

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 OMNIPARSER, 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.

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-the-arts and creating widely usable agents that can work across multiple platforms, *e.g.* Windows/MacOS, IOS/Android and multiple applications (Web browser Office365, PhotoShop, Adobe), with most previous work focusing on limiting applications or platforms.

While large multimodal models like GPT-4V and other models trained on UI data (HWL⁺²³; YZS⁺²⁴; CSC⁺²⁴) have demonstrated abilities to understand basic elements of the UI screenshot, action grounding remains one of the key challenges in converting the actions predicted by LLMs to the actual actions on screen in terms of keyboard/mouse movement or API call (ZGK⁺²⁴). It has been noted that GPT-4V is unable to produce the exact x-y coordinate of the button location, Set-of-Mark prompting (YZL⁺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 set-of-marks prompting, GPT-4V is able to ground the action into a specific bounding box which has ground truth location instead of a specific xy coordinate value, which greatly improves the robustness of the action grounding (ZGK⁺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 platforms and applications.

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.¹

2 RELATED WORKS

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), ActionBERT (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 dataset that on UI screen understanding. Most notably the Rico dataset (DHF⁺17), which contains more than 66k unique UI screens and its view hierarchies. Later (SWL⁺22) augments Rico by providing 500k human annotations on the original 66k RICO 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. Same on mobile platform, PixelHelp (LHZ⁺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⁺23), MiniWob++(LGP⁺18), Visual-WebArena (KLJ⁺24; ZXZ⁺24), and OS-World (XZC⁺24) that provide simulated environment, but does not provide dataset explicitly for general screen understanding tasks such as interactable icon detection on real world websites.

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 available 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.

2.2 AUTONOMOUS GUI AGENT

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

¹Github repository and Huggingface model links will be made available after the review.

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

3 METHODS

To complete a complex task, T on graphical user interface, the process can usually be broken down into several steps of state-action pairs $(S_0, A_0), \dots, (S_n, A_n)$, where n is the number of steps. Each 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 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:

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

Here S_i encapsules the current information of the screen, which varies according to the model π ’s input types and the operating environment. For example, in the web tasks (ZGK⁺24), $S_i = \{h_i, \text{Img}_i\}$, where h_i is the HTML information at step i , and Img_i is the screenshot of the GUI at step i . In Android tasks (RLR⁺23), h_i is the view hierarchy information. In these cases, the model π is required to extract information directly from 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 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 detection model, a finetuned icon description model, and an OCR module. This combination produces a structured state informaton S_i , which includes a DOM-like representation of the UI and a screenshot overlaid with bounding boxes for potential interactable elements

$$\begin{aligned} a_{i+1} &= \pi(T, S_i, [(S_0, A_0), \dots, (S_{i-1}, A_{i-1})]) \\ S_i &= \{\text{Img}_i^{\text{som}}, \text{LS}_i, \text{OCR}_i^{\text{txt}}\} \\ \text{Img}_i^{\text{som}}, \text{LS}_i, \text{OCR}_i^{\text{txt}} &= \text{OMNIPARSER}(\text{Img}_i) \end{aligned} \quad (2)$$

Here $\text{Img}_i^{\text{som}}$ 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 the OMNIPARSER in more details for the rest of the section.

3.1 INTERACTABLE REGION DETECTION

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

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

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

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

166 3.2 INCORPORATING LOCAL SEMANTICS OF FUNCTIONALITY

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

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

184 4 EXPERIMENTS AND RESULTS

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

192 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
198 each task refers to one of the detected bounding boxes, e.g. ‘click on ‘settings’’, ‘click on the
199 minimize button’. During evaluation, GPT-4V is prompted to predict the bounding box ID associated
200 to it. Detailed prompt are specified in Appendix. The task screenshot images are sampled from the
201 ScreenSpot (CSC⁺24) benchmark, where they are labeled with set of marks using OMNIPARSER.
202 The tasks are further divided into 3 sub-categories by difficulty: easy (less than 10 bounding boxes),
203 medium (10-40 bounding boxes) and hard (more than 40 bounding boxes).

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

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

213 4.2 EVALUATION ON SCREENSPOT

214 ScreenSpot dataset (CSC⁺24) is a benchmark dataset that includes over 600 interface screenshots
215 from mobile (iOS, Android), desktop (macOS, Windows), and web platforms. The task instructions

216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269

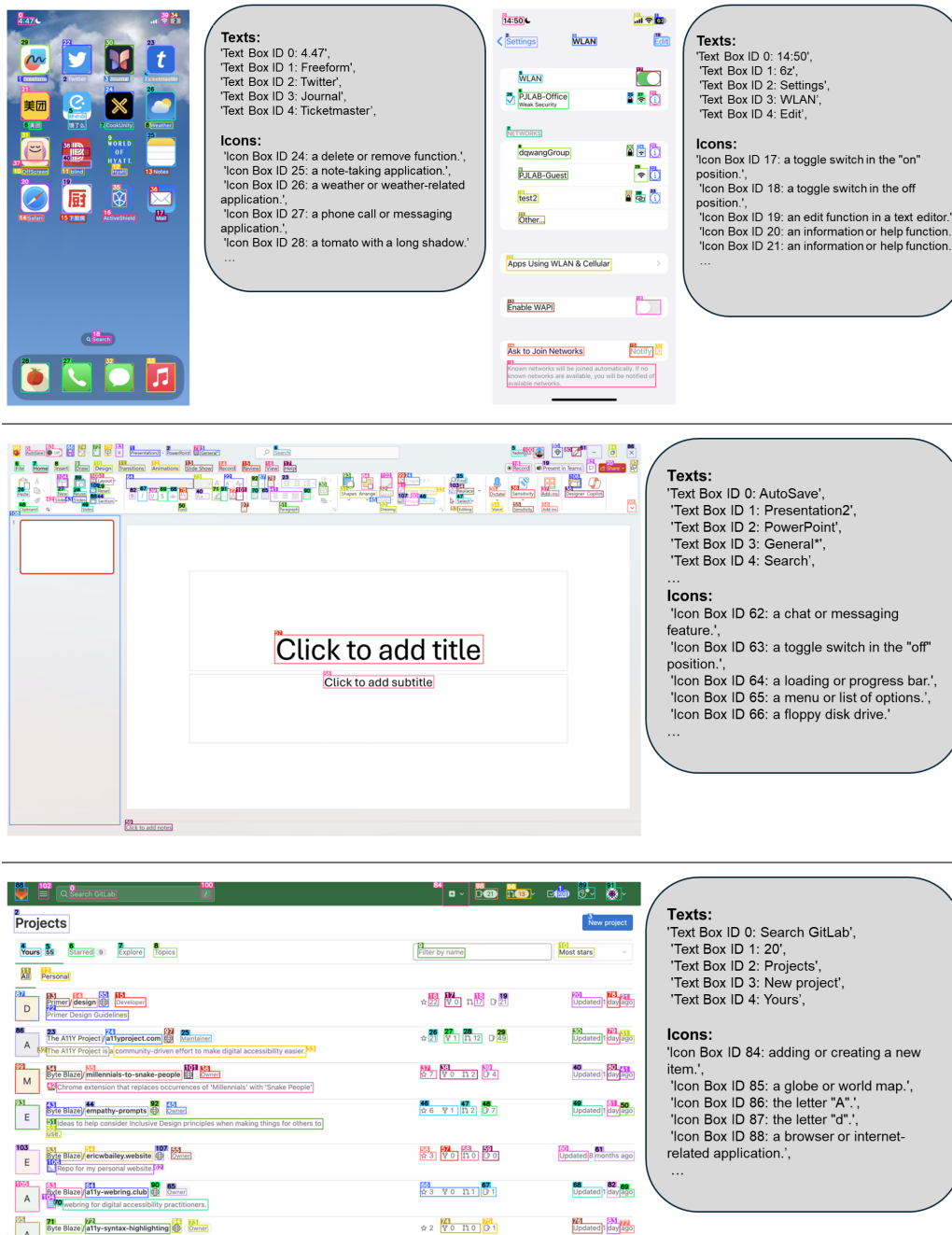


Figure 1: Examples of parsed screenshot image and local semantics by OMNIPARSER. The inputs to OmniParse are user task and UI screenshot, from which it will produce: 1) parsed screenshot image with bounding boxes and numeric IDs overlaid, and 2) local semantics containing both text extracted and icon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

270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323

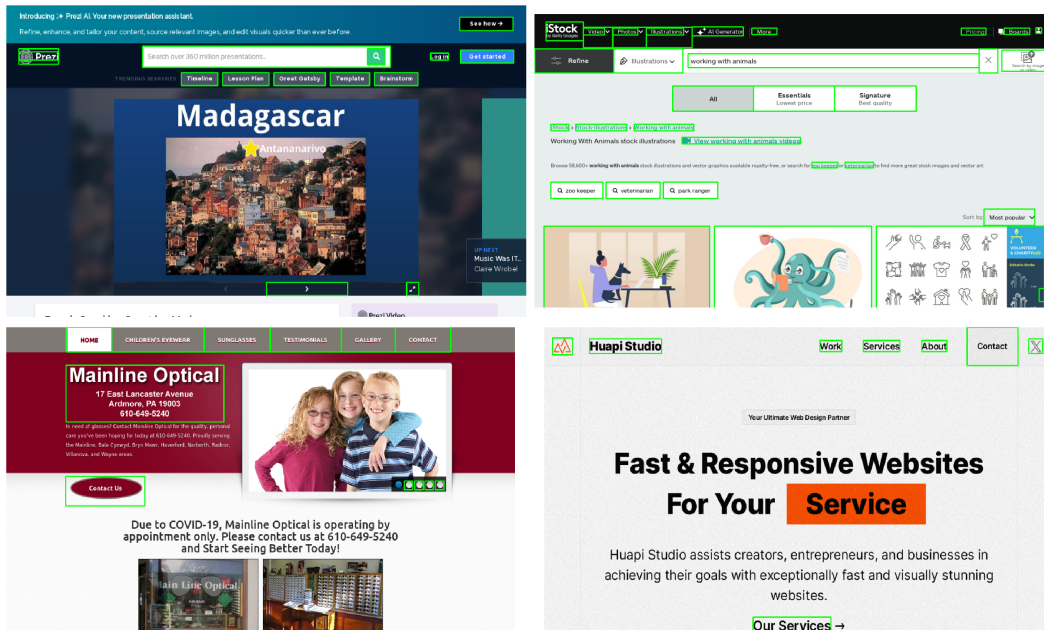


Figure 2: Examples from the Interactable Region Detection dataset. The bounding boxes are based on the interactable region extracted from the DOM tree of the webpage.

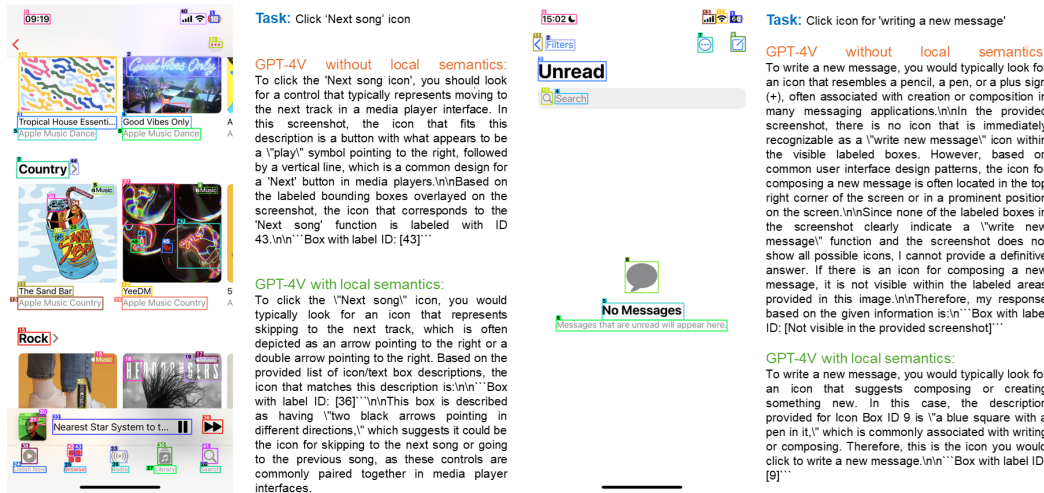


Figure 3: Examples from the SeeAssign evaluation. We can see that fine-grain local semantics improves the GPT-4V’s ability to assign correct labels to the referred icon.

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

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

Methods	Model Size	Mobile		Desktop		Web		Average
		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.

4.3 EVALUATION ON MIND2WEB

In order to test how OMNIPARSER is helpful to the web navigation scenario, We evaluate on (DGZ⁺23) benchmark. There are 3 different categories of task in the test set: Cross-Domain, Cross-Website, and Cross-Tasks. We used a cleaned version of Mind2Web tests set processed from the raw HTML dump which eliminates a small number of samples that has incorrect bounding boxes. In 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 history as text prompt, and SOM labeled screenshot to GPT-4V similar to the prompting strategy in (YYZ⁺23; ZGK⁺24). Following the original paper, we perform offline evaluation focusing on the element accuracy, Operation F1 and step success rate averaged across the task.

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⁺24; CSC⁺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 Appendix 8.4.

In the second section of the table (row 4-9) we report numbers from Mind2web paper (DGZ⁺23) and SeeAct (ZGK⁺24) paper. In this section, all of the approaches use the HTML elements selected by a finetuned element proposal model on Mind2Web training set which produces top 50 relevant elements 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. In GPT-4V+SOM, the set of mark (SOM) boxes are selected from the element proposal model, and 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 bounding boxes on top of screenshot. The better performance of textual choice corroborates with the experiment results in 4.1.

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 an accurate action prediction. Lastly, OMNIPARSER with GPT-4V significantly outperforms all the other trained models using only UI screenshot such as SeeClick and Qwen-VL.

Methods	Input Types		Cross-Website			Cross-Domain			Cross-Task		
	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	✓	✓	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	×	✓	-	-	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 addition to evaluation on multi-step web browsing tasks, we assess OMNIPARSER on the mobile navigating benchmark AITW (RLR⁺23), which contains 30k instructions and 715k trajectories. We use the same train/test split as in (CSC⁺24) based on instructions, which retain only one trajectory for each instruction 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 finetuning.

In table 4, we report the GPT-4V baseline in (YYZ⁺23) paper, which corresponds to the best performing set up (GPT-4V ZS+history) that uses UI elements detected by IconNet (SWL⁺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⁺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⁺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.

Methods	Models	Office	Web Browser	Windows System	Coding	Media & Video	Windows Utils	Total
Pytesseract + DOM + Grounding DINO	Phi3-V	0%	0%	4.2%	4.3%	0.0%	0.0%	1.3%
	GPT-4o	0.0%	0.0%	29.2%	0.0%	5.0%	0.0%	5.2%
	GPT-4V-1106	0.0%	10.3%	21.3%	12.5%	9.8%	0.0%	8.6%
Omniparser	Phi3-V	0.0%	0.0%	8.6%	0.0%	5.0%	0.0%	2.0%
	GPT-4o	0.0%	6.7%	30.3%	4.3%	15.3%	8.3%	9.4%
	GPT-4V-1106	2.3%	23.6%	20.8%	8.3%	20.0%	0.0%	12.5%
Human performance	-	75.8%	76.7%	83.3%	68.4%	42.8%	91.7%	74.5%

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

5 ABLATIONS

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+24) and Llama-3.2-V (DJP+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 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 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.

6 DISCUSSIONS

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

Repeated Icons/Texts From analysis of the the GPT-4V’s response log, we found that GPT-4V often fails to make the correct prediction when the results of the OMNIPARSER contains multiple repeated icons/texts, which will lead to failure if the user task requires clicking on one of the buttons. This is illustrated by the figure 7 (Left) in the Appendix. A potential solution to this is to add finer grain descriptions to the repeated elements in the UI screenshot, so that the GPT-4V is aware of the existence of repeated elements and take it into account when predicting next action.

Corase Prediction of Bounding Boxes One common failure case of OMNIPARSER is that it fails to detect the bounding boxes with correct granularity. In figure 7 (Right), the task is to click on the text ‘MORE’. The OCR module of OMNIPARSER detects text bounding box 8 which encompass the desired text. But since it uses center of the box as predicted click point, it falls outside of the ground truth bounding box. This is essentially due to the fact that the OCR module we use does not have a notion of which text region are hyperlink and clickable. Hence we plan to train a model that combines OCR and interactable region detection into one module so that it can better detect the clickable text/hyperlinks.

Icon Misinterpretation We found that in some cases the icon with similar shape can have different meanings depending on the UI screenshot. For example, in figure 8, the task is to find button related

486 to 'More information', where the ground truth is to click the three dots icon in the upper right part
487 of the screenshot. OMNIPARSER successfully detects all the relevant bounding boxes, but the icon
488 description model interpret it as: "a loading or buffering indicator". We think this is due to the fact
489 that the icon description model is only able to see each icon cropped from image, while not able to
490 see the whole picture during both training and inference. So without knowing the full context of the
491 image, a symbol of three dots can indeed mean loading buffer in other scenarios. A potential fix to
492 this is to train an icon description model that is aware of the full context of the image.

493 494 7 CONCLUSION

495
496 In this paper, We propose OMNIPARSER, a general vision only approach that parse UI screenshots
497 into structured elements. OMNIPARSER encompasses two finetuned models: an icon detection model
498 and a functional description models. To train them, we curated an interactable region detection
499 dataset using popular webpages, and an icon functional description dataset. We demonstrate that with
500 the parsed results, the performance of GPT-4V is greatly improved on ScreenSpot benchmarks. It
501 achieves better performance compared to GPT-4V agent that uses HTML extracted information on
502 Mind2Web, and outperforms GPT-4V augmented with specialized Android icon detection model on
503 AITW benchmark. We hope OMNIPARSER can serve as a general and easy-to-use tool that has the
504 capability to parse general user screen across both PC and mobile platforms without any dependency
505 on extra information such as HTML and view hierarchy in Android.
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539

REFERENCES

- 540
541
542 [AAA⁺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 Jianfeng Gao, Mei Gao, Min Gao, Amit Garg, Allie Del Giorno, Abhishek Goswami,
548 Suriya Gunasekar, Emman Haider, Junheng Hao, Russell J. Hewett, Wenxiang Hu,
549 Jamie Huynh, Dan Iter, Sam Ade Jacobs, Mojan Javaheripi, Xin Jin, Nikos Karam-
550 patziakis, Piero Kauffmann, Mahoud Khademi, Dongwoo Kim, Young Jin Kim, Lev
551 Kurilenko, James R. Lee, Yin Tat Lee, Yuanzhi Li, Yunsheng Li, Chen Liang, Lars Li-
552 den, Xihui Lin, Zeqi Lin, Ce Liu, Liyuan Liu, Mengchen Liu, Weishung Liu, Xiaodong
553 Liu, Chong Luo, Piyush Madan, Ali Mahmoudzadeh, David Majercak, Matt Mazzola,
554 Caio César Teodoro Mendes, Arindam Mitra, Hardik Modi, Anh Nguyen, Brandon
555 Norick, Barun Patra, Daniel Perez-Becker, Thomas Portet, Reid Pryzant, Heyang Qin,
556 Marko Radmilac, Liliang Ren, Gustavo de Rosa, Corby Rosset, Sambudha Roy, Olatunji
557 Ruwase, Olli Saarikivi, Amin Saied, Adil Salim, Michael Santacroce, Shital Shah, Ning
558 Shang, Hiteshi Sharma, Yelong Shen, Swadheen Shukla, Xia Song, Masahiro Tanaka,
559 Andrea Tupini, Praneetha Vaddamanu, Chunyu Wang, Guanhua Wang, Lijuan Wang,
560 Shuhang Wang, Xin Wang, Yu Wang, Rachel Ward, Wen Wen, Philipp Witte, Haiping
561 Wu, Xiaoxia Wu, Michael Wyatt, Bin Xiao, Can Xu, Jiahang Xu, Weijian Xu, Jilong
562 Xue, Sonali Yadav, Fan Yang, Jianwei Yang, Yifan Yang, Ziyi Yang, Donghan Yu,
563 Lu Yuan, Chenruidong Zhang, Cyril Zhang, Jianwen Zhang, Li Lyna Zhang, Yi Zhang,
564 Yue Zhang, Yunan Zhang, and Xiren Zhou. Phi-3 technical report: A highly capable
565 language model locally on your phone, 2024.
- 566 [BEH⁺23] Rohan Bavishi, Erich Elsen, Curtis Hawthorne, Maxwell Nye, Augustus Odena, Arushi
567 Somani, and Saĝnak Taşırlar. Introducing our multimodal models, 2023.
- 568 [BZB⁺24] Rogerio Bonatti, Dan Zhao, Francesco Bonacci, Dillon Dupont, Sara Abdali, Yinheng
569 Li, Yadong Lu, Justin Wagle, Kazuhito Koishida, Arthur Bucker, Lawrence Jang, and
570 Zack Hui. Windows agent arena: Evaluating multi-modal os agents at scale, 2024.
- 571 [BZX⁺21] Chongyang Bai, Xiaoxue Zang, Ying Xu, Srinivas Sunkara, Abhinav Rastogi, Jindong
572 Chen, and Blaise Aguera y Arcas. Uibert: Learning generic multimodal representations
573 for ui understanding, 2021.
- 574 [CSC⁺24] Kanzhi Cheng, Qiushi Sun, Yougang Chu, Fangzhi Xu, Yantao Li, Jianbing Zhang, and
575 Zhiyong Wu. Seeclck: Harnessing gui grounding for advanced visual gui agents, 2024.
- 576 [DGZ⁺23] Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Samuel Stevens, Boshi Wang, Huan
577 Sun, and Yu Su. Mind2web: Towards a generalist agent for the web, 2023.
- 578 [DHF⁺17] Biplab Deka, Zifeng Huang, Chad Franzen, Joshua Hibsman, Daniel Afergan, Yang
579 Li, Jeffrey Nichols, and Ranjitha Kumar. Rico: A mobile app dataset for building
580 data-driven design applications. In *Proceedings of the 30th Annual ACM Symposium*
581 *on User Interface Software and Technology*, UIST '17, page 845–854, New York, NY,
582 USA, 2017. Association for Computing Machinery.
- 583 [DJP⁺24] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-
584 Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, Anirudh
585 Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Ko-
586 renev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurelien Rodriguez, Austen Gregerson,
587 Ava Spataru, Baptiste Roziere, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte
588 Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller,
589 Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Niko-
590 laidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, David Esiobu,
591 Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hup-
592 kes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael
593 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 Bil-
598 lock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu,
599 Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua
600 Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Kartikeya Upasani, Kate
601 Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, Khalid El-Arini, Krithika Iyer, Kshitiz
602 Malik, Kuenley Chiu, Kunal Bhalla, Lauren Rantala-Yearly, Laurens van der Maaten,
603 Lawrence Chen, Liang Tan, Liz Jenkins, Louis Martin, Lovish Madaan, Lubo Malo,
604 Lukas Blecher, Lukas Landzaat, Luke de Oliveira, Madeline Muzzi, Mahesh Pasupuleti,
605 Mannat Singh, Manohar Paluri, Marcin Kardas, Mathew Oldham, Mathieu Rita, Maya
606 Pavlova, Melanie Kambadur, Mike Lewis, Min Si, Mitesh Kumar Singh, Mona Has-
607 san, Naman Goyal, Narjes Torabi, Nikolay Bashlykov, Nikolay Bogoychev, Niladri
608 Chatterji, Olivier Duchenne, Onur Çelebi, Patrick Alrassy, Pengchuan Zhang, Peng-
609 wei Li, Petar Vasic, Peter Weng, Prajjwal Bhargava, Pratik Dubal, Praveen Krishnan,
610 Punit Singh Koura, Puxin Xu, Qing He, Qingxiao Dong, Ragavan Srinivasan, Raj
611 Ganapathy, Ramon Calderer, Ricardo Silveira Cabral, Robert Stojnic, Roberta Raileanu,
612 Rohit Girdhar, Rohit Patel, Romain Sauvestre, Ronnie Polidoro, Roshan Sumbaly,
613 Ross Taylor, Ruan Silva, Rui Hou, Rui Wang, Saghar Hosseini, Sahana Chennabas-
614 appa, Sanjay Singh, Sean Bell, Seohyun Sonia Kim, Sergey Edunov, Shaoliang Nie,
615 Sharan Narang, Sharath Rapparthi, Sheng Shen, Shengye Wan, Shruti Bhosale, Shun
616 Zhang, Simon Vandenhende, Soumya Batra, Spencer Whitman, Sten Sootla, Stephane
617 Collot, Suchin Gururangan, Sydney Borodinsky, Tamar Herman, Tara Fowler, Tarek
618 Sheasha, Thomas Georgiou, Thomas Scialom, Tobias Speckbacher, Todor Mihaylov,
619 Tong Xiao, Ujjwal Karn, Vedanuj Goswami, Vibhor Gupta, Vignesh Ramanathan, Viktor
620 Kerkez, Vincent Gonguet, Virginie Do, Vish Vogeti, Vladan Petrovic, Weiwei Chu,
621 Wenhan Xiong, Wenyin Fu, Whitney Meers, Xavier Martinet, Xiaodong Wang, Xiao-
622 qing Ellen Tan, Xinfeng Xie, Xuchao Jia, Xuwei Wang, Yaelle Goldschlag, Yashesh
623 Gaur, Yasmine Babaei, Yi Wen, Yiwen Song, Yuchen Zhang, Yue Li, Yuning Mao,
624 Zacharie Delpierre Coudert, Zheng Yan, Zhengxing Chen, Zoe Papakipos, Aaditya
625 Singh, Aaron Grattafiori, Abha Jain, Adam Kelsey, Adam Shajnfeld, Adithya Gangidi,
626 Adolfo Victoria, Ahuva Goldstand, Ajay Menon, Ajay Sharma, Alex Boesenberg, Alex
627 Vaughan, Alexei Baevski, Allie Feinstein, Amanda Kallet, Amit Sangani, Anam Yunus,
628 Andrei Lupu, Andres Alvarado, Andrew Caples, Andrew Gu, Andrew Ho, Andrew Poul-
629 ton, Andrew Ryan, Ankit Ramchandani, Annie Franco, Aparajita Saraf, Arkabandhu
630 Chowdhury, Ashley Gabriel, Ashwin Bharambe, Assaf Eisenman, Azadeh Yazdan,
631 Beau James, Ben Maurer, Benjamin Leonhardi, Bernie Huang, Beth Loyd, Beto De
632 Paola, Bhargavi Paranjape, Bing Liu, Bo Wu, Boyu Ni, Braden Hancock, Bram Wasti,
633 Brandon Spence, Brani Stojkovic, Brian Gamido, Britt Montalvo, Carl Parker, Carly
634 Burton, Catalina Mejia, Changhan Wang, Changkyu Kim, Chao Zhou, Chester Hu,
635 Ching-Hsiang Chu, Chris Cai, Chris Tindal, Christoph Feichtenhofer, Damon Civin,
636 Dana Beaty, Daniel Kreymer, Daniel Li, Danny Wyatt, David Adkins, David Xu, Davide
637 Testuggine, Delia David, Devi Parikh, Diana Liskovich, Didem Foss, Dingkan Wang,
638 Duc Le, Dustin Holland, Edward Dowling, Eissa Jamil, Elaine Montgomery, Eleonora
639 Presani, Emily Hahn, Emily Wood, Erik Brinkman, Esteban Arcaute, Evan Dunbar,
640 Evan Smothers, Fei Sun, Felix Kreuk, Feng Tian, Firat Ozgenel, Francesco Caggioni,
641 Francisco Guzmán, Frank Kanayet, Frank Seide, Gabriela Medina Florez, Gabriella
642 Schwarz, Gada Badeer, Georgia Swee, Gil Halpern, Govind Thattai, Grant Herman,
643 Grigory Sizov, Guangyi, Zhang, Guna Lakshminarayanan, Hamid Shojanazeri, Han Zou,
644 Hannah Wang, Hanwen Zha, Haroun Habeeb, Harrison Rudolph, Helen Suk, Henry
645 Aspegren, Hunter Goldman, Ibrahim Damla, Igor Molybog, Igor Tufanov, Irina-Elena
646 Veliche, Itai Gat, Jake Weissman, James Geboski, James Kohli, Japhet Asher, Jean-
647 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,
Kam Hou U, Karan Saxena, Karthik Prasad, Kartikay Khandelwal, Katayoun Zand,
Kathy Matosich, Kaushik Veeraraghavan, Kelly Michelena, Keqian Li, Kun Huang,
Kunal Chawla, Kushal Lakhota, 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 Khabisa, 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 Rohan Maheswari, Russ Howes, Ruty Rinott, Sai Jayesh Bondu, Samyak Datta, Sara
 662 Chugh, Sara Hunt, Sargun Dhillon, Sasha Sidorov, Satadru Pan, Saurabh Verma, Seiji
 663 Yamamoto, Sharadh Ramaswamy, Shaun Lindsay, Shaun Lindsay, Sheng Feng, Sheng-
 664 hao Lin, Shengxin Cindy Zha, Shiva Shankar, Shuqiang Zhang, Shuqiang Zhang, Sinong
 665 Wang, Sneha Agarwal, Soji Sajuyigbe, Soumith Chintala, Stephanie Max, Stephen Chen,
 666 Steve Kehoe, Steve Satterfield, Sudarshan Govindaprasad, Sumit Gupta, Sungmin Cho,
 667 Sunny Virk, Suraj Subramanian, Sy Choudhury, Sydney Goldman, Tal Remez, Tamar
 668 Glaser, Tamara Best, Thilo Kohler, Thomas Robinson, Tianhe Li, Tianjun Zhang, Tim
 669 Matthews, Timothy Chou, Tzook Shaked, Varun Vontimitta, Victoria Ajayi, Victoria
 670 Montanez, Vijai Mohan, Vinay Satish Kumar, Vishal Mangla, Vitor Albiero, Vlad
 671 Ionescu, Vlad Poenaru, Vlad Tiberiu Mihailescu, Vladimir Ivanov, Wei Li, Wenchen
 672 Wang, Wenwen Jiang, Wes Bouaziz, Will Constable, Xiaocheng Tang, Xiaofang Wang,
 673 Xiaojian Wu, Xiaolan Wang, Xide Xia, Xilun Wu, Xinbo Gao, Yanjun Chen, Ye Hu,
 674 Ye Jia, Ye Qi, Yenda Li, Yilin Zhang, Ying Zhang, Yossi Adi, Youngjin Nam, Yu, Wang,
 675 Yuchen Hao, Yundi Qian, Yuzi He, Zach Rait, Zachary DeVito, Zef Rosnbrick, Zhaoduo
 676 Wen, Zhenyu Yang, and Zhiwei Zhao. The llama 3 herd of models, 2024.
- 675 [FLN⁺24] Hiroki Furuta, Kuang-Huei Lee, Ofir Nachum, Yutaka Matsuo, Aleksandra Faust,
 676 Shixiang Shane Gu, and Izzeddin Gur. Multimodal web navigation with instruction-
 677 finetuned foundation models, 2024.
- 678 [GFH⁺24] Izzeddin Gur, Hiroki Furuta, Austin Huang, Mustafa Safdari, Yutaka Matsuo, Dou-
 679 glas Eck, and Aleksandra Faust. A real-world webagent with planning, long context
 680 understanding, and program synthesis, 2024.
- 681 [HSZ⁺21] Zecheng He, Srinivas Sunkara, Xiaoxue Zang, Ying Xu, Lijuan Liu, Nevan Wichers,
 682 Gabriel Schubiner, Ruby Lee, Jindong Chen, and Blaise Agüera y Arcas. Actionbert:
 683 Leveraging user actions for semantic understanding of user interfaces, 2021.
- 684 [HWL⁺23] Wenyi Hong, Weihang Wang, Qingsong Lv, Jiazheng Xu, Wenmeng Yu, Junhui Ji, Yan
 685 Wang, Zihan Wang, Yuxuan Zhang, Juanzi Li, Bin Xu, Yuxiao Dong, Ming Ding, and
 686 Jie Tang. Cogagent: A visual language model for gui agents, 2023.
- 687 [KLJ⁺24] Jing Yu Koh, Robert Lo, Lawrence Jang, Vikram Duvvur, Ming Chong Lim, Po-
 688 Yu Huang, Graham Neubig, Shuyan Zhou, Ruslan Salakhutdinov, and Daniel Fried.
 689 Visualwebarena: Evaluating multimodal agents on realistic visual web tasks. *arXiv
 preprint arXiv:2401.13649*, 2024.
- 690 [LGP⁺18] Evan Zheran Liu, Kelvin Guu, Panupong Pasupat, Tianlin Shi, and Percy Liang. Rein-
 691 forcement learning on web interfaces using workflow-guided exploration. In *International
 692 Conference on Learning Representations (ICLR)*, 2018.
- 693 [LHZ⁺20] Yang Li, Jiacong He, Xin Zhou, Yuan Zhang, and Jason Baldridge. Mapping natural
 694 language instructions to mobile UI action sequences. In Dan Jurafsky, Joyce Chai,
 695 Natalie Schluter, and Joel Tetreault, editors, *Proceedings of the 58th Annual Meeting of
 696 the Association for Computational Linguistics*, pages 8198–8210, Online, July 2020.
 697 Association for Computational Linguistics.

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

8 APPENDIX

8.1 DETAILS OF ICON-DESCRIPTION DATASET

In figure 4, we see that the original BLIP-2 model tend to focus on describing shapes and colors of app icons, while struggling to recognize the semantics of the icon. This motivates us to finetune this model on an icon description dataset. For the dataset, we use the result of parsed icon bounding boxes inferred by the interactive icon detection model on the ScreenSpot dataset since it contains screenshots on both mobile and PC. For the description, we ask GPT-4o whether the object presented in the parsed bounding box is an app icon. If GPT-4o decides the image is an icon, it outputs one-sentence description of the icon about the potential functionality. And if not, GPT-4o will output 'this is not an icon', while still including this in the dataset. In the end, we collected 7185 icon-description pairs for finetuning.

We finetune BLIP-2 model for 1 epoch on the generated dataset with constant learning rate of $1e^{-5}$, no weight decay and Adam optimizer. We show a few of the qualitative examples of finetuned model vs the original model in figure 4.







	Before	After
	an app icon with a pie chart on it	a presentation or screen sharing application
	the microsoft office logo is shown in a circle	Microsoft Outlook, an email application.
	an iphone app with an image of a flower	Photos, a photo-sharing application.
	an orange and white logo with a smiley face	Discord, a messaging and voice chat application.
	a blue app icon with a person on it	a location or location-related function.
	a grey and white image of a gear wheel	Settings.

Figure 4: Example comparisons of icon description model using BLIP-2 (Left) and its finetuned version (Right). Original BLIP-2 model tend to focus on describing shapes and colors of app icons. After finetuning on the functionality semantics dataset, the model is able to show understanding of semantics of some common app icons.

8.2 TRAINING DETAILS OF INTERACTABLE ICON REGION DETECTION MODEL

As introduced in 3.1, we train a YOLOv8 model on the interactable icon region detection dataset. We collect in total of 66990 samples where we split 95% (63641) for training, and 5% (3349) for validation. We train for 20 epochs with batch size of 256, learning rate of $1e^{-3}$, and the Adam optimizer on 4 GPUs. We show the training curve in figure 5.

8.3 DETAILS OF SEEASSIGN EVALUATION

8.3.1 PROMPT USED FOR GPT-4V

GPT-4V without local semantics:

Here is a UI screenshot image with bounding boxes and corresponding labeled ID overlaid on top of it, your task is {task}. Which icon box label you should operate on? Give a brief analysis, then put your answer in the format of \n```\nBox with label ID: [xx]```\n

GPT-4V with local semantics:

810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863

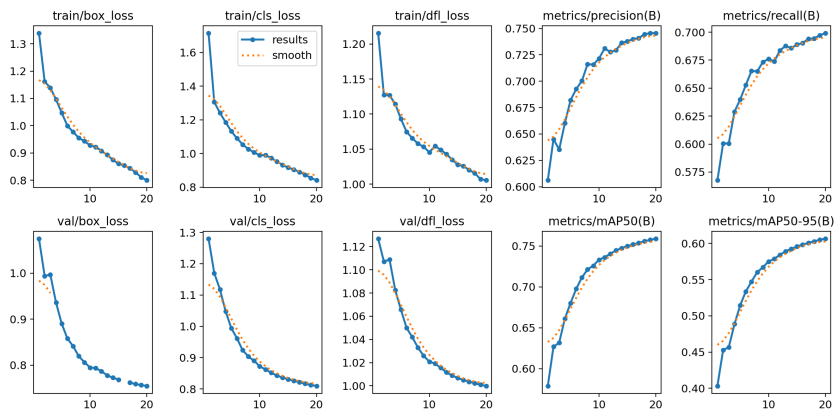


Figure 5: Training curves of interactive icon region detection model.

Here is a UI screenshot image with bounding boxes and corresponding labeled ID overlaid on top of it, and here is a list of icon/text box description: {parsed_local_semantics}. Your task is {task}. Which bounding box label you should operate on? Give a brief analysis, then put your answer in the format of `\n```\nBox with label ID: [xx]```\n`

8.4 DETAILS OF MIND2WEB EVALUATION

Here we list more details of each baseline in table 3.

SeeClick, QWen-VL SeeClick is a finetuned version of Qwen-VL on the Mind2Web training set and we report both of their numbers in their paper.

CogAgent CogAgent number is taken from the SEEAct paper (ZGK⁺24), where they report cogagent-chat-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 the Mind2Web (DGZ⁺23) paper, where they use HTML information to augment the corresponding web agent.

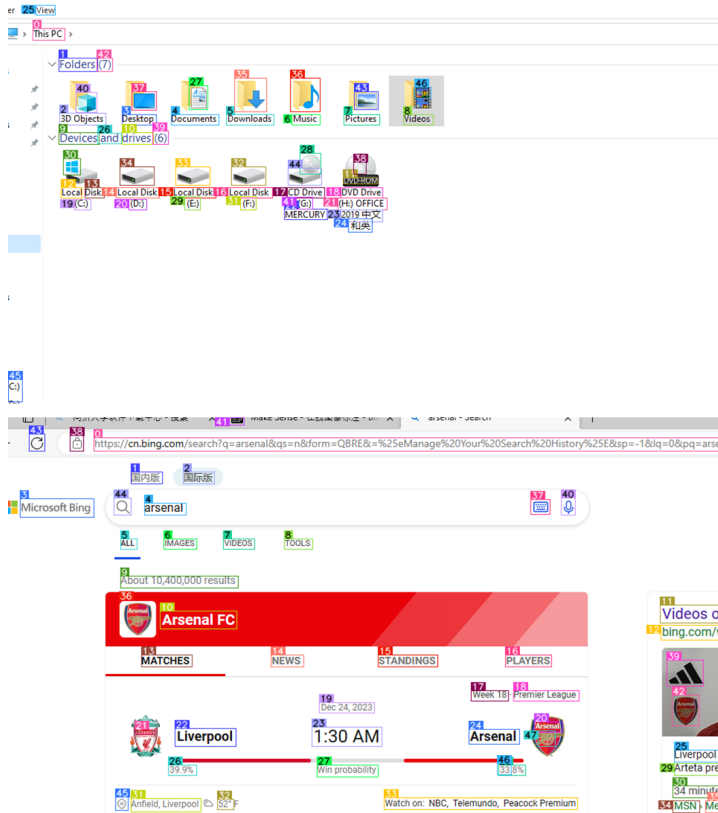
GPT-4V+som This model corresponds to the image annotation grounding method in SeeAct paper, where the som boxes extracted from the selected HTML elements are provided to GPT-4V to make action prediction.

GPT-4V+textual choice This corresponds to the best performing scenario in SeeAct paper (except the Oracle), that uses the selected HTML elements 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 6.

864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917



Task: Click 'Next song' icon

GPT-4V 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 bounding box labeled with ID 1. Therefore, the answer is: "Box with label ID: [1]"

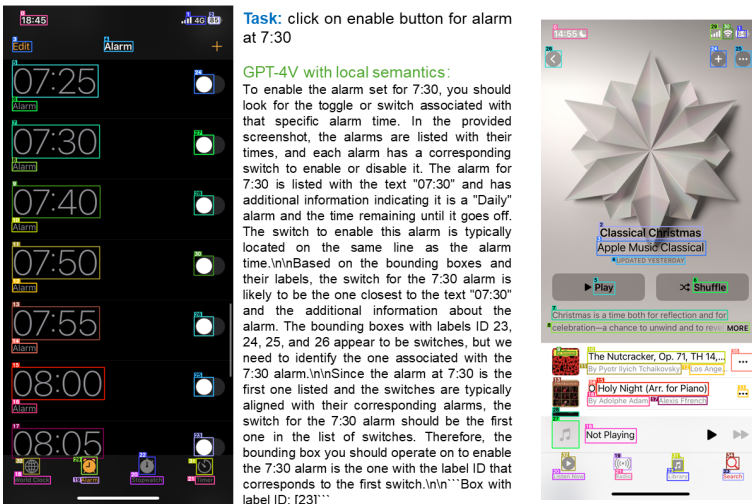
GPT-4V without local semantics:
To complete the task of clicking "This PC," I need to identify the bounding box that corresponds to the "This PC" icon or text. According to the provided list, "Text Box ID 0: This PC" is the description for the "This PC" element. Therefore, the answer is: "Box with label ID: [0]"

Task: Click 'voice search icon'

GPT-4V 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. Therefore, the answer is: "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." Therefore, the bounding box label to operate on is: "Box with label ID: [40]"

Figure 6: More examples of local semantics of icon functionality help with GPT-4V in grounding actions



Task: click on enable button for alarm at 7:30

GPT-4V with local semantics:
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 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. The switch to enable this alarm is typically located on the same line as the alarm time. Based 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 need to identify the one associated with the 7:30 alarm. Since the alarm at 7:30 is the first one listed and the switches are typically aligned with their corresponding alarms, the switch for the 7:30 alarm should be the first 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 corresponds to the first switch. "Box with label ID: [23]"

Task: More information

GPT-4V local semantics:
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 Christmas." 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. To achieve the task, the bounding box with the description that most likely corresponds to the "MORE" link 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. In summary, the next action I will perform is: "Click ID: 8"

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

918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971

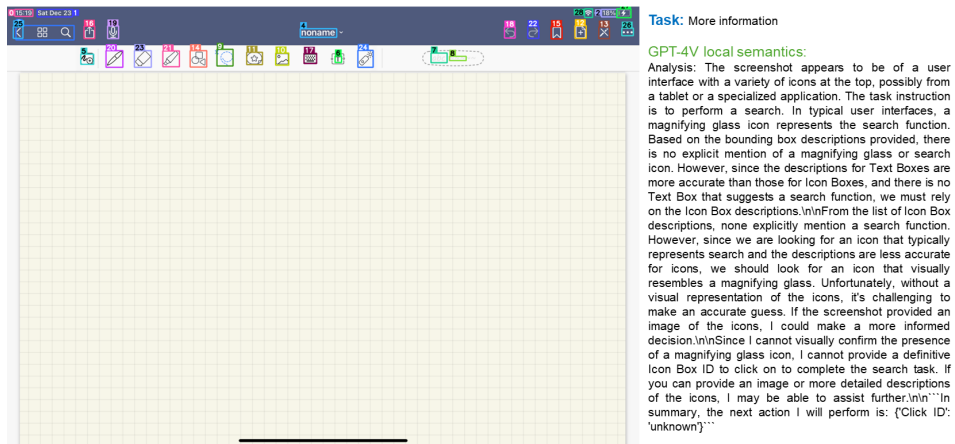


Figure 8: Analysis of failure cases. The task is to find button related to 'More information', and the ground truth is to click the three dots icon in the upper right part of the screenshot. The the icon functional description model does not take into account the context of this page and interpret it as: "a loading or buffering indicator" which causes the failure.