When GUI Grounding Fails: Entropy-Based Analysis and Training-Free Refinement

Chengxin Liu [1] **Moon Ye-Bin** [2] **Tae-Hyun Oh** [1]

Abstract

Graphical User Interface (GUI) agents have recently emerged as powerful tools for enabling automated operation across diverse platforms, with the potential to alleviate human workload. Despite promising progress, GUI grounding remains a critical bottleneck for achieving precise interaction, as incorrect localization of UI elements can lead to task failure. Therefore, it is crucial to identify when the model is likely to fail. In this work, we demonstrate that the entropy of output tokens is strongly correlated with model failures. Based on this observation, we propose an entropy-aware, training-free method to improve GUI grounding performance. Experiments on two GUI grounding benchmarks show that the proposed method achieves consistent improvements over state-of-the-art models.

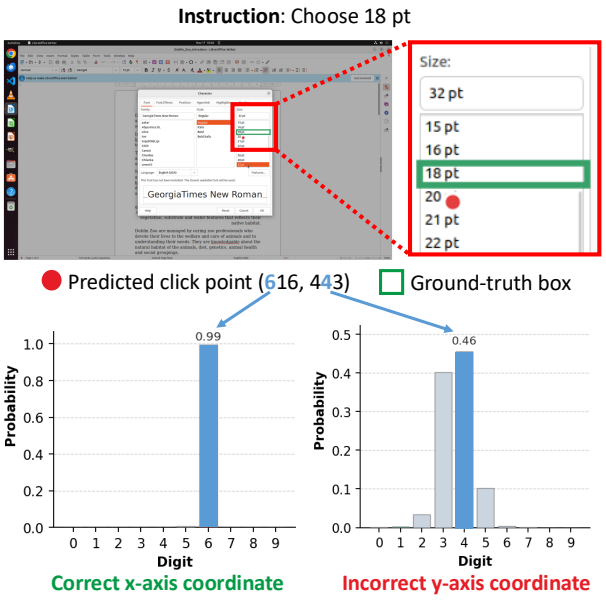


Figure 1. Model failures are often associated with numeric uncertainty in GUI grounding. The digit-level probabilities of the y-axis coordinate show high uncertainty, leading to incorrect grounding.

1. Introduction

Recent advances in multimodal large language models (MLLMs) have enabled GUI agents to operate autonomously across various real-world environments, such as the web (Deng et al., 2023; Zhou et al., 2024), desktops (Xie et al., 2024), and mobile devices (Rawles et al., 2025). The emergence of such agents substantially alleviates human workload and facilitates more efficient workflows.

While recent GUI agents have demonstrated promising performance across a wide range of platforms, GUI grounding (Cheng et al., 2024), i.e., accurately localizing UI elements corresponding to a given instruction, remains a major bottleneck for precise interaction. In practice, incorrect grounding will result in an agent being stuck, ultimately leading to task failure. Therefore, it is important to identify when the model is likely to fail. Early detection of such failures could enhance the reliability and practicality of GUI agents in real-world applications. However, it remains challenging to predict model failures.

Interestingly, recent studies (Wang et al., 2025a; Fu et al., 2026; Xu et al., 2026b) suggest that entropy is an effective indicator of model uncertainty. Inspired by this line of work, we use token entropy as a metric to characterize model outputs. As illustrated in Fig. 1, we observe that model failures in GUI grounding are often accompanied by numeric uncertainty. Building on this observation, we further show a strong correlation between output token entropy and model failures (Fig. 2). This provides a new perspective on identifying potential failures and improving GUI grounding.

Based on our analysis, we propose an entropy-aware inference strategy to improve accuracy. The key idea is to trigger probability-based point prediction when the model output is uncertain. By detecting high-uncertainty predictions and refining them accordingly, the proposed method improves GUI grounding accuracy over state-of-the-art models without additional training.

[1]KAIST [2]POSTECH. Correspondence to: Chengxin Liu <cxliu@kaist.ac.kr>.

Accepted to the 2nd Workshop on Compositional Learning at ICML 2026, Seoul, South Korea. Copyright 2026 by the author(s).

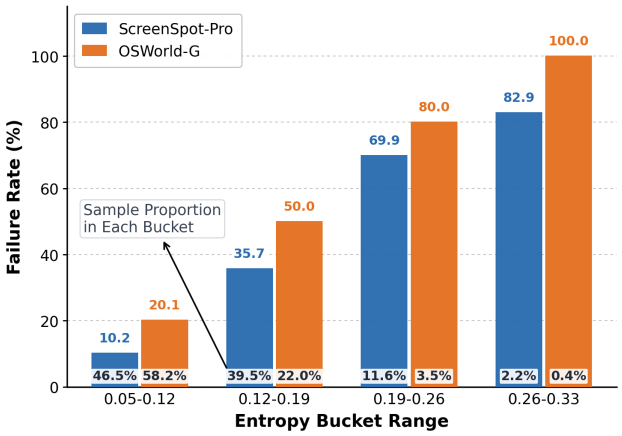


Figure 2. The correlation between output token entropy and failure rate for GUI-Owl-1.5 (Xu et al., 2026a) on OSWorld-G (Xie et al., 2024) and ScreenSpot-Pro (Li et al., 2025). The failure rate increases monotonically with entropy across both datasets, indicating that output token entropy is strongly correlated with GUI grounding failure.

The contributions of this paper are summarized as follows:

- We show that output token entropy is strongly correlated with model failure in GUI grounding.
- We propose a training-free, entropy-aware method that improves the performance of state-of-the-art models on GUI grounding benchmarks.

2. Related Work

GUI agents. Existing GUI agents can be broadly categorized into naive GUI agents (Tang et al., 2025; Wu et al., 2025; Xie et al., 2024; Qin et al., 2025; Anthropic, 2025; Xu et al., 2025) and two-stage GUI agents (Wang et al., 2024; Cheng et al., 2024; Zhang et al., 2025b; Gou et al., 2025; Zhang et al., 2025a). The former handle user tasks in an end-to-end manner, whereas the latter decompose tasks into separate planning and action stages. Benefiting from the recent advances in MLLMs, native GUI agents often achieve substantially higher accuracy than their two-stage counterparts, thereby attracting extensive attention from the community. Nevertheless, GUI grounding remains a critical performance bottleneck. Recent work (Zhou et al., 2025b) suggests that techniques such as chain-of-thought reasoning (Wei et al., 2022) do not necessarily improve GUI grounding. In this work, we demonstrate that failures in GUI grounding are strongly correlated with output token entropy, offering a new perspective for improving GUI agents.

3. Entropy-Based Analysis

3.1. Token Entropy as Uncertainty Measurement

Preliminary. Given a GUI screenshot and text instruction as input, existing native GUI agents (Xu et al., 2026a; Lu et al., 2026), denoted as M, predict actions and coordinates via next-token prediction. The probability of the generated text token at time step t is written as follows:

$$p_t = M(\cdot \mid \mathbf{x}, y_{<t}) \in \Delta^{|\mathcal{V}|-1}, \tag{1}$$

where $\mathbf{x}$ is the multimodal input, $y_{<t} = (y_1, y_2, \ldots, y_{t-1})$ represents all previously generated text tokens, $\mathcal{V}$ is the vocabulary, and $\Delta^{|\mathcal{V}|-1}$ denotes the $(|\mathcal{V}|-1)$-dimensional probability simplex.

Uncertainty measurement. We adopt token-level entropy to quantify model uncertainty at time step t. The entropy is defined as:

$$\mathrm{Ent}_t = -\sum_v p_t[v] \log p_t[v], \tag{2}$$

where $p_t[v]$ denotes the probability of token v in the vocabulary $\mathcal{V}$. A high entropy value indicates that several tokens have similar probabilities, reflecting greater uncertainty and the presence of multiple possible reasoning paths. In contrast, a low value of Ent_t occurs when a single token dominates the probability mass, indicating that the model converges to a deterministic prediction.

3.2. Token Entropy Correlates with Failures

We investigate the correlation between token entropy and model failures on GUI grounding benchmarks.

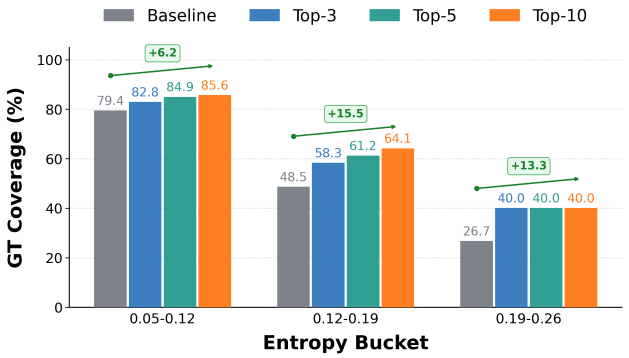
Experimental setup. For each image-text pair, we compute Ent_t for all generated tokens and average them to obtain a single entropy score for the sample. We conduct the analysis using a state-of-the-art method GUI-Owl-1.5 (Xu et al., 2026a) on two GUI grounding datasets, namely OSWorld-G (Xie et al., 2024) and ScreenSpot-Pro (Li et al., 2025).

Results. We first partition the samples into groups based on their entropy values, and then compute the failure rate for each group. As shown in Fig. 2, one can observe that: i) output token entropy is strongly correlated with failure rate; ii) the positive correlation consistently holds across both benchmarks.

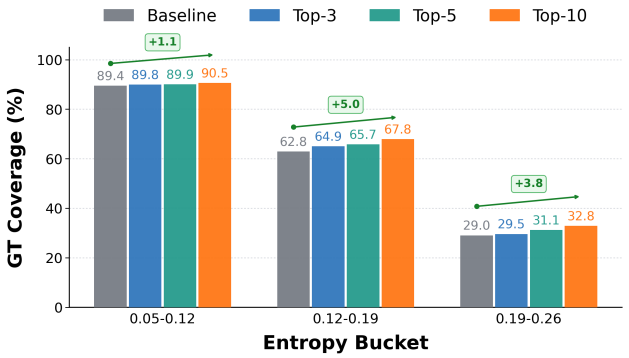
Insights. The strong and consistent positive correlation between output token entropy and failure rate suggests that entropy can serve as an effective proxy for model uncertainty in GUI grounding. This offers a new perspective to identify model failures without access to ground truth, enabling adaptive strategies during inference.

3.3. Ground-Truth Coverage Analysis

We further explore the model's potential capability to predict correct grounding points.



(a) OSWorld-G



(b) ScreenSpot-Pro

Figure 3. GT coverage of the top-K candidate points across different entropy buckets, where $K = 1$ (baseline), 3, 5, and 10. Coverage consistently improves as the number of candidate points increases across both datasets.

Experimental setup. We extract logit probabilities from model predictions and select the top-K candidate points to evaluate ground-truth (GT) coverage.

Specifically, for a predicted 3-digit coordinate (as shown in Fig. 1), we decompose the value into three token-level distributions corresponding to the hundreds, tens, and ones digits. The probability of a candidate value is computed as: $P(\text{value}) = P(\text{hundreds}) \times P(\text{tens}) \times P(\text{ones})$. Based on these distributions, multiple candidate values are sampled according to their joint probabilities. An example is shown in Appendix A. For each image, the prediction is considered correct if at least one of the top-K candidates falls within the GT bounding box. This metric reflects the model's ability to predict spatially aligned points, even when the baseline prediction may be incorrect.

Results. As shown in Fig. 3, the GT coverage rate consistently improves as the number of candidate points increases. The gain is particularly noticeable when expanding from the baseline to larger K values, indicating that the model often ranks correct candidate points close to the top, even if they are not selected as the final prediction. We also show an example of GT coverage in Fig. 4. Although the top-1 prediction (i.e., baseline) fails, the correct target may still be captured within the top-K candidates.

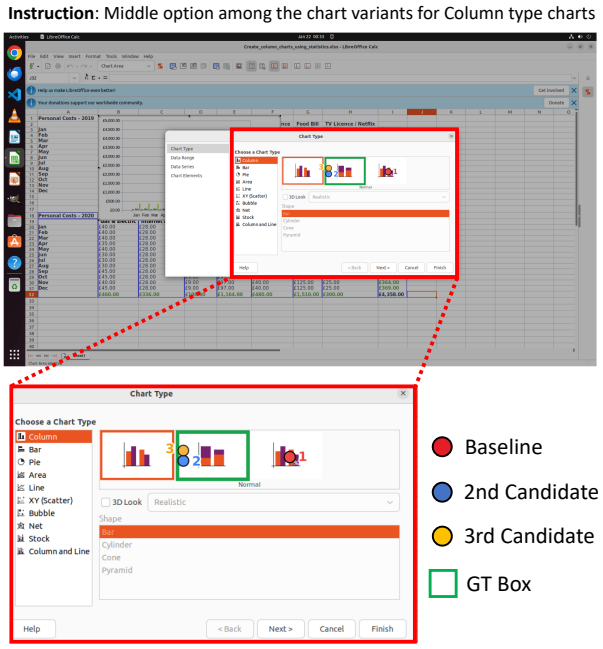


Figure 4. An example of GT coverage with top-3 candidate points. Although the top-1 (baseline) prediction is incorrect, other high-ranked candidate points correctly fall within the GT region.

Insights. This observation suggests that prediction errors are not solely due to a lack of relevant visual grounding, but may also arise from suboptimal selection among high-probability candidate points.

4. Entropy-Aware GUI Grounding

Method overview. Inspired by our findings, we propose an entropy-aware inference strategy to improve GUI grounding. As illustrated in Fig. 5, given an input image and a textual instruction, the GUI agent first predicts the corresponding action and coordinate. If the output token entropy exceeds a predefined threshold θ, probability-based point prediction is applied to refine the results based on a set of high-probability candidate points. Otherwise, the original prediction is retained.

Probability-based point prediction. Following the analysis in Sec. 3.3, we leverage the candidate set to make a more robust prediction. Specifically, we enumerate the top-K coordinate candidates by estimating their joint probabilities, forming a ranked candidate pool. These candidate points are then grouped using K-means. We select the most confident cluster according to the aggregated probabilities of points within the cluster, and compute its centroid as the final prediction. This strategy enables more reliable grounding than solely relying on the top-1 output.

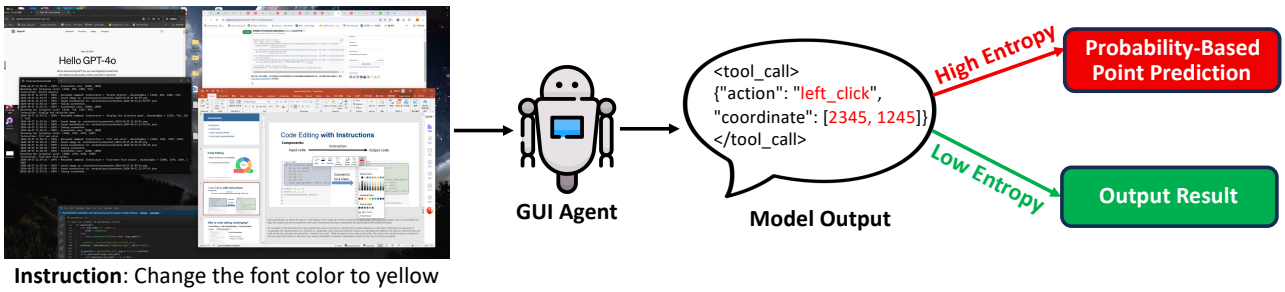


Figure 5. Given an input image and instruction, the GUI agent outputs the action and coordinate. If the output token entropy exceeds a predefined threshold, probability-based point prediction is applied to refine the result. Otherwise, the original prediction is retained.

Table 1. Comparison with state-of-the-art methods on OSWorld-G and OSWorld-G-Refine datasets. The best result is highlighted in **bold**, while the second-best result is underlined.

Agent Model	OSG	OSG-R
General Models		
Operator (OpenAI, 2025)	40.6	57.8
Qwen3-VL-8B-Instruct (Bai et al., 2025)	54.8	64.4
Qwen3-VL-32B-Instruct (Bai et al., 2025)	65.1	69.0
GUI Models		
UGround-7B (Gou et al., 2025)	36.4	-
UI-TARS-1.5-7B (Qin et al., 2025)	52.8	64.2
JEDI-7B (Xie et al., 2024)	54.1	63.8
GTA1-7B (Yang et al., 2026)	55.1	67.7
MAI-UI-8B (Zhou et al., 2025a)	60.1	-
MAI-UI-32B (Zhou et al., 2025a)	67.6	-
OpenCUA-32B (Wang et al., 2025b)	59.6	70.2
Ours		
GUI-Owl-1.5-8B-Instruct (Xu et al., 2026a)	65.8	69.3
+ Entropy-Aware Decoding (Ours)	**68.1**	**70.7**

Table 2. Comparison on the ScreenSpot-Pro dataset.

Agent Model	Acc
General Models	
Claude 3.7 Sonnet (Anthropic, 2025)	27.7
Operator (OpenAI, 2025)	36.6
Qwen3-VL-8B-Instruct (Bai et al., 2025)	54.6
Qwen3-VL-32B-Instruct (Bai et al., 2025)	57.9
GUI Models	
JEDI-7B (Xie et al., 2024)	39.5
GUI-G^2-7B (Tang et al., 2026)	47.5
GTA1-New-7B (Yang et al., 2026)	55.5
MAI-UI-8B (Zhou et al., 2025a)	65.8
EvoCUA-8B (Xue et al., 2026)	45.4
UI-TARS-72B (Qin et al., 2025)	38.1
UGround-v1-72B (Gou et al., 2025)	34.5
GTA1-32B (Yang et al., 2026)	53.6
MAI-UI-32B (Zhou et al., 2025a)	67.9
EvoCUA-32B (Xue et al., 2026)	49.7
UI-TARS-1.5 (Qin et al., 2025)	61.6
Ours	
GUI-Owl-1.5-8B-Instruct (Xu et al., 2026a)	71.1
+ Entropy-Aware Decoding (Ours)	**71.4**

5. Experiment

5.1. Datasets and Implementation Details

Datasets and evaluation metrics. We evaluate our method on OSWorld-G (Xie et al., 2024), OSWorld-G-Refine (Xie et al., 2024), and ScreenSpot-Pro (Li et al., 2025) datasets. OSWorld-G-Refine shares the same images as OSWorld-G, but uses refined instructions. For simplicity, we denote OSWorld-G and OSWorld-G-Refine as OSG and OSG-R, respectively. Following prior work (Xie et al., 2024; Li et al., 2025), we use the grounding accuracy as the evaluation metric.

Implementation details. We use the top-10 candidate points for probability-based point prediction and set the number of clusters in K-means to 2. The entropy threshold θ is set to 0.05. Please refer to Appendix A for more details.

5.2. Quantitative Results

OSWorld-G. Table 1 shows the comparisons on the OSWorld-G and OS-World-G-Refine datasets. Our entropy-aware decoding consistently improves the baseline from 65.8 to 68.1 on OSG and from 69.3 to 70.7 on OSG-R, achieving the best overall performance. These results validate the effectiveness of the proposed entropy-aware strategy. We also show the qualitative results in Appendix B.

ScreenSpot-Pro. Table 2 reports the results on the ScreenSpot-Pro dataset. Our method improves the performance of baseline from 71.1 to 71.4. Although the gain is modest, the result demonstrates that the proposed method could enhance GUI grounding without additional training.

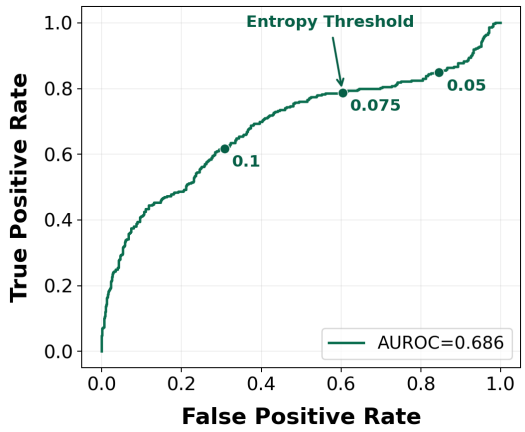


Figure 6. ROC curve for detecting GUI grounding failures using output token entropy on OSWorld-G.

Table 3. Ablation study on the sensitivity of entropy threshold.

Threshold θ	0.05	0.075	0.1	0.125
OSWorld-G	68.1	68.1	67.9	67.6

Table 4. Ablation study on top-K candidates.

Top-K	4	6	10	14
OSWorld-G	67.7	68.1	68.1	67.9

5.3. Ablation Study

Quantitative uncertainty analysis. Fig. 6 shows the ROC curve for using output token entropy to detect GUI grounding failures. The AUROC of 0.686 indicates that entropy provides useful information for distinguishing correct predictions from failures. Although entropy is not a perfect predictor, higher-entropy predictions are generally more likely to be incorrect, suggesting that entropy serves as a practical uncertainty signal for identifying high-risk cases and guiding selective refinement.

The sensitivity of entropy threshold. Table 3 shows that the proposed method is relatively robust to the choice of entropy threshold, with accuracy varying from 67.6% to 68.1% across thresholds $\{0.05, 0.075, 0.1, 0.125\}$. In particular, thresholds of 0.05 and 0.075 achieve the best performance. Larger thresholds result in only a marginal drop in accuracy and still consistently outperform the baseline.

Impact of top-K candidates. Table 4 shows the effect of varying the number of top-K candidates. The performance first improves and then slightly declines as K increases, suggesting that a moderate value of K is preferable. Intuitively, a larger K may introduce lower-quality candidates, which could affect the refinement process.

Table 5. Comparisons with different refinement methods.

Refinement Method	OSWorld-G
Mean Top-K Points	66.5
Uniformly Weighted K-means	67.6
Probability-Weighted K-means (Ours)	**68.1**

Table 6. Result on OSWorld-G with Qwen3-VL.

Method	OSWorld-G
Qwen3-VL-8B-Instruct	59.4
+ Entropy-Aware Decoding (Ours)	**60.5**

Comparison with different refinement methods. Table 5 compares different refinement strategies. One can observe that: i) averaging the entropy-triggered top-K candidate points achieves a decent accuracy of 66.5; ii) applying uniformly weighted K-means clustering further boosts performance to 67.6%; iii) the proposed entropy-triggered, probability-weighted K-means achieves the best result of 68.1%. This demonstrates the effectiveness of both spatial clustering and probability-aware candidate aggregation.

Generalization to other model. Table 6 evaluates the proposed method on Qwen3-VL (Bai et al., 2025). Our method improves the grounding accuracy from 59.4% to 60.5%. This improvement demonstrates that the proposed method can generalize to other model.

Efficiency. We measure the inference cost of GUI-Owl-1.5 on OSWorld-G using an NVIDIA A100 GPU. A single model inference takes approximately 1.6 seconds, whereas our probability-based point prediction takes less than 3 milliseconds, making the cost negligible.

6. Conclusion

We have demonstrated that output token entropy is strongly correlated with model failures in GUI grounding. Experiments on two GUI grounding benchmarks show that entropy-aware inference can improve the performance of state-of-the-art models. We hope our findings inspire further research on enhancing the reliability of GUI agents.

Acknowledgement

This work was supported by Samsung Electronics Co., Ltd (Project Code: IO260114-15161-01), the InnoCORE program of the Ministry of Science and ICT (N10250156, KAIST InnoCore LLM) (50%), and Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government(MSIT) (No. RS-2024-00457882, National AI Research Lab Project) (50%).

References

Anthropic. Claude 3.7 sonnet. https://www.anthropic.com/news/claude-3-7-sonnet, 2025.

Bai, S., Cai, Y., Chen, R., Chen, K., Chen, X., Cheng, Z., Deng, L., Ding, W., Gao, C., Ge, C., Ge, W., Guo, Z., Huang, Q., Huang, J., Huang, F., Hui, B., Jiang, S., Li, Z., Li, M., Li, M., Li, K., Lin, Z., Lin, J., Liu, X., Liu, J., Liu, C., Liu, Y., Liu, D., Liu, S., Lu, D., Luo, R., Lv, C., Men, R., Meng, L., Ren, X., Ren, X., Song, S., Sun, Y., Tang, J., Tu, J., Wan, J., Wang, P., Wang, P., Wang, Q., Wang, Y., Xie, T., Xu, Y., Xu, H., Xu, J., Yang, Z., Yang, M., Yang, J., Yang, A., Yu, B., Zhang, F., Zhang, H., Zhang, X., Zheng, B., Zhong, H., Zhou, J., Zhou, F., Zhou, J., Zhu, Y., and Zhu, K. Qwen3-vl technical report. *arXiv preprint arXiv: 2511.21631*, 2025.

Cheng, K., Sun, Q., Chu, Y., Xu, F., YanTao, L., Zhang, J., and Wu, Z. SeeClick: Harnessing GUI grounding for advanced visual GUI agents. In *ACL*, pp. 9313–9332, 2024.

Deng, X., Gu, Y., Zheng, B., Chen, S., Stevens, S., Wang, B., Sun, H., and Su, Y. Mind2web: towards a generalist agent for the web. In *NeurIPS*, pp. 28091 – 28114, 2023.

Fu, Y., Wang, X., Zhang, H., Tian, Y., and Zhao, J. Deep think with confidence. In *ICLR*, 2026.

Gou, B., Wang, R., Zheng, B., Xie, Y., Chang, C., Shu, Y., Sun, H., and Su, Y. Navigating the digital world as humans do: Universal visual grounding for GUI agents. In *ICLR*, 2025.

Li, K., Meng, Z., Lin, H., Luo, Z., Tian, Y., Ma, J., Huang, Z., and Chua, T.-S. Screenspot-pro: Gui grounding for professional high-resolution computer use. In *ACM MM*, pp. 8778–8786, 2025.

Lu, Z., Chai, Y., Guo, Y., Yin, X., Liu, L., Wang, H., Xiao, H., Ren, S., Zhao, P., Liu, G., Xiong, G., and Li, H. Ui-r1: Enhancing efficient action prediction of gui agents by reinforcement learning. *AAAI*, pp. 17608–17616, 2026.

OpenAI. Computer-using agent: Introducing a universal interface for ai to interact with the digital world. https://openai.com/index/computer-using-agent, 2025.

Qin, Y., Ye, Y., Fang, J., Wang, H., Liang, S., Tian, S., Zhang, J., Li, J., Li, Y., Huang, S., Zhong, W., Li, K., Yang, J., Miao, Y., Lin, W., Liu, L., Jiang, X., Ma, Q., Li, J., Xiao, X., Cai, K., Li, C., Zheng, Y., Jin, C., Li, C., Zhou, X., Wang, M., Chen, H., Li, Z., Yang, H., Liu, H., Lin, F., Peng, T., Liu, X., and Shi, G. Ui-tars: Pioneering automated gui interaction with native agents. *arXiv preprint arXiv: 2501.12326*, 2025.

Rawles, C., Clinckemaillie, S., Chang, Y., Waltz, J., Lau, G., Fair, M., Li, A., Bishop, W. E., Li, W., Campbell-Ajala, F., Toyama, D. K., Berry, R. J., Tyamagundlu, D., Lillicrap, T. P., and Riva, O. Androidworld: A dynamic benchmarking environment for autonomous agents. In *ICLR*, 2025.

Tang, F., Shen, Y., Zhang, H., Chen, S., Hou, G., Zhang, W., Zhang, W., Song, K., Lu, W., and Zhuang, Y. Think twice, click once: Enhancing gui grounding via fast and slow systems. *arXiv preprint arXiv: 2503.06470*, 2025.

Tang, F., Gu, Z., Lu, Z., Liu, X., Shen, S., Meng, C., Wang, W., Zhang, W., Shen, Y., Lu, W., Xiao, J., and Zhuang, Y. GUI-G²: Gaussian reward modeling for GUI grounding. In *AAAI*, pp. 33214–33222, 2026.

Wang, J., Xu, H., Ye, J., Yan, M., Shen, W., Zhang, J., Huang, F., and Sang, J. Mobile-agent: Autonomous multi-modal mobile device agent with visual perception. In *ICLR Workshop*, 2024.

Wang, S., Yu, L., Gao, C., Zheng, C., Liu, S., Lu, R., Dang, K., Chen, X.-H., Yang, J., Zhang, Z., Liu, Y., Yang, A., Zhao, A., Yue, Y., Song, S., Yu, B., Huang, G., and Lin, J. Beyond the 80/20 rule: High-entropy minority tokens drive effective reinforcement learning for LLM reasoning. In *NeurIPS*, 2025a.

Wang, X., Wang, B., Lu, D., Yang, J., Xie, T., Wang, J., Deng, J., Guo, X., Xu, Y., Wu, C. H., Shen, Z., Li, Z., Li, R., Li, X., Chen, J., Boyuan, Z., PEIHANG, L., Lei, F., Cao, R., Fu, Y., Shin, D., Shin, M., Jiarui, H., Wang, Y., Chen, J., Ye, Y., Zhang, D., Wang, Y., Wang, H., Yang, D., Zhong, V., Y.Charles, Yang, Z., and Yu, T. OpenCUA: Open foundations for computer-use agents. In *NeurIPS*, 2025b.

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Ichter, B., Xia, F., Chi, E. H., Le, Q. V., and Zhou, D. Chain-of-thought prompting elicits reasoning in large language models. In *NeurIPS*, pp. 24824 – 24837, 2022.

Wu, Z., Wu, Z., Xu, F., Wang, Y., Sun, Q., Jia, C., Cheng, K., Ding, Z., Chen, L., Liang, P. P., and Qiao, Y. OS-ATLAS: Foundation action model for generalist GUI agents. In *ICLR*, 2025.

Xie, T., Zhang, D., Chen, J., Li, X., Zhao, S., Cao, R., Hua, T. J., Cheng, Z., Shin, D., Lei, F., Liu, Y., Xu, Y., Zhou, S., Savarese, S., Xiong, C., Zhong, V., and Yu, T. Osworld: benchmarking multimodal agents for open-ended tasks in real computer environments. In *NeurIPS*, pp. 52040 – 52094, 2024.

Xu, H., Zhang, X., Liu, H., Wang, J., Zhu, Z., Zhou, S., Hu, X., Gao, F., Cao, J., Wang, Z., Chen, Z., Liao, J., Zheng,

Q., Zeng, J., Xu, Z., Bai, S., Lin, J., Zhou, J., and Yan, M. Mobile-agent-v3.5: Multi-platform fundamental gui agents. *arXiv preprint arXiv: 2602.16855*, 2026a.

Xu, Y., Wang, Z., Wang, J., Lu, D., Xie, T., Saha, A., Sahoo, D., Yu, T., and Xiong, C. Aguvis: unified pure vision agents for autonomous gui interaction. In *ICML*, pp. 69772 – 69805, 2025.

Xu, Z., Wang, Z., Qian, Z., Shi, D., Tang, F., Hu, M., Su, S., Zou, X., Feng, W., Mahapatra, D., Peng, Y., Lin, M., and Ge, Z. Thinking in uncertainty: Mitigating hallucinations in mlrms with latent entropy-aware decoding. In *CVPR*, pp. 11064–11075, June 2026b.

Xue, T., Peng, C., Huang, M., Guo, L., Han, T., Wang, H., Wang, J., Zhang, X., Yang, X., Zhao, D., Ding, J., Ma, X., Xie, Y., Pei, P., Cai, X., and Qiu, X. Evocua: Evolving computer use agents via learning from scalable synthetic experience. *arXiv preprint arXiv: 2601.15876*, 2026.

Yang, Y., Li, D., Dai, Y., Yang, Y., Luo, Z., Zhao, Z., Hu, Z., HUANG, J., Saha, A., Chen, Z., Xu, R., Pan, L., Xiong, C., and Li, J. GTA1: GUI test-time scaling agent. In *ICLR*, 2026.

Zhang, C., Huang, H., Ni, C., Mu, J., Qin, S., He, S., Wang, L., Yang, F., Zhao, P., Du, C., et al. Ufo2: The desktop agentos. *arXiv preprint arXiv:2504.14603*, 2025a.

Zhang, C., Li, L., He, S., Zhang, X., Qiao, B., Qin, S., Ma, M., Kang, Y., Lin, Q., Rajmohan, S., Zhang, D., and Zhang, Q. UFO: A UI-focused agent for windows OS interaction. In *NAACL*, pp. 597–622, 2025b.

Zhou, H., Zhang, X., Tong, P., Zhang, J., Chen, L., Kong, Q., Cai, C., Liu, C., Wang, Y., Zhou, J., and Hoi, S. Mai-ui technical report: Real-world centric foundation gui agents. *arXiv preprint arXiv: 2512.22047*, 2025a.

Zhou, S., Xu, F. F., Zhu, H., Zhou, X., Lo, R., Sridhar, A., Cheng, X., Ou, T., Bisk, Y., Fried, D., Alon, U., and Neubig, G. Webarena: A realistic web environment for building autonomous agents. In *ICLR*, 2024.

Zhou, Y., Dai, S., Wang, S., Zhou, K., Jia, Q., and Xu, J. GUI-g1: Understanding r1-zero-like training for visual grounding in GUI agents. In *NeurIPS*, 2025b.

System Prompt

Tools
You may call one or more functions to assist with the user query.
You are provided with function signatures within <tools></tools> XML tags:

```
<tools>
{
  "type": "function",
  "function": {
    "name": "computer_use",
    "description": "Use a mouse to interact with a computer.
    * The screen's resolution is 1000x1000.
    * Make sure to click any buttons, links, icons, etc.
      with the cursor tip in the center of the element.
    * Do not use any other computer-use tool.
    * Only use left_click and mouse_move.
    * If the element cannot be found, terminate the task
      and report failure.",
    "parameters": {
      "properties": {
        "action": {
          "enum": ["mouse_move", "left_click"],
          "type": "string"
        },
        "coordinate": {
          "type": "array"
        }
      },
      "required": ["action"],
      "type": "object"
    }
  }
}
</tools>
```

For each function call, return a JSON object with function name and arguments within <tool_call></tool_call> XML tags:

```
<tool_call>
{
  "name": <function-name>,
  "arguments": <args-json-object>
}
</tool_call>
```

Figure 7. Full system prompt of GUI-Owl-1.5 used in evaluation.

A. More Implementation Details

System prompt. To ensure a fair compatison, we adopt the same system prompt following GUI-Owl-1.5 (Xu et al., 2026a). The full system prompt is provided in Fig. 7.

Decoding setting and candidate extraction. We use the same decoding settings as GUI-Owl-1.5, with detailed settings provided in Table 7. Regarding candidate extraction, we record the model's output token probability during decoding and

Table 7. Generation configuration used in all experiments.

Parameter	Value
Top-p	0.01
Top-k	1
Temperature	0.01
Repetition Penalty	1.0
Maximum Sequence Length	2048

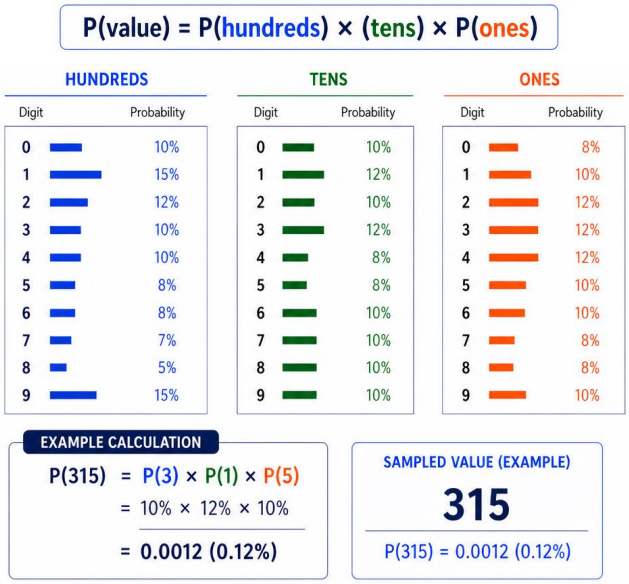


Figure 8. An example of probability-based point sampling.

independently derive digit-level coordinate probabilities from the corresponding output token. Since these probabilities are already produced by the model, this process introduces negligible computational overhead.

Probability-based point sampling. Based on the extracted token probabilities, Fig. 8 shows an example of probability-based point sampling. Specifically, the probability of coordinate 315 is computed by multiplying the probabilities of digits 3 (hundreds), 1 (tens), and 5 (ones). Based on the joint probability, we select the top-K candidates for final point prediction.

K-means clustering-based prediction. Given the top candidates K accompanied by their probabilities, we apply K-means clustering ($K = 2$) to obtain candidate clusters. The confidence of a cluster C is defined as:

$$\mathrm{Conf}(C) = \frac{\sum_{p_i \in C} w_i}{1 + \frac{1}{|C|} \sum_{p_i \in C} \|p_i - c\|_2}, \tag{3}$$

where p_i denotes the candidate point in cluster C, w_i is its probability, and c is the weighted cluster center. This metric favors clusters with high cumulative probability and low spatial dispersion. The final prediction is obtained by selecting the cluster with the highest confidence score and returning its weighted center.

B. Qualitative Results

Fig. 9 shows the visualization results. Our method outputs a prediction closer to the ground-truth region compared to the baseline, demonstrating improved GUI grounding accuracy.

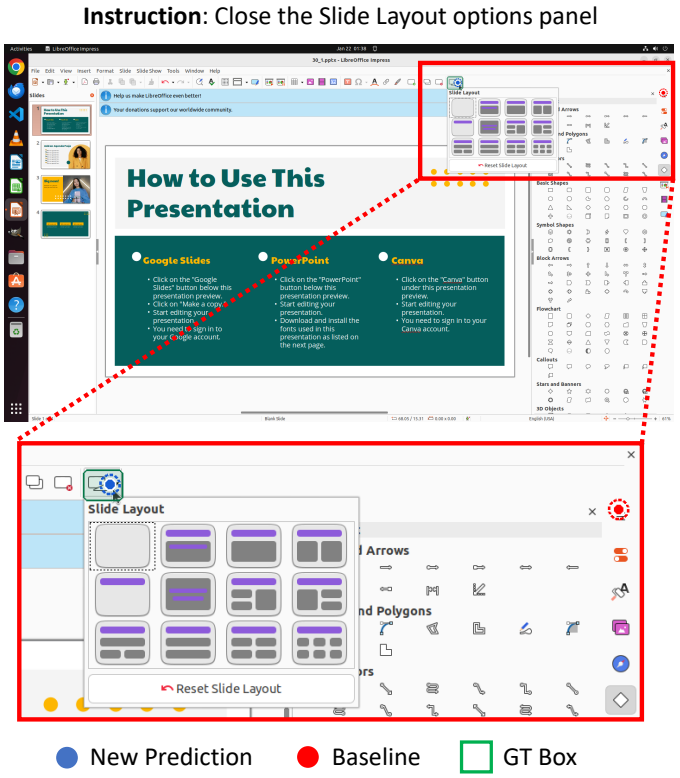


Figure 9. Visualization of GUI grounding prediction. The blue and red dots denote the predictions of entropy-aware decoding and the baseline, respectively. The green bounding box represents ground-truth region.

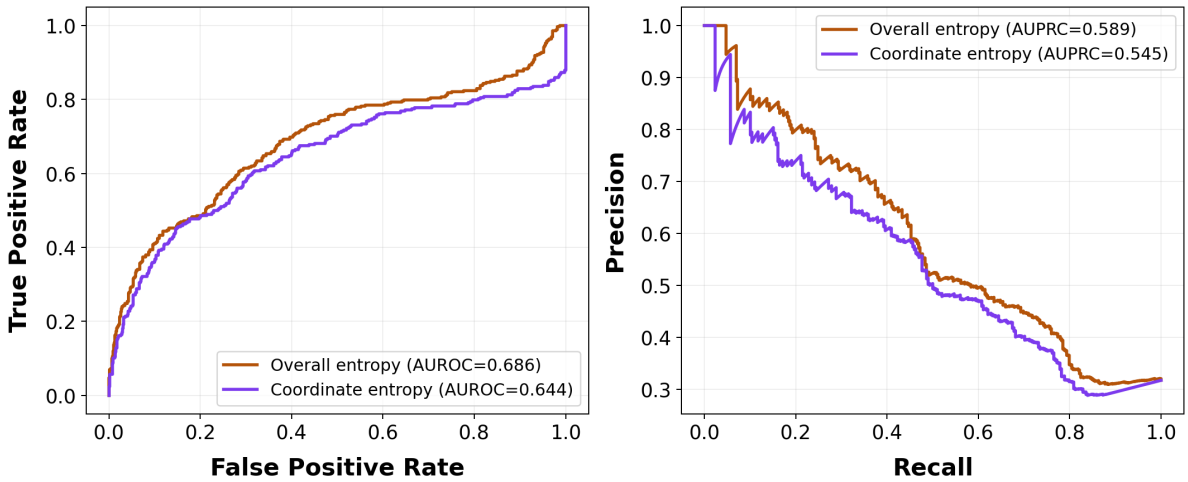


Figure 10. ROC and precision–recall curves for GUI grounding failure prediction using overall entropy and coordinate entropy.

C. Discussion on Token Entropy

The proposed method uses overall token entropy to determine whether refinement should be triggered. A natural question is whether coordinate-token entropy alone would provide a better uncertainty signal. Fig. 10 compares overall entropy and coordinate entropy as uncertainty signals for GUI grounding failure prediction. Overall entropy achieves higher AUROC (0.686 vs. 0.644) and AUPRC (0.589 vs. 0.545), indicating a stronger ability to distinguish model failures. This suggests that uncertainty information from non-coordinate tokens also contributes to identifying grounding errors. Therefore, focusing exclusively on coordinate entropy does not necessarily provide a more effective failure indicator in our setting.

D. Limitation

While our approach achieves reasonable improvements over baselines, the performance gain is inherently bounded by the model's candidate quality. If none of the top-K candidates fall within the ground-truth region, the method may still fail to predict the correct point. Additionally, the gain on ScreenSpot-Pro is marginal, suggesting that additional techniques may be required to address high-resolution GUI grounding.