Reasoning for Mobile User Experience with Multimodal LLMs: Task, Benchmark, and Approach

Supplementary Material

Ruichao Mao, Zhou Fang, Teng Guo, Hao Yang, Yaping Li, Shaohua Peng, Maji Huang, Xiaoyu Lin, Shuoyang Liu, Xuepeng Li, Yuyu Zhang, Hai Rao

Ant Group

{maoruichao.mrc, yingnan.fz, gt447200, charles.yh, pinky.lyp, shaohua.psh}@antgroup.com
{huangmaji.hmj, axiao.lxy, liushuoyang.lsy, healy.lxp, yuyu.zyy, raohai.rh}@antgroup.com

1. Overview

We detail the UXBench construction pipeline, including sample collection, preprocessing, and MLLM annotation prompts. Additional experiments on VisualWebBench and inference efficiency analysis on UXBench are provided to support the main paper.

2. Detailed Data Pipeline of UXBench

This section provides an in-depth description of the data construction process for UXBench, elaborating on the methodology outlined in Section 3.2 of the main paper to ensure transparency and reproducibility.

2.1. Data Collection and Preprocessing

The UXBench dataset is constructed from two primary sources: real-world user feedback for positive samples and systematic device-level collection for negative samples.

2.2. Data Collection and Preprocessing

The UXBench dataset comprises positive samples from real-world user feedback and negative samples from systematic device captures.

Positive samples. We collected 13M textual feedback entries (Jan-Sep 2025), of which 3M included screenshots. After filtering non-UI content and removing personally identifiable information (PII) containing images, we retained 22,354 unique UI screenshots exhibiting UX issues for annotation.

Negative samples. We systematically captured UI states across Android, iOS, and HarmonyOS platforms using automated scripts to navigate popular apps and websites, forming our negative sample pool of well-designed interfaces.

2.3. MLLM-Assisted Annotation Prompts

BubbleOcclT (Bubble Occludes Text). You are a senior UI/UX expert specializing in information hierarchy, visual occlusion, and readability. Determine whether a non-modal dismissible text overlay occludes textual content based solely on a single UI screenshot.

Definition. A non-modal dismissible text overlay is a non-blocking informational layer (e.g., tooltip, guide bubble, notification banner) requiring explicit dismissal via clicking "×" or "Close". While theoretically allowing interaction with underlying content, it may physically occlude page text and impair readability.

Analysis. (1) Identify whether the screenshot contains such an overlay with visible close control. (2) Locate static or dynamic textual content (titles, labels, descriptions) on the main interface. (3) Determine if the overlay's visual region overlaps with page text pixel-wise, causing partial or complete obstruction.

Options. A. No dismissible text overlay present. B. Overlay present but does not occlude text. C. Overlay present and occludes text.

Output. Provide analysis and return assessment as $\boxed{X}$ where $X \in \{A, B, C\}$.

BubbleOcclBtn (Bubble Occludes Button). Evaluate whether a non-modal dismissible text overlay occludes interactive clickable elements.

Definition. Similar to BubbleOcclT, but focuses on blocking interactive hotspots (buttons, links, input fields, icons, form controls).

Analysis. (1) Identify overlay with close control. (2) Locate all clickable elements. (3) Assess whether overlay's pixel region overlaps with any interactive hit area, preventing user access.

Options. A. No dismissible text overlay present. B. Overlay present but does not occlude clickable elements.

C. Overlay present and occludes clickable elements.

Output. Provide analysis and return $\boxed{X}$ where $X \in \{A, B, C\}$.

PopupNoClose (Popup Without Close Control). Assess whether a popup provides an explicit, discoverable close control.

Definition. An explicit close control comprises: (1) a clearly identifiable "×" icon (typically top-right/left corner), or (2) a text button labeled "Close" or equivalent. Other forms (e.g., "Cancel", "Back", ESC key, backdrop dismiss) do not qualify.

Analysis. (1) Identify whether a modal popup exists. (2) Search for qualifying close controls. (3) Verify control is visually clear and position-appropriate. (4) If any popup lacks such control, classify as non-compliant.

Options. A. No modal popup present. B. Modal popup exists without explicit close control. C. Modal popup exists with explicit close control.

Output. Provide analysis and return $\boxed{X}$ where $X \in \{A, B, C\}$.

PopupBlockClose (Popup Blocks Native Close). Analyze whether a popup's overlay blocks the native close button in H5 mini-program applications.

Core Principle. H5 mini-programs layer native controls (e.g., capsule button with "..." and "×") above page content (webview-rendered elements including popups and semi-transparent masks). Native controls remain physically clickable regardless of visual darkening by underlying masks due to superior z-index.

Analysis. (1) Identify H5 characteristics: presence of capsule button or circular close icon in navigation bar. (2) Locate popups and semi-transparent masks. (3) Apply core principle: even if visually covered, native close buttons remain clickable.

Options. A. Not an H5 application screenshot. B. Clickable. C. Not clickable.

Output. Provide analysis and return $\boxed{X}$ where $X \in \{A, B, C\}$.

PopupStack (Multiple Stacked Popups). Determine whether two or more modal dialogs exist simultaneously.

Definition. A modal dialog blocks main interface interaction, featuring: backdrop overlay, explicit dismissal requirements, interaction prevention outside dialog, and visual elevation (shadow, centering). Two or more such instances (nested or parallel) constitute stacking.

Analysis. (1) Identify components matching modal characteristics. (2) Count distinct modal instances, excluding non-modal overlays (Toast, Tooltip, Banner). (3) Classify as stacked if ≥ 2 independent modals detected.

Options. A. Yes (multiple modals present). B. No.

Output. Provide analysis and return $\boxed{X}$ where $X \in \{A, B\}$.

MismatchBadge (Badge-Landing Mismatch). Determine consistency between promotional badges (e.g., app home grid icons) and landing pages.

Task. Given a landing page screenshot and promotional copy, identify inconsistencies.

Criteria. (1) Classify copy type: Action-oriented (explicit behavior/benefit, e.g., "Get Red Packet") requires strict matching; Promotional (branding/atmosphere, e.g., "Safe Guardian") allows thematic relevance. (2) Judge whether landing page first-screen content (text, visuals, theme) effectively matches promotional copy.

Category A: Inconsistent. Landing page fails to match promotional copy through: no relevant content/visual elements on first screen, or core benefit mismatch (for action-oriented copy).

Category B: Consistent. First screen clearly matches copy via precise text/benefit matching, visual element correspondence, or thematic/functional alignment (for promotional copy).

Output. Analyze screenshot and promotional copy, return $\boxed{X}$ where $X \in \{A, B\}$.

MismatchContent (Service Name Mismatch). Evaluate consistency between service page UI screenshot and service name.

Criteria. Classify as Inconsistent (A) if: (1) Complete service name does not appear verbatim in screenshot text content, or (2) Service name contains unrelated terms absent from screenshot.

Options. A. Inconsistent. B. Consistent.

Output. Analyze screenshot and service name, return $\boxed{X}$ where $X \in \{A, B\}$.

MismatchFunc (Service Description Mismatch). Evaluate consistency between service page UI screenshot and service description.

Criteria. Classify as Inconsistent (A) if: (1) Core information in description does not align with overall functionality shown in screenshot, or (2) Key claims (service items, included content, usage conditions) lack explicit support in screenshot.

Options. A. Inconsistent. B. Consistent.

Output. Analyze screenshot and description, return $\boxed{X}$ where $X \in \{A, B\}$.

3. More Experimental Results

This section presents additional experimental results that provide further insights into the performance and efficiency of our proposed UI-UX model, complementing the analysis in Section 5.2 of the main paper.

3.1. Performance on VisualWebBench

To evaluate the impact of our training strategy, particularly the MultiUI dataset, on model generalization, we con-

duct experiments on the VisualWebBench benchmark. This benchmark assesses web UI understanding across captioning, QA, OCR, and grounding tasks. We compare three variants: the base model (Qwen3-VL-4B-Thinking), UI-UX trained *without* MultiUI, and UI-UX trained *with* MultiUI. Results are summarized in Table 1.

The comparison reveals consistent advantages of incorporating MultiUI. While the variant without MultiUI shows marginal or degraded performance (e.g., -7.8 in Act-Grd) versus the base model, UI-UX with MultiUI demonstrates significant gains across most tasks. Notably, it achieves $+9.5$ improvement in Web Captioning and $+3.8$ in Element Grounding. These results indicate that diverse UI scenarios in MultiUI prevent catastrophic forgetting during fine-tuning on UXBench's specialized tasks. It acts as an effective regularizer, enabling the model to retain and enhance broad UI comprehension abilities, validating our mixed-task training approach.

Table 1. Performance on UI understanding tasks. Web: captioning (Cap), WebQA (QA), HeadOCR. Elem: OCR, grounding (Grd). Act: prediction (Pred), grounding. M: multi-task.

Model	Web			Elem		Act	
	Cap	QA	OCR	OCR	Grd	Pred	Grd
qwen3-vl-4b-thinking	19.7	**83.2**	70.2	94.5	80.2	38.1	74.8
UI-UX (w/o MultiUI)	21.7	83.7	67.8	93.7	79.2	35.9	67.0
UI-UX (w/ MultiUI)	**29.2**	83.0	**71.2**	**95.3**	**84.0**	**39.5**	**75.7**

3.2. Comprehensive Performance Metrics on UXBench

To enable assessment under real-world imbalanced distributions where UX issues may be relatively rare, we evaluate models using precision, recall, and F1 metrics in addition to accuracy. Table 2 presents comprehensive performance comparisons across all models on the UXBench test set.

Table 2. Model performance comparison on UXBench with comprehensive metrics.

Model	ACC.	Prec.	Recall	F1
Qwen3-VL-235B-Instruct	0.5600	0.6130	0.7975	0.6176
Qwen3-VL-235B-Thinking	0.5854	0.5956	0.8108	0.6256
Claude-4.5-Sonnet	0.6550	0.6276	0.8160	0.6838
Qwen3-VL-4B-Instruct + RLVR	0.6100	0.6430	0.7663	0.6147
Qwen3-VL-4B-Thinking + SFT	0.6384	0.6598	0.7821	0.7158
UI-UX (ours)	**0.7963**	**0.8152**	**0.8350**	**0.8250**

Our UI-UX model achieves substantial improvements across all metrics. Most notably, UI-UX attains 0.8152 precision and 0.8350 recall, yielding an F1 score of 0.8250—significantly outperforming the best baseline (Claude-4.5-Sonnet with F1=0.6838) by 14.12 percentage points. The balanced precision-recall trade-off demonstrates that UI-UX not only identifies UX issues accurately but also maintains low false positive rates, which is critical for practical deployment where false alarms can erode user trust. The 21.13 percentage point improvement in accuracy over Claude-4.5-Sonnet further validates the effectiveness of our Asymmetric Transition Reward mechanism and multi-task training strategy.

3.3. Inference Latency Comparison on UXBench

A key advantage of UI-UX is its high inference efficiency, which is crucial for practical deployment. We measure the average inference latency (in seconds) and the mean output length (in tokens) for various models on the UXBench test set. All evaluations are conducted under identical conditions using a single NVIDIA A100 80GB GPU with vLLM v0.11.0 inference framework and temperature set to 0. Note that Claude-4.5-Sonnet latency is measured via API calls.

Table 3. Inference latency and output length comparison on UXBench. Claude-4.5-Sonnet is evaluated via API.

Model	Params	Latency (s)	Tokens
MimoVL-0528	7B	22.9	2,902
Qwen3-VL-Thinking	4B	25.3	3,490
Claude-4.5-Sonnet†	-	34.3	2,840
UI-UX (ours)	4B	**2.3**	**334**

†Evaluated via API call.

As shown in Table 3, UI-UX exhibits remarkably low inference latency of only 2.3 seconds per sample, which is approximately $10\times$ faster than MimoVL-0528 and Qwen3-VL-Thinking, and nearly $15\times$ faster than Claude-4.5-Sonnet. Furthermore, this speed is achieved with a drastically shorter mean output length of 334 tokens, compared to thousands of tokens generated by other reasoning models. This dramatic reduction in verbosity, over $8\times$ fewer tokens than competing approaches, is a direct result of the Asymmetric Transition Reward mechanism, which penalizes redundant reasoning steps while preserving accuracy. The combination of compact outputs and efficient inference makes UI-UX particularly suitable for real-world deployment scenarios requiring rapid UX issue detection, such as continuous integration pipelines or real-time user testing environments.