OS tasks. The dataset encompasses a variety of desktop applications and web tasks across different operating systems. It contains 9802 tasks, each containing natural language instructions, UI screens, a list of UI elements (labels and bounding box coordinates), and corresponding PyAutoGUI code scripts.

The baseline method DetACT extracts UI elements and their coordinates from screenshots using OCR, icon matching, and color detection modules, filters them for task relevance, and then passes them to LLMs or MLLMs to generate a sequence of actions in PyAutoGUI code. We incorporate UGround by replacing input with pure screenshots, and let MLLMs generate element descriptions instead of coordinates when calling PyAutoGUI functions. Then we call UGround to obtain the coordinates from the screen. Following the method in (Kapoor et al., 2024), we keep the same prompt and five in-context examples retrieved from the training set by task similarity. We report the action score, which measures the correctness of action sequences along with penalizing wrong click points, and type or press values.

3.4 ONLINE AGENT EVALUATION

We further evaluate our model in an end-to-end manner, on two online benchmarks from Android and Web. Generally, without backup grounding methods as a hybrid approach, pure visual grounding can be much harder than SoM or other text-only approaches, because at least these methods ensure effective actions at each step. In contrast, a failure of the grounding model can lead to a repetitive clicking to a blank area, making agents stuck at a point.

Online Web Agents: We use the test set of Mind2Web-Live (Pan et al., 2024) as our evaluation on web. It adds functional evaluation metrics to tasks in Mind2Web (Deng et al., 2023). Specifically, key nodes (sub-steps) are annotated to the tasks, to verify the completion of the tasks lively. The baseline agent is text-only, perceives and interacts with webpages by hundreds of textual HTML elements. To completely avoid using HTML, we make the following changes to the action space: 1) We only allow the agent to see the current viewport, and add `scroll_up` and `scroll_down` to the action space. 2) In the baseline agent, an additional judgment model is used to determine whether to press enter after input through HTML¹. We remove this judgment and change the actions to `type` and `press_enter`, letting the agent make its own decisions. 3) We disable API-based select, which forces agents to select options merely through clicking and makes the action more challenging. We report completion rate and task success rate here, which measure the completion of key nodes, and the full completion of tasks.

Table 5: Action Scores (AS) on OmniACT. ‘I’ and ‘T’ stand for images and textual elements respectively.

| Inputs | Planner | Grounding | AS |
|--------|---------|-----------|--------------|
| T | GPT-4 | DetACT | 11.60 |
| T + I | GPT-4 | DetACT | 17.02 |
| Image | GPT-4 | SeeClick | 28.92 |
| | GPT-4 | UGround | 31.07 |
| | GPT-4o | SeeClick | 29.59 |
| | GPT-4o | UGround | 32.77 |

Table 6: Completion Rate and Task Success Rate (SR) on Mind2Web-Live.

| Inputs | Planner | Grounding | Completion Rate | Task SR |
|--------|---------|-----------|-----------------|-------------|
| HTML | GPT-4 | Choice | 44.3 | 21.1 |
| | GPT-4o | Choice | 47.6 | 22.1 |
| Image | GPT-4 | UGround | 55.3 | 23.1 |
| | GPT-4o | UGround | 56.0 | 19.2 |

Table 7: Task Success Rates (SR) on AndroidWorld. ‘I’, ‘T’, ‘R’, ‘S’ stand for images, textual elements, raw images, and SoM images respectively.

| Input | Planner | Grounding | SR |
|-------------|---------|-----------|-------------|
| Acc. Tree | GPT-4 | Choice | 30.6 |
| I (R&S) + T | GPT-4 | SoM | 25.4 |
| Image | GPT-4 | UGround | 31.0 |
| | GPT-4o | UGround | 32.8 |

¹ The original agent leverages an LLM to decide between `fill_form` and `fill_search` in the original action space, where the difference is whether to perform a `press_enter` after typing.

Online Android Agents: We use AndroidWorld (Rawles et al., 2024), an online benchmark running in Android emulators as our online evaluation on Mobile UI. It spreads 116 tasks across 20 apps, and provides accurate evaluation according to the states of the device. The baseline agent M3A receives both raw and SoM images, together with textual UI elements and element status as input to reason and decide the next action in ReAct-style (Yao et al., 2023). It also introduces self-reflection (Shinn et al., 2024) in the pipeline to instruct agents to summarize the current move and facilitate the following steps. We also compare with the text-only variation of M3A, which is the prior SOTA on this benchmark. It removes the screenshot input, only using textual UI elements as the input.

4 RESULTS AND ANALYSIS

4.1 RESULTS

Universal GUI Grounding. As shown in Table 2, UGround demonstrates state-of-the-art performance on the ScreenSpot benchmark, surpassing previous GUI grounding models across all platforms. This superiority is particularly pronounced with icon elements in desktop and web user interfaces, where the icons are usually small. UGround also demonstrates flexible acceptance to either original GUI commands, or referring expressions generated by MLLMs on ScreenSpot, both substantially outperform SeeClick by large margins.

Offline Agent Evaluation. As shown in Table 3, Table 4, and Table 5, UGround achieves SOTA performance on the three environments, outperforming prior text-only or SoM baselines with pure raw screenshots. Compared with unified models, UGround with GPT-4 largely outperforms the finetuned SeeClick-Mind2Web model, but fails to outperform CogAgent on Mind2Web. It shows that with a larger model size and sufficient training, unified models still is possible to work better than UGround. But we do observe that training on Mind2Web limits the flexibility of the models in realistic use, since the models always generate action for a screenshot, never suggesting scrolling down. This hinders the realistic use of such finetuned models.

Online Agent Evaluation. Online end-to-end evaluation is the hardest setting for all grounding methods, as it assesses the success of whole tasks. However, with UGround as the visual grounding model, we get either higher or comparable performance on Mind2Web-Live and AndroidWorld, as shown in Table 6 and Table 7. Specifically, it outperforms SoM method in AndroidWorld by a large margin, even though Android environments have the least dense UI layout which should be suitable for the SoM method.

4.2 ERROR ANALYSIS

We investigate action-wise error types in our experiments on ScreenSpot (GPT-4o as the planner), AndroidControl, and Multimodal-Mind2Web, to understand the performance of the three platforms. We sample 60 failure cases from each benchmark (or benchmark sub-split), to analyze whether an error is caused by wrong element descriptions, or grounding failures of UGround. For the convenience of statistics, we skip errors caused by ambiguous or wrong ground truth answers, or an alternative action at the step, only considering the two error types to understand the grounding performance.

As shown in Figure 4, planning errors are the main reasons for the failures on every benchmark. The most common cases are the planner generating element descriptions of other elements, showing poor understanding of the given tasks and the elements in the pages. Some other errors include hallucinating on the screenshot, generating non-existent elements; generating overly generic descriptions, and not precisely describing the target area.

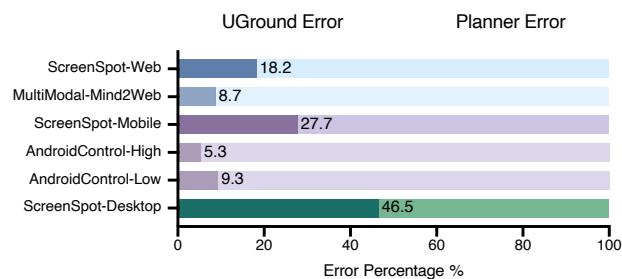


Figure 4: Error analysis of planning and grounding errors on ScreenSpot (with GPT-4o generated referring expressions), AndroidControl and Multimodal-Mind2Web.

In addition, although UGround performs strongly on ScreenSpot-Mobile and ScreenSpot-Desktop, many failures from these two platforms are grounding errors, not planning errors. As discussed in subsection 2.2, functionalities of elements, especially those of less-frequent elements, are one of the main difficulties when building the model. OS has the most icons, where many of them are special to UGround because of a lack of OS data. Hence UGround performs relatively poorly on OS and Mobile on ScreenSpot. However, when running for realistic Android tasks, most of the actions turn out to be not very hard, leading to very few grounding errors caused by UGround, which aligns with the fact that the model works well and still outperforms previous SOTA on AndroidWorld.

4.3 DATA ANALYSIS

We analyze two key aspects of our training data: the scale of the web synthetic data and the diversity of data sources. Our objective is to understand the impact of various training data types on the performance of GUI visual grounding.

Synthetic Data Scaling. To investigate the effect of our web synthetic data, we train UGround on randomly sampled 50K, 100K, 200K, 400K, and the entire data, and compare with their performance on ScreenSpot with GPT-4o generated referring expressions. The results on the three platforms and their average are illustrated in Figure 5, as well as the baseline of SeeClick.

The average performance keeps increasing with data scaling up. But the main improvement happens from 0 to 100K data. And with only 50K webpages of synthetic data, UGround already largely surpasses SeeClick by above 10%. To conclude, with only 100K synthetic data, the carefully curated synthetic data is able to teach UGround to understand referring expressions to a remarkable level. We observe that the remaining data mainly improves precision on less frequent element types, like radio, checkbox, very small text links, etc.

Training Data Sources. To further dissect the influence of training data, we compare the performance of UGround when trained with only web synthetic data, only data generated by humans and GPT-4, and a combination of both (full model). We compare the performance differences on ScreenSpot with GPT-4o-generated referring expressions.

As shown in Table 8. The analysis reveals that while human and GPT-4 data are generally perceived as high-quality inputs, their standalone performance in GUI visual grounding does not surpass that of 100K synthetic data. Specifically, the model trained only on synthetic data (UGround-Synthetic) achieved a performance score of 76.9%, with notably higher scores on icon examples. This highlights the effectiveness of leveraging abundant ARIA labels on the web, and the effectiveness of our referring expression generation template.

The combination of both data types (UGround) yields the highest performance at 81.4%. This suggests a synergistic effect where the diversity of training examples from both human, GPT-4 and synthetic sources enhances the model’s ability to generalize across different GUI elements and actions.

ACKNOWLEDGMENTS

We are grateful for the collaboration with the Orby AI team (particularly Sanjari Srivastava, Peng Qi, and Will Lu) for their contribution on data collection and analysis, as well as for providing computing resources. We would also like to extend our appreciation to Kanzhi Cheng, Yulu Guo, Lizi Yang and colleagues from the OSU NLP group for their insightful comments. Special thanks to Yichen Pan,

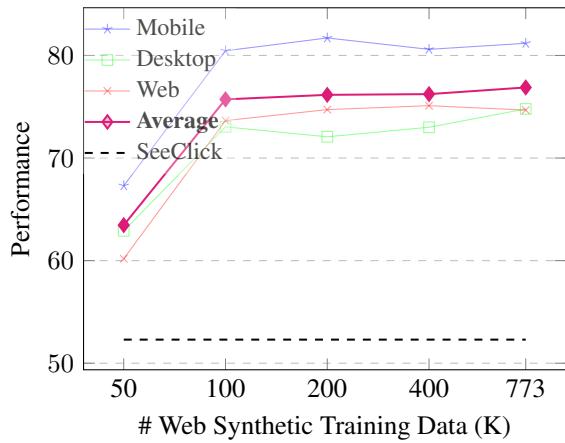


Figure 5: Performance of UGround trained on different data sizes of web synthetic data.

Table 8: Results on ScreenSpot of UGround trained with different data sources.

| Model | Mobile | | Desktop | | Web | | Average |
|-------------------|--------|-------------|---------|-------------|------|-------------|---------|
| | Text | Icon/Widget | Text | Icon/Widget | Text | Icon/Widget | |
| UGround-Synthetic | 89.0 | 73.4 | 88.1 | 61.4 | 84.8 | 64.6 | 76.9 |
| UGround-Human/GPT | 92.3 | 71.2 | 84.5 | 46.4 | 87.0 | 59.2 | 73.4 |
| UGround | 93.4 | 76.9 | 92.8 | 67.9 | 88.7 | 68.9 | 81.4 |

Christopher Rawles, Dehan Kong, Alice Li, and Raghav Kapoor for their assistance with evaluation. This work is supported in part by Orby AI and ARL W911NF2220144. The views and conclusions contained herein are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. government. The U.S. government is authorized to reproduce and distribute reprints for government purposes notwithstanding any copyright notice herein.

REFERENCES

- AI@Meta. Llama 3 model card, 2024. URL https://github.com/meta-llama/llama3/blob/main/MODEL_CARD.md.
- Chongyang Bai, Xiaoxue Zang, Ying Xu, Srinivas Sunkara, Abhinav Rastogi, Jindong Chen, and Blaise Agüera y Arcas. Uibert: Learning generic multimodal representations for ui understanding. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, pp. 1705–1712, 2021.
- Wentong Chen, Junbo Cui, Jinyi Hu, Yujia Qin, Junjie Fang, Yue Zhao, Chongyi Wang, Jun Liu, Guirong Chen, Yupeng Huo, et al. Guicourse: From general vision language models to versatile gui agents. *arXiv preprint arXiv:2406.11317*, 2024.
- Kanzhi Cheng, Qiushi Sun, Yougang Chu, Fangzhi Xu, Yantao Li, Jianbing Zhang, and Zhiyong Wu. Seeclick: Harnessing gui grounding for advanced visual gui agents. *arXiv preprint arXiv:2401.10935*, 2024.
- Siyuan Cheng, Bozhong Tian, Qingbin Liu, Xi Chen, Yongheng Wang, Huajun Chen, and Ningyu Zhang. Can we edit multimodal large language models? In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 13877–13888, 2023.
- Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Sam Stevens, Boshi Wang, Huan Sun, and Yu Su. Mind2web: Towards a generalist agent for the web. In *Advances in Neural Information Processing Systems*, volume 36, pp. 28091–28114, 2023.
- Xiaoyi Dong, Pan Zhang, Yuhang Zang, Yuhang Cao, Bin Wang, Linke Ouyang, Songyang Zhang, Haodong Duan, Wenwei Zhang, Yining Li, et al. Internlm-xcomposer2-4khd: A pioneering large vision-language model handling resolutions from 336 pixels to 4k hd. *arXiv preprint arXiv:2404.06512*, 2024.
- Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.
- Peng Gao, Renrui Zhang, Chris Liu, Longtian Qiu, Siyuan Huang, Weifeng Lin, Shitian Zhao, Shijie Geng, Ziyi Lin, Peng Jin, et al. Sphinx-x: Scaling data and parameters for a family of multi-modal large language models. *arXiv preprint arXiv:2402.05935*, 2024.
- Izzeddin Gur, Hiroki Furuta, Austin Huang, Mustafa Safdari, Yutaka Matsuo, Douglas Eck, and Aleksandra Faust. A real-world webagent with planning, long context understanding, and program synthesis. In *International Conference on Learning Representations*, 2024.
- Hongliang He, Wenlin Yao, Kaixin Ma, Wenhao Yu, Yong Dai, Hongming Zhang, Zhenzhong Lan, and Dong Yu. Webvoyager: Building an end-to-end web agent with large multimodal models. *arXiv preprint arXiv:2401.13919*, 2024.

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. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14281–14290, 2024.

Peter C Humphreys, David Raposo, Tobias Pohlen, Gregory Thornton, Rachita Chhaparia, Alistair Muldal, Josh Abramson, Petko Georgiev, Adam Santoro, and Timothy Lillicrap. A data-driven approach for learning to control computers. In *International Conference on Machine Learning*, pp. 9466–9482. PMLR, 2022.

Subbarao Kambhampati, Karthik Valmeekam, Lin Guan, Kaya Stechly, Mudit Verma, Siddhant Bhambri, Lucas Saldyt, and Anil Murthy. Llms can't plan, but can help planning in llm-modulo frameworks. *arXiv preprint arXiv:2402.01817*, 2024.

Raghav Kapoor, Yash Parag Butala, Melisa Russak, Jing Yu Koh, Kiran Kamble, Waseem Alshikh, and Ruslan Salakhutdinov. Omniact: A dataset and benchmark for enabling multimodal generalist autonomous agents for desktop and web. *arXiv preprint arXiv:2402.17553*, 2024.

Jing Yu Koh, Robert Lo, Lawrence Jang, Vikram Duvvur, Ming Chong Lim, Po-Yu Huang, Graham Neubig, Shuyan Zhou, Ruslan Salakhutdinov, and Daniel Fried. Visualwebarena: Evaluating multimodal agents on realistic visual web tasks. In *ICLR 2024 Workshop on Large Language Model (LLM) Agents*, 2024.

Zhengfeng Lai, Haotian Zhang, Wentao Wu, Haoping Bai, Aleksei Timofeev, Xianzhi Du, Zhe Gan, Jiulong Shan, Chen-Nee Chuah, Yinfei Yang, et al. From scarcity to efficiency: Improving clip training via visual-enriched captions. *arXiv preprint arXiv:2310.07699*, 2023.

Wei Li, William Bishop, Alice Li, Chris Rawles, Folawiyo Campbell-Ajala, Divya Tyamagundlu, and Oriana Riva. On the effects of data scale on computer control agents. *arXiv preprint arXiv:2406.03679*, 2024.

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

Yang Li, Gang Li, Luheng He, Jingjie Zheng, Hong Li, and Zhiwei Guan. Widget captioning: Generating natural language description for mobile user interface elements. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 5495–5510, 2020b.

Haotian Liu, Chunyuan Li, Yuheng Li, Bo Li, Yuanhan Zhang, Sheng Shen, and Yong Jae Lee. Llava-next: Improved reasoning, ocr, and world knowledge, January 2024. URL <https://llava-v1.github.io/blog/2024-01-30-llava-next/>.

Yichen Pan, Dehan Kong, Sida Zhou, Cheng Cui, Yifei Leng, Bing Jiang, Hangyu Liu, Yanyi Shang, Shuyan Zhou, Tongshuang Wu, et al. Webcanvas: Benchmarking web agents in online environments. *arXiv preprint arXiv:2406.12373*, 2024.

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*, pp. 8748–8763. PMLR, 2021.

Christopher Rawles, Alice Li, Daniel Rodriguez, Oriana Riva, and Timothy Lillicrap. Android in the wild: A large-scale dataset for android device control. In *Advances in Neural Information Processing Systems*, volume 36, pp. 59708–59728, 2023.

Christopher Rawles, Sarah Clinckemaillie, Yifan Chang, Jonathan Waltz, Gabrielle Lau, Marybeth Fair, Alice Li, William Bishop, Wei Li, Folawiyo Campbell-Ajala, et al. Androidworld: A dynamic benchmarking environment for autonomous agents. *arXiv preprint arXiv:2405.14573*, 2024.

-
- Peter Shaw, Mandar Joshi, James Cohan, Jonathan Berant, Panupong Pasupat, Hexiang Hu, Urvashi Khandelwal, Kenton Lee, and Kristina N Toutanova. From pixels to ui actions: Learning to follow instructions via graphical user interfaces. *Advances in Neural Information Processing Systems*, 36: 34354–34370, 2023.
- Tianlin Shi, Andrej Karpathy, Linxi Fan, Jonathan Hernandez, and Percy Liang. World of bits: An open-domain platform for web-based agents. In *International Conference on Machine Learning*, pp. 3135–3144. PMLR, 2017.
- Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*, 36, 2024.
- Junyang Wang, Haiyang Xu, Jiabo Ye, Ming Yan, Weizhou Shen, Ji Zhang, Fei Huang, and Jitao Sang. Mobile-agent: Autonomous multi-modal mobile device agent with visual perception. In *ICLR 2024 Workshop on Large Language Model (LLM) Agents*, 2024.
- WebAIM. The WebAIM Million. <https://webaim.org/projects/million/>, 2024. Accessed: 2024-08-04.
- Zhiyong Wu, Chengcheng Han, Zichen Ding, Zhenmin Weng, Zhoumianze Liu, Shunyu Yao, Tao Yu, and Lingpeng Kong. Os-copilot: Towards generalist computer agents with self-improvement. In *ICLR 2024 Workshop on Large Language Model (LLM) Agents*, 2024.
- Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh Jing Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, et al. Osworld: Benchmarking multimodal agents for open-ended tasks in real computer environments. *arXiv preprint arXiv:2404.07972*, 2024.
- Ruyi Xu, Yuan Yao, Zonghao Guo, Junbo Cui, Zanlin Ni, Chunjiang Ge, Tat-Seng Chua, Zhiyuan Liu, Maosong Sun, and Gao Huang. Llava-uhd: an lmm perceiving any aspect ratio and high-resolution images. *arXiv preprint arXiv:2403.11703*, 2024.
- An Yan, Zhengyuan Yang, Wanrong Zhu, Kevin Qinghong Lin, Linjie Li, Jianfeng Wang, Jianwei Yang, Yiwu Zhong, Julian J. McAuley, Jianfeng Gao, Zicheng Liu, and Lijuan Wang. Gpt-4v in wonderland: Large multimodal models for zero-shot smartphone gui navigation. *ArXiv*, abs/2311.07562, 2023.
- Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. Webshop: Towards scalable real-world web interaction with grounded language agents. *Advances in Neural Information Processing Systems*, 35:20744–20757, 2022.
- Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*, 2023.
- Chaoyun Zhang, Liqun Li, Shilin He, Xu Zhang, Bo Qiao, Si Qin, Minghua Ma, Yu Kang, Qingwei Lin, Saravan Rajmohan, et al. Ufo: A ui-focused agent for windows os interaction. *arXiv preprint arXiv:2402.07939*, 2024a.
- Jiwen Zhang, Jihao Wu, Yihua Teng, Minghui Liao, Nuo Xu, Xiao Xiao, Zhongyu Wei, and Duyu Tang. Android in the zoo: Chain-of-action-thought for gui agents. *arXiv preprint arXiv:2403.02713*, 2024b.
- Boyuan Zheng, Boyu Gou, Jihyung Kil, Huan Sun, and Yu Su. Gpt-4v (ision) is a generalist web agent, if grounded. In *Forty-first International Conference on Machine Learning*, 2024.
- Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, Hao Zhang, Joseph E Gonzalez, and Ion Stoica. Judging llm-as-a-judge with mt-bench and chatbot arena. In *Advances in Neural Information Processing Systems*, volume 36, pp. 46595–46623, 2023.
- Shuyan Zhou, Frank F Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, et al. Webarena: A realistic web environment for building autonomous agents. In *The Twelfth International Conference on Learning Representations*, 2023.