## OMNIPARSER FOR PURE VISION BASED GUI AGENT

Anonymous authors

Paper under double-blind review

#### ABSTRACT

The recent advancements of large vision language models shows their great potential in driving the agent system operating on user interfaces. However, we argue that the power multimodal models like GPT-4V as a general agent on multiple operating systems across different applications is largely underestimated due to the lack of a robust screen parsing technique capable of: 1) reliably identifying interactable icons within the user interface, and 2) understanding the semantics of various elements in a screenshot and accurately associate the intended action with the corresponding region on the screen. To fill these gaps, we introduce OMNI-PARSER, a comprehensive method for parsing general user interface screenshots into structured elements, which significantly enhances the ability of GPT-4V to generate actions that can be accurately grounded in the corresponding regions of the interface. We first curated an interactable icon detection dataset using popular webpages and an icon description dataset. These datasets were utilized to fine-tune specialized models: a detection model to parse interactable regions on the screen and a caption model to extract the functional semantics of the detected elements. OMNIPARSER significantly improves GPT-4V's performance on ScreenSpot benchmark. And on Mind2Web and AITW benchmark, OMNIPARSER with screenshot only input outperforms the GPT-4V baselines requiring additional information outside of screenshot. We further demonstrate that OMNIPARSER can seamlessly integrate with other vision language models, significantly enhancing their agentic capabilities.

028 029

031

000

001 002 003

004

006 007

008 009

010

011

012

013

014

015

016

017

018

019

021

024

025

026

027

#### 1 INTRODUCTION

Large language models have shown great success in their understanding and reasoning capabilities. More recent works have explored the use of large vision-language models (VLMs) as agents to perform complex tasks on the user interface (UI) with the aim of completing tedious tasks to replace human efforts (YZL+23; YYZ+23; DGZ+23; ZGK+24; HWL+23; YZS+24; WXJ+24; GFH+24; CSC+24). Despite the promising results, there remains a significant gap between current state-of-thearts and creating widely usable agents that can work across multiple platforms, *e.g.* Windows/MacOS, IOS/Android and multiple applications (Web broswer Office365, PhotoShop, Adobe), with most previous work focusing on limiting applications or platforms.

040 While large multimodal models like GPT-4V and other models trained on UI data (HWL<sup>+</sup>23; 041 YZS<sup>+24</sup>; CSC<sup>+24</sup>) have demonstrated abilities to understand basic elements of the UI screenshot, 042 action grounding remains one of the key challenges in converting the actions predicted by LLMs 043 to the actual actions on screen in terms of keyboard/mouse movement or API call (ZGK<sup>+</sup>24). It 044 has been noted that GPT-4V is unable to produce the exact x-y coordinate of the button location, Set-of-Mark prompting (YZL<sup>+</sup>23) proposes to overlay a group of bounding boxes each with unique numeric IDs on to the original image, as a visual prompt sent to the GPT-4V model. By applying 046 set-of-marks prompting, GPT-4V is able to ground the action into a specific bounding box which has 047 ground truth location instead of a specific xy coordinate value, which greatly improves the robustness 048 of the action grounding (ZGK<sup>+</sup>24). However, the current solutions using SoM relies on parsed HTML information to extract bounding boxes for actionable elements such as buttons, which limits its usage to web browsing tasks. We aim to build a general approach that works on a variety of 051 platforms and applications. 052

In this work, we argue that previous pure vision-based screen parsing techniques are not satisfactory, which lead to significant underestimation of GPT-4V model's understanding capabilities. And a

reliable vision-based screen parsing method that works well on general user interface is a key to
 improve the robustness of the agentic workflow on various operating systems and applications. We
 present OMNIPARSER, a general screen parsing tool to extract information from UI screenshot into
 structured bounding box and labels which enhances GPT-4V's performance in action prediction in a
 variety of user tasks.

- We summarize our contributions as follows:
  - We curate a interactable region detection dataset using bounding boxes extracted from DOM tree of popular webpages.
  - We propose OmniParser, a pure vision-based user interface screen parsing method that combines multiple finetuned models for better screen understanding and easier grounded action generation.
  - We evaluate OmniParser on the ScreenSpot, Mind2Web, and AITW benchmarks, demonstrating significant improvement over the GPT-4V baseline, using only screenshots as input.
  - We show that OMNIPARSER operates seamlessly and serves as an easy-to-integrate tool for a variety of state-of-the-art public vision language models. And we open-source both our code and model to facilitate further development and integration of OMNIPARSER to other vision language models. <sup>1</sup>
- 071 072 073 074

075

061

062

063

064

065

067

068

069

070

## 2 RELATED WORKS

## 076 2.1 UI SCREEN UNDERSTANDING

There has been a line of modeling works focusing on detailed understanding of UI screens, such as Screen2Words (WLZ+21), UI-BERT (BZX+21), WidgetCaptioning (LLH+20), Action-BERT (HSZ+21). These works demonstrated effective usage of multimodal models for extracting semantics of user screen. But these models rely on additional information such as view hierarchy, or are trained for visual question answering tasks or screen summary tasks.

There are also a couple publicly available datasets on UI screen understanding. Most notably the 083 Rico dataset (DHF $^+$ 17), which contains more than 66k unique UI screens and its view hierarchies. 084 Later (SWL<sup>+</sup>22) auguments Rico by providing 500k human annotations on the original 66k RICO 085 screens identifying various icons based on their shapes and semantics, and associations between selected general UI elements (like icons, form fields, radio buttons, text inputs) and their text labels. 087 Same on mobile platform, PixelHelp (LHZ<sup>+</sup>20) provides a dataset that contains UI elements of screen spanning across 88 common tasks. In the same paper they also released RicoSCA which is a cleaned version of Rico. For the web and general OS domain, there are several works such Mind2Web (DGZ<sup>+</sup>23), MiniWob++(LGP<sup>+</sup>18), Visual-WebArena (KLJ<sup>+</sup>24; ZXZ<sup>+</sup>24), and OS-090 World (XZC $^+24$ ) that provide simulated environment, but does not provide dataset explicitly for 091 general screen understanding tasks such as interactable icon detection on real world websites. 092

To address the absence of a large-scale, general web UI understanding dataset, and to keep pace with the rapid evolution of UI design, we curated an icon detection dataset using the DOM information from popular URLs availabe on the Web. This dataset features the up-to-date design of icons and buttons, with their bounding boxes retrieved from the DOM tree, providing ground truth locations.

097 098

## 2.2 AUTONOMOUS GUI AGENT

099 Recently there has been a lot of works on designing autonomous GUI agent to perform tasks in 100 place of human users. One line of work is to train an end-to-end model to directly predict the next 101 action, representative works include: Pixel2Act (SJC+23), WebGUM(FLN+24) in web domain, 102 Ferret (YZS<sup>+</sup>24), CogAgent (HWL<sup>+</sup>23), and Fuyu (BEH<sup>+</sup>23) in Mobile domain. Another line 103 of works involve leveraging existing multimodal models such as GPT-4V to perform user tasks. 104 Representative works include MindAct agent (DGZ<sup>+</sup>23), SeeAct agent (ZGK<sup>+</sup>24) in web domain 105 and agents in (YYZ<sup>+</sup>23; WXY<sup>+</sup>24; RLR<sup>+</sup>23) for mobile domain. These work often leverages 106 the DOM information in web browser, or the view hierarchies in mobile apps to get the ground

<sup>&</sup>lt;sup>1</sup>Github repository and Huggingface model links will be made available after the review.

108 truth position of interactable elements of the screen, and use Set-Of-Marks(YZL+23) to overlay the 109 bounding boxes on top of the screenshot then feed into the vision-language models. However, ground 110 truth information of interactable elements may not always be available when the goal is to build a 111 general agent for cross-platforms and cross-applications tasks. Therefore, we focus on providing a 112 systematic approach for getting structured elements from general user screens.

#### 3 METHODS

116 To complete a complex task, T on graphical user interface, the process can usually be broken down 117 into several steps of state-action pairs  $(S_0, A_0), \dots, (S_n, A_n)$ , where n is the number of steps. Each 118 step requires the model's (e.g. GPT-4V) ability to: 1) understand the state of the UI screen information  $S_i$  in the current step, i.e. analyzing what is the screen content overall, what are the functions of 119 120 detected icons that are labeled with numeric ID, and 2) predict what is the next action  $A_i$  on the current screen that is likely to help completing the whole task. Normally this can be formulated as: 121

122

113 114

115

123 124

125

127

129

130

131

133

138 139

140 141

142

143

146

147

 $\begin{aligned} a_{i+1} &= \pi(T, S_i, [(S_0, A_0), ..., (S_{i-1}, A_{i-1})] \\ S_i &= \{h_i, \mathrm{Img}_i\} \end{aligned}$ (1)

Here  $S_i$  encapsules the current information of the screen, which varies according to the model 126  $\pi$ 's input types and the operating environment. For example, in the web tasks (ZGK<sup>+</sup>24),  $S_i =$  $\{h_i, \text{Img}_i\}$ , where  $h_i$  is the HTML information at step i, and  $\text{Img}_i$  is the screenshot of the GUI at step 128 i. In Android tasks (RLR<sup>+</sup>23),  $h_i$  is the view hierarchy information. In these cases, the model  $\pi$  is required to extract information directly from  $\text{Img}_i$  and at the same time generate an action prediction. Instead of trying to accomplish the two goals in one call, we found it beneficial to extract some of the information such as semantics in the screen parsing stage, to alleviate the burden of GPT-4V so that it 132 can leverages more information from the parsed screen and focus more on the action prediction.

Hence we propose OMNIPARSER, which integrates the outputs from a finetuned interactable icon 134 detection model, a finetuned icon description model, and an OCR module. This combination produces 135 a structured state information  $S_i$ , which includes a DOM-like representation of the UI and a screenshot 136 overlaid with bounding boxes for potential interactable elements 137

$$a_{i+1} = \pi(T, S_i, [(S_0, A_0), ..., (S_{i-1}, A_{i-1})]$$

$$S_i = \{ \text{Img}_i^{som}, \text{LS}_i, \text{OCR}_i^{txt} \}$$
(2)

 $\operatorname{Img}_{i}^{som}, \operatorname{LS}_{i}, \operatorname{OCR}_{i}^{txt} = \operatorname{OMNIPARSER}(\operatorname{Img}_{i})$ Here  $\operatorname{Img}_{i}^{som}$  is the set-of-mark image labeled by the finetuned interactable icon detection model,  $LS_{i}$ 

is the local semantics output by the finetuned icon description model. We discuss each component of

144 145

#### 3.1 INTERACTABLE REGION DETECTION

the OMNIPARSER in more details for the rest of the section.

Identifying interactable regions from the UI screen is a crucial step to reason about what actions 148 should be performed given a user tasks. Instead of directly prompting GPT-4V to predict the 149 xy coordinate value of the screen that it should operate on, we follow previous works to use the 150 Set-of-Marks approach (YZL<sup>+</sup>23) to overlay bounding boxes of interactable icons on top of UI 151 screenshot, and ask GPT-4V to generate the bounding box ID to perform action on. However, 152 different from (ZGK+24; KLJ+24) which uses the ground truth button location retrieved from 153 DOM tree in web browswer, and (YYZ<sup>+</sup>23) which uses labeled bounding boxes in the AITW 154 dataset (RLR<sup>+</sup>23), we finetune a detection model to extract interactable icons/buttons. 155

Specifically, we curate a dataset of interactable icon detection dataset, containing 67k unique screen-156 shot images, each labeled with bounding boxes of interactable icons derived from DOM tree. We 157 first took a 100k uniform sample of popular publicly availabe urls on the web (OXL+22), and collect 158 bounding boxes of interactable regions of the webpage from the DOM tree of each urls. Some 159 examples of the webpage and the interactable regions are shown in 2. 160

Apart from interactable region detection, we also have a OCR module to extract bounding boxes of 161 texts. Then we merge the bounding boxes from OCR detection module and icon detection module

while removing the boxes that have high overlap (we use 90% as a threshold). For every bounding box, we label it with a unique ID next to it using a simple algorithm to minimizing the overlap between numeric labels and other bounding boxes.

165 166 167

## 3.2 INCORPORATING LOCAL SEMANTICS OF FUNCTIONALITY

We found in a lot of cases where only inputting the UI screenshot overlayed with bounding boxes and associated IDs can be misleading to GPT-4V. We argue the limitation stems from GPT-4V's constrained ability to simultaneously perform the composite tasks of identifying each icon's semantic information and predicting the next action on a specific icon box. This has also been observed by several other works (YYZ<sup>+</sup>23; ZGK<sup>+</sup>24). To address this issue, we incorporate the local semantics of functionality into the prompt, i.e. for each icons detected by the interactable region detection model, we use a finetuned model to generate description of functionality to the icons, and for each text boxes, we use the detected texts and its label.

We perform more detailed analysis for this topic in section 4.1. To the best of our knowledge, there is 176 no public model that is specifically trained for up-to-date UI icon description, and is suitable for our 177 purpose to provide fast and accurate local semantics for the UI screenshot. Therefore we curate a 178 dataset of 7k icon-description pairs using GPT-4o, and finetune a BLIP-v2 model (LLSH23) on this 179 dataset. Details of dataset and training can be found in Appendix 8.1. After finetuning, we found the 180 model is much more reliable in its description to common app icons. Examples can be seen in figure 181 5. And in figure 3, we show it is helpful to incorporate the semantics of local bounding boxes in the 182 form of text prompt along with the UI screenshot visual prompt. 183

184 185

187

188

189

## 4 EXPERIMENTS AND RESULTS

We conduct experiments on several benchmarks to demonstrate the effectiveness of OMNIPARSER. We start by a motivating experiments showing that current GPT-4V model with set of mark prompting (YZL<sup>+</sup>23) is prone to incorrectly assigning label ID to the referred bounding boxes. Then we evaluate on Seeclick benchmark and Mind2Web to further showcase OMNIPARSER with local semantics can improve the GPT-4V's performance on real user tasks on different platforms and applications.

#### 4.1 EVALUATION ON SEEASSIGN TASK

194 To test the ability of correctly predicting the label ID given the description of the bounding boxes for 195 GPT-4v models, We handcrafted a dataset SeeAssign that contains 112 tasks consisting of samples 196 from 3 different platforms: Mobile, Desktop and Web Browser. Each task includes a concise task 197 description and a screenshot image. The task descriptions are manually created and we make sure each task refers to one of the detected bounding boxes, e.g. 'click on 'settings", 'click on the 198 minimize button'. During evaluation, GPT-4V is prompted to predict the bounding box ID associated 199 to it. Detailed prompt are specified in Appendix. The task screenshot images are sampled from the 200 ScreenSpot (CSC<sup>+</sup>24) benchmark, where they are labeled with set of marks using OMNIPARSER. 201 The tasks are further divided into 3 sub-categories by difficulty: easy (less than 10 bounding boxes), 202 medium (10-40 bounding boxes) and hard (more than 40 bounding boxes). 203

From table 1, we see that GPT-4V often mistakenly assign the numeric ID to the table especially when there are a lot of bounding boxes over the screen. And by adding local semantics including texts within the boxes and short descriptions of the detected icons, GPT-4V's ability of correctly assigning the icon improves from 0.705 to 0.938.

From figure 3, we see that without the description of the referred icon in the task, GPT-4V often fails to link the icon required in the task and the ground truth icon ID in the SoM labeled screenshot, which leads to hallucination in the response. With fine-grain local semantics added in the text prompt, it makes it much easier for GPT-4V to find the correct icon ID for the referred icon.

212

- 213 4.2 EVALUATION ON SCREENSPOT
- 215 ScreenSpot dataset (CSC<sup>+</sup>24) is a benchmark dataset that includes over 600 interface screenshots from mobile (iOS, Android), desktop (macOS, Windows), and web platforms. The task instructions



Figure 1: Examples of parsed screenshot image and local semantics by OMNIPARSER. The inputs to OmniParse are user task and UI screenshot, from which it will produce: 1) parsed screenshot image with bounding boxes and numeric IDs overlayed, and 2) local semantics containing both text extracted and icon description.

are manually created so that each instruction corresponds to an actionable elements on the UI screen.
We first evaluate the performance of OMNIPARSER using the this benchmark. In table 2, we can see
across the 3 different platforms: Mobile, Desktop and Web, OMNIPARSER significantly improves the
GPT-4V baseline from 16.2% to 73.0%. Noticeably, OMNIPARSER's performance even surpasses
models that are specifically finetuned on GUI dataset including SeeClick, CogAgent and Fuyu by
a large margin. We also note that incorporating the local semantics (OMNIPARSER w. LS in the



Figure 2: Examples from the Interactable Region Detection dataset. The bounding boxes are based on the interactable region extracted from the DOM tree of the webpage.



Figure 3: Examples from the SeeAssign evaluation. We can see that fine-grain local semantics improves the GPT-4V's ability to assign correct labels to the referred icon.

table) further improves the overall performance. This corroborates with the finds in section 4.1 that incorporating local semantics of the UI screenshot in text format, i.e. adding OCR text and descriptions of the icon bounding boxes further helps GPT-4V to accurately identify the correct element to operate on. Furthermore, our findings indicate that the interactable region detection (ID) model we finetuned improves overall accuracy by an additional 4.3% compared to using the raw Grounding DINO model. This underscores the importance of accurately detecting interactable elements for the success of UI tasks. Overall, the results demonstrate that the UI screen understanding capability of GPT-4V is significantly underestimated and can be greatly enhanced with more accurate interactable elements detection and the incorporation of functional local semantics. More ablations across other open-source vision language models can be found in section 5.

	Easy	Medium	Hard	Overall
GPT-4V w.o. local semantics	0.913	0.692	0.620	0.705
GPT-4V w. local semantics	1.00	0.949	0.900	0.938

328

324

Table 1: Comparison of GPT-4V with and without local semantics

Methods	Model Size		Mobile		Desktop		Web	
		Text	Icon/Widget	Text	Icon/Widget	Text	Icon/Widget	
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%
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%
OmniParser (GPT-4V)	-	93.9%	57.0%	91.3%	63.6%	81.3	51.0%	73.0%
- w.o. ID	-	94.8%	53.7%	89.3%	44.9%	83.0%	45.1%	68.7%
- w.o. ID and w.o. LS	-	92.7%	49.4%	64.9%	26.3%	77.3%	39.7%	58.38%

Table 2: Comparison of different approaches on ScreenSpot Benchmark. LS is short for local semantics of functionality and ID is short for the interactable region detection model we finetune. The setting w.o. ID means we replace the ID model with original Grounding DINO model not finetuned on our data, and with local semantics. The setting w.o. ID and w.o LS means we use Grounding DINO model, and further without using the icon description in the text prompt.

343 344 345

346

340

341

342

#### 4.3 EVALUATION ON MIND2WEB

347 In order to test how OMNIPARSER is helpful to the web navigation secnario, We evaluate on 348 (DGZ<sup>+</sup>23) benchmark. There are 3 different categories of task in the test set: Cross-Domain, Cross-349 Website, and Cross-Tasks. We used a cleaned version of Mind2Web tests set processed from the raw 350 HTML dump which eliminates a small number of samples that has incorrect bounding boxes. In 351 total we have 867, 167, 242 tasks in the test set from Cross-Domain, Cross-Website, and Cross-Tasks category respectively. During evaluation, we feed both the parsed screen results and the action 352 history as text prompt, and SOM labeled screenshot to GPT-4V similar to the prompting strategy 353 in (YYZ<sup>+</sup>23; ZGK<sup>+</sup>24). Following the original paper, we perform offline evaluation focusing on the 354 element accuracy, Operation F1 and step success rate averaged across the task. 355

In the first section of the table (row 1-3), We report numbers from a set of open source VL models as it appears in (ZGK<sup>+</sup>24; CSC<sup>+</sup>24). Here CogAgent and Qwen-VL are not finetuned on the Mind2Web training set. More detailed information about model settings can be found in the Appendix8.4.

359 In the second section of the table (row 4-9) we report numbers from Mind2web paper (DGZ $^+23$ ) and 360 SeeAct (ZGK<sup>+</sup>24) paper. In this section, all of the approaches use the HTML elements selected by a 361 finetuned element proposal model on Mind2Web training set which produces top 50 relevant elements 362 on the HTML page based on the user task. Additionally, GPT-4V+SOM and GPT-4V+textual choices corresponds to the SeeAct with image annotation, and textual choices grounding methods respectively. 363 In GPT-4V+SOM, the set of mark (SOM) boxes are selected from the element proposal model, and 364 are labeled with the ground truth location extracted from HTML. In contrast, GPT-4V+textual uses DOM information of the selected relevant elements directly in the text prompt, rather than overlaying 366 bounding boxes on top of screenshot. The better performance of textual choice corroborates with the 367 experiment results in 4.1. 368

In the last section (row 10), we report numbers of OMNIPARSER using GPT-4V as the action prediction model. We observe OMNIPARSER achieves comparable performance to model that uses additional HTML information and additionally finetuned element proposal model.

In summary, without using parsed HTML information, OMNIPARSER is able to outperform GPT-4's performance that uses HTML in every sub-category by a significant margin, suggesting the substantial benefit of the screen parsing results provided by OMNIPARSER. Additionally, OMNIPARSER outperforms the GPT-4V+SOM approach by a large margin where the set of marks coordinates are extracted from the html. This suggests that OMNIPARSER's interactable detection model and the icon description model provides useful information to perform the task. Compared to GPT-4V+textual choices, OMNIPARSER significantly outperforms in Cross-Website and Cross-Domain category (+4.1% and +5.2%), while underperforming (-0.8%) slightly in the Cross-Task category, which
 indicates that OMNIPARSER provides higher quality information compared to ground truth element
 information from DOM and top-k relevant element proposal used by the GPT-4V+textual choices
 set-up, and make the GPT-4V easier to make a accurate action prediction. Lastly, OMNIPARSER
 with GPT-4V significantly outperform all the other trained models using only UI screenshot such as
 SeeClick and Qwen-VL.

Methods	Input Ty	pes	Ci	ross-Webs	site	Cı	oss-Dom	ain	(	Cross-Tas	ik 🛛
	HTML free	image	Ele.Acc	Op.F1	Step SR	Ele.Acc	Op.F1	Step SR	Ele.Acc	Op.F1	Step SR
CogAgent	✓	<	18.4	42.2	13.4	20.6	42.0	15.5	22.4	53.0	17.6
Qwen-VL	$\checkmark$	<	13.2	83.5	9.2	14.1	84.3	12.0	14.1	84.3	12.0
SeeClick	✓	<	21.4	80.6	16.4	23.2	84.8	20.8	28.3	87.0	25.5
MindAct (gen)	×	×	13.9	44.7	11.0	14.2	44.7	11.9	14.2	44.7	11.9
MindAct	×	×	42.0	65.2	38.9	42.1	66.5	39.6	42.1	66.5	39.6
GPT-3.5-Turbo	×	×	19.3	48.8	16.2	21.6	52.8	18.6	21.6	52.8	18.6
GPT-4	×	×	35.8	51.1	30.1	37.1	46.5	26.4	41.6	60.6	36.2
GPT-4V+som	×	<ul><li>✓</li></ul>	-	-	32.7	-	-	23.7	-	-	20.3
GPT-4V+textual choice	×	< ✓	38.0	67.8	32.4	42.4	69.3	36.8	46.4	73.4	40.2
OmniParser (GPT-4V)	✓	<	41.0	84.8	36.5	45.5	85.7	42.0	42.4	87.6	39.4

Table 3: Comparison of different methods across various categories on Mind2Web benchmark.

#### 4.4 EVALUATION ON ANDROID-IN-THE-WILD

In additional to evaluation on multi-step web browsing tasks, we assess OMNIPARSER on the mobile navigating benchmark AITW (RLR<sup>+</sup>23), which contains 30k instructions and 715k trajectories. We use the same train/test split as in (CSC<sup>+</sup>24) based on instructions, which retain only one trajectory for each instructions and no intersection between train and test. For a fair comparison, we only use their test split for evaluation and discard the train set as our method does not require finetuing.

In table 4, we report the GPT-4V baseline in (YYZ<sup>+</sup>23) paper, which corresponds to the best performing set up (GPT-4V ZS+history) that uses UI elements detected by IconNet (SWL<sup>+</sup>22) through set-of-marks prompting (YZL+23) for each screenshot at every step of the evaluation. The detected UI elements consist of either OCR-detected text or an icon class label, which is one of the 96 possible icon types identified by IconNet. Additionally, action history is also incorporated at each step's prompt as well. We used the exact same prompt format in (YYZ<sup>+</sup>23) except the results from the IconNet model is replaced with the output of the finetuned interactable region detection (ID) model. Interestingly, we found that the ID model can generalize well to mobile screen. By replacing the IconNet with the interactable region detection (ID) model we finetuned on the collected webpages, and incorporating local semantics of icon functionality (LS), we find OMNIPARSER delivers significantly improved performance across most sub-categories, and a 4.7% increase in the overall score compared to the best performing GPT-4V + history baseline. 

Methods	Modality	General	Install	GoogleApps	Single	WebShopping	Overall
ChatGPT-CoT	Text	5.9	4.4	10.5	9.4	8.4	7.7
PaLM2-CoT	Text	-	-	-	-	-	39.6
GPT-4V image-only	Image	41.7	42.6	49.8	72.8	45.7	50.5
GPT-4V + history	Image	43.0	46.1	49.2	78.3	48.2	53.0
OmniParser (GPT-4V)	Image	48.3	57.8	51.6	77.4	52.9	57.7

Table 4: Comparison of different methods across various tasks and overall performance in AITW benchmark.

#### 4.5 EVALUATION ON WINDOWSAGENTARENA

In this section, we report numbers in the concurrent work Windows Agent Arena (BZB<sup>+</sup>24) paper in table 5. Windows Agent Arena is a general environment for Windows operating system (OS) where the models are evaluated on 150+ user tasks in a real Windows OS and use a wide range of applications, tools, and web browsers. For cleaner comparison, we consider the cases that do not use additional UIA tree information from the windows environment. The baseline Pytesseract+DOM+Grounding DINO approach uses Pytesseract python package for OCR, Grounding DINO for icon detection and additionally incorporating DOM tree information. We observe that OmniParser achieves better performance than the baseline by only using screenshot consistently across different models.

M 1.1

Off

Web Provision

436 437

438 439 440

441 442

443

444

445

446

447

wicui	ous	moucis	onnee	WCD DIOWSCI	windows System	Counig	Micula & Muco	windows outs	10141
		Phi3-V	0%	0%	4.2%	4.3%	0.0%	0.0%	1.3%
Pytesseract	+ DOM +	GPT-40	0.0%	0.0%	29.2%	0.0%	5.0%	0.0%	5.2%
Grounding	g DINO	GPT-4V-1106	0.0%	10.3%	21.3%	12.5%	9.8%	0.0%	8.6%
		Phi3-V	0.0%	0.0%	8.6%	0.0%	5.0%	0.0%	2.0%
Omnin	orcor	GPT-40	0.0%	6.7%	30.3%	4.3%	15.3%	8.3%	9.4%
Ommp	aisei	GPT-4V-1106	2.3%	23.6%	20.8%	8.3%	20.0%	0.0%	12.5%
Human per	formance	-	75.8%	76.7%	83.3%	68.4%	42.8%	91.7%	74.5%

Windows System

Coding

Modio & Video

Windows Utik

Table 5: Comparison of different methods' performance on Windows Agent Arena benchmark.

#### 5 ABLATIONS

M 41 1

To further demonstrate OMNIPARSER is a plugin choice for off-the-shelf vision language models, we show the performance of OMNIPARSER combined with recently announced vision language models: Phi-3.5-V (AAA<sup>+</sup>24) and Llama-3.2-V (DJP<sup>+</sup>24). As seen in table 6, our finetuned interactable region detection (ID) model significantly improves the task performance compared to grounding dino model (GD) with local semantics across all subcategories for GPT-4V, Phi-3.5-V and Llama-3.2-V. In addition, the local semantics of icon functionality helps significantly with the performance for every vision language model.

Methods	Models	Model Size	Mobile	Desktop	Web	Average
OmniParser	GPT-4V	-	75.5%	77.5%	66.2%	73.0%
- w.o. ID	GPT-4V	-	74.3%	67.1%	64.1%	68.7%
- w.o. ID and w.o LS	GPT-4V	-	71.1%	45.6%	58.5%	58.4%
OmniParser	Phi-3.5-V	4.2B	39.4%	39.2%	24.0%	34.2%
- w.o. ID	Phi-3.5-V	4.2B	38.1%	32.5%	22.2%	30.9%
- w.o. ID and w.o LS	Phi-3.5-V	4.2B	32.9%	31.0%	18.5%	27.5%
OmniParser	Llama-3.2-V	11B	47.6%	48.1%	37.4%	44.4%
- w.o. ID	Llama-3.2-V	11B	45.7%	44.6%	37.5%	42.6%
- w.o. ID and w.o LS	Llama-3.2-V	11B	38.5%	37.3%	31.2%	35.6%

Table 6: Ablation study of OMNIPARSER performance using different vision language models on 460 ScreenSpot Benchmark. LS is short for local semantics of icon functionality, ID is short for the interactable region detection model we finetune. The setting w.o. ID means we replace the ID model 462 with original Grounding DINO model not finetuned on our data, and with local semantics. The setting w.o. ID and w.o LS means we use Grounding DINO model, and further without using the 463 icon description in the text prompt.

461

#### 6 DISCUSSIONS

468 In this section, we discuss a couple of common failure cases of OMNIPARSER with examples and 469 potential approach to improve. 470

Repeated Icons/Texts From analysis of the the GPT-4V's response log, we found that GPT-4V 471 often fails to make the correct prediction when the results of the OMNIPARSER contains multiple 472 repeated icons/texts, which will lead to failure if the user task requires clicking on one of the buttons. 473 This is illustrated by the figure 8 (Left) in the Appendix. A potential solution to this is to add finer 474 grain descriptions to the repeated elements in the UI screenshot, so that the GPT-4V is aware of the 475 existence of repeated elements and take it into account when predicting next action. 476

Corase Prediction of Bounding Boxes One common failure case of OMNIPARSER is that it fails 477 to detect the bounding boxes with correct granularity. In figure 8 (Right), the task is to click on the 478 text 'MORE'. The OCR module of OMNIPARSER detects text bounding box 8 which encompass 479 the desired text. But since it uses center of the box as predicted click point, it falls outside of the 480 ground truth bounding box. This is essentially due to the fact that the OCR module we use does 481 not have a notion of which text region are hyperlink and clickable. Hence we plan to train a model 482 that combines OCR and interactable region detection into one module so that it can better detect the 483 clickable text/hyperlinks. 484

**Icon Misinterpretation** We found that in some cases the icon with similar shape can have different 485 meanings depending on the UI screenshot. For example, in figure 9, the task is to find button related to 'More information', where the ground truth is to click the three dots icon in the upper right part
of the screenshot. OMNIPARSER successfully detects all the relevant bounding boxes, but the icon
description model interpret it as: "a loading or buffering indicator". We think this is due to the fact
that the icon description model is only able to see each icon cropped from image, while not able to
see the whole picture during both training and inference. So without knowing the full context of the
image, a symbol of three dots can indeed mean loading buffer in other scenarios. A potential fix to
this is to train an icon description model that is aware of the full context of the image.

## 7 CONCLUSION

In this paper, We propose OMNIPARSER, a general vision only approach that parse UI screenshots into structured elements. OMNIPARSER encompasses two finetuned models: an icon detection model and a functional description models. To train them, we curated an interactable region detection dataset using popular webpages, and an icon functional description dataset. We demonstrate that with the parsed results, the performance of GPT-4V is greatly improved on ScreenSpot benchmarks. It achieves better performance compared to GPT-4V agent that uses HTML extracted information on Mind2Web, and outperforms GPT-4V augmented with specialized Android icon detection model on AITW benchmark. We hope OMNIPARSER can serve as a general and easy-to-use tool that has the capability to parse general user screen across both PC and mobile platforms without any dependency on extra information such as HTML and view hierarchy in Android. 

# 540 REFERENCES

0-11						
542	[AAA <sup>+</sup> 24]	Marah Abdin, Jyoti Aneja, Hany Awadalla, Ahmed Awadallah, Ammar Ahmad Awan,				
543		Nguyen Bach, Amit Bahree, Arash Bakhtiari, Jianmin Bao, Harkirat Behl, Alon Ben-				
544		haim, Misha Bilenko, Johan Bjorck, Sébastien Bubeck, Martin Cai, Qin Cai, Vishrav				
545		Chaudhary, Dong Chen, Dongdong Chen, Weizhu Chen, Yen-Chun Chen, Yi-Ling Chen,				
546		Hao Cheng, Parul Chopra, Xiyang Dai, Matthew Dixon, Ronen Eldan, Victor Fragoso,				
547		Jiameng Gao, Mei Gao, Mili Gao, Amit Gaig, Ame Dei Giomo, Admistek Goswalmi, Suriya Gunasakar Emman Haider Junhang Hao, Russell I. Hewett, Wenyiang Hu				
548		Jamie Huvnh Dan Iter Sam Ade Jacobs Moian Javaberini Xin Jin Nikos Karam-				
549		patziakis, Piero Kauffmann, Mahoud Khademi, Dongwoo Kim, Young Jin Kim, Lev				
550		Kurilenko, James R. Lee, Yin Tat Lee, Yuanzhi Li, Yunsheng Li, Chen Liang, Lars Li-				
551		den, Xihui Lin, Zeqi Lin, Ce Liu, Liyuan Liu, Mengchen Liu, Weishung Liu, Xiaodong				
552		Liu, Chong Luo, Piyush Madan, Ali Mahmoudzadeh, David Majercak, Matt Mazzola,				
553		Caio César Teodoro Mendes, Arindam Mitra, Hardik Modi, Anh Nguyen, Brandon				
554		Norick, Barun Patra, Daniel Perez-Becker, Thomas Portet, Reid Pryzant, Heyang Qin,				
555		Marko Radmilac, Liliang Ren, Gustavo de Rosa, Corby Rosset, Sambudha Roy, Olatunji				
556		Kuwase, Olii Saarikivi, Amin Saled, Adii Salim, Michael Santacroce, Siniai Shan, Ning Shang Hiteshi Sharma Valong Shan Swadheen Shukla Yia Song Masahiro Tanaka				
557		Andrea Tupini Praneetha Vaddamanu Chunyu Wang Guanhua Wang Lijuan Wang				
558		Shuohang Wang, Xin Wang, Yu Wang, Rachel Ward, Wen Wen. Philipp Witte Haining				
559		Wu, Xiaoxia Wu, Michael Wyatt, Bin Xiao, Can Xu, Jiahang Xu, Weijian Xu, Jilong				
560		Xue, Sonali Yadav, Fan Yang, Jianwei Yang, Yifan Yang, Ziyi Yang, Donghan Yu,				
561		Lu Yuan, Chenruidong Zhang, Cyril Zhang, Jianwen Zhang, Li Lyna Zhang, Yi Zhang,				
562		Yue Zhang, Yunan Zhang, and Xiren Zhou. Phi-3 technical report: A highly capable				
563		language model locally on your phone, 2024.				
564	[BFH+23]	Rohan Bayishi Erich Elsen Curtis Hawthorne Maxwell Nye Augustus Odena Arushi				
565	[BEII 23]	Somani, and Sağnak Taşırlar. Introducing our multimodal models. 2023.				
566		Somani, and Sugnati Taşınar. Introducing our matimodal models, 2020.				
567	[BZB <sup>+</sup> 24]	Rogerio Bonatti, Dan Zhao, Francesco Bonacci, Dillon Dupont, Sara Abdali, Yinheng Li, Yadong Lu, Justin Wagle, Kazuhito Koishida, Arthur Bucker, Lawrence Jang, and				
568						
569		Zack Hui. Windows agent arena: Evaluating multi-modal os agents at scale, 2024.				
570	[BZX+21]	Chongyang Bai Xiaoxue Zang Ying Xu Sriniyas Sunkara Abbinay Rastogi Jindong				
571	[2211 -1]	Chen, and Blaise Aguera y Arcas. Uibert: Learning generic multimodal representations				
572		for ui understanding, 2021.				
5/3						
574	$[CSC^+24]$	Kanzhi Cheng, Qiushi Sun, Yougang Chu, Fangzhi Xu, Yantao Li, Jianbing Zhang, and				
575		Zniyong wu. Seeclick: Harnessing gui grounding for advanced visual gui agents, 2024.				
570	[DGZ <sup>+</sup> 23]	Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Samuel Stevens, Boshi Wang, Huan				
570	L ]	Sun, and Yu Su. Mind2web: Towards a generalist agent for the web, 2023.				
570						
580	$[DHF^+17]$	Biplab Deka, Zifeng Huang, Chad Franzen, Joshua Hibschman, Daniel Afergan, Yang				
581		LI, JEITTEY INICIOIS, and Kanjina Kumar. KICO: A mobile app dataset for building data driven design applications. In <i>Proceedings of the 30th Annual ACM Symposium</i>				
582		on User Interface Software and Technology IIIST '17 page 845-854 New York NY				
583		USA 2017 Association for Computing Machinery				
584						
585	[DJP <sup>+</sup> 24]	Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-				
586		Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, Anirudh				
587		Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Ko-				
588		renev, Arunur Hinsvark, Arun Kao, Aston Zhang, Aurelien Kodriguez, Austen Gregerson,				
589		Ava Spataru, Dapuste Koziere, Deutany Diron, Binn Tang, Boddle Unern, Unarlotte Caucheteux Chava Navak Chloe Bi Chris Marra Chris McConnell Christian Keller				
590		Christophe Touret, Chunyang Wu, Corinne Wong Cristian Canton Ferrer Cyrus Niko-				
591		laidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits. David Esiobu				
592		Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hup-				
593		kes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis				

594 Anderson, Graeme Nail, Gregoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, 595 Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel 596 Kloumann, Ishan Misra, Ivan Evtimov, Jade Copet, Jaewon Lee, Jan Geffert, Jana 597 Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Kartikeya Upasani, Kate 600 Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, Khalid El-Arini, Krithika Iyer, Kshitiz 601 Malik, Kuenley Chiu, Kunal Bhalla, Lauren Rantala-Yeary, Laurens van der Maaten, 602 Lawrence Chen, Liang Tan, Liz Jenkins, Louis Martin, Lovish Madaan, Lubo Malo, 603 Lukas Blecher, Lukas Landzaat, Luke de Oliveira, Madeline Muzzi, Mahesh Pasupuleti, 604 Mannat Singh, Manohar Paluri, Marcin Kardas, Mathew Oldham, Mathieu Rita, Maya 605 Pavlova, Melanie Kambadur, Mike Lewis, Min Si, Mitesh Kumar Singh, Mona Hassan, Naman Goyal, Narjes Torabi, Nikolay Bashlykov, Nikolay Bogoychev, Niladri 607 Chatterji, Olivier Duchenne, Onur Çelebi, Patrick Alrassy, Pengchuan Zhang, Peng-608 wei Li, Petar Vasic, Peter Weng, Prajjwal Bhargava, Pratik Dubal, Praveen Krishnan, 609 Punit Singh Koura, Puxin Xu, Qing He, Qingxiao Dong, Ragavan Srinivasan, Raj Ganapathy, Ramon Calderer, Ricardo Silveira Cabral, Robert Stojnic, Roberta Raileanu, 610 Rohit Girdhar, Rohit Patel, Romain Sauvestre, Ronnie Polidoro, Roshan Sumbaly, 611 Ross Taylor, Ruan Silva, Rui Hou, Rui Wang, Saghar Hosseini, Sahana Chennabas-612 appa, Sanjay Singh, Sean Bell, Seohyun Sonia Kim, Sergey Edunov, Shaoliang Nie, 613 Sharan Narang, Sharath Raparthy, Sheng Shen, Shengye Wan, Shruti Bhosale, Shun 614 Zhang, Simon Vandenhende, Soumya Batra, Spencer Whitman, Sten Sootla, Stephane 615 Collot, Suchin Gururangan, Sydney Borodinsky, Tamar Herman, Tara Fowler, Tarek 616 Sheasha, Thomas Georgiou, Thomas Scialom, Tobias Speckbacher, Todor Mihaylov, 617 Tong Xiao, Ujjwal Karn, Vedanuj Goswami, Vibhor Gupta, Vignesh Ramanathan, Viktor 618 Kerkez, Vincent Gonguet, Virginie Do, Vish Vogeti, Vladan Petrovic, Weiwei Chu, 619 Wenhan Xiong, Wenyin Fu, Whitney Meers, Xavier Martinet, Xiaodong Wang, Xiao-620 qing Ellen Tan, Xinfeng Xie, Xuchao Jia, Xuewei Wang, Yaelle Goldschlag, Yashesh Gaur, Yasmine Babaei, Yi Wen, Yiwen Song, Yuchen Zhang, Yue Li, Yuning Mao, 621 Zacharie Delpierre Coudert, Zheng Yan, Zhengxing Chen, Zoe Papakipos, Aaditya 622 Singh, Aaron Grattafiori, Abha Jain, Adam Kelsey, Adam Shajnfeld, Adithya Gangidi, 623 Adolfo Victoria, Ahuva Goldstand, Ajay Menon, Ajay Sharma, Alex Boesenberg, Alex 624 Vaughan, Alexei Baevski, Allie Feinstein, Amanda Kallet, Amit Sangani, Anam Yunus, 625 Andrei Lupu, Andres Alvarado, Andrew Caples, Andrew Gu, Andrew Ho, Andrew Poul-626 ton, Andrew Ryan, Ankit Ramchandani, Annie Franco, Aparajita Saraf, Arkabandhu 627 Chowdhury, Ashley Gabriel, Ashwin Bharambe, Assaf Eisenman, Azadeh Yazdan, Beau James, Ben Maurer, Benjamin Leonhardi, Bernie Huang, Beth Loyd, Beto De Paola, Bhargavi Paranjape, Bing Liu, Bo Wu, Boyu Ni, Braden Hancock, Bram Wasti, 630 Brandon Spence, Brani Stojkovic, Brian Gamido, Britt Montalvo, Carl Parker, Carly 631 Burton, Catalina Mejia, Changhan Wang, Changkyu Kim, Chao Zhou, Chester Hu, Ching-Hsiang Chu, Chris Cai, Chris Tindal, Christoph Feichtenhofer, Damon Civin, 632 Dana Beaty, Daniel Kreymer, Daniel Li, Danny Wyatt, David Adkins, David Xu, Davide 633 Testuggine, Delia David, Devi Parikh, Diana Liskovich, Didem Foss, Dingkang Wang, 634 Duc Le, Dustin Holland, Edward Dowling, Eissa Jamil, Elaine Montgomery, Eleonora 635 Presani, Emily Hahn, Emily Wood, Erik Brinkman, Esteban Arcaute, Evan Dunbar, 636 Evan Smothers, Fei Sun, Felix Kreuk, Feng Tian, Firat Ozgenel, Francesco Caggioni, 637 Francisco Guzmán, Frank Kanayet, Frank Seide, Gabriela Medina Florez, Gabriella 638 Schwarz, Gada Badeer, Georgia Swee, Gil Halpern, Govind Thattai, Grant Herman, 639 Grigory Sizov, Guangyi, Zhang, Guna Lakshminarayanan, Hamid Shojanazeri, Han Zou, 640 Hannah Wang, Hanwen Zha, Haroun Habeeb, Harrison Rudolph, Helen Suk, Henry 641 Aspegren, Hunter Goldman, Ibrahim Damlaj, Igor Molybog, Igor Tufanov, Irina-Elena 642 Veliche, Itai Gat, Jake Weissman, James Geboski, James Kohli, Japhet Asher, Jean-Baptiste Gaya, Jeff Marcus, Jeff Tang, Jennifer Chan, Jenny Zhen, Jeremy Reizenstein, Jeremy Teboul, Jessica Zhong, Jian Jin, Jingyi Yang, Joe Cummings, Jon Carvill, Jon Shepard, Jonathan McPhie, Jonathan Torres, Josh Ginsburg, Junjie Wang, Kai Wu, 645 Kam Hou U, Karan Saxena, Karthik Prasad, Kartikay Khandelwal, Katayoun Zand, 646 Kathy Matosich, Kaushik Veeraraghavan, Kelly Michelena, Keqian Li, Kun Huang, 647 Kunal Chawla, Kushal Lakhotia, Kyle Huang, Lailin Chen, Lakshya Garg, Lavender A,

648		Leandro Silva, Lee Bell, Lei Zhang, Liangpeng Guo, Licheng Yu, Liron Moshkovich,
649		Luca Wehrstedt, Madian Khabsa, Manav Avalani, Manish Bhatt, Maria Tsimpoukelli,
650		Martynas Mankus, Matan Hasson, Matthew Lennie, Matthias Reso, Maxim Groshev,
651		Maxim Naumov, Maya Lathi, Meghan Keneally, Michael L. Seltzer, Michal Valko,
652		Michelle Restrepo, Mihir Patel, Mik Vyatskov, Mikayel Samvelyan, Mike Clark, Mike
653		Macey, Mike Wang, Miquel Jubert Hermoso, Mo Metanat, Mohammad Rastegari, Mun-
654		ish Bansal, Nandhini Santhanam, Natascha Parks, Natasha White, Navyata Bawa, Nayan
655		Singhal, Nick Egebo, Nicolas Usunier, Nikolay Pavlovich Laptev, Ning Dong, Ning
656		Zhang, Norman Cheng, Oleg Chernoguz, Olivia Hart, Omkar Salpekar, Ozlem Kalinli,
657		Parkin Kent, Parth Parekh, Paul Saab, Pavan Balaji, Pedro Rittner, Philip Bontrager,
658		Pierre Roux, Piotr Dollar, Polina Zvyagina, Prashant Ratanchandani, Pritish Yuvraj,
659		Qian Liang, Rachad Alao, Rachel Rodriguez, Rafi Ayub, Raghotham Murthy, Raghu
660		Nayani, Rahul Mitra, Raymond Li, Rebekkah Hogan, Robin Battey, Rocky Wang,
661		Konan Maneswari, Russ Howes, Ruly Rinou, Sai Jayesn Bondu, Samyak Dalla, Sara Chugh Sara Hunt Saraun Dhillon Sacha Sideroy Satadru Dan Saurah Varma Saiii
662		Vamamoto Sharadh Ramaswamy Shaun Lindsay Shaun Lindsay Sheng Feng Sheng-
663		hao Lin Shengxin Cindy Zha Shiya Shankar Shuqiang Zhang
664		Wang Sneha Agarwal Soii Saiuvighe Soumith Chintala Stephanie Max Stephen Chen
665		Steve Kehoe, Steve Satterfield, Sudarshan Govindaprasad, Sumit Gupta, Sugmin Cho.
888		Sunny Virk, Surai Subramanian, Sy Choudhury, Sydney Goldman, Tal Remez, Tamar
667		Glaser, Tamara Best, Thilo Kohler, Thomas Robinson, Tianhe Li, Tianjun Zhang, Tim
669		Matthews, Timothy Chou, Tzook Shaked, Varun Vontimitta, Victoria Ajayi, Victoria
000		Montanez, Vijai Mohan, Vinay Satish Kumar, Vishal Mangla, Vítor Albiero, Vlad
009		Ionescu, Vlad Poenaru, Vlad Tiberiu Mihailescu, Vladimir Ivanov, Wei Li, Wenchen
070		Wang, Wenwen Jiang, Wes Bouaziz, Will Constable, Xiaocheng Tang, Xiaofang Wang,
671		Xiaojian Wu, Xiaolan Wang, Xide Xia, Xilun Wu, Xinbo Gao, Yanjun Chen, Ye Hu,
672		Ye Jia, Ye Qi, Yenda Li, Yilin Zhang, Ying Zhang, Yossi Adi, Youngjin Nam, Yu, Wang,
673		Yuchen Hao, Yundi Qian, Yuzi He, Zach Rait, Zachary DeVito, Zef Rosnbrick, Zhaoduo
674		Wen, Zhenyu Yang, and Zhiwei Zhao. The Ilama 3 herd of models, 2024.
675	[ELN]+241	Hiroli Furnta Kuang Huai Laa Ofir Naahum Vutaka Matsua Alaksandra Faust
676	[FLIN 24]	Shiving Shane Gu and Izzeddin Gur Multimodal web pavigation with instruction
677		finetuned foundation models 2024
678		iniciality roundation models, 2024.
679	[GFH <sup>+</sup> 24]	Izzeddin Gur, Hiroki Furuta, Austin Huang, Mustafa Safdari, Yutaka Matsuo, Dou-
680	1 1	glas Eck, and Aleksandra Faust. A real-world webagent with planning, long context
681		understanding, and program synthesis, 2024.
682		
683	[HSZ <sup>+</sup> 21]	Zecheng He, Srinivas Sunkara, Xiaoxue Zang, Ying Xu, Lijuan Liu, Nevan Wichers,
684		Gabriel Schubiner, Ruby Lee, Jindong Chen, and Blaise Agüera y Arcas. Actionbert:

[HWL<sup>+</sup>23] Wenyi Hong, Weihan Wang, Qingsong Lv, Jiazheng Xu, Wenmeng Yu, Junhui Ji, Yan Wang, Zihan Wang, Yuxuan Zhang, Juanzi Li, Bin Xu, Yuxiao Dong, Ming Ding, and Jie Tang. Cogagent: A visual language model for gui agents, 2023.

685 686

687

688

689 690

691

692

693 694

695

696

697

Leveraging user actions for semantic understanding of user interfaces, 2021.

- [KLJ<sup>+</sup>24] 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. arXiv preprint arXiv:2401.13649, 2024.
- [LGP<sup>+</sup>18] Evan Zheran Liu, Kelvin Guu, Panupong Pasupat, Tianlin Shi, and Percy Liang. Reinforcement learning on web interfaces using workflow-guided exploration. In *International Conference on Learning Representations (ICLR)*, 2018.
- [LHZ+20] Yang Li, Jiacong He, Xin Zhou, Yuan Zhang, and Jason Baldridge. Mapping natural language instructions to mobile UI action sequences. In Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel Tetreault, editors, *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8198–8210, Online, July 2020. Association for Computational Linguistics.

702 703 704	[LLH <sup>+</sup> 20]	Yang Li, Gang Li, Luheng He, Jingjie Zheng, Hong Li, and Zhiwei Guan. Widget captioning: Generating natural language description for mobile user interface elements, 2020.
705 706 707	[LLSH23]	Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. Blip-2: Bootstrapping language- image pre-training with frozen image encoders and large language models, 2023.
708 709 710	[OXL+22]	Arnold Overwijk, Chenyan Xiong, Xiao Liu, Cameron VandenBerg, and Jamie Callan. Clueweb22: 10 billion web documents with visual and semantic information, 2022.
710 711 712	[RLR+23]	Christopher Rawles, Alice Li, Daniel Rodriguez, Oriana Riva, and Timothy Lillicrap. Android in the wild: A large-scale dataset for android device control, 2023.
713 714 715	[SJC <sup>+</sup> 23]	Peter Shaw, Mandar Joshi, James Cohan, Jonathan Berant, Panupong Pasupat, Hexiang Hu, Urvashi Khandelwal, Kenton Lee, and Kristina Toutanova. From pixels to ui actions: Learning to follow instructions via graphical user interfaces, 2023.
716 717 718 719	[SWL+22]	Srinivas Sunkara, Maria Wang, Lijuan Liu, Gilles Baechler, Yu-Chung Hsiao, Jindong Chen, Abhanshu Sharma, and James Stout. Towards better semantic understanding of mobile interfaces. <i>CoRR</i> , abs/2210.02663, 2022.
720 721	[WLZ <sup>+</sup> 21]	Bryan Wang, Gang Li, Xin Zhou, Zhourong Chen, Tovi Grossman, and Yang Li. Screen2words: Automatic mobile ui summarization with multimodal learning, 2021.
722 723 724 725	[WXJ <sup>+</sup> 24]	Junyang Wang, Haiyang Xu, Haitao Jia, Xi Zhang, Ming Yan, Weizhou Shen, Ji Zhang, Fei Huang, and Jitao Sang. Mobile-agent-v2: Mobile device operation assistant with effective navigation via multi-agent collaboration, 2024.
726 727 728	[WXY <sup>+</sup> 24]	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, 2024.
729 730 731 732 733	[XZC <sup>+</sup> 24]	Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh Jing Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, Yitao Liu, Yiheng Xu, Shuyan Zhou, Silvio Savarese, Caiming Xiong, Victor Zhong, and Tao Yu. Osworld: Benchmarking multimodal agents for open-ended tasks in real computer environments, 2024.
734 735 736 737 738	[YYZ <sup>+</sup> 23]	An Yan, Zhengyuan Yang, Wanrong Zhu, Kevin Lin, Linjie Li, Jianfeng Wang, Jianwei Yang, Yiwu Zhong, Julian McAuley, Jianfeng Gao, Zicheng Liu, and Lijuan Wang. Gpt-4v in wonderland: Large multimodal models for zero-shot smartphone gui navigation, 2023.
739 740	[YZL+23]	Jianwei Yang, Hao Zhang, Feng Li, Xueyan Zou, Chunyuan Li, and Jianfeng Gao. Set-of-mark prompting unleashes extraordinary visual grounding in gpt-4v, 2023.
741 742 743 744	[YZS <sup>+</sup> 24]	Keen You, Haotian Zhang, Eldon Schoop, Floris Weers, Amanda Swearngin, Jeffrey Nichols, Yinfei Yang, and Zhe Gan. Ferret-ui: Grounded mobile ui understanding with multimodal llms, 2024.
745 746	[ZGK+24]	Boyuan Zheng, Boyu Gou, Jihyung Kil, Huan Sun, and Yu Su. Gpt-4v(ision) is a generalist web agent, if grounded, 2024.
747 748 749 750 751 752 753 754 755	[ZXZ <sup>+</sup> 24]	Shuyan Zhou, Frank F Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xi- anyi Cheng, Yonatan Bisk, Daniel Fried, Uri Alon, et al. Webarena: A realistic web environment for building autonomous agents. <i>ICLR</i> , 2024.

#### 756 8 APPENDIX 757

4  ...

. .

-

#### 758 8.1 DETAILS OF ICON-DESCRIPTION DATASET 759

In figure 5, we see that the original BLIP-2 model tend to focus on describing shapes and colors 760 of app icons, while struggling to recognize the semantics of the icon. This motivates us to finetune 761 this model on an icon description dataset. For the dataset, we use the result of parsed icon bounding 762 boxes inferenced by the interactable icon detection model on the ScreenSpot dataset since it contains 763 screenshots on both mobile and PC. For the description, we ask GPT-40 whether the object presented 764 in the parsed bounding box is an app icon. If GPT-40 decides the image is an icon, it outputs one-765 sentence description of the icon about the potential functionality. And if not, GPT-40 will output 'this 766 is not an icon', while still including this in the dataset. In the end, we collected 7185 icon-description 767 pairs for finetuning. 768

We manually inspected the dataset to include icon-description pairs across a wide range of functions. 769 These icons include system icons and popular software/app icons. After deduplication, we have 174 770 app icons in the pc platform, and 170 app icons in mobile platform. Further, we leveraged GPT-40 771 and conducted an analysis of the distribution of the icons. We first summarize the icons into the 772 following types: 773

....

....

c

11.11

774	1. Functional Icons. These icons represent actions or functionalities users can perform.
775	Subcategory:
776	• Navigation Icons: Back, forward, home, refresh, menu.
777	• Action Icons: Add (+), delete (trash), edit (pencil), search (magnifying glass), share,
778	upload, download.
779	• System Actions: Lock, log out, power off, settings (gear icon).
780	2. Informational Icons. These icons convey information or statuses. Examples include:
781	Subcategory:
782	• Notification Icons: Alerts, messages, updates.
783	• Status Icons: Battery level, network signal, Wi-Fi, Bluetooth, processing/loading
784	(spinner).
785	• Error or Warning Icons: Exclamation marks, red crosses, or triangles.
786	3. App-Specific Icons. These icons are unique to a specific app or service.
787	4. Media Control Icons. These icons control media playback. Examples include play, pause.
788	stop, fast forward, rewind, volume up, volume down.
789	
790	The detail distribution of each sub-category is presented in figure 4. With this dataset, the resulting
791	model demonstrates strong generalization on varied benchmarks and real-world applications.
792	We finetune BLIP-2 model for 1 epoch on the generated dataset with constant learning rate of $1e^{-5}$ ,
793	no weight decay and Adam optimizer. We show a few of the qualitative examples of finetuned model
794	vs the original model in figure 5.
795	
790	8.2 TRAINING DETAILS OF INTERACTABLE ICON REGION DETECTION MODEL
797	As introduced in 3.1, we train a VOI Ov8 model on the interactable icon region detection dataset
790	We collect in total of 66990 samples where we split 95% (63641) for training and 5% (3349) for
800	validation. We train for 20 epochs with batch size of 256, learning rate of $1e^{-3}$ , and the Adam
801	optimizer on 4 GPUs. We show the training curve in figure 6.
802	
802	8.3 DETAILS OF SEEASSIGN EVALUATION
804	
805	8.3.1 PROMPT USED FOR GPT-4V
806	GPT-4V without local semantics:
807	Here is a III screenshot image with bounding boxes and corresponding
808	labeled ID overlayed on top of it, your task is {task}. Which icon
809	box label you should operate on? Give a brief analysis, then put your
	answer in the format of $n^{*}Box$ with label ID: $[xx]^{*}n$



- **CogAgent** CogAgent number is taken from the SEEAct paper (ZGK<sup>+</sup>24), where they report cogagentchat-hf checkpoint that is not fine-tuned on Mind2Web for experiments.
  - MindAct(Gen), MindAct, GPT-3.5-Turbo, GPT-4 The numbers for these baseline are taken from



Figure 6: Training curves of interactable icon region detection model.

the Mind2Web (DGZ<sup>+</sup>23) paper, where they use HTML information to augument the corresponding
 web agent.

GPT-4V+som This model corresponds to the image annotation grounding method in SeeAct paper,
 where the som boxes extracted from the selected HTML elements are provided to GPT-4V to make
 action prediction.

**GPT-4V+textual choice** This corresponds to the best performing scenario in SeeAct paper (except the Oracle), that uses the selected HTML elments information in a multi-choice question format as input to the GPT-4V agent.

#### 8.4.1 QUALITATIVE EXAMPLES

We list a few more examples to demonstrate local semantics of icon function description helps GPT-4V make better action prediction in figure 7.

923

924

925 926

927 928

929

930 931

932

933

934

935

936



Under review as a conference paper at ICLR 2025

Task: Click 'Next song' icon

without local semantics To click on \"This PC.\" you should operate on the icon that is labeled with the ID that corresponds to the \"This PC\" text in the screenshot. In this image, the \"This PC\" text is within the boundi box labeled with ID 1.\n\n```Box with label ID:

#### GPT-4V without local semantics:

To complete the task of clicking "This PC," I need to identify the bounding box that corresponds to the "This PC!" icon or text. According to the provided list, "Text Box ID 0: This PC' is the description for the \"This PC\ element.\n\nTherefore, the answer is:\n```Box with label ID: [0]

Task: Click 'voice search icon'

without local semantics To perform a voice search, you would typically look for a microphone icon, which is commonly used to represent voice input functionality. In the screenshot provided, the icon with a microphone symbol is located in the search bar area, which is labeled with the number 44.\n\nTherefore, the answer is:\n```Box with label ID: [44]```"

#### GPT-4V without local semantics:

The \"voice search icon\" is typically represented by a microphone icon. Looking at the list of icon/text box descriptions, the description that matches this representation is 'Icon Box ID 40: a microphone icon on a white background' \n\nTherefore the bounding box label to operate on is:\n\n```Box with label ID:

Figure 7: More examples of local semantics of icon functionality help with GPT-4V in grounding actions

Task: click on enable button for alarm 18:45 46 88 at 7:30 Alarm ~ GPT-4V local semantics GPT-4V with local semantics 25 To enable the alarm set for 7:30, you should look for the toggle or switch associated with that specific alarm time. In the provided screenshot, the alarms are listed with their times, and each alarm has a corresponding :30 switch to enable or disable it. The alarm for 7:30 is listed with the text "07:30" and has additional information indicating it is a "Daily" alarm and the time remaining until it goes off 07:40 The switch to enable this alarm is typically located on the same line as the alarm 07:50 time.\n\nBased on the bounding boxes and their labels, the switch for the 7:30 alarm is likely to be the one closest to the text "07:30" and the additional information about the alarm. The bounding boxes with labels ID 23, 24, 25, and 26 appear to be switches, but we 07:55 Nutcracker, Op. 71, TH 14,.. need to identify the one associated with the 7:30 alarm.\n\nSince the alarm at 7:30 is the 08:00 first one listed and the switches are typically aligned with their corresponding alarms, the switch for the 7:30 alarm should be the first light (Arr. for Piano) one in the list of switches. Therefore, the bounding box you should operate on to enable the 7:30 alarm is the one with the label ID that Ö ((+)) , Ö corresponds to the first switch.\n\n```Box with label ID: [23]```

#### Task: More information

Analysis: The screenshot appears to be from a music streaming application, possibly Apple Music, given the text "Apple Music Classical" in Text Box ID 3. The task instruction is to find more information about "Classical Christmass". In the screenshot, there is a section with the title "Classical Christmas" (Text Box ID 2) and a brief description (Text Box ID 7) with a "MORE" link at the end (Text Box ID 8), suggesting that clicking on this link will provide additional information about the "Classical Christmas" section.in/ITO achieve the task, the bounding box with the description that most likely corresponds to the "MORE" link should be clicked. Based on the Analysis: The screenshot appears to be from a description that most likely corresponds to the "MORE" links should be clicked. Based on the bounding boxes and their descriptions, Text Box ID 8 with the text "MORE" is the most relevant and accurate choice for this action.hn/n"."In summary, the next action I will perform is: {\'Click ID': 8)```

965 Figure 8: Analysis of failure cases. All the bounding boxes are labeled by which relies only on the 966 screenshot. Left: There are in total 7 similar enable button for 7 different alarm times in the parsed 967 screenshot. And the correct Icon ID corresponding to alarm 7:30 is 27. GPT-4V fails to make the 968 correct prediction. **Right:** The ground truth region to click is the text 'MORE' inside bounding box 8. 969 We can see that the OCR fails to detect the text 'MORE' in bold, and only detects the bounding box 970 8, which encompasses 'MORE'. Since the predicts the click point as the center of the box, so it the 971 predicted click point falls outside of the ground truth region, which leads to failure in this task.

18



Figure 9: Analysis of failure cases. The task is to find button related to 'More information', and the ground truth is to click the three dots icon in the upper right part of the screenshot. The the icon functional description model does not take into account the context of this page and interpret it as: "a loading or buffering indicator" which causes the failure.