CONCLICK: ELEMENT-WISE CONTRASTIVE LEARNING IMPROVES GUI GROUNDING

Anonymous authors
Paper under double-blind review

ABSTRACT

GUI grounding, the task of locating the interface element specified by a user instruction, is a fundamental capability of GUI agents. However, existing grounding models remain error-prone, often causing downstream task failures. We systematically evaluate 10 grounding models across five datasets and identify a major error pattern: most incorrect clicks hit non-target interface elements rather than the background, with different models repeatedly selecting the same distractors. Motivated by these findings, we propose CONCLICK, a contrastive learning framework that teaches models to distinguish the target element from plausible distractors. We construct CONCLICK-DATA, a dataset containing 4,292 challenging target–distractor pairs, and train the model using complementary listwise ranking and pairwise margin objectives. CONCLICK improves the grounding accuracy of Qwen3-VL-8B from 54.71% to 66.03% on ScreenSpot-Pro, from 83.72% to 86.28% on MMBench-GUI L2, and from 63.33% to 70.20% on OSWorld-G, outperforming prior methods built on the same or similarly sized backbones. We further introduce RECLICK, an inference-time strategy that revisits promising regions and compares predictions across competing candidates, yielding an additional 2.96 percentage-point improvement in average accuracy across the three benchmarks. Together, our analysis, dataset, and methods provide both empirical insights and practical tools for building more reliable GUI agents. Data and code are available at https://anonymous.4open.science/r/ConClick-review-B321.

1 INTRODUCTION

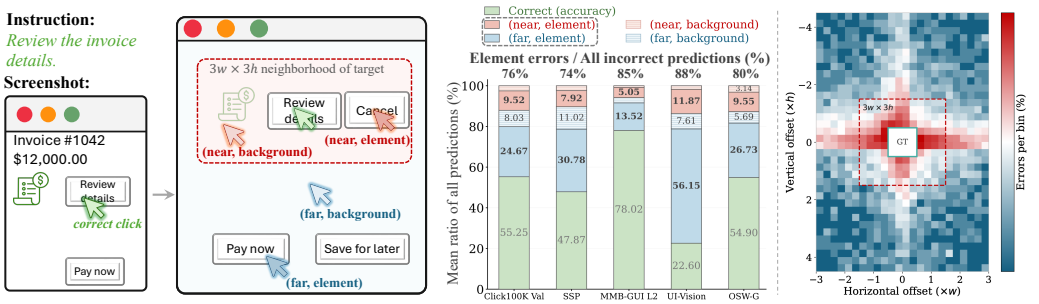
GUI grounding maps instructions and screenshots to click locations[1], providing a core capability for GUI agents. Substantial effort has been devoted to improving GUI grounding through stronger multimodal backbones (Bai et al., 2025; Wang et al., 2025a), larger datasets (Gou et al., 2025; Wu et al., 2025; Xie et al., 2026; Qin et al., 2025; Wang et al., 2026b), and reinforcement learning (Zhou et al., 2026; Yuan et al., 2025; Tang et al., 2026b; Liu et al., 2026; Gu et al., 2025). Inference methods yield further improvements by searching regions, enlarging crops, repeating predictions, or re-encoding selected regions (Luo et al., 2025; Zhang et al., 2026b; Jiang et al., 2025; Lin et al., 2026; Du et al., 2026). All these methods are evaluated using click-point accuracy, the percentage of correct clicks, defined as those falling within the annotated target region (Fig. 1(a)). However, this criterion treats all off-target clicks equally and does not reveal where incorrect clicks land. For example, when instructed to review the invoice details in Fig. 1, clicking "Pay now" instead of "Review details" can initiate an unintended payment, whereas clicking the background may have no effect. Wrong-button clicks may also delete files or expose confidential information, potentially causing irreversible harm that accuracy alone does not capture. Distinguishing error types is therefore important for assessing the potential consequences of grounding failures.

To bridge this gap, we examine incorrect clicks along two dimensions: element selection and spatial proximity. The element-level dimension distinguishes clicks on a non-target GUI element from those on the background. The spatial dimension distinguishes clicks near or far from the target element according to whether they fall inside or outside a target-centered neighborhood.[2] Combining

[1]Throughout this paper, we use "click" to refer to the predicted location.

[2]The neighborhood is a target-centered rectangle, such as one of size $3w \times 3h$, where w and h are the target width and height. Near and far are relative to this neighborhood, so far need not imply a large absolute distance.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.



(a) GUI grounding task (b) Four types of incorrect clicks (c) Clicks distributions from 10 models (d) Spatial distribution of incorrect clicks on Click100K Val

Figure 1: **Characterizing incorrect clicks.** (a) GUI grounding predicts a click within the target region given an instruction and a screenshot. (b) Incorrect clicks are categorized by whether they fall on an element or background and inside or outside the target-centered $3w \times 3h$ neighborhood. (c) Mean accuracy and four error rates across 10 models on five datasets, each using all predictions as the denominator. (d) Spatial distribution of incorrect clicks pooled across 10 models on Click100K Val. Offsets are normalized by target width and height; colors show the percentage of all predictions per 0.25×0.25 bin on a logarithmic scale. Red indicates more errors.

these dimensions yields four error categories: *near-element*, *near-background*, *far-element*, and *far-background* (Fig. 1(b)). We use this categorization to characterize incorrect clicks, with a particular focus on **element selection errors**. We evaluate 10 models across five datasets: a validation set of Click100K (Awadalla et al., 2025) and four test sets, ScreenSpot-Pro (Li et al., 2025), MMBench-GUI L2 (Wang et al., 2025b), UI-Vision (Nayak et al., 2025), and OSWorld-G (Xie et al., 2026).

Fig. 1(c) shows accuracy and the four error rates over all predictions, averaged across 10 models per dataset. These distributions reveal consistent error patterns across datasets. Specifically, *near-element* and *far-element* errors, i.e., clicks on non-target GUI elements, together account for approximately 74%–88% of all incorrect predictions across the five datasets. The predominance of element errors also holds when the 10 models are analyzed separately (Fig. 7). We conduct further analysis of incorrect element selection. Results show that element coverage and uniform random clicking cannot explain the predominance of element errors. We refer to non-target GUI elements selected by incorrect predictions as distractor elements and observe that different models can select the same distractor for the same instruction and screenshot. Our spatial analysis also shows that errors are denser inside the target neighborhood despite being more numerous outside it (Fig. 1(d)).

These results on element selection suggest that GUI models still struggle to distinguish the target from distractor elements, motivating us to explore supervision that explicitly accounts for distractors. Traditional coordinate supervision is effective for learning to predict target coordinates, but does not explicitly contrast the target with the distractor it is likely to select. Existing training examples with target-only grounding annotations do not provide the corresponding distractor elements needed for this supervision. To support it, we first construct ConClick-Data, a small dataset of about 8k examples from Click100K: half contain target–distractor annotations, and half are ordinary grounding examples sampled across source categories to maintain data diversity. We use two GUI models, GTA1-7B (Yang et al., 2026b) and GUI-Libra-8B (Yang et al., 2026a), to identify observed distractors by matching their incorrect predictions to non-target GUI elements.

For training, we introduce ConClick, an element-wise contrastive training strategy that learns to prefer the target over observed distractors for the same instruction. Specifically, a listwise ranking loss encourages the model to rank the target above candidate distractors, while a pairwise margin loss encourages a stronger preference for the target over the annotated distractor. We train the model with a coordinate loss on all examples and an additional selection loss combining these two terms on target–distractor examples, directly addressing the observed element selection errors.

As a natural extension, we introduce ReClick to further improve target selection at inference time by comparing competing candidate clicks. It uses the region preferences learned during training to select regions for revisiting and obtain candidate predictions. Motivated in part by the higher error density near the target, it also examines the area around the initial click. Each candidate is

evaluated by the support it receives from other views, measured by cross-view likelihood, and its spatial agreement with other candidates. We combine these two scores with equal weights to select the final prediction, using the same comparison rule on all benchmarks.

Experiments show that ConClick improves grounding accuracy across the models and datasets. On ScreenSpot-Pro, it raises Qwen3-VL-8B accuracy from 54.71% to 66.03%, with ReClick further improving it to 72.36%. Together, ConClick and ReClick reduce the element error rate across all three benchmarks. On ScreenSpot-Pro, ReClick also reduces near-target error density and achieves higher accuracy than MVP or ZoomClick applied to the same trained model.

Our contributions are summarized as follows:

- We dissect GUI grounding errors across models and datasets, revealing that most incorrect clicks hit other elements and that different models repeatedly select the same distractors.
- We introduce ConClick, an element-wise contrastive training strategy that strengthens discrimination between target and distractor elements. We also introduce ReClick to guide region revisits and compare candidate clicks at inference time.
- Experiments show that element-wise contrastive training improves grounding and reduces clicks on non-target elements across models and datasets, with improvements from ReClick.

2 Dissecting Incorrect Clicks in GUI Grounding

2.1 Error Decomposition Setup

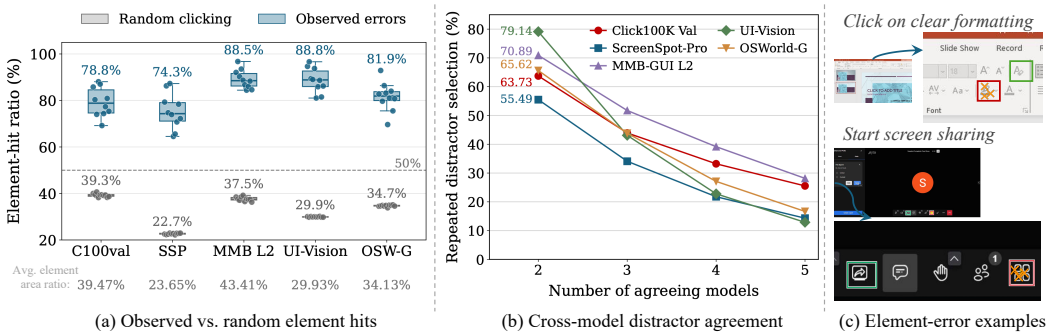
Model and data. We evaluate 10 GUI grounding models: GTA1-7B (Yang et al., 2026b), GUI-Libra-8B (Yang et al., 2026a), Qwen3-VL-8B (Bai et al., 2025), V2P-7B (Chen et al., 2025), OpenCUA-7B (Wang et al., 2026b), GUI-G2-7B (Tang et al., 2026b), UI-TARS-1.5-7B (Seed, 2025), SE-GUI-7B (Yuan et al., 2025), GUI-Actor-7B (Wu et al., 2026), and InternVL3.5-8B (Wang et al., 2025a), and closely reproduce reported accuracies using the official model configurations. Finally, 161,640 predictions are collected across the 10 models on a 5,000-example validation set of Click100K and four benchmarks: ScreenSpot-Pro, MMBench-GUI L2, UI-Vision, and OSWorld-G. For OSWorld-G, we use only target-bearing examples, excluding refusal examples. In addition to the ground-truth target annotations, we use OmniParser-v2 (Lu et al., 2025) to detect GUI element regions in each screenshot. Manual inspection indicates that the detections are generally reliable (App. A.1). We assign an incorrect click to the element category if it falls within a detected GUI element region and to the background category otherwise.

2.2 Error Patterns and Further Analysis

Incorrect element selections dominate across individual models. Across all 50 model–dataset combinations, near-element and far-element errors together account for 63.68%–95.43% of incorrect predictions, with far-element consistently the largest error category (see App. Fig. 7 for all results). The predominance of these errors is concerning because **clicking the wrong element can trigger harmful actions that are difficult or impossible to reverse**. For example, an agent may submit a payment instead of reviewing an invoice, permanently delete an important file, or send confidential information to the wrong recipient. These actions can cause financial loss, data loss, or privacy breaches, and correcting a later prediction may not undo the damage. A background click may leave the interface unchanged, yet click-point accuracy counts both outcomes as equally incorrect and does not capture this difference in potential harm.

Element coverage and uniform random clicking cannot explain the predominance of element errors. For each model, we analytically compute the random click expectation as the mean fraction of non-target screen area covered by GUI elements over its incorrect examples, counting overlaps only once. Across the five datasets, median observed element-hit proportions range from 74.3% to 88.8%, compared with 22.7%–39.3% under uniform random clicking (Fig. 2(a)). Across all 50 model–dataset combinations, the observed proportion of element errors exceeds 50%, while the random-click expectation remains below 50%.

Different models select the same distractors. Fig. 2(b) summarizes the results across the five datasets: 55.49%–79.14% of examples with at least two incorrect predictions contain a distractor



(a) Observed vs. random element hits (b) Cross-model distractor agreement (c) Element-error examples

Figure 2: **Further analysis of incorrect clicks.** (a) Observed versus random element-hit proportions across 10 models and five datasets. Points represent models; random click expectations vary with each model's error set. Box lines and labels show medians. Bottom percentages show dataset-wide mean element coverage. (b) Agreement of at least k models on the same distractor, among examples with at least k incorrect predictions. (c) Element-error examples. Green boxes mark targets, red boxes mark distractors, and orange crosses indicate incorrect element clicks from different models.

selected by multiple models. The proportions decrease as more models are required to agree, but among examples with at least five incorrect predictions, 12.88%–28.15% contain a distractor selected by at least five models. Fig. 2(c) illustrates two such cases of different models selecting the same distractor. For the Qwen3-VL-8B error cases from Click100K Val examined in Fig. 5(b), attention scores do not consistently favor the target over the distractor.

These findings motivate explicitly contrasting the target with observed distractors to strengthen target selection. Section 3 introduces ConClick, an element-wise contrastive training strategy, and ReClick as a natural inference-time extension.

3 Method

3.1 Element-wise Contrastive Training: ConClick

Fig. 3 shows how ConClick combines element-wise contrastive learning on observed target–distractor examples with coordinate supervision for GUI grounding. The contrastive objective operates on region selection scores, which are derived from attention and indicate the model preference for each candidate region under the given instruction. We compare the target with distractors in the same screenshot, encouraging a higher score for the target.

Constructing ConClick-Data for contrastive training. Each standard GUI grounding example contains a screenshot, an instruction q, and an annotated target box bbox^+, but no distractor annotations. We add a distractor box bbox^- to form target–distractor pairs for contrastive supervision, treating the target as the positive and the distractor as a hard negative. Motivated by the repeated selection of the same non-target elements across models, we use incorrect model predictions on training examples to identify observed distractors. Specifically, we use Click100K as the source of our training data. We obtain observed distractors from incorrect predictions of GTA1-7B and GUI-Libra-8B, matching their click locations to non-target GUI elements detected by OmniParser-v2. This procedure yields 4,292 target–distractor pairs. We combine these pairs with 4,292 ordinary grounding examples sampled across source categories to maintain data diversity, forming ConClick-Data. App. B provides more details.

Coordinate and contrastive selection losses. For all examples, we follow standard coordinate-supervised training, using the center of the annotated target box as the target click and applying a token-level cross-entropy loss $\mathcal{L}_{\mathrm{coord}}$ over the target coordinate sequence. For paired examples, we add a contrastive selection loss that encourages a preference for the target over distractor elements.

Attention-based GUI grounding and visual-score calibration have been explored in prior work (Wu et al., 2026; Chen et al., 2025). Here, we learn a region-selection readout supervised by observed target–distractor pairs. To compare the target with distractor elements, we learn weights to combine attention scores across layers and heads and aggregate them into region selection scores. We use pre-

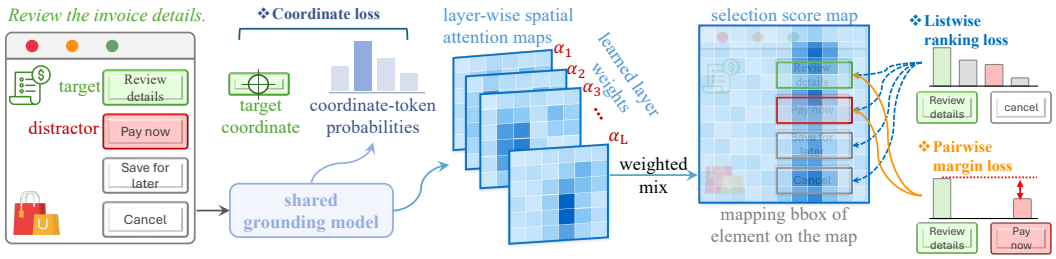


Figure 3: **Element-wise contrastive training with ConClick.** In the example on the left, the target is "Review details" and the distractor is "Pay now." Learned weights combine spatial attention maps from the shared grounding model into a selection score map for GUI element regions. The listwise ranking loss contrasts the target with candidate distractors, while the pairwise margin loss encourages a higher score for the target than for the annotated distractor.

softmax attention scores from the final prompt token to each visual token (Vaswani et al., 2017). Let $A_{l,h,i}$ denote this score for visual token i in layer l and head h. Learned weights $\alpha_{l,h}$ are normalized by a softmax across layers and heads to produce token selection scores s_i, which are aggregated within each region bbox using log-mean-exp:

$$s_i = \sum_{l,h} \alpha_{l,h} A_{l,h,i}, \qquad S(\mathrm{bbox}) = \log\left(\frac{1}{N_{\mathrm{bbox}}} \sum_{i \in \mathrm{bbox}} \exp(s_i)\right). \tag{1}$$

Here $S(\mathrm{bbox})$ is the score of the entire region, obtained by aggregating its token scores. N_{bbox} is the number of visual tokens within the box, and $i \in \mathrm{bbox}$ denotes these tokens.

The annotated distractor bbox^- remains fixed for each training example. To introduce additional hard negatives as the model learns, we select up to three other regions with the highest current selection scores, excluding regions whose visual-token masks overlap the target or annotated distractor. The candidate set $\mathcal{B}$ therefore contains bbox^+, bbox^-, and these additional high-scoring regions. The selection loss $\mathcal{L}_{\mathrm{sel}}$ implements element-wise contrastive supervision. We combine a softmax listwise ranking objective with a margin-augmented pairwise logistic objective, with the pairwise term weighted by λ_{pair}:

$$\mathcal{L}_{\mathrm{sel}} = \underbrace{-\log \frac{\exp(S(\mathrm{bbox}^+))}{\sum_{\mathrm{bbox} \in \mathcal{B}} \exp(S(\mathrm{bbox}))}}_{\mathcal{L}_{\mathrm{list}}\ \text{(Listwise ranking)}} + \lambda_{\mathrm{pair}} \underbrace{\log\big(1 + \exp\big(m + S(\mathrm{bbox}^-) - S(\mathrm{bbox}^+)\big)\big)}_{\mathcal{L}_{\mathrm{pair}}\ \text{(Pairwise margin)}}. \tag{2}$$

The listwise ranking loss $\mathcal{L}_{\mathrm{list}}$ contrasts the positive target with all negative distractors in $\mathcal{B}$ through softmax normalization, encouraging a higher score for the target. The pairwise margin loss $\mathcal{L}_{\mathrm{pair}}$ directly compares the target and annotated distractor, using a softplus penalty to encourage the target score to exceed the distractor score by a margin m. For paired examples, we add the selection loss weighted by λ_{sel} to the coordinate loss. Ordinary examples use only $\mathcal{L}_{\mathrm{coord}}$. Parsed regions are required only to construct the data and compute the training loss.

3.2 Region Revisiting and Candidate Comparison: ReClick

Fig. 4 illustrates the procedure of ReClick. Region revisits are guided by selection scores learned through contrastive training; for other grounders, these scores can instead be derived from frozen model attention (App. C.2). Given only a screenshot and an instruction, ReClick first produces an initial prediction, then uses these scores to select regions that may contain competing elements and also examines a crop around the initial click. It compares candidate clicks decoded from these enlarged views and the full screenshot using cross-view likelihood and spatial agreement.

Score regions, revisit crops, and decode clicks. The full screenshot pass produces the initial prediction p_0 and token scores from the attention readout. We define four overlapping crop regions that cover the screenshot, score each region using Eq. (1), and select the two highest-scoring regions as potential locations of competing elements (Crop1 and Crop3 in Fig. 4). We also include a crop centered on p_0 (Crop* in Fig. 4) to examine the area around the initial click. We enlarge the three crops by $2\times$ and decode one point from each. Mapping these points back to the original screenshot

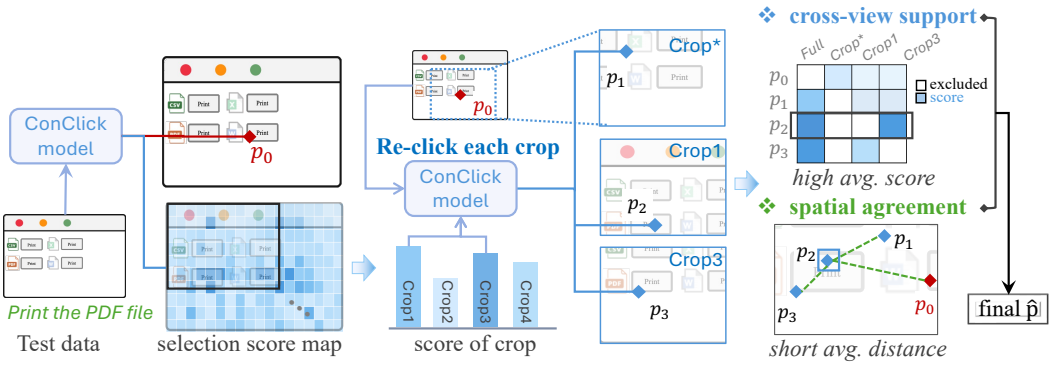


Figure 4: **RECLICK at inference time.** Given a screenshot and an instruction, CONCLICK produces an initial click p_0 and a selection score map. RECLICK selects two high-scoring regions (Crop1 and Crop3) and a crop centered on p_0 (Crop*), then predicts clicks p_1, p_2, p_3 from the enlarged crops. It compares all four candidates using cross-view support and spatial agreement. Cross-view support measures coordinate likelihood in other views containing the candidate, while spatial agreement favors candidates with smaller mean distances to the others. The final click $\hat{p}$ maximizes the equally weighted sum of the two standardized scores.

gives candidates $\mathcal{P} = \{p_0, p_1, p_2, p_3\}$, where p_1 comes from the crop around the initial click and p_2, p_3 come from the two score-guided crops. The full screenshot and the three enlarged crops form four views for candidate prediction and comparison.

Cross-view candidate evidence. For each candidate p, we compute the log-likelihood of its coordinate sequence in every view containing that location and standardize the scores within each view. We then exclude its generating view and denote the remaining views containing its location by $\mathcal{V}_p$. For each view $v \in \mathcal{V}_p$, $\bar{\ell}_v(p)$ denotes the standardized log-likelihood of the coordinate sequence for p in that view. The cross-view support score is

$$S_{\text{cross}}(p) = \frac{1}{|\mathcal{V}_p|} \sum_{v \in \mathcal{V}_p} \bar{\ell}_v(p). \tag{3}$$

Excluding candidates without cross-view scores gives the remaining candidate set $\mathcal{P}'$.

Choosing a candidate. We measure spatial agreement as the negative mean distance to other candidates in $\mathcal{P}'$, using coordinates $\tilde{p} = (p_x/W, p_y/H)$ normalized by screenshot width and height:

$$S_{\text{agree}}(p) = -\frac{\sum_{p' \in \mathcal{P}' \setminus \{p\}} \|\tilde{p} - \tilde{p}'\|_2}{\max(1, |\mathcal{P}'| - 1)}. \tag{4}$$

Higher scores indicate greater spatial agreement; the score is zero when only one candidate remains. We standardize both scores across $\mathcal{P}'$, denoted by $\hat{S}_{\text{cross}}$ and $\hat{S}_{\text{agree}}$, and select the candidate with the highest equally weighted sum:

$$\hat{p} = \arg\max_{p \in \mathcal{P}'} \left[\hat{S}_{\text{cross}}(p) + \hat{S}_{\text{agree}}(p)\right]. \tag{5}$$

We use the same comparison rule across all three benchmarks and retain the initial prediction if $\mathcal{P}'$ is empty. The procedure uses up to four visual grounding passes plus cached-prefix candidate continuations. App. C provides more implementation details.

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUPS

Training data and backbone. Section 3.1 describes the training data. Our primary backbone is Qwen3-VL-8B. We also evaluate InternVL3.5-8B and Qwen3-VL-30B-A3B-Instruct to verify

Table 1: Performance of different models. Acc. is point accuracy, and E-err. is the element error rate over all predictions. ConClick-8B and ConClick-8B + ReClick use the same trained checkpoint. For other models, * denotes published accuracies reported in the original publications; all other results are re-evaluated using released weights and implementations. † marks ScreenSpot-Pro accuracy within one percentage point of the published result, supporting the fidelity of the reproduction used to compute E-err. Avg. is the mean across the three benchmarks. All values are percentages. **Best** and **second-best** results are in **blue bold** and **black bold**, respectively.

	ScreenSpot-Pro		OSWorld-G		MMB-GUI L2		Avg.	
Method	Acc.↑	E-err.↓	Acc.↑	E-err.↓	Acc.↑	E-err.↓	Acc.↑	E-err.↓
GUI-Actor-7B (Wu et al., 2026)	44.09†	43.83	43.92	42.35	69.59	19.37	52.53	35.18
SE-GUI-7B (Yuan et al., 2025)	46.74†	37.51	49.02	42.16	76.74	19.89	57.50	33.19
GUI-G2-7B (Tang et al., 2026b)	48.39†	33.14	40.59	41.37	79.27	17.84	56.08	30.78
OpenCUA-7B (Wang et al., 2026b)	49.46†	34.98	60.98	31.96	81.02	16.17	63.82	27.70
UI-TARS-1.5-7B (Seed, 2025)	49.65†	33.02	60.78	31.76	75.43	20.67	61.95	28.48
V2P-7B (Chen et al., 2025)	52.25†	41.68	58.24	36.08	80.30	18.42	63.60	32.06
GUI-Libra-8B (Yang et al., 2026a)	54.84†	38.20	63.53	33.33	79.60	19.09	65.99	30.21
GTA1-7B (Yang et al., 2026b)	55.34†	35.36	64.71	29.22	83.86	14.83	67.97	26.47
ZwZ-8B (Wei et al., 2026)	56.80*	-	-	-	-	-	-	-
Propose-then-Critic-8B (Wang et al., 2026a)	58.70*	-	-	-	78.40*	-	-	-
Qwen3-VL-8B (Bai et al., 2025) (baseline)	54.71	33.84	63.33	29.22	83.72	14.66	67.25	25.91
ConClick-8B (Ours)	**66.03**	**30.30**	**70.20**	**24.12**	**86.28**	**13.08**	**74.17**	**22.50**
ConClick-8B + ReClick (Ours)	**72.36**	**25.87**	**71.76**	**23.73**	**87.28**	**12.10**	**77.13**	**20.57**

cross-backbone applicability. We use LoRA (Hu et al., 2022) with bfloat16 for every backbone. App. B provides the detailed training settings for ConClick-8B.

Evaluation. Following prior work, we report accuracy (Acc.), the percentage of predictions inside the target region. We also report the element error rate (E-err.), the percentage of all predictions that fall outside the target region and on a parsed GUI element. We evaluate on ScreenSpot-Pro, MMBench-GUI L2, and OSWorld-G, with 1,581, 3,594, and 510 examples, respectively.

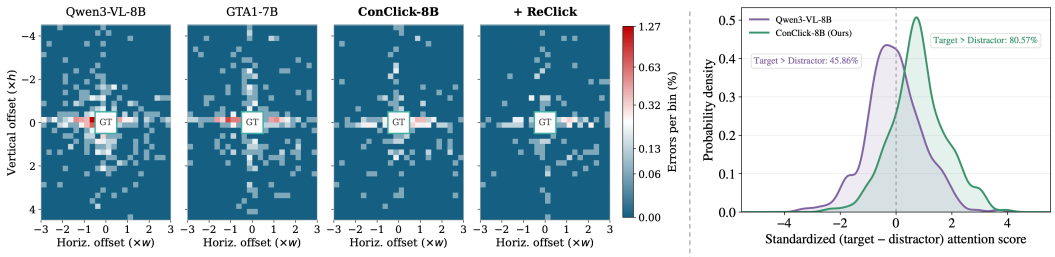
Inference baselines. We compare ReClick with MVP (Zhang et al., 2026b) and ZoomClick (Jiang et al., 2025) on the same ConClick-8B checkpoint. MVP uses its released prompts and preprocessing, while ZoomClick uses its released search procedure. App. C provides implementation details, including minimum crop sizes and boundary handling.

4.2 Main Results

GUI Model Comparisons. Table 1 shows that ConClick-8B improves accuracy and reduces element error rates over the Qwen3-VL-8B backbone on all three benchmarks. It achieves accuracies of 66.03%, 86.28%, and 70.20% on ScreenSpot-Pro, MMBench-GUI L2, and OSWorld-G, respectively, improving over the baseline by 11.32, 2.56, and 6.87 percentage points. Compared with other locally evaluated models in Table 1, ConClick-8B achieves the highest average accuracy (74.17%) and the lowest element error rates on all three benchmarks, averaging 22.50%; error-rate comparisons are limited to methods with available results. These results support our training strategy for improving grounding accuracy and reducing element selection errors. Subset accuracies for locally evaluated models appear in App. Tables A6 (ScreenSpot-Pro) and A7 (MMBench-GUI L2).

ReClick further improves grounding at inference time. Compared with direct inference, ReClick increases accuracy by 6.33, 1.00, and 1.57 percentage points on these three datasets, respectively, while reducing element error rates on all three benchmarks (Table 1).

Higher grounding accuracy does not necessarily imply fewer clicks on wrong elements. Across the nine prior models with both metrics in Table 1, the Spearman correlations between accuracy and element error rate are -0.117, -0.883, and -0.820 on ScreenSpot-Pro, MMBench-GUI L2, and OSWorld-G, respectively. Although the two metrics are negatively correlated on the latter two benchmarks, their weak correlation on ScreenSpot-Pro shows that accuracy alone does not reliably reflect element selection errors across models. For example, GUI-Libra-8B has higher accuracy than UI-TARS-1.5-7B on ScreenSpot-Pro (54.84% vs. 49.65%) but also a higher element error rate



(a) Spatial distributions of incorrect clicks on ScreenSpot-Pro

(b) Attention-score gaps on corrected cases

Figure 5: **Spatial errors and attention-score gaps.** (a) Incorrect clicks on ScreenSpot-Pro, with target-centered offsets normalized by target width and height. Red indicates more errors per bin as a percentage of all predictions. (b) Standardized target–distractor score gaps on Click100K Val examples where Qwen3-VL-8B fails and ConClick-8B succeeds.

Table 2: Ablations of training data and loss components on Qwen3-VL-8B. Colors are normalized within each column, with darker green indicating better performance; bold marks the best results.

Data		Loss			ScreenSpot-Pro		OSWorld-G		MMB-GUI L2		Avg.	
Random	Our	$\mathcal{L}_{\text{coord}}$	$\mathcal{L}_{\text{list}}$	$\mathcal{L}_{\text{pair}}$	Acc.↑	E-err.↓	Acc.↑	E-err.↓	Acc.↑	E-err.↓	Acc.↑	E-err.↓
					54.71	33.84	63.33	29.22	83.72	14.66	67.25	25.91
✓		✓			59.90	32.64	67.65	26.08	84.47	14.16	70.67	24.29
	✓	✓			62.43	32.26	68.63	25.49	84.92	13.72	71.99	23.82
	✓	✓	✓		64.26	31.82	**70.20**	25.10	85.81	13.41	73.42	23.44
	✓	✓		✓	64.20	32.07	69.41	25.29	85.92	13.36	73.18	23.57
	✓	✓	✓	✓	**66.03**	**30.30**	**70.20**	**24.12**	**86.28**	**13.08**	**74.17**	**22.50**

(38.20% vs. 33.02%). We therefore recommend reporting error types, particularly element errors, alongside accuracy when evaluating GUI grounding models.

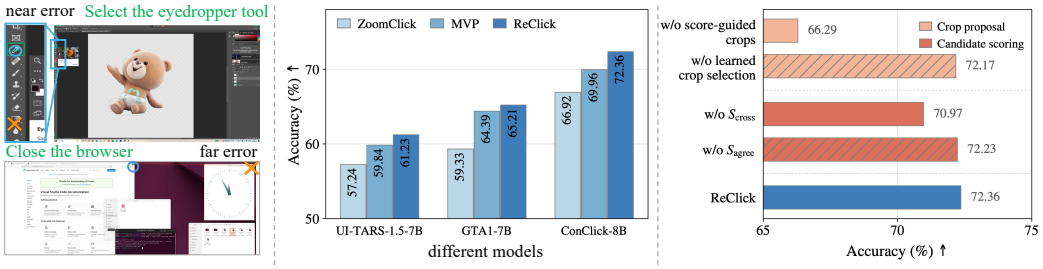
ConClick and ReClick also reduce near-target error density. Fig. 5(a) compares the spatial distributions of incorrect clicks on ScreenSpot-Pro. Compared with the Qwen3-VL-8B baseline, our training strategy lowers error density within the target neighborhood, and ReClick further reduces it at inference time. ConClick-8B exhibits lower near-target error density than GTA1-7B. These results show that improvements extend beyond element selection to reducing errors near the target.

Attention scores favor the target on corrected examples. Fig. 5(b) examines examples from Click100K Val on which Qwen3-VL-8B predicts incorrectly but ConClick-8B predicts correctly. We compute the attention-score difference between the target and a fixed distractor for each model, standardized by the standard deviation of its visual-token scores. Compared with Qwen3-VL-8B, the distribution for ConClick-8B shifts toward more positive values, indicating a stronger preference for the target over the distractor on these corrected examples. These results support the effectiveness of ConClick in strengthening the preference for the target over distractors.

4.3 Ablation Studies

Ablations for ConClick. Table 2 examines the contributions of training data and contrastive losses. With coordinate-only training, replacing equally sized random data with our data raises average accuracy from 70.67% to 71.99% and lowers the average element error rate from 24.29% to 23.82%. Adding the full contrastive objective further improves these metrics to 74.17% and 22.50%, respectively. On the same data, combining the listwise ranking loss $\mathcal{L}_{\text{list}}$ and pairwise margin loss $\mathcal{L}_{\text{pair}}$ (Eq. (2)) achieves higher average accuracy than either term alone (74.17% vs. 73.42% and 73.18%, respectively), with the lowest element error rates across all three benchmarks. These results support the effectiveness of both our data construction strategy and the combined contrastive objective. App. B details the random-data sampling procedure, and App. B.2 reports additional training ablations on InternVL3.5-8B and Qwen3-VL-30B-A3B-Instruct.

Test-time method comparisons and ReClick ablations. Fig. 6(a) shows ReClick correcting initial clicks both near and far from the target, illustrating its benefits extend beyond nearby errors.



(a) Clicks corrected by ReClick (b) Comparison of inference methods (c) Ablation of ReClick components

Figure 6: **Examples, comparisons, and ablations of ReClick.** (a) Examples where ReClick corrects initial clicks near and far from the target. Orange crosses mark initial incorrect clicks, and blue circles mark corrected predictions. (b) Accuracy comparison of ZoomClick, MVP, and ReClick across three backbones. (c) Ablations of crop proposal and candidate scoring components using the same ConClick-8B checkpoint. Results in (b,c) are on ScreenSpot-Pro.

Fig. 6(b) shows ReClick achieving higher accuracy than ZoomClick and MVP on ScreenSpot-Pro across three GUI grounding models. On ConClick-8B, it reaches 72.36% accuracy, compared with 69.96% for MVP and 66.92% for ZoomClick. Applying ReClick to other backbones requires adapting the attention readout. Implementation details appear in App. C.2. Fig. 6(c) examines the components of ReClick on ScreenSpot-Pro. Removing score-guided crops or using a fixed attention readout to select them lowers average accuracy across the three benchmarks. Combining cross-view support and spatial agreement yields the best average accuracy (App. Table A4).

5 Related Work

GUI grounding training. Prior work develops coordinate pretraining and grounding supervision across platforms (Cheng et al., 2024; Gou et al., 2025; Wu et al., 2025; Xie et al., 2026; Feizi et al., 2026), attention based action prediction and visual scoring (Wu et al., 2026; Chen et al., 2025), and reinforcement learning or spatial and semantic alignment (Zhou et al., 2026; Yuan et al., 2025; Tang et al., 2026b; Liu et al., 2026). VISTA and UI-Ins explore target preserving views and multi-perspective instructions, respectively (Qiu et al., 2026; Chen et al., 2026). Related studies examine instruction sensitivity and hallucinations (Jandial et al., 2026; Zhang et al., 2026c). Motivated by our error analysis, ConClick contrasts target and observed distractor region scores alongside coordinate supervision to improve element selection.

Inference-time improvements. Prior methods use region search, cropping, and candidate selection (Li et al., 2025; Luo et al., 2025; Tang et al., 2026a; Jiang et al., 2025; Tao et al., 2025; Zhang et al., 2026a), or reprocess selected visual tokens (Ouyang et al., 2026; Lin et al., 2026). MVP clusters predictions across attention guided views (Zhang et al., 2026b), while GUI-RC uses spatial voting over sampled predictions (Du et al., 2026). UI-AGILE uses a vision language model to select crops, and Measure Twice trains a visual critic (Lian et al., 2026; Wang et al., 2026a). Our ReClick requires no external model beyond the trained grounder; it revisits selected regions and combines cross-view likelihood with spatial agreement to select the final click.

6 Conclusion

This paper analyzes GUI grounding errors across ten models and five datasets, showing that most incorrect clicks hit non-target elements and different models repeatedly select the same distractors. Motivated by these findings, we construct ConClick-Data and introduce ConClick, an element-wise contrastive training framework that learns to distinguish targets from observed distractors. As a natural extension, ReClick uses learned region preferences to guide revisits and compares candidate predictions across views. Experiments show improved grounding accuracy and fewer clicks on non-target elements, with further gains from ReClick. Our findings highlight the importance of distinguishing error types alongside accuracy, since wrong-element clicks can trigger harmful actions, and show how error analysis can guide training and inference.

AI USE STATEMENT

We used generative AI tools to assist with coding the methods and experiments designed by the authors, verifying training data, reviewing grounding errors, polishing the manuscript, checking for typographical errors, and checking the consistency of mathematical notation. The authors reviewed the AI-assisted work, including the code, data verification, and error analysis, and verified the experimental outputs. We take responsibility for the final content, including text, claims, and code.

REPRODUCIBILITY STATEMENT

Our code, CONCLICK-DATA, and trained model checkpoints are publicly available at https://anonymous.4open.science/r/ConClick-review-B321 to support reproduction and further research. We train CONCLICK-8B on a single compute node using two NVIDIA H200 GPUs. The appendix describes data construction and sampling, training configurations, evaluation protocols, and the implementation of RECLICK. For the locally evaluated baselines in Table 1, we use the official implementations released by the respective authors. Where comparable published results are available, our reproduced accuracies closely match the reported results; on ScreenSpot-Pro, they are within one percentage point.

REFERENCES

Anas Awadalla, Dhruba Ghosh, Aylin Akkus, Yuhui Zhang, Marianna Nezhurina, Jenia Jitsev, Yejin Choi, and Ludwig Schmidt. Gelato — from data curation to reinforcement learning: Building a strong grounding model for computer-use agents. https://github.com/mlfoundations/gelato, 2025.

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, et al. Qwen3-vl technical report. *arXiv preprint arXiv:2511.21631*, 2025.

Jikai Chen, Long Chen, Dong Wang, Qinglin Su, Zhixuan Chu, Bingguang Hao, Leilei Gan, Chenyi Zhuang, and Jinjie Gu. V2p: Visual attention calibration for gui grounding via background suppression and center peaking. *arXiv preprint arXiv:2508.13634*, 2025.

Liangyu Chen, Hanzhang Zhou, Jianan Zhang, Panrong Tong, Xu Zhang, Quyu Kong, Chen Liu, Yuqi Liu, Wenxuan Wang, Yue Wang, et al. Ui-ins: Enhancing gui grounding with multi-perspective instruction as reasoning. In *International Conference on Learning Representations*, volume 2026, pp. 54418–54444, 2026.

Kanzhi Cheng, Qiushi Sun, Yougang Chu, Fangzhi Xu, Li YanTao, Jianbing Zhang, and Zhiyong Wu. Seeclick: Harnessing gui grounding for advanced visual gui agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 9313–9332, 2024.

Yong Du, Yuchen Yan, Fei Tang, Zhengxi Lu, Chang Zong, Weiming Lu, Shengpei Jiang, and Yongliang Shen. Test-time reinforcement learning for gui grounding via region consistency. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 30593–30601, 2026.

Aarash Feizi, Shravan Nayak, Xiangru Jian, Kevin Qinghong Lin, Kaixin Li, Rabiul Awal, Xing Han Lù, Johan S Obando Ceron, Juan A Rodriguez, Nicolas Chapados, et al. Grounding computer use agents on human demonstrations. In *International Conference on Learning Representations*, volume 2026, pp. 5181–5215, 2026.

Google DeepMind. Gemini 3.5 Flash model card, May 2026. URL https://deepmind.google/models/model-cards/gemini-3-5-flash/. Published May 19, 2026. Accessed September 21, 2026.

Boyu Gou, Demi Ruohan Wang, Boyuan Zheng, Yanan Xie, Cheng Chang, Yiheng Shu, Huan Sun, and Yu Su. Navigating the digital world as humans do: Universal visual grounding for gui agents. In *International Conference on Learning Representations*, volume 2025, pp. 30851–30883, 2025.

Zhangxuan Gu, Zhengwen Zeng, Zhenyu Xu, Xingran Zhou, Shuheng Shen, Yunfei Liu, Beitong Zhou, Changhua Meng, Tianyu Xia, Weizhi Chen, et al. Ui-venus technical report: Building high-performance ui agents with rft. *arXiv preprint arXiv:2508.10833*, 2025.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

JaidedAI. EasyOCR. GitHub repository, 2020. URL https://github.com/JaidedAI/EasyOCR. Software. Accessed September 21, 2026.

Surgan Jandial, Yinheng Li, Justin Wagle, and Kazuhito Koishida. Do gui grounders truly understand ui elements? In *Findings of the Association for Computational Linguistics: EACL 2026*, pp. 2772–2785, 2026.

Zhiyuan Jiang, Shenghao Xie, Wenyi Li, Wenqiang Zu, Peihang Li, Jiahao Qiu, Siqi Pei, Lei Ma, Tiejun Huang, Mengdi Wang, et al. Zoom in, click out: Unlocking and evaluating the potential of zooming for gui grounding. *arXiv preprint arXiv:2512.05941*, 2025.

Kaixin Li, Ziyang Meng, Hongzhan Lin, Ziyang Luo, Yuchen Tian, Jing Ma, Zhiyong Huang, and Tat-Seng Chua. Screenspot-pro: Gui grounding for professional high-resolution computer use. In *Proceedings of the 33rd ACM International Conference on Multimedia*, pp. 8778–8786, 2025.

Shuquan Lian, Yuhang Wu, Jia Ma, Yifan Ding, Zihan Song, Bingqi Chen, Xiawu Zheng, Hui Li, and Rongrong Ji. Ui-agile: Advancing gui agents with effective reinforcement learning and precise inference-time grounding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8575–8584, 2026.

Jiaping Lin, Fei Shen, Junzhe Li, Ping Nie, Fei Yu, Ming Li, and Haizhou Li. What happens before decoding? prefill determines gui grounding in vlms. *arXiv preprint arXiv:2605.12549*, 2026.

Yuhang Liu, Zeyu Liu, Shuanghe Zhu, Pengxiang Li, Congkai Xie, Jiasheng Wang, Xueyu Hu, Xiaotian Han, Jianbo Yuan, Xinyao Wang, et al. Infigui-g1: Advancing gui grounding with adaptive exploration policy optimization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 32267–32275, 2026.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

Y Lu, T Dhome-Casanova, J Yang, and A Awadallah. Omniparser v2: Turning any llm into a computer use agent. *Microsoft Research. Retrieved March*, 18:2025, 2025.

Tiange Luo, Lajanugen Logeswaran, Justin Johnson, and Honglak Lee. Visual test-time scaling for gui agent grounding. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 19989–19998, 2025.

Shravan Nayak, Xiangru Jian, Kevin Qinghong Lin, Juan A Rodriguez, Montek Kalsi, Rabiul Awal, Nicolas Chapados, M Tamer Özsu, Aishwarya Agrawal, David Vazquez, et al. Ui-vision: A desktop-centric gui benchmark for visual perception and interaction. *arXiv preprint arXiv:2503.15661*, 2025.

Mingyu Ouyang, Kevin Qinghong Lin, Mike Zheng Shou, and Hwee Tou Ng. Focusui: Efficient ui grounding via position-preserving visual token selection. *arXiv preprint arXiv:2601.03928*, 2026.

Yujia Qin, Yining Ye, Junjie Fang, Haoming Wang, Shihao Liang, Shizuo Tian, Junda Zhang, Jiahao Li, Yunxin Li, Shijue Huang, et al. Ui-tars: Pioneering automated gui interaction with native agents. *arXiv preprint arXiv:2501.12326*, 2025.

Xinyu Qiu, Yunzhu Zhang, Heng Jia, Shuheng Shen, Changhua Meng, and Linchao Zhu. Vista: View-consistent self-verified training for gui grounding. *arXiv preprint arXiv:2606.14579*, 2026.

ByteDance Seed. Ui-tars-1.5. https://seed-tars.com/1.5, 2025.

Fei Tang, Bofan Chen, Zhengxi Lu, Tongbo Chen, Songqin Nong, Tao Jiang, Wenhao Xu, Weiming Lu, Jun Xiao, Yueting Zhuang, et al. Ui-zoomer: Uncertainty-driven adaptive zoom-in for gui grounding. *arXiv preprint arXiv:2604.14113*, 2026a.

Fei Tang, Zhangxuan Gu, Zhengxi Lu, Xuyang Liu, Shuheng Shen, Changhua Meng, Wen Wang, Wenqi Zhang, Yongliang Shen, Weiming Lu, et al. Gui-g^2: Gaussian reward modeling for gui grounding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 33214–33222, 2026b.

Xingjian Tao, Yiwei Wang, Yujun Cai, Zhicheng Yang, and Jing Tang. Understanding gui agent localization biases through logit sharpness. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pp. 23361–23374, 2025.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Weiyun Wang, Zhangwei Gao, Lixin Gu, Hengjun Pu, Long Cui, Xingguang Wei, Zhaoyang Liu, Linglin Jing, Shenglong Ye, Jie Shao, et al. Internvl3. 5: Advancing open-source multimodal models in versatility, reasoning, and efficiency. *arXiv preprint arXiv:2508.18265*, 2025a.

Wenkai Wang, Xiyun Li, Hongcan Guo, Wenhao Yu, Tianqing Fang, Haitao Mi, Dong Yu, and Shengyu Zhang. Measure twice, click once: Co-evolving proposer and visual critic via reinforcement learning for gui grounding. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 20964–20984, 2026a.

Xinyuan Wang, Bowen Wang, Dunjie Lu, Junlin Yang, Tianbao Xie, Junli Wang, Jiaqi Deng, Xiaole Guo, Yiheng Xu, Chen Wu, et al. Opencua: Open foundations for computer-use agents. *Advances in Neural Information Processing Systems*, 38:139756–139806, 2026b.

Xuehui Wang, Zhenyu Wu, JingJing Xie, Zichen Ding, Bowen Yang, Zehao Li, Zhaoyang Liu, Qingyun Li, Xuan Dong, Zhe Chen, et al. Mmbench-gui: Hierarchical multi-platform evaluation framework for gui agents. *arXiv preprint arXiv:2507.19478*, 2025b.

Lai Wei, Liangbo He, Jun Lan, Lingzhong Dong, Yutong Cai, Siyuan Li, Huijia Zhu, Weiqiang Wang, Linghe Kong, Yue Wang, et al. Zooming without zooming: Region-to-image distillation for fine-grained multimodal perception. *arXiv preprint arXiv:2602.11858*, 2026.

Qianhui Wu, Kanzhi Cheng, Rui Yang, Chaoyun Zhang, Jianwei Yang, Huiqiang Jiang, Jian Mu, Baolin Peng, Bo Qiao, Reuben Tan, et al. Gui-actor: Coordinate-free visual grounding for gui agents. *Advances in Neural Information Processing Systems*, 38:15101–15128, 2026.

Zhiyong Wu, Zhenyu Wu, Fangzhi Xu, Yian Wang, Qiushi Sun, Chengyou Jia, Kanzhi Cheng, Zichen Ding, Liheng Chen, Paul Pu Liang, et al. Os-atlas: Foundation action model for generalist gui agents. In *International Conference on Learning Representations*, volume 2025, pp. 5090–5108, 2025.

Tianbao Xie, Jiaqi Deng, Xiaochuan Li, Junlin Yang, Haoyuan Wu, Jixuan Chen, Wenjing Hu, Xinyuan Wang, Yuhui Xu, Zekun Wang, et al. Scaling computer-use grounding via user interface decomposition and synthesis. *Advances in Neural Information Processing Systems*, 38, 2026.

Rui Yang, Qianhui Wu, Zhaoyang Wang, Hanyang Chen, Ke Yang, Hao Cheng, Huaxiu Yao, Baolin Peng, Huan Zhang, Jianfeng Gao, et al. Gui-libra: Training native gui agents to reason and act with action-aware supervision and partially verifiable rl. *arXiv preprint arXiv:2602.22190*, 2026a.

Yan Yang, Dongxu Li, Yutong Dai, Yuhao Yang, Ziyang Luo, Zirui Zhao, Zhiyuan Hu, Junzhe Huang, Amrita Saha, Zeyuan Chen, et al. Gta1: Gui test-time scaling agent. In *International Conference on Learning Representations*, volume 2026, pp. 48295–48323, 2026b.

Xinbin Yuan, Jian Zhang, Kaixin Li, Zhuoxuan Cai, Lujian Yao, Jie Chen, Enguang Wang, Qibin Hou, Jinwei Chen, Peng-Tao Jiang, et al. Enhancing visual grounding for gui agents via self-evolutionary reinforcement learning. *arXiv preprint arXiv:2505.12370*, 2025.

Borui Zhang, Bo Zhang, Bo Wang, Wenzhao Zheng, Yuhao Cheng, Liang Tang, Yiqiang Yan, Jie Zhou, and Jiwen Lu. Bami: Training-free bias mitigation in gui grounding. *arXiv preprint arXiv:2605.06664*, 2026a.

Yunzhu Zhang, Zeyu Pan, Zhengwen Zeng, Shuheng Shen, Changhua Meng, and Linchao Zhu. Mvp: Multiple view prediction improves gui grounding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 27482–27492, 2026b.

Zicheng Zhang, Hongyi Jing, Rui Lv, Shuo Fang, Shiai Zhu, Junying Wang, Chunyi Li, Xiaohong Liu, Chenguang Ma, and Guangtao Zhai. Exposing and evaluating hallucinations for gui grounding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 40208–40223, 2026c.

Yuqi Zhou, Sunhao Dai, Shuai Wang, Kaiwen Zhou, Qinglin Jia, and Jun Xu. Gui-g1: Understanding r1-zero-like training for visual grounding in gui agents. *Advances in Neural Information Processing Systems*, 38:95683–95705, 2026.

A Diagnostic Analysis Details

A.1 Predictions and Element Labels

The benchmark subsets described in Section 2.1 contain 1,581 ScreenSpot-Pro examples, 3,594 MMBench-GUI L2 examples, 5,479 UI-Vision examples, and 510 OSWorld-G examples. Fig. 7 shows prediction distributions for all ten evaluated models.

Parser settings. OmniParser-v2 icon detection uses confidence threshold 0.05 and intersection-over-union threshold 0.7. The detector preserves aspect ratio, using the original longer side rounded up to a multiple of 32. EasyOCR (JaidedAI, 2020) uses English recognition with text-detection threshold 0.8. We retain text and icon boxes with area at least 16 pixels and at most 35% of the screenshot area. For shared-element analysis, a click in overlapping regions is assigned to the smallest containing region, with original region order breaking ties. Element coverage is computed as the area of the union of retained regions.

Manual inspection of parsing results. Two annotators inspect the detections on 100 randomly selected screenshots. A screenshot meets the quality criterion when incorrectly detected elements account for at most 5% of all GUI elements in that screenshot. Missed elements are not counted as detection errors in this inspection, so the criterion assesses detection correctness rather than coverage. Overall, 87% of the inspected screenshots meet this criterion.

Manual verification of non-target clicks. To verify the accuracy of automated error categorization, two annotators manually reviewed 100 cases automatically identified as non-target clicks based on model predictions and parser results. Human inspection confirmed that 92% (92/100) of these cases were true non-target clicks.

Model-assisted review of distractor errors. We randomly sample 500 screenshots from examples classified as element errors in the diagnostic analysis, where a grounding model clicks outside the ground-truth target region and on another parsed GUI element. The review aims to distinguish actual selections of the wrong element from disagreements caused by incorrect ground-truth annotations or cases where both the annotated target and the model-selected element are valid choices, for example because multiple actions are acceptable or the instruction is ambiguous. For each sampled example, we provide Gemini 3.5 Flash (Google DeepMind, 2026) with the instruction, a clean screenshot, a marked full screenshot, and a marked zoomed view, using green for the annotated target and red for the model-selected element. The prompt below asks Gemini to judge which regions match the instruction: only the annotated target (A), only the model-selected element (B), both (C), or neither (D). Option A indicates an actual element-selection error, option B suggests an incorrect ground-truth annotation, and option C indicates that the model-selected element is also acceptable and should not be treated as a wrong-element selection. Option D shows that neither candidate aligns with the instructions, leaving the target undefined. Gemini selects option A in 94% of cases reviewed, indicating that most instances are due to element-selection errors rather than annotation mistakes or valid alternatives. The following prompt is provided to Gemini:

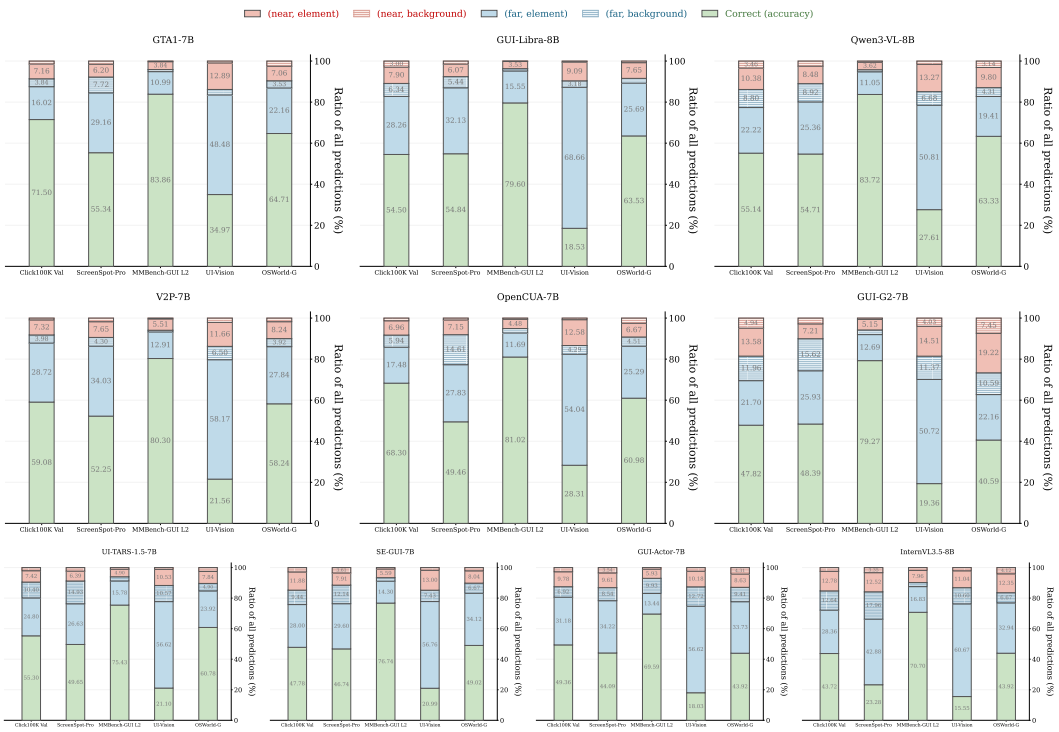


Figure 7: Prediction distributions for ten baseline models across five datasets. Each bar reports accuracy and four error rates over all predictions, with a $3w \times 3h$ neighborhood.

> You are checking a GUI grounding example.
> The two candidate targets are the green bounding box and the red bounding box.
>
> Instruction: [instruction]
>
> Which candidate bounding box corresponds to the instruction?
>
> A: only the green bounding box corresponds to the instruction.
> B: only the red bounding box corresponds to the instruction.
> C: both bounding boxes correspond to the instruction, and either click would work.
> D: neither bounding box corresponds to the instruction.
> Return JSON only as {`"answer":"A"`}.

Counting conventions. In Fig. 2(b), the denominator at threshold k counts examples with at least k valid incorrect predictions. The numerator counts examples with at least k models selecting the same parsed element, counting each example once. In Fig. 2(a), the observed element-hit proportion and random-click expectation use the same set of valid incorrect predictions for each model.

A.2 Neighborhood Density and Scale Sensitivity

We test whether the higher error density near the target persists across models, datasets, and neighborhood scales. We vary the neighborhood size $rw \times rh$ with $r \in \{2, 3, 4, 5\}$ for all ten models and five datasets.

For each example, we clip the target-centered neighborhood to the screenshot. For polygon targets, w and h are the dimensions of the enclosing box. Let a_r be the neighborhood area as a fraction of screenshot area, averaged over all examples in a dataset and including the target area. Let e_r be the fraction of valid incorrect clicks inside the neighborhood. We compute e_r for each model separately and for predictions pooled across models. The near/far error density ratio is

$$R_r = \frac{e_r / a_r}{(1 - e_r)/(1 - a_r)}. \tag{6}$$

A ratio above one indicates higher error density inside the neighborhood. Fig. 8 summarizes the results. The ratio exceeds one in all 200 model–dataset–scale combinations, with a minimum of

2.70. Thus, the higher near-target error density persists across the evaluated models, datasets, and neighborhood sizes.

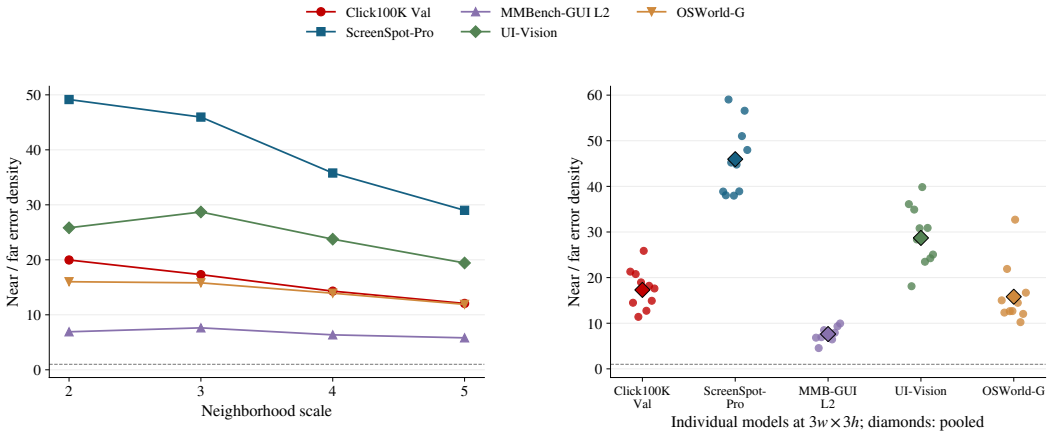


Figure 8: Neighborhood-scale sensitivity across five datasets (left) and individual-model density ratios at $r = 3$ (right). Small points represent ten models; diamonds represent pooled errors. The dashed line marks equal near and far density.

Spatial heatmaps. Fig. 9 extends the main-text heatmap to all five datasets, pooling predictions across the ten models. Offsets from the target center are normalized by target width and height, with bins of size 0.25×0.25. Colors show incorrect clicks per bin as a percentage of all predictions, using a shared logarithmic scale. The dashed box marks the $3w \times 3h$ neighborhood.

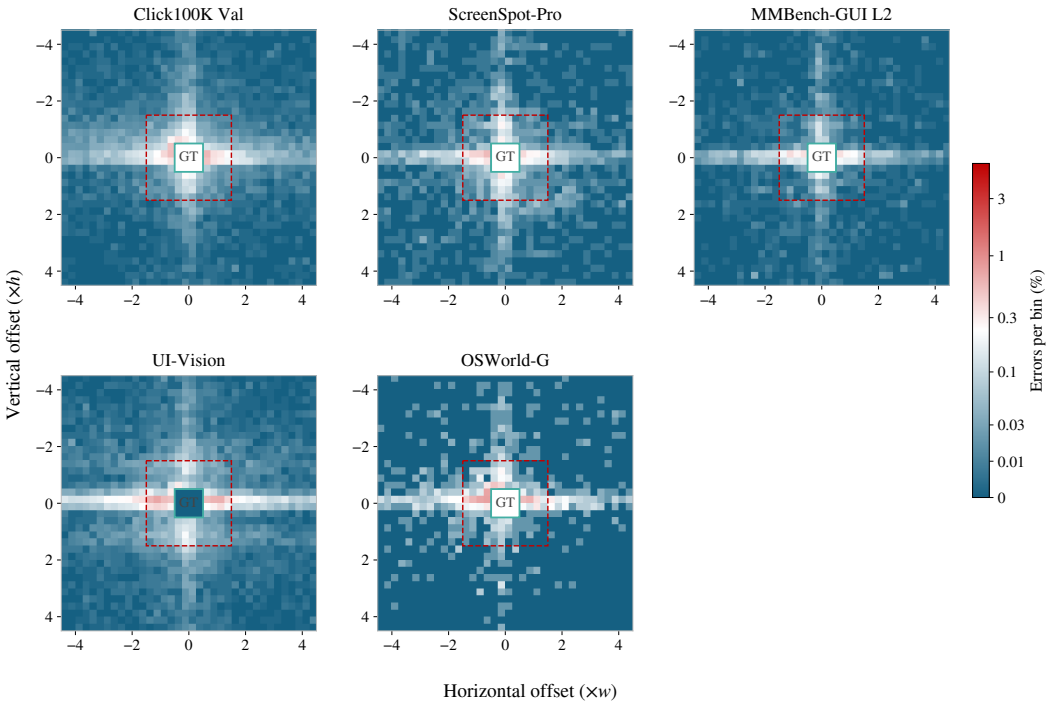


Figure 9: Pooled spatial distributions of incorrect clicks across five datasets, extending the main-text heatmap. Colors give the percentage of all predictions per bin on a shared logarithmic scale; red denotes more errors.

B ConClick Data and Training Details

Training data collection. We collect training examples from Click100K, using the original screenshots, instructions, and annotated target boxes. For each example, GTA1-7B and GUI-Libra-8B predict a click location from the screenshot and instruction. We match incorrect clicks to GUI element regions detected by OmniParser-v2. An example becomes a candidate when both models select the same non-target element, and examples with multiple candidate distractors are discarded. We pair the selected element box with the annotated target box.

Agreement on a non-target element may reflect an incorrect ground-truth annotation. We therefore use Gemini 3.5 Flash (Google DeepMind, 2026) to inspect the screenshot and instruction and assess whether the ground-truth annotation is correct. We retain only pairs for which this review confirms that the annotated target is correct and the shared model prediction is an incorrect selection. This verification step yields 4,292 paired training examples. To maintain coverage of the different Click100K subsets, we additionally sample ordinary grounding examples across the 86 source subsets using an equal per-subset sampling budget. We match the total number of ordinary examples to the 4,292 mined target–distractor pairs, giving 8,584 training examples in total. Our goal is to evaluate learning from observed distractors, not optimize data selection.

The following prompt specifies the pair-verification task, with the annotated target marked in green and the shared model-selected element marked in red:

> You are reviewing a candidate GUI grounding training pair. Given the screenshot and instruction, determine whether the green-box element is a valid target and the red-box element is an incorrect choice. The green box is the dataset annotation; the red box is the element selected by two grounding models. Neither the annotation nor model agreement guarantees correctness.
>
> Instruction: [instruction]
>
> Judge each candidate using the visible interface and the instruction, including the requested action and any constraints on location or context. Do not reject the red-box element merely because it differs from the annotation. If both elements would independently satisfy the instruction, including when multiple actions are acceptable or the wording permits both choices, classify both as valid. If the screenshot does not provide enough evidence to decide, report uncertainty rather than guessing.
>
> Choose exactly one option:
> A: The green-box element satisfies the instruction, and the red-box element does not.
> B: The red-box element satisfies the instruction, and the green-box element does not.
> C: Both elements satisfy the instruction; either choice is acceptable.
> D: Neither element satisfies the instruction.
> E: There is insufficient evidence to determine which element satisfies the instruction.
>
> Return JSON only, with an `answer` field containing A, B, C, D, or E and a `reason` field briefly explaining the decision using evidence from the screenshot and instruction.

Under this decision rule, only option A qualifies a candidate for retention as a target–distractor pair.

Random training data. We retain Click100K training examples with complete instructions, image information, and target boxes, exclude the fixed 5,000-example validation set by row ID, and uniformly sample 8,584 examples without replacement from the remaining 96,313 candidates sorted by row ID, using random seed 20260808.

Validation and evaluation. We use a held-out set of 5,000 Click100K examples, separate from the training data, for hyperparameter tuning and checkpoint selection. This is the same Click100K validation set used for the error analysis in Section 2.1. We also tune the hyperparameters of ReClick for the grounding models on this validation set. The selected checkpoint is evaluated on ScreenSpot-Pro, MMBench-GUI L2, and OSWorld-G. We omit UI-Vision from downstream evaluation because Click100K includes some UI-Vision data.

Training configuration. The following training recipe is for ConClick-8B, built on Qwen3-VL-8B. We fine-tune the instruction-tuned backbone with LoRA rank 64, scaling factor 128, and dropout 0.05. Adapters are applied to the attention and feed-forward projections, and the visual merger remains frozen. We use AdamW (Loshchilov & Hutter, 2017) with zero weight decay, gradient clipping at 1.0, and an effective batch size of 128. Table A1 lists the stage-specific settings. Refinement continues from the initial checkpoint with a new optimizer and learning-rate schedule.

Table A1: Training settings for CONCLICK-8B. Both stages use cosine learning-rate decay.

Setting	Initial training	Refinement
Learning rate	3.45×10^{-5}	1×10^{-6}
Schedule length (updates)	320	30
Warmup updates	10	10
Selected checkpoint (update)	110	10
Selection loss weight λ_{sel}	0.1	0.05

The attention readout combines decoder layers 18–23 (zero-based) and is trained jointly with the adapters at learning rate 10^{-4}. The pairwise loss uses margin $m = 0.3$ and weight $\lambda_{\text{pair}} = 0.5$. Visual tokens are assigned to regions by image-patch overlap. Tokens shared by the target and distractor masks are removed from both masks, and pairs with an empty mask contribute only coordinate loss.

Direct inference. Coordinate-only training and CONCLICK use the same prompt requesting a point $[x, y]$ on a 0–1000 scale. Images retain their aspect ratio, with a pixel budget between 3,136 and 8,847,360. Decoding is greedy with a maximum of 32 generated tokens. The pretrained Qwen3-VL-8B baseline uses its released computer-use prompt and pixel coordinates.

B.1 SENSITIVITY TO THE PAIRWISE MARGIN

We evaluate the pairwise margin $m \in \{0.1, 0.2, 0.3, 0.4, 0.5\}$ on Click100K Val. Fig. 10(a) shows accuracies between 72.18% and 73.22%, with the highest observed accuracy at $m = 0.3$.

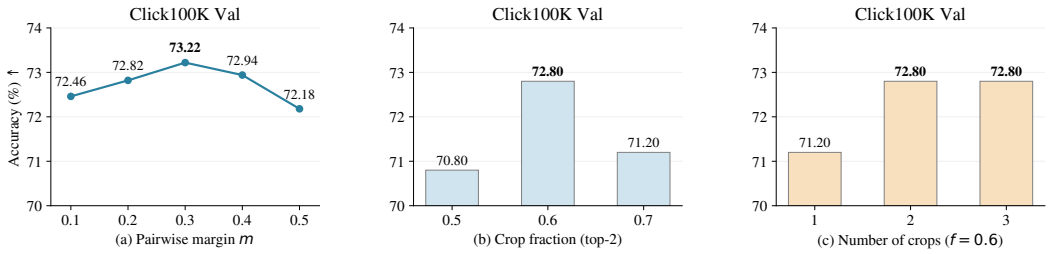


Figure 10: Sensitivity to training and inference parameters. (a) Pairwise margin on Click100K Val. (b) RECLICK crop fraction with two score-guided crops and (c) number of score-guided crops at crop fraction 0.6, evaluated on Click100K Val. Labels show accuracy (%); bold marks the highest value in each panel.

B.2 TRAINING ABLATIONS ON ADDITIONAL BACKBONES

Table A2 extends the training ablations to InternVL3.5-8B and Qwen3-VL-30B-A3B-Instruct. For each backbone, we compare coordinate-only training on random data, coordinate-only training on our data, and training with the full objective. Using our data improves accuracy on all three benchmarks for both backbones, and the full objective yields further improvements.

C RECLICK IMPLEMENTATION AND EXTERNAL GROUNDERS

Crop configuration. For screenshot dimensions W, H and crop fraction f, the crop width and height are

$$w_f = \min\big(W, \max(320, \text{round}(fW))\big), \\ h_f = \min\big(H, \max(320, \text{round}(fH))\big). \tag{7}$$

The initial-click crop uses $f = 0.4$ and is centered on the initial prediction before boundary adjustment. The four candidate grid crops use $f = 0.6$, with normalized centers at $(0.3, 0.3)$, $(0.7, 0.3)$, $(0.3, 0.7)$, and $(0.7, 0.7)$. Crops extending beyond the screenshot are shifted inside its boundaries

Table A2: Training ablations on additional backbones. All results are accuracy (%). Darker green indicates better performance; bold marks the best results within each backbone.

Data		Training objective	ScreenSpot-Pro	OSWorld-G	MMB-GUI L2	Avg.
Random	Our		Acc.↑	Acc.↑	Acc.↑	Acc.↑
InternVL3.5-8B						
✓		Coordinate-only	25.62	46.67	71.45	47.91
	✓	Coordinate-only	25.81	47.65	72.65	48.70
	✓	Full objective	**26.69**	**49.22**	**72.95**	**49.62**
Qwen3-VL-30B-A3B-Instruct						
✓		Coordinate-only	62.37	69.02	84.53	71.97
	✓	Coordinate-only	63.82	70.78	85.42	73.34
	✓	Full objective	**64.77**	**72.55**	**85.61**	**74.31**

while preserving the computed width and height. Each selected crop is enlarged by $2\times$ using Lanczos resampling. A visual token contributes to a crop score when its center lies inside the crop. Fig. 10(b,c) reports crop-parameter ablations on Click100K Val. With two score-guided crops, crop fractions 0.5, 0.6, and 0.7 yield accuracies of 70.80%, 72.80%, and 71.20%, respectively. At crop fraction 0.6, selecting one, two, or three score-guided crops yields 71.20%, 72.80%, and 72.80% accuracy, respectively.

Candidate scoring. For candidate p in view v, the coordinate log-likelihood $\ell_v(p)$ sums token log-probabilities over the native response format, excluding the end-of-sequence token. Within each view, identical serialized coordinate responses are counted once when computing the mean μ_v and population standard deviation σ_v over candidates whose locations lie in that view. The standardized log-likelihood is

$$\bar{\ell}_v(p) = \frac{\ell_v(p) - \mu_v}{\max(\sigma_v, 10^{-6})}. \tag{8}$$

The generating view participates in this normalization, but cross-view support averages only the standardized scores from the other views containing the candidate, as in Eq. (3). We then standardize cross-view support and spatial agreement separately across the remaining candidate set $\mathcal{P}'$ for the current example. For either score $F \in \{S_{\text{cross}}, S_{\text{agree}}\}$, this gives

$$\hat{F}(p) = \frac{F(p) - \mu_F}{\max(\sigma_F, 10^{-6})}, \tag{9}$$

where μ_F and σ_F are the mean and population standard deviation of F over $\mathcal{P}'$. Both stages floor the standard deviation at 10^{-6}. The final comparison maximizes $\hat{S}_{\text{cross}}(p) + \hat{S}_{\text{agree}}(p)$, as in Eq. (5). Ties are resolved in the order of the full-image prediction, the initial-click crop, and the score-guided crops ranked by region score.

Inference baselines. On ConClick-8B, MVP uses attention from the coordinate-comma token at decoder layer 24 and makes up to two crop predictions. It uses the same image budget as direct inference. ZoomClick uses search depth 3, crop ratio 0.5, minimum crop size 768 pixels, a 2×2 initial grid, and a 50-pixel stopping threshold. Crop centers are adjusted to keep crops inside the screenshot.

C.1 Inference Comparisons and Ablations

Tables A3 and A4 provide the complete three-benchmark results corresponding to Fig. 6(b) and (c), respectively.

C.2 External Point Grounders

Attention readouts. Table A5 summarizes the attention readout used by ReClick for each model. All three use the last prompt token as the query. For ConClick-8B, learned weights combine pre-softmax query–key scores across all 32 heads in layers 18–23. For GTA1-7B and UI-TARS-1.5-7B, we obtain crop scores from frozen model attention. GTA1 takes the maximum attention probability across all heads in layers 12–25. These probabilities are normalized over all keys, restricted to

Table A3: Comparison of inference methods across three backbones and benchmarks. Acc. denotes accuracy and E-err. denotes the element error rate, both in percentages. Within each backbone, **blue bold** and **black bold** mark the best and second-best results, respectively.

		ScreenSpot-Pro		OSWorld-G		MMB-GUI L2	
Backbone	Method	Acc.↑	E-err.↓	Acc.↑	E-err.↓	Acc.↑	E-err.↓
UI-TARS-1.5-7B	MVP	**59.84**	**32.51**	**60.78**	**30.98**	**76.27**	**20.17**
	ZoomClick	57.24	35.48	57.25	37.06	73.57	21.87
	ReClick	**61.23**	**30.55**	**65.88**	**30.59**	**77.74**	**19.59**
GTA1-7B	MVP	**64.39**	**32.38**	**69.02**	**26.47**	**83.70**	**15.41**
	ZoomClick	59.33	37.38	65.49	28.43	80.75	17.58
	ReClick	**65.21**	**30.30**	**70.59**	**25.10**	**84.67**	**14.50**
ConClick-8B	MVP	**69.96**	**27.39**	**70.78**	**24.90**	**86.39**	**12.94**
	ZoomClick	66.92	29.60	65.88	28.04	83.95	15.14
	ReClick	**72.36**	**25.87**	**71.76**	**23.73**	**87.28**	**12.10**

Table A4: Component ablations of ReClick. Acc. and E-err. are percentages over all predictions. Avg. is the unweighted mean across the three benchmarks. All variants use the same ConClick-8B checkpoint. Bold marks the best value in each column.

	ScreenSpot-Pro		OSWorld-G		MMB-GUI L2		Avg.	
Variant	Acc.↑	E-err.↓	Acc.↑	E-err.↓	Acc.↑	E-err.↓	Acc.↑	E-err.↓
ReClick	**72.36**	25.87	**71.76**	**23.73**	87.28	12.10	**77.13**	**20.57**
w/o score-guided crops	66.29	30.42	69.80	25.10	86.28	13.05	74.12	22.86
w/o learned crop selection	72.17	**25.68**	70.59	24.90	87.26	12.27	76.67	20.95
w/o S_{cross}	70.97	26.31	71.57	23.92	87.15	12.19	76.56	20.81
w/o S_{agree}	72.23	26.12	71.18	24.51	**87.48**	**12.08**	76.96	20.90

image tokens, and converted to log space before region pooling. UI-TARS uses pre-softmax query–key scores from head 25 in layer 24. Layer and head indices are zero-based. The readout for each model is selected by accuracy on the held-out Click100K validation set and fixed across all three benchmarks. Crop construction and candidate scoring follow the procedure above.

Table A5: Model-specific attention readouts for ReClick. QK denotes pre-softmax query–key scores. Layer and head indices are zero-based. External readouts use temperature 1. Each configuration is shared across all three evaluation benchmarks.

Model	Layers	Heads	Attention scores	Aggregation
ConClick-8B	18–23	All 32	QK	Learned weighted sum
GTA1-7B	12–25	All	Probabilities	Maximum
UI-TARS-1.5-7B	24	25	QK	Single head

Shared parameter	Value
Initial-click crop fraction	0.4
Grid crop fraction	0.6
Number of score-guided crops	2
Minimum crop dimension	320 pixels
Crop enlargement factor	2×
Cross-view support coefficient	1
Spatial agreement coefficient	1

Model interfaces and baselines. For UI-TARS, all methods use the released grounding prompt, a 12,845,056-pixel image budget, and greedy decoding, sharing the same initial prediction. GTA1 uses method-specific point-model interfaces, so its initial predictions can differ between methods. Candidate likelihoods use coordinates mapped to each resized view and the native response format of each model. MVP uses coordinate-comma attention at layer 20 and up to four crop predictions for these external models. Its ScreenSpot-Pro procedure is also used for MMBench-GUI L2, while OSWorld-G uses the corresponding benchmark-specific procedure. The UI-TARS implementation adapts MVP attention extraction to the shared model interface. ZoomClick uses the search settings

specified above with a point-model adapter. All methods use the same accuracy and element-error evaluation, with failed predictions included in the denominator.

D Detailed Benchmark Results

Table A6 reports accuracy and element error rates by domain on ScreenSpot-Pro, and Table A7 reports subset accuracies on MMBench-GUI L2. Both use the same locally evaluated models and predictions as Table 1.

Table A6: Performance by domain on ScreenSpot-Pro (%). Acc. is point accuracy; E-err. is the percentage of all predictions in each domain that fall outside the target region but inside a parsed GUI element. Each domain pools Text and Icon targets. Overall pools all examples and matches Table 1; all results use the same locally evaluated models and predictions. † retains the Table 1 marker on overall accuracy. **Blue bold** and **black bold** mark the best and second-best values in each column, respectively; ties receive the same marking.

Method	CAD		Dev		Creative		Scientific		Office		OS		Overall	
	Acc.↑	E-err.↓	Acc.↑	E-err.↓	Acc.↑	E-err.↓	Acc.↑	E-err.↓	Acc.↑	E-err.↓	Acc.↑	E-err.↓	Acc.↑	E-err.↓
GUI-Actor-7B	31.03	52.11	39.13	46.49	43.70	43.99	48.03	46.46	60.87	28.26	44.90	43.37	44.09†	43.83
SE-GUI-7B	41.76	41.38	43.81	42.81	37.83	39.88	54.72	34.65	70.43	22.17	35.20	41.84	46.74†	37.51
GUI-G2-7B	48.28	37.55	46.49	38.13	40.47	**36.36**	53.54	31.10	68.70	20.00	34.69	32.14	48.39†	33.14
OpenCUA-7B	32.95	39.85	44.82	41.14	45.16	42.23	58.66	**29.53**	71.74	22.17	47.96	**28.57**	49.46†	34.98
UI-TARS-1.5-7B	45.98	32.57	46.15	40.47	44.28	36.95	55.12	32.68	69.57	20.00	38.78	31.12	49.65†	33.02
V2P-7B	49.81	45.21	46.49	44.48	43.11	51.61	57.48	37.40	73.48	23.48	48.47	42.35	52.25†	41.68
GUI-Libra-8B	49.81	44.44	51.17	43.14	51.91	40.47	55.51	39.76	72.17	19.57	51.02	38.27	54.84†	38.20
GTA1-7B	50.19	44.83	51.84	37.79	49.85	38.12	55.51	39.37	78.26	18.26	50.00	**29.08**	55.34†	35.36
Qwen3-VL-8B (baseline)	49.43	39.08	52.51	36.12	50.44	38.12	55.51	33.46	72.61	16.96	50.51	36.22	54.71	33.84
ConClick-8B (Ours)	**65.13**	**29.89**	**63.88**	**32.44**	**56.01**	38.71	**66.14**	30.71	**85.65**	**12.61**	**64.80**	33.16	**66.03**	**30.30**
ConClick-8B + ReClick (Ours)	**71.26**	**26.44**	**69.57**	**28.76**	**65.40**	**31.38**	**72.44**	**26.77**	**89.57**	**10.00**	**69.90**	**28.57**	**72.36**	**25.87**

Table A7: Subset accuracy on MMBench-GUI L2 (%). All results use the locally evaluated models and predictions from Table 1. Each platform is divided into Basic and Advanced (Adv.) instructions. Overall pools examples across all subsets; All matches the corresponding accuracy in Table 1. **Blue bold** and **black bold** mark the best and second-best values in each column, respectively; ties receive the same marking.

Method	Windows		macOS		Linux		iOS		Android		Web		Overall		
	Basic	Adv.	Basic	Adv.	Basic	Adv.	Basic	Adv.	Basic	Adv.	Basic	Adv.	Basic	Adv.	All
GUI-Actor-7B	61.99	55.51	59.71	59.83	49.74	41.84	90.45	82.73	70.79	79.72	89.68	72.08	71.80	67.40	69.59
SE-GUI-7B	79.34	56.25	75.07	58.67	65.45	44.39	95.22	83.64	94.66	85.35	90.97	71.10	84.89	68.68	76.74
GUI-G2-7B	80.44	56.99	81.45	66.76	68.06	48.47	95.86	82.42	95.79	84.51	92.90	76.95	87.24	71.39	79.27
OpenCUA-7B	90.77	62.13	84.06	65.03	81.68	58.16	95.22	80.00	95.22	79.72	94.19	76.30	90.77	71.39	81.02
UI-TARS-1.5-7B	76.75	56.62	73.62	60.98	70.16	51.53	90.45	80.00	92.42	80.00	85.48	72.40	82.48	68.46	75.43
V2P-7B	84.50	59.19	83.77	69.36	71.73	57.14	94.27	85.15	92.98	83.10	91.94	74.68	87.69	72.99	80.30
GUI-Libra-8B	86.72	63.60	86.38	70.23	64.92	48.98	91.08	80.30	94.38	83.10	91.61	73.38	87.47	71.83	79.60
GTA1-7B	89.67	68.01	**88.99**	71.39	79.58	60.71	95.86	82.42	**96.91**	84.79	94.84	**80.52**	91.89	75.93	83.86
Qwen3-VL-8B (baseline)	87.82	67.65	86.96	68.79	75.39	61.22	**96.82**	**86.36**	96.63	86.20	**96.45**	80.19	91.16	76.37	83.72
ConClick-8B (Ours)	**91.51**	**69.85**	86.96	**78.61**	**82.20**	**66.84**	**97.13**	**88.79**	**97.47**	**90.70**	**95.81**	77.60	**92.56**	**80.08**	**86.28**
ConClick-8B + ReClick (Ours)	**93.73**	**75.37**	**90.72**	**80.92**	**82.72**	**71.94**	96.50	86.06	95.51	**87.89**	**95.81**	**81.17**	**93.17**	**81.46**	**87.28**