This CVPR paper is the Open Access version, provided by the Computer Vision Foundation.
Except for this watermark, it is identical to the accepted version;
the final published version of the proceedings is available on IEEE Xplore.

UI-Lens: Assessing General MLLMs' Potential to Automate UI Display Quality Assurance

Wei Xiang, Yexinrui Wu, Xinli Chen, Xinran Li, Shi Chen†
Zhejiang University
wxiang@zju.edu.cn, yexinruiwu@zju.edu.cn

Abstract

The goal of multimodal large language models (MLLMs) is not simply recognition, but visual discernment. This includes the critical capability to distinguish whether content is partially rendered or merely obscured by another element. Current models, which excel at object identification, fundamentally struggle with this ability, limiting their robustness in complex, layered environments. This challenge is acutely evident in User Interface (UI) display defect detection, which requires fine-grained element boundary understanding, missing-content detection, and reasoning about sequential interface semantic consistency. However, the capabilities of multimodal large language models (MLLMs) and vision-language models (VLMs) for detecting UI defect in realistic and complex interfaces have not been systematically validated. To fill this gap, we present ***UI-Lens****, a UI display defect detection benchmark for multilingual UI scenarios. The dataset consists of 4,759 Chinese and 3,392 English interfaces meticulously annotated by design experts, covering six display defect categories. We conduct a systematic evaluation of 9 mainstream models (7 closed-source, 2 open-source). Results show clear shortcomings in current models: for tasks requiring fine-grained element boundary understanding, performance is near-random, with task-average F1 scores of 22.19% and 33.75% on Text Overflow and Container Overlap, respectively; for sequential interface semantic consistency, the task-average F1 score is only 11.44%, indicating severe underperformance. We release UI-Lens to catalyze research toward robust UI display defect detection with fine-grained boundary awareness in realistic interfaces. The dataset is available on this link.*

1. Introduction

While general-purpose MLLMs [46, 48] and VLMs [49] demonstrate remarkable success in understanding normal user interfaces (UIs) [18], their ability to identify defective UIs remains critically underexplored [6]. This reflects a systematic tendency in MLLMs, which are fundamentally biased towards what an object is, rather than how it is presented or its contextual state. UI display defects—such as text overflow, cropped content, or container overlap—are prevalent in complex, large-scale commercial applications [1, 24, 25, 27, 31, 34, 51]. These display defects severely degrade interface usability and user trust [4, 33], yet existing MLLM research, often focused on tasks like OCR [5, 11, 21] or UI understanding [2, 41, 44] on clean pages, fails to address this pressing challenge.

UI display defect detection demands a model's comprehensive capabilities in fine-grained element boundary understanding [10], content-missing awareness [13], and semantic consistency judgment [4]. As illustrated in Figure 1, display defects such as Cropped Content can involve only a handful of occluded pixels, whereas Text Inconsistency often demands reasoning across multiple pages or components [14]. These demands place the task beyond the coverage of existing UI benchmarks.

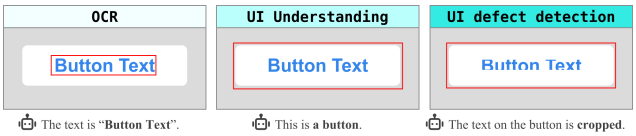


Figure 1. Differences among OCR, UI understanding, and UI display defect detection tasks: OCR focuses on text recognition; UI understanding specializes in identifying and locating UI components; UI display defect detection requires perceiving and assessing display defects.

The research gap is further exacerbated by the scarcity of realistic, challenging datasets. Existing evaluations of mobile UI display defect detection predominantly rely on data from older open-source datasets (*e.g.*, RICO [9]) [25, 26], which tend to exhibit relatively simple interaction logic and do not adequately capture the diverse styling display defects prevalent in contemporary commercial-grade applications. Consequently, the actual capabilities of foundation models for detecting real-world UI displays remain largely unknown.

†Corresponding author.

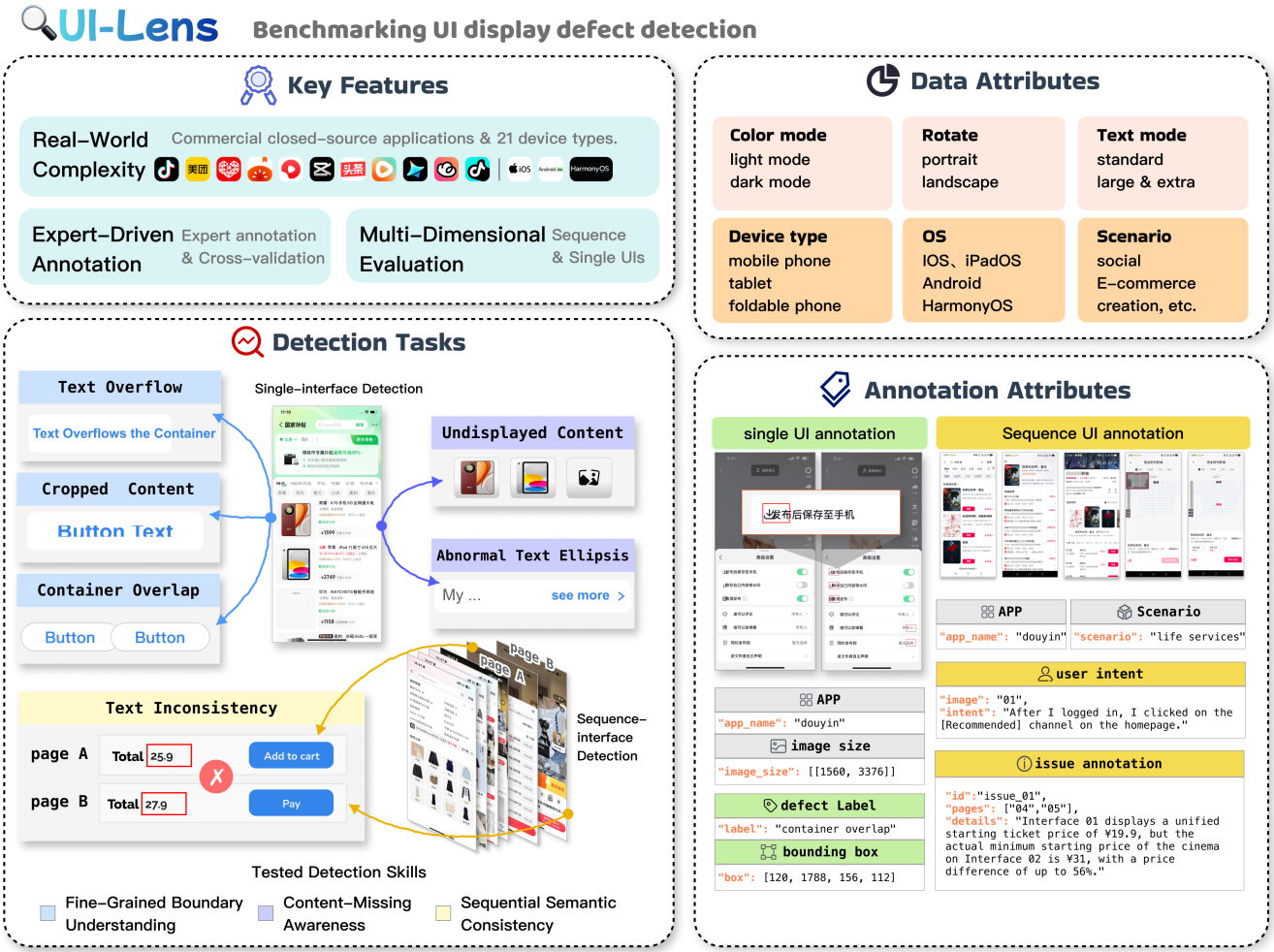


Figure 2. **Overview of UI-Lens.** The benchmark covers 11 commercial applications across 21 device types with diverse attributes (color mode, rotation, text mode, OS, scenario). It includes six defect detection tasks with expert annotations providing multi-dimensional labels for single UI and sequence UI detection.

As shown in Figure 2, to fill this gap, we introduce and release **UI-Lens**, a multi-faceted benchmark for evaluating mobile UI display defect detection in commercial applications under real-world, complex conditions. We collected real-world UI screens from 11 leading commercial applications across 21 mainstream devices, covering critical modes such as large-font and dark mode and including categories like social and e-commerce. Because genuine display defect cases are scarce, we collaborated with eight UI/UX design experts to synthesize an additional 1,030 positive examples that closely mimic real display defects (for the Chinese version of the test set). In total, the Chinese version of the benchmark comprises 5,356 systematically annotated instances spanning six key display defect subtypes: *Text Overflow*, *Cropped Content*, *Container Overlap*, *Undisplayed Content*, *Abnormal Text Ellipsis*, and *Text Inconsistency*, comprehensively encompassing typical display defect scenarios caused by text, containers, and semantic content.

Our comprehensive evaluation of 9 mainstream MLLMs and VLMs (including 7 closed-source and 2 open-source [36, 43]) on UI-Lens reveals significant, previously undocumented weaknesses. As shown in Figure 3 (a), overall accuracy on single-interface defect detection tasks is poor, and performance plummets on fine-grained boundary-related single-interface defect: average accuracy for Text Overflow (50.04%), Cropped Content (54.07%), and Container Overlap (54.14%) is near random guessing. This result exposes a critical failure in current SOTA models' fine-grained perceptual abilities on UI display defects. Figure Fig. 3 (b) reveals significant differences in F1 Score across models (as the harmonic mean of precision and recall, F1 Score is particularly sensitive to overly low values). Notably, *Seed1.6-Vison* [35] and *Gemini-2.5-Pro* [8] demonstrate superior stability, achieving consistently high F1 scores across all five

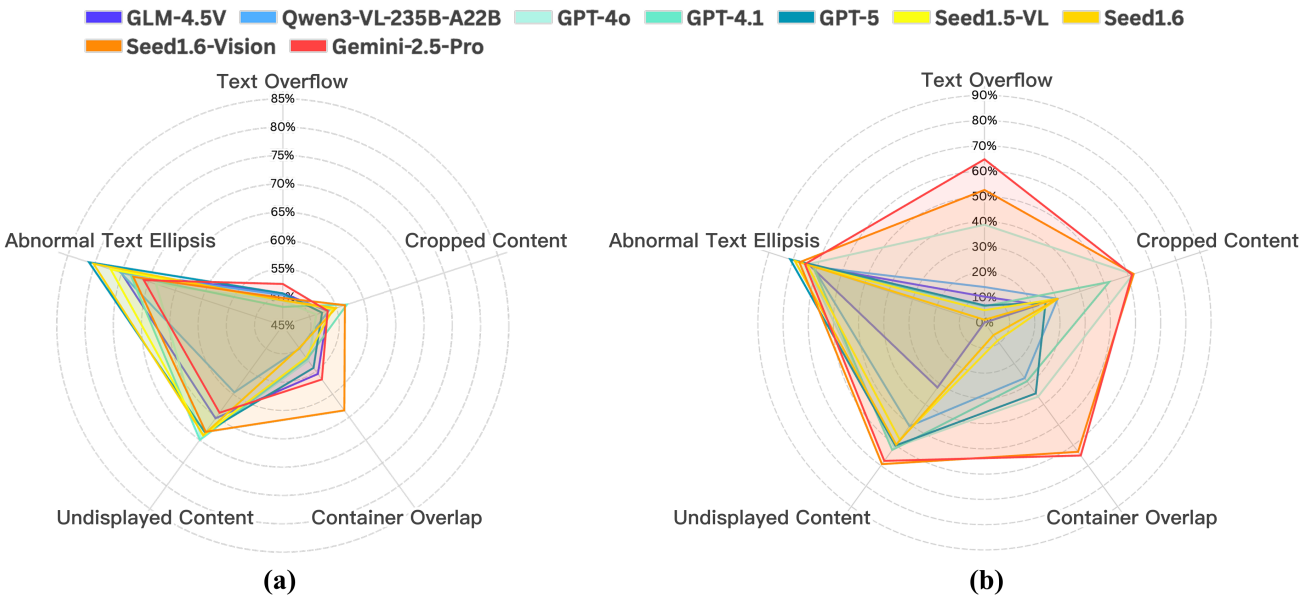


Figure 3. **Overview of single-interface defect detection performance.** (a) Accuracy; (b) F1 scores.

single-interface defect types.

Our main contributions are as follows:

- We construct UI-Lens, a large-scale, high-quality dataset comprising 4,759 Chinese pages. It is sourced from 11 latest commercial applications and 21 devices, ensuring data timeliness, diversity, and real-world complexity.
- We formulate challenging tasks for UI display defect detection, systematically including six key subtypes that rigorously evaluate models' capabilities in fine-grained boundary understanding, content-missing awareness, and semantic consistency judgment.
- Through extensive experiments on 9 models, we provide the first systematic analysis of their capabilities in UI display defect detection, revealing their critical failures in fine-grained boundary understanding and identifying clear directions for future optimization.

To promote the development of robust and reliable UI display defect detection techniques, we publicly release UI-Lens at this link. Additionally, we will open-source an English version consisting of 3,392 English interfaces.

2. UI-Lens

In this section, we present UI-Lens. We first describe the data collection process in section 2.1, followed by the data annotation process in section 2.2. We then provide an overview of the sample distribution of UI-Lens in section 2.3.

2.1. Data Collection

The data collection follows the principles of diversity and authenticity. Appendix A contains detailed metadata, including the model of the collection device, application names, modes, and examples of corresponding interface names. We first manually collected user interfaces from 11 leading commercial applications across domains like social networking, e-commerce, and authoring tools. The data spans 21 distinct devices (from smartphones to tablets) and four operating systems (including iOS and HarmonyOS). Crucially, we explicitly included challenging scenarios such as large-font mode and dark mode to evaluate the robustness of the model.

We observed that UI display defects are naturally scarce in these high-quality commercial applications. This scarcity makes it challenging to build a large-scale evaluation set purely from real samples and confirms the difficulty of the task. To address this scarcity while maintaining authenticity, we collaborated with 8 senior UI/UX design experts (with an average of 7.8 years of experience). Based on the collected real UI interfaces, these experts generated an additional 1,030 positive sample images for the Chinese dataset. These synthesized data were generated via professional image manipulation in Figma (similar to Photoshop), and the detailed pipeline are illustrated in Appendix A. Within the Chinese dataset, quantitative evaluation on 35,308 predictions confirms the consistency between synthetic and real-world datasets: the average accuracy is comparable (62.82% vs. 61.81%), further corroborated by a high Pearson correlation [3] (r=0.88) across defect categories. This demonstrates that these synthesized examples closely mimic real-world display defect scenarios. Together, these data provide high-value, challenging cases that are difficult to capture at scale.

2.2. Data Categorization and Annotation

UI display defect types. We define a fine-grained hierarchy of UI display defects, organizing 6 key subtypes into three major categories. Visual examples for each category are presented in Figure 4:

- **Element-boundary defect** (see Figure 4 (a)): covering Text Overflow, Cropped Content, and Container Overlap. The display defect regions in such tasks are typically subtle, and determination requires analyzing spatial relationships between multiple element boundaries.
- **Content-missing defect** (see Figure 4 (b)): covering Undisplayed Content and Abnormal Text Ellipsis. These tasks aim to detect cases where information presentation in the UI is incomplete, posing challenges to models in capturing missing content and determining information integrity.
- **Semantic-inconsistency defect** (see Figure 4 (c)): covering Text Inconsistency. This defect usually occurs in sequence-interface scenarios, requiring context understanding from multiple pages, such as verifying consistent pricing during a purchase flow. For this subtype, we construct a dedicated answer set, providing structured annotations for each instance of Text Inconsistency, including the specific interface where the display defects occurs, the violated specification details, and a precise problem description. This answer set serves as a unified reference for the subsequent evaluation stage.

Annotation process. To ensure annotation quality, the entire process was managed by the 8 UI/UX design experts. We first jointly formulated detailed annotation guidelines for all 6 display defect types (see Appendix B for details of the guidelines). The process then involved a rigorous two-stage process: (1) Initial Annotation: Experts identified and marked display defect regions using bounding boxes; (2) Cross-Validation and Arbitration: A different expert reviewed the annotations. Any disagreements were discussed and resolved by the expert panel to ensure high consistency and accuracy. To measure inter-annotator reliability, we computed Krippendorff's alpha [17] on a subset of 1,000 instances annotated by multiple experts, resulting in a score of 0.8417 (which exceeds the widely accepted threshold of 0.80, demonstrating robust reliability). This expert-driven, high-quality annotation process is a key differentiator of our benchmark and is essential for evaluating fine-grained tasks where non-expert annotation would be unreliable.

2.3. Benchmark Statistics and Analysis

To visually present the scale and distribution characteristics of UI-Lens, this section conducts a statistical analysis from three perspectives: overall data scale, distribution of display defect points, and application coverage distribution.

Overall data scale. The Chinese version of the UI-Lens dataset contains 4,759 high-quality UI images, within which 5,356 display defect instances are annotated. This scale provides significant coverage in the field of mobile UI display defect detection. While dataset for general UI understanding (*e.g.*, WebUI [41]) prioritize massive scale via automated collection, our contribution is a high-density, expert-annotated defect benchmark. It is deliberately curated to test the limits of SOTA models on difficult, fine-grained tasks, rather than to serve as a large-scale training corpus. The scale reflects the high cost of expert annotation and the inherent scarcity of real-world display defects.

UI-Lens distribution. The sample distribution of the 6 UI display defect types is shown in Table 1. These statistics are specific to the Chinese subset of the benchmark.

- **Element boundary and content missing display defects**: In total, there are 4,156 images, of which 2,078 are positive samples, containing 4,621 display defect instances in total, with an average of 2.22 display defects per image. We maintain an approximate 1:1 positive-to-negative ratio to ensure fairness of evaluation across display defect types. For rare but high-value display defect cases in real-world scenarios, additional synthetic samples are incorporated to maintain balance in type distribution and completeness in task coverage. This balanced design is crucial for reliable evaluation, ensuring that a 50% accuracy score genuinely represents random-guess performance.
- **Semantic-inconsistency display defect**: We collected 89 sequences, each with an average of 6.8 interfaces and 8.1 annotated display defect instances, reaching a total of 735 display defect instances. It is worth noting that a display defect instance appears across several interfaces, reaching a compatible instance number with other display defects. The *Text Inconsistency* task often occurs in sequence-interface UIs, collection and annotation are performed based on continuous UI interface sequences, ensuring that display defect instances originate from genuine content inconsistency within real interaction flows.

Application coverage distribution. The Chinese subset covers 11 leading applications and we prioritizes comprehensive scenario coverage. We systematically cover five major UI scenarios: (1) shooting and creation (*e.g.*, CapCut and Jimeng); (2) e-commerce (*e.g.*, PDD); (3) life services (*e.g.*, Meituan); (4) social short videos (*e.g.*, Douyin; and (5) text-image reading (*e.g.*, Hongguo and fanqie). This scenario-centric design ensures that our benchmark captures the distinct UI display defect patterns characteristic of different functional contexts. By sampling multiple applications within each scenario, we validate the generalizability of defect patterns across implementations while maintaining scenario-specific depth through Douyin's multi-functional architecture.



Figure 4. **UI display defect types.** (a) Element-boundary: Text Overflow, Cropped Content, Container Overlap; (b) Content-missing: Undisplayed Content, Abnormal Text Ellipsis; (c) Semantic-inconsistency: Text Inconsistency.

Display Defect Types	Statistics			Tasks	
	Samples	Pos. Samp	Instances	Bbox	Img seqs
Single-interface defects					
Text Overflow	716	358	835	✓	✗
Cropped Content	790	395	831	✓	✗
Container Overlap	872	436	921	✓	✗
Undisplayed Content	850	425	878	✓	✗
Abnormal Text Ellipsis	928	464	1,156	✓	✗
Sequence-interface defects					
Text Inconsistency	603	N/A	735	✗	✓
Total	**4,759**	**2,078**	**5,356**	N/A	N/A

Table 1. **Overview of UI display defect types and their annotations.** Display defects are categorized into single-interface display defects (detected within individual screens) and sequence-interface display defects (requiring multiple screens). Statistics include the number of samples, positive samples (Pos. Samp), and instances. Task annotations indicate whether bounding box (Bbox) and image sequences (Img seqs) are available for each display defect type. The total row summarizes the overall statistics.

3. Experiment

This section introduces evaluation of 9 multimodal models on UI-Lens. Section 3.1 introduces the experimental setup. Section 3.2 presents the quantitative results.

3.1. Experiment Setup

Evaluation protocol. To ensure a fair comparison and assess general ability, all evaluations are conducted using a zero-shot CoT [40] approach. The prompt engineering adaptation for all models is kept consistent across display defect detection sub-tasks. For each model, we perform two runs under each setting and report the average score. Detailed prompts are provided in Appendix C.

Evaluated models. We systematically evaluate 9 mainstream multimodal models, covering both closed-source and open-source models. The closed-source models include the Seed series (*Seed1.6* [35], *Seed1.6-Vison* [35], *Seed1.5-VL* [12]), the GPT series (*GPT-5* [30], *GPT-4.1* [29], *GPT-4o* [16]), and the Gemini series (*Gemini-2.5-Pro* [8]); the open-source models include *Qwen3-VL-235B-A22B* [43] and the *GLM-4.5V* [36]. Detailed configurations, inference parameters, and additional experimental information for each model are provided in Appendix D.

Evaluation tasks. We design two evaluation tasks tailored to the characteristics of the display defect types.

1. **Single-interface display defect detection:** We formulate this as a binary classification task for each defect type. As shown in Figure 5 (a), this task focuses on UI display defect detection within individual UI interfaces, encompassing all display defect types except Text Inconsistency. Given our dataset's balanced 1:1 positive-to-negative ratio (as described in Sec 3.3), Accuracy (Eq. 1) serves as a primary metric, where 50% represents random-guess performance. We also report Precision (Eq. 2), Recall (Eq. 3), and the F1 Score (Eq. 4). As both false positives and false negatives are critical in UI testing, the F1 score provides a comprehensive measure of a model's overall performance.
2. **Sequence-interface display defect detection:** This task focuses on Text Inconsistency display defects, which requires reasoning across multiple UI pages. Formulated as an open-ended recall task, we introduce a judge model to evaluate the consistency between model outputs and our ground truth answer set. The evaluation workflow is illustrated in Figure 5 (b): we first extract the standard answer from the answer set, then extract the corresponding output from the model output set, sequentially perform pages intersection judgment and issue details consistency verification. For this task, we also rely on precision, re-

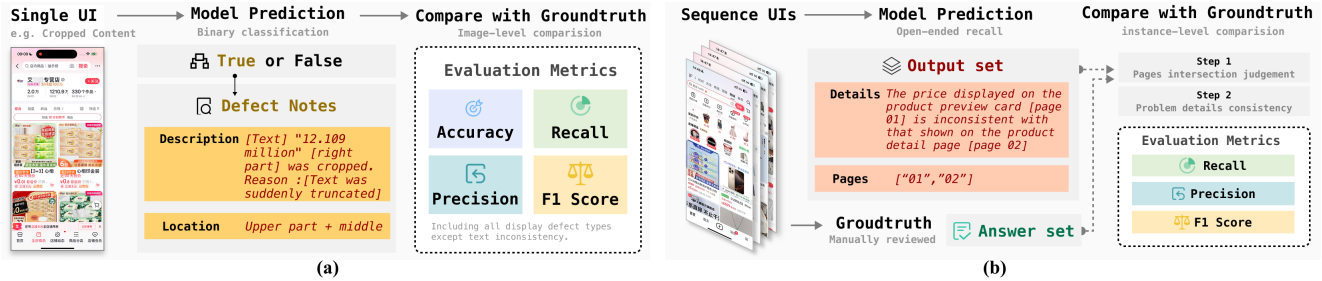


Figure 5. **Evaluation workflow.** (a) Single-interface display defect detection; (b) Sequence-interface display defect detection.

call, and F1 score. The Accuracy metric is not applicable here, as an open-ended task lacks a clearly defined set of "true negatives" (*i.e.*, all possible correct statements a model didn't make). Our evaluation focuses on whether the model recalls the ground-truth answers (Recall) and whether the answers it does provide are correct (Precision).

To formalize the evaluation for both tasks, the core metrics are calculated based on the counts of true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN) as follows:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{1}$$

$$\text{Precision} = \frac{TP}{TP + FP} \tag{2}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{3}$$

$$\text{F1 Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{4}$$

Detailed computation rules are in Appendix E.

3.2. Results

Figure 6 reports task-average accuracy, precision, recall, and F1 score on UI-Lens. Table 2 presents per-task accuracy and F1 score for each model, summarizing model-average accuracy and F1 score.

Finding 1: Existing models perform poorly on UI display defect detection tasks, with fine-grained tasks dropping to random-guess performance. Task-average accuracy ranging from 50%-75%, with nearly all models approaching the random-guessing baseline of 50% (due to our 1:1 balanced dataset). Even the best overall model, such as *Seed1.6-Vison* [35] and *Gemini-2.5-Pro* [8], only achieves approximately 64%-66% model-average F1 score. As shown in Table 2, the average accuracy on fine-grained boundary tasks plummets to the level of a coin flip: 50.04% for Text Overflow, 54.07% for Cropped Content, and 54.14% for Container Overlap. Taken together, these quantitative results indicate that current MLLMs and VLMs lack the specialized visual reasoning capabilities required to support robust UI quality assurance in real-world, complex interfaces.

Finding 2: Models exhibit a pronounced precision–recall trade-off. As shown in figure 6 (a), models adopt polarized strategies. Aggressive models, such as *Gemini-2.5-Pro* [8], pursue high recall (81.41% model-average recall, ranked first) for comprehensive coverage but suffer from low precision (56.85% model-average precision, ranked last), requiring substantial human review resources. Conservative models, such as *Seed1.5-VL* [12], prefer avoiding false positives even at the cost of missed detections; for example, *Seed1.5-VL* [12] achieves 75.7% model-average precision (ranked first) but only 27.07% model-average recall, rendering it unusable in scenarios requiring high coverage. However, as shown in Figure 6 (b), due to the open-ended recall nature of the sequence-interface semantic inconsistency task, models generally demonstrate higher precision but lower recall. Moreover, all three metrics perform poorly on Text Inconsistency, reflecting the inherent difficulty in comprehensively identifying all semantic inconsistencies across multiple interface states.

Finding 3: Fine-grained boundary understanding and sequence-level interface semantic consistency auditing pose severe challenges, while content-absence tasks are relatively manageable. On display defect detection tasks involving boundary understanding, average F1 scores are uniformly low—Text Overflow at 22.19%, Container Overlap at 33.75%, and Cropped Content at 42.05%—approaching random discrimination. Sequence-level semantic consistency auditing is even more demanding: for Text Inconsistency, the average F1 across models is only 11.44%, highlighting weak cross-interface reasoning. In contrast, models perform relatively better on tasks involving content absence, with task-average F1 scores of 57.78% for Undisplayed Content and 75.24% for Abnormal Text Ellipsis.

Finding 4: Advanced prompting strategies yield marginal gains, highlighting a fundamental lack of UI cognitive models. We evaluated diverse prompting paradigms (self-correction [38], FGVP [45], and one-shot CoT [40]) on SOTA models (Table 3). While one-shot prompting outperformed self-reasoning methods by providing explicit do-

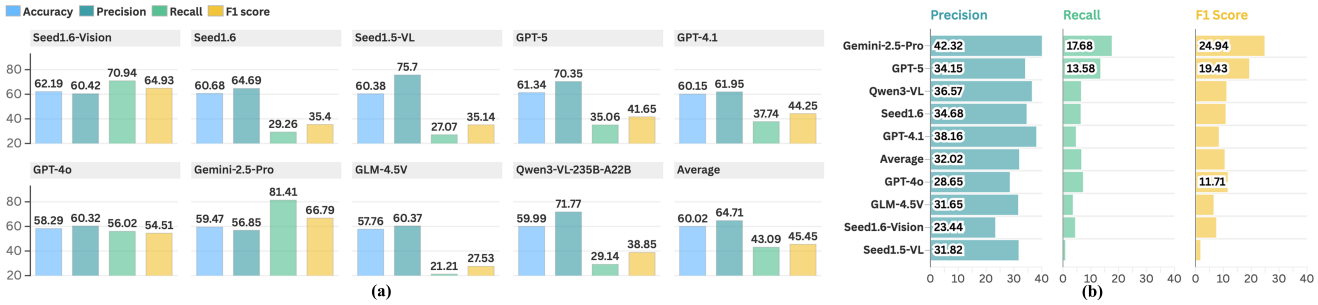


Figure 6. **Evaluation results of accuracy, precision, recall, and F1 score.** (a) Results for single-interface tasks; (b) Results for sequence-interface tasks.

Model	Accuracy					F1 Scores					
	TO	CC	CO	UC	ATE	TO	CC	CO	UC	ATE	TI
Seed1.5-VL [12]	49.42	54.39	52.03	69.04	77.02	4.91	30.62	8.83	57.28	74.06	1.92
Seed1.6-Vison [35]	49.85	56.56	63.53	68.21	72.82	52.54	62.31	63.23	69.26	77.30	7.59
Seed1.6 [35]	49.42	54.90	49.94	68.92	80.22	1.14	30.86	6.10	59.69	79.21	10.96
GPT-5 [30]	50.67	52.31	54.26	68.40	81.07	6.80	25.51	34.65	60.18	81.13	19.43
GPT-4.1 [29]	48.28	56.92	52.34	69.96	73.24	6.50	52.12	28.72	62.33	71.57	8.54
GPT-4o [16]	49.33	51.25	52.70	69.60	68.59	38.77	62.01	36.24	61.11	74.42	11.71
***Gemini-2.5-Pro* [8]**	52.32	53.36	56.78	64.05	70.86	64.67	61.60	65.05	67.68	74.97	24.94
GLM-4.5V [36]	50.37	53.62	50.06	59.59	75.17	10.30	22.81	N/A	31.77	72.80	6.64
Qwen3-VL-235B-A22B [43]	50.67	53.36	55.58	65.26	75.06	14.10	30.58	27.17	50.68	71.72	11.25
ALL	50.04	54.07	54.14	67.00	74.89	22.19	42.05	33.75	57.78	75.24	11.44

Table 2. **Model performance evaluation.** The table reports accuracy and F1 scores for 9 models across six evaluation tasks (TO: Text Overflow; CC: Cropped Content; CO: Container Overlap; UC: Undisplayed Content; ATE: Abnormal Text Ellipsis; TI: Text Inconsistency). Cell shading encodes each model's relative rank within a task: purple denotes the top four (darker indicates a better rank), and yellow denotes the bottom four (darker indicates a worse rank). The ALL row presents the average performance per task across all models.

main context, overall performance improvements remained marginal (+0.60 to +2.48 in F1). Consistent with prior research [6], purely prompt-based approaches inherently struggle to resolve subtle visual nuances. Consequently, current MLLMs lack a cognitive model of common UI conventions and reason only about surface-level visual cues, exhibiting weak "atomic defect detection" capabilities that prompt engineering alone cannot overcome.

Model	0 shot	Self correction	FGVP	One shot
Gemini-2.5-Pro	66.79	66.89 (+0.10)	67.08 (+0.29)	67.39 (+0.60)
Seed1.6-Vison	64.93	63.94 (-0.99)	66.21 (+1.28)	67.41 (+2.48)

Table 3. Performance F1 scores for SOTA models across four prompting strategies.

4. Error Analysis

This section analyzes errors in UI display defect detection. For single-interface detection error analysis in subsection 4.1 , we concentrate on the best-performing model, *Gemini-2.5-Pro* [8], by classifying its errors and attributing their causes. For sequential-interface detection error analysis in subsection 4.2, we examine the features of recalled and non-recalled samples to pinpoint the underlying constraints of the current detection paradigm.

4.1. Single-Interface Detection Error Analysis

We conducted a statistical analysis of errors produced by Gemini-2.5-Pro in single-interface detection tasks, identifying six principal error types (see Figure 7). These error types reveal critical shortcomings in the model's perceptual and reasoning capabilities when applied to UI defect detection.

Based on the error analysis above, we identify the following key capability deficiencies in current MLLMs:

Bottleneck in fine-grained perception. The prevalence of "Minor Defect" errors—often involving sub-pixel text truncation—reveals the model's limited sensitivity to subtle visual anomalies. This suggests that current vision encoders, possibly optimized for object-centric tasks, lack the boundary-level precision to detect fine layout deformations essential in UIs.

Failure in layout comprehension. "Hallucination" errors indicate a failure in scene understanding. Instead of

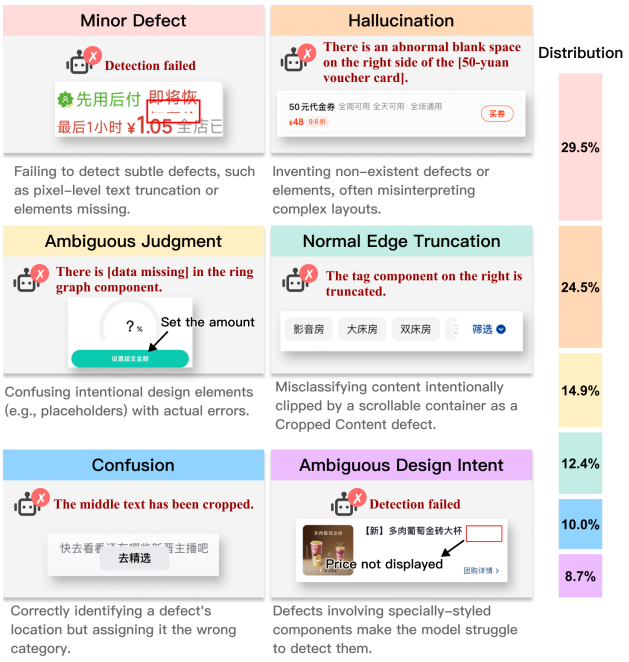


Figure 7. **Case examples and distribution of error types identified in single-interface detection tasks.** Among them, *Minor defect* accounts for the highest proportion (29.5%), primarily manifesting as failures in detecting pixel-level text truncation or missing elements.

faithfully parsing the visual evidence and recognizing standard UI conventions, the model invents non-existent defects (e.g., misinterpreting intentional spacing on a voucher card as an abnormal blank space). This reveals an inability to properly contextualize complex layouts, leading the model to misrepresent a normal structural design as anomalous.

Inability to infer design intent. "Ambiguous Judgment" and "Ambiguous Design Intent" errors show models struggle to distinguish deliberate designs from actual defects. For example, they may falsely flag an intentional placeholder (*e.g.*, a "?" in a ring graph) as missing data, or conversely, fail to detect genuinely missing information within specially-styled components.

Mismatch between static analysis and interactive context. "Normal Edge Truncation" errors reveal that models apply static reasoning to dynamic UIs. Lacking an understanding of interactive affordances (*i.e.*, "is-scrollable"), they incorrectly classify naturally clipped content in scrollable containers (*e.g.*, a carousel) as Cropped Content defects.

4.2. Sequential Interface Detection Error Analysis

Sequential detection requires models to track semantic relationships and information flow across multiple steps, a significant leap from single-screen analysis. Our analysis reveals that current models lack robust capabilities for this cross-page contextual reasoning.

Dominance of shallow reasoning. We find that models perform best on localized inconsistencies. 85% of the top-20 recalled samples involve simple numeric mismatches—such as price discrepancies. This strongly suggests that models default to literal, token-level comparisons and struggle with inconsistencies that require deeper semantic integration or contextual inference across interface states.

Absence of path-level state tracking. Failures often occur across user journeys (*e.g.*, a promotional banner vanishing during checkout). While models can compare static elements, they lack the dynamic memory required for path-level consistency, exposing a deep deficit in procedural and interactive reasoning.

5. Related Works

In recent years, multimodal large language models (MLLMs) and vision–language models (VLMs) have achieved significant advances in user interface understanding [19, 20, 39, 47]. Studies show that even general-purpose foundation models (*e.g.* Gemini-2.5-Pro) demonstrate strong visual–structural parsing and semantic reasoning capabilities: they can accurately recognize UI components, interpret textual content, and infer interface functionality [8], thereby laying the groundwork for more complex UI tasks.

Meanwhile, research interest in leveraging multimodal visual reasoning for UI quality assurance has been steadily increasing [7, 15, 23]. Early systems such as Owl Eyes [25] and Nighthawk [26], built on visual understanding methods, can localize display defects in mobile applications, including element misalignment and occlusion. Recent works related to UI quality assurance further include: (1) end-to-end model-based approaches that directly detect UI defects, exemplified by AutoConsis [14], which combines multimodal models with large language models (LLMs) to detect data inconsistencies across application interfaces; (2) using MLLMs as judges to generate UI feedback aligned with human aesthetic and user-experience (UX) ratings [28, 32, 37, 42]. On the benchmarking side, WebRSSBench [22] evaluates MLLMs' color robustness and safety-critical detection capabilities on web interfaces, while GUI Testing Arena [50] assesses the ability of Graphical UI (GUI) agents to discover UI defects during execution.

6. Conclusion

We introduce UI-Lens, a benchmark for UI display defect detection in commercial apps, comprising 4,759 expert-labeled pages across 6 defect types. Systematic evaluation of mainstream models reveals critical gaps in fine-grained boundary perception and cross-interface semantic reasoning, guiding future work toward advanced detection methods.

Acknowledgments

This work was supported by the National Natural Science Foundation of China under Grant No. 62407038.

References

[1] Asif Ali, Yuanqing Xia, Qamar Navid, Zohaib Ahmad Khan, Javed Ali Khan, Eman Abdullah Aldakheel, and Doaa Khafaga. Mobile-UI-repair: a deep learning based UI smell detection technique for mobile user interface. *PeerJ Computer Science*, 10:e2028, 2024. 1

[2] Chongyang Bai, Xiaoxue Zang, Ying Xu, Srinivas Sunkara, Abhinav Rastogi, Jindong Chen, et al. UIBert: Learning generic multimodal representations for UI understanding, 2021. 1

[3] Jacob Benesty, Jingdong Chen, Yiteng Huang, and Israel Cohen. Pearson correlation coefficient. In *Noise reduction in speech processing*, pages 1–4. Springer, 2009. 3

[4] Chun-Fu Chen, Marco Pistoia, Conglei Shi, Paolo Girolami, Joseph W Ligman, and Yong Wang. UI X-Ray: Interactive mobile UI testing based on computer vision. In *Proceedings of the 22nd International Conference on Intelligent User Interfaces*, pages 245–255, 2017. 1

[5] Song Chen, Xinyu Guo, Yadong Li, Tao Zhang, Mingan Lin, Dongdong Kuang, Youwei Zhang, Lingfeng Ming, Fengyu Zhang, Yuran Wang, et al. Ocean-OCR: Towards general ocr application via a vision-language model. *arXiv preprint arXiv:2501.15558*, 2025. 1

[6] Ayoung Choi and Jongwook Jeong. Can LLMs Detect Display Issues? Uncovering the Impact of Prompting Techniques. In *Adjunct Proceedings of the 38th Annual ACM Symposium on User Interface Software and Technology*, pages 1–3, 2025. 1, 7

[7] Ayoung Choi and Jongwook Jeong. Can LLMs Detect Display Issues? Uncovering the Impact of Prompting Techniques. In *Adjunct Proceedings of the 38th Annual ACM Symposium on User Interface Software and Technology*, pages 1–3, 2025. 8

[8] Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, et al. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*, 2025. 2, 5, 6, 7, 8

[9] Biplab Deka, Zifeng Huang, Chad Franzen, Joshua Hibschman, Daniel Afergan, Yang Li, Jeffrey Nichols, and Ranjitha Kumar. Rico: A mobile app dataset for building data-driven design applications. In *Proceedings of the 30th annual ACM symposium on user interface software and technology*, pages 845–854, 2017. 1

[10] Xiaoyu Deng, Zhengjian Kang, Xintao Li, Yongzhe Zhang, and Tianmin Guo. Covis: A collaborative framework for fine-grained graphic visual understanding. *arXiv preprint arXiv:2411.18764*, 2024. 1

[11] Yihao Ding, Siwen Luo, Yue Dai, Yanbei Jiang, Zechuan Li, Geoffrey Martin, and Yifan Peng. A Survey on MLLM-based Visually Rich Document Understanding: Methods, Challenges, and Emerging Trends. *arXiv preprint arXiv:2507.09861*, 2025. 1

[12] Dong Guo, Faming Wu, Feida Zhu, Fuxing Leng, Guang Shi, Haobin Chen, Haoqi Fan, Jian Wang, Jianyu Jiang, Jiawei Wang, et al. Seed1.5-vl technical report, 2025. 5, 6, 7

[13] Dennis Guse, Sebastian Schuck, Oliver Hohlfeld, Alexander Raake, and Sebastian Möller. Subjective quality of webpage loading: The impact of delayed and missing elements on quality ratings and task completion time. In *2015 Seventh international workshop on quality of multimedia experience (QoMEX)*, pages 1–6. IEEE, 2015. 1

[14] Yongxiang Hu, Hailiang Jin, Xuan Wang, Jiazhen Gu, Shiyu Guo, Chaoyi Chen, Xin Wang, and Yangfan Zhou. AutoConsis: Automatic GUI-driven Data Inconsistency Detection of Mobile Apps. In *Proceedings of the 46th International Conference on Software Engineering: Software Engineering in Practice*, page 137–146, New York, NY, USA, 2024. Association for Computing Machinery. 1, 8

[15] Yongxiang Hu, Xuan Wang, Yingchuan Wang, Yu Zhang, Shiyu Guo, Chaoyi Chen, Xin Wang, and Yangfan Zhou. Auitestagent: Automatic requirements oriented gui function testing. *arXiv preprint arXiv:2407.09018*, 2024. 8

[16] Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*, 2024. 5, 7

[17] Klaus Krippendorff. Computing krippendorff's alpha-reliability. 2011. 4

[18] Newton Lee, editor. *UI, User Interface*, pages 1923–1923. Springer International Publishing, Cham, 2024. 1

[19] Hongxin Li, Jingfan Chen, Jingran Su, Yuntao Chen, Li Qing, and Zhaoxiang Zhang. AutoGUI: Scaling GUI grounding with automatic functionality annotations from LLMs. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10323–10358, Vienna, Austria, 2025. Association for Computational Linguistics. 8

[20] Zhangheng Li, Keen You, Haotian Zhang, Di Feng, Harsh Agrawal, Xiujun Li, Mohana Prasad Sathya Moorthy, Jeff Nichols, Yinfei Yang, and Zhe Gan. Ferret-UI 2: Mastering universal user interface understanding across platforms. *arXiv preprint arXiv:2410.18967*, 2024. 8

[21] Junpeng Liu, Tianyue Ou, Yifan Song, Yuxiao Qu, Wai Lam, Chenyan Xiong, Wenhu Chen, Graham Neubig, and Xiang Yue. Harnessing webpage UIs for text-rich visual understanding. *arXiv preprint arXiv:2410.13824*, 2024. 1

[22] Junliang Liu, Jingyu Xiao, Wenxin Tang, Wenxuan Wang, Zhixian Wang, Minrui Zhang, and Shuanghe Yu. Benchmarking mllm-based web understanding: Reasoning, robustness and safety. *arXiv preprint arXiv:2509.21782*, 2025. UI feedback. 8

[23] Wei Liu, Feng Lin, Linqiang Guo, Tse-Hsun Peter Chen, and Ahmed E Hassan. Guiwatcher: Automatically detecting gui lags by analyzing mobile application screencasts. In *2025 IEEE/ACM 47th International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*, pages 46–56. IEEE, 2025. 8

[24] Zhe Liu. Discovering UI display issues with visual understanding. In *Proceedings of the 35th IEEE/ACM International Conference on Automated Software Engineering*, page 1373–1375, New York, NY, USA, 2021. Association for Computing Machinery. 1

[25] Zhe Liu, Chunyang Chen, Junjie Wang, Yuekai Huang, Jun Hu, and Qing Wang. Owl eyes: spotting UI display issues via visual understanding. In *Proceedings of the 35th IEEE/ACM International Conference on Automated Software Engineering*, page 398–409, New York, NY, USA, 2021. Association for Computing Machinery. 1, 8

[26] Zhe Liu, Chunyang Chen, Junjie Wang, Yuekai Huang, Jun Hu, and Qing Wang. Nighthawk: Fully automated localizing UI display issues via visual understanding. *IEEE Transactions on Software Engineering*, 49(1):403–418, 2022. 1, 8

[27] Zhijie Liu, Yutian Tang, Meiyun Li, Xin Jin, Yunfei Long, Liang Feng Zhang, and Xiapu Luo. LLM-compdroid: Repairing configuration compatibility bugs in android apps with pre-trained large language models. *ACM Transactions on Software Engineering and Methodology*, 2024. 1

[28] Reuben A Luera, Ryan Rossi, Franck Dernoncourt, Samyadeep Basu, Sungchul Kim, Subhojyoti Mukherjee, Puneet Mathur, Ruiyi Zhang, Jihyung Kil, Nedim Lipka, et al. MLLM as a UI judge: Benchmarking multimodal LLMs for predicting human perception of user interfaces. *arXiv preprint arXiv:2510.08783*, 2025. 8

[29] OpenAI. Introducing GPT-4.1 in the API, 2025. 5, 7

[30] OpenAI. GPT-5 system card. Technical report, n.d. 5, 7

[31] Šarūnas Packevičius, Andrej Ušaniov, Šarūnas Stanskis, and Eduardas Bareiša. The testing method based on image analysis for automated detection of UI defects intended for mobile applications. In *International Conference on Information and Software Technologies*, pages 560–576. Springer, 2015. 1

[32] Seokhyeon Park, Yumin Song, Soohyun Lee, Jaeyoung Kim, and Jinwook Seo. Leveraging multimodal llm for inspirational user interface search. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems*, pages 1–22, 2025. 8

[33] Lumpapun Punchoojit and Nuttanont Hongwarittorrn. Usability studies on mobile user interface design patterns: a systematic literature review. *Advances in Human-Computer Interaction*, 2017(1):6787504, 2017. 1

[34] Antu Saha, Yang Song, Junayed Mahmud, Ying Zhou, Kevin Moran, and Oscar Chaparro. Toward the automated localization of buggy mobile app uis from bug descriptions. In *Proceedings of the 33rd ACM SIGSOFT International Symposium on Software Testing and Analysis*, page 1249–1261, New York, NY, USA, 2024. Association for Computing Machinery. 1

[35] ByteDance Seed Team. Introduction to techniques used in seed1.6, 2025. 2, 5, 6, 7

[36] V Team, Wenyi Hong, Wenmeng Yu, Xiaotao Gu, Guo Wang, Guobing Gan, Haomiao Tang, Jiale Cheng, Ji Qi, Junhui Ji, et al. Glm-4.5v and glm-4.1v-thinking: Towards versatile multimodal reasoning with scalable reinforcement learning, 2025. 2, 5, 7

[37] Yuekai Wang and Franceska Xhakaj. AIHeurEval: Generating Heuristic Evaluations on Multiple UI Screens with Multimodal Large Language Models. In *International Conference on Human-Computer Interaction*, pages 228–237. Springer, 2025. 8

[38] Yifei Wang, Yuyang Wu, Zeming Wei, Stefanie Jegelka, and Yisen Wang. A theoretical understanding of self-correction through in-context alignment. *Advances in Neural Information Processing Systems*, 37:89869–89912, 2024. 6

[39] Ziwei Wang, Weizhi Chen, Leyang Yang, Sheng Zhou, Shengchu Zhao, Hanbei Zhan, Jiongchao Jin, Liangcheng Li, Zirui Shao, and Jiajun Bu. MP-GUI: Modality Perception with MLLMs for GUI Understanding. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 29711–29721, 2025. 8

[40] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022. 5, 6

[41] Jason Wu, Siyan Wang, Siman Shen, Yi-Hao Peng, Jeffrey Nichols, and Jeffrey P Bigham. WebUI: A dataset for enhancing visual UI understanding with web semantics. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, pages 1–14, 2023. 1, 4

[42] Jason Wu, Yi-Hao Peng, Xin Yue Amanda Li, Amanda Swearngin, Jeffrey P Bigham, and Jeffrey Nichols. UIClip: a data-driven model for assessing user interface design. In *Proceedings of the 37th Annual ACM Symposium on User Interface Software and Technology*, pages 1–16, 2024. 8

[43] An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025. 2, 5, 7

[44] Hao Yang, Weijie Qiu, Ru Zhang, Zhou Fang, Ruichao Mao, Xiaoyu Lin, Maji Huang, Zhaosong Huang, Teng Guo, Shuoyang Liu, and Hai Rao. UI-UG: A unified MLLM for UI understanding and generation. 1

[45] Lingfeng Yang, Yueze Wang, Xiang Li, Xinlong Wang, and Jian Yang. Fine-grained visual prompting. *Advances in Neural Information Processing Systems*, 36:24993–25006, 2023. 6

[46] Shukang Yin, Chaoyou Fu, Sirui Zhao, Ke Li, Xing Sun, Tong Xu, and Enhong Chen. A survey on multimodal large language models. *National Science Review*, 11(12):nwae403, 2024. 1

[47] 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. In *European Conference on Computer Vision*, pages 240–255. Springer, 2024. 8

[48] Duzhen Zhang, Yahan Yu, Jiahua Dong, Chenxing Li, Dan Su, Chenhui Chu, and Dong Yu. MM-LLMs: Recent advances in multimodal large language models. *arXiv preprint arXiv:2401.13601*, 2024. 1

[49] Jingyi Zhang, Jiaxing Huang, Sheng Jin, and Shijian Lu. Vision-language models for vision tasks: A survey. *IEEE*

transactions on pattern analysis and machine intelligence, 46 (8):5625–5644, 2024. 1

[50] Kangjia Zhao, Jiahui Song, Leigang Sha, Haozhan Shen, Zhi Chen, Tiancheng Zhao, Xiubo Liang, and Jianwei Yin. GUI Testing Arena: A unified benchmark for advancing autonomous gui testing agent. *arXiv preprint arXiv:2412.18426*, 2024. 8

[51] Mingyuan Zhong, Ruolin Chen, Xia Chen, James Fogarty, and Jacob O. Wobbrock. ScreenAudit: Detecting screen reader accessibility errors in mobile apps using large language models. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems*, pages 1–19. Association for Computing Machinery. 1