VisCritic: Visual State Comparison as Process Reward for GUI Agents

Jiachen Qian

City University of Hong Kong
72510756@cityu-dg.edu.cn

Abstract. GUI agents powered by vision-language models show strong potential for automating digital tasks, yet frequently fail in long-horizon scenarios due to the absence of step-level verification. Existing process reward models verify actions through textual reasoning alone, missing the visual nature of GUI state changes. We introduce **VisCritic**, a visual process reward framework that verifies agent actions by directly comparing pre-action and post-action screenshots in visual feature space. VisCritic employs a Siamese vision transformer to extract change-aware representations, coupled with an Action-Aware Critic Head that jointly evaluates action success, task progress, and error type. A critic-training data construction pipeline generates weakly supervised samples from existing trajectories without additional human labels for critic training. Experiments and offline analyses across five benchmarks demonstrate that VisCritic serves as a plug-and-play enhancement for diverse GUI agents, generally improving benchmark metrics while providing visual diagnostic cues.

Keywords: GUI Agent · Visual Process Reward · VLM · Action Verification

1 Introduction

Graphical User Interface (GUI) agents [7, 12, 21] powered by multimodal large language models [6, 22, 25, 28] automate digital tasks by interpreting screenshots and generating executable actions such as clicking, typing, and scrolling. Despite rapid advances in visual grounding [7, 39, 45] and action prediction, these agents remain fragile in long-horizon tasks, where a single erroneous action can cascade into task failure [35, 36]. Recent studies further show that screenshot-based and multimodal agents can be compromised by visual perturbations or poisoned visual memories [26, 27], reinforcing the need for verification mechanisms grounded in visual state evidence.

A fundamental limitation of current GUI agents is the absence of reliable *step-level verification*: after executing an action, the agent has no mechanism to confirm whether the action produced its intended visual effect. Consider the example in Fig. 1: the agent intends to click an "Order" button but instead clicks an adjacent element. Without verification, the agent proceeds based on a false assumption, inevitably leading to failure.

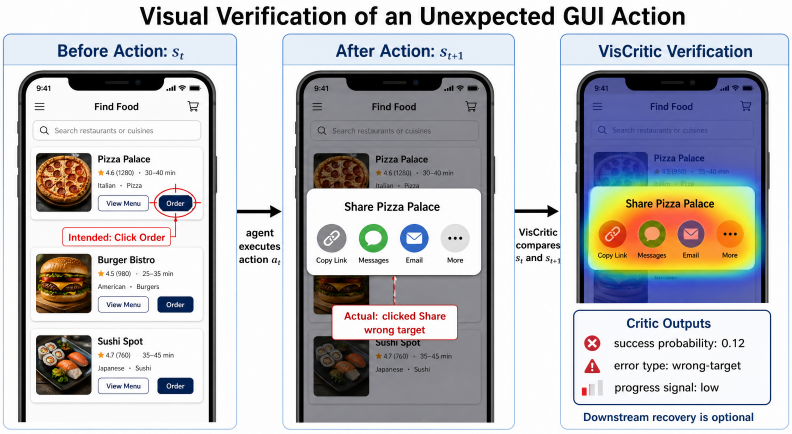

Fig. 1: Motivation example (schematic illustration). A GUI agent accidentally clicks "Share" instead of "Order" (left→middle). VisCritic detects the error by comparing screenshots: the attention map highlights the unexpected popup, and the critic predicts low success with "wrong target" classification (right).

Recent work has begun to address this gap through process reward models [17, 37] and error detection mechanisms [34, 35]. However, these approaches share a critical limitation: *their verification signal is largely mediated through textual reasoning, tool calls, or structured checklists.* This verification paradigm is fundamentally misaligned with the visual nature of GUI interactions, where action outcomes manifest as pixel-level state changes—a button changes color, a dialog appears, a page scrolls to reveal new content.

In this paper, we propose **VisCritic**, a visual process reward framework that bridges this gap by directly comparing pre-action and post-action screenshots to verify GUI agent actions (Fig. 2). Our key observation is that *visual differences between the screen before and after an action are often an informative signal when action outcomes manifest visually.* VisCritic realizes this idea with a Siamese Visual Difference Encoder, an Action-Aware Critic Head, and a critic-training data construction pipeline that weakly supervises training from existing trajectories. It operates as a plug-and-play inference-time module for screenshot-based GUI agents that expose action traces and pre-/post-action screenshots, without modifying the agent architecture or requiring additional training of the agent itself.

Evaluation across five benchmarks spanning web, mobile, and desktop environments demonstrates improvements in most tested interactive settings and outperforms text-based PRM baselines in the majority of evaluated settings. VisCritic also provides qualitative diagnostic cues through change-region attention maps. Our contributions are summarized as follows:

1. We identify the fundamental mismatch between textual verification and visual GUI state changes, and propose visual state comparison as a more natural and effective process reward signal.
2. We design VisCritic, a plug-and-play visual critic framework with a Siamese-based Visual Difference Encoder and multi-task Critic Head that jointly predicts action success, task progress, and error type.
3. We introduce a weakly supervised data pipeline that builds critic-training samples from existing trajectories without additional human labels.
4. Extensive experiments and offline analyses on five benchmarks show that VisCritic generally improves diverse GUI agents and outperforms text-based verification approaches in the majority of evaluated settings.

2 Related Work

GUI Agents. GUI agents automate digital tasks by perceiving screen content and executing interface actions. Early approaches rely on structured representations such as HTML or accessibility trees [9], but recent work has shifted toward purely visual agents that operate directly on screenshots. SeeClick [7] demonstrates that GUI grounding is the performance bottleneck. UGround [12] proposes scalable synthetic data for visual grounding. ShowUI [21] unifies vision-language-action modeling with UI-guided token selection. Concurrently, LLM-based agents [14,33,41] demonstrate autonomous GUI navigation. Despite these advances, many current GUI agents still rely primarily on single-pass action prediction and lack reliable step-level verification of executed action outcomes.

Process Reward Models for GUI Agents. PRMs provide intermediate supervision to guide agent behavior during multi-step tasks [20]. GUI-PRA [37] introduces a process reward agent with dynamic memory and adaptive UI perception, explicitly addressing the "lost in the middle" problem and lack of UI changing awareness; however, its verification operates through textual tool calls. CRAFT-GUI [23] designs task-specific reward functions by integrating rule-based verification with model-predicted evaluation.

More recently, GUI-Shepherd [4] provides dense step-by-step process supervision trained on 52K human-annotated interactions, serving both as an RL reward provider and inference-time verifier. Web-Shepherd [3] proposes the first PRM specifically for web agents, using structured subgoal checklists. ADMIRE [42] addresses the reward fidelity–density trade-off by anchoring trajectories to dynamically distilled milestones. While these PRMs achieve notable improvements, their verification signal is largely mediated through textual state descriptions, tool calls, or structured checklists—thus inheriting the modality gap between textual verification and the visual nature of GUI state changes. VisCritic provides a complementary visual verification signal that could be combined with textual PRMs for multi-modal process supervision.

Error Detection and Recovery. BacktrackAgent [35] introduces a four-component framework for error detection and recovery through backtracking,

where the verifier is rule-based and the judger is a trained VLM. GUI-Critic-R1 [34] takes a "look before you leap" approach, pre-judging actions before execution. The CVPR 2025 work on VLM self-correction [19] explores whether VLMs can correct their own grounding errors through feedback. Minitap [11] achieves 100% success on AndroidWorld via a multi-agent architecture with task decomposition and deterministic post-validation, but its per-action-type validators are limited to action types with programmatically verifiable outcomes. VisCritic generalizes verification to a broad range of GUI action types by operating in visual feature space, complementing deterministic validators.

Visual Change Detection, World Models, and Tree Search. Visual change detection has been extensively studied in remote sensing [8,31] and video surveillance, where Siamese architectures [1] have proven effective for identifying meaningful differences between temporally separated images. We apply this paradigm to a novel context: using visual state differences as a reward signal for agent decision-making. Concurrent GUI world models—MobileDreamer [2], CUWM [13], and Dyna-Mind [38]—address state *prediction* rather than verification. VisCritic is complementary: while world models predict *what will happen*, VisCritic verifies *what actually happened* without depending on the fidelity of learned dynamics.

Tree search methods such as LATS [43] and Agent Alpha [32] improve *action selection* by exploring multiple trajectories via MCTS. VisCritic addresses the orthogonal problem of *action verification*—confirming whether a chosen action produced its intended visual effect.

3 Method

3.1 Problem Formulation

We formulate GUI task automation as a sequential decision process. At each time step t, the agent observes a screenshot s_t, receives a task instruction l, and generates an action a_t (*e.g.*, `CLICK(x,y)`, `TYPE("text")`, `SCROLL(direction)`). The environment executes a_t and transitions to a new state, producing screenshot s_{t+1}. The task is complete when the agent reaches a goal state or exceeds a maximum number of steps.

Visual Critic is defined as a function $\mathcal{C} : (s_t, a_t, s_{t+1}, l) \to (\hat{y}_{\text{suc}}, \hat{y}_{\text{prog}}, \hat{y}_{\text{err}})$ that takes the before-after screenshot pair (s_t, s_{t+1}), the executed action a_t, and the task instruction l, and outputs:

- $\hat{y}_{\text{suc}} \in [0, 1]$: action success probability,
- $\hat{y}_{\text{prog}} \in [-1, 1]$: task progress score,
- $\hat{y}_{\text{err}} \in \{\text{success, no-op, wrong-target, page-error, timeout}\}$: error type.

When $\hat{y}_{\text{suc}} < \gamma$ (a success threshold), the critic can optionally expose its verification result to downstream recovery policies, such as retrying, backtracking, or requesting a textual recovery suggestion.

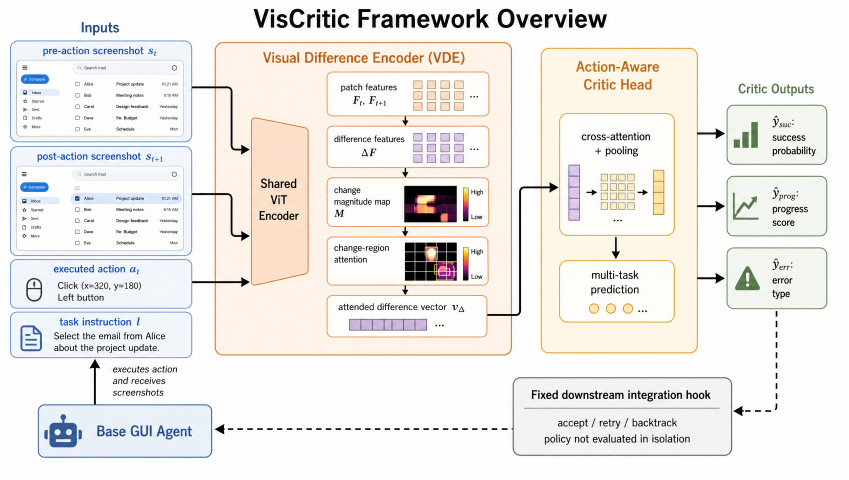

Fig. 2: Overview of the VisCritic framework. Given a pre-action screenshot s_t and post-action screenshot s_{t+1}, the Visual Difference Encoder (VDE) extracts patch-level semantic differences via a shared ViT encoder, computes a change magnitude map, and applies change region attention to produce the attended difference vector $\mathbf{v}_\Delta$. The Action-Aware Critic Head fuses $\mathbf{v}_\Delta$ with the action a_t and task instruction l to predict action success, task progress, and error type. The critic output can be passed to a fixed downstream integration hook for accept, retry, or backtrack decisions.

3.2 Visual Difference Encoder

The Visual Difference Encoder (VDE) is the core component responsible for extracting meaningful visual changes between consecutive screenshots. Rather than relying on pixel-level image differencing, which is sensitive to rendering noise, animation artifacts, and minor layout shifts, VDE operates in a learned semantic feature space.

Siamese Feature Extraction. Given screenshots s_t and s_{t+1}, a shared-weight ViT [10] encoder $\mathcal{E}_\theta$ extracts patch-level features:

$$F_t = \mathcal{E}_\theta(s_t) \in \mathbb{R}^{P\times d}, \quad F_{t+1} = \mathcal{E}_\theta(s_{t+1}) \in \mathbb{R}^{P\times d}, \tag{1}$$

where P is the number of visual patches and d is the feature dimension. The Siamese architecture [1] ensures that both screenshots are encoded in the same feature space, enabling meaningful comparison.

Difference Feature Computation. We compute the patch-level difference features:

$$\Delta F = F_{t+1} - F_t \in \mathbb{R}^{P\times d}. \tag{2}$$

Additionally, we compute a *change magnitude map* $M \in \mathbb{R}^P$:

$$M_i = \|\Delta F_i\|_2, \quad i = 1, \ldots, P, \tag{3}$$

which indicates the degree of semantic change at each patch position.

Change Region Attention. We introduce change region attention to focus on regions with meaningful UI changes:

$$\alpha_i = \frac{\exp(M_i/\beta)}{\sum_{j=1}^{P} \exp(M_j/\beta)}, \tag{4}$$

where β is a learnable temperature parameter. The attended difference representation is:

$$\mathbf{v}_{\Delta} = \sum_{i=1}^{P} \alpha_i \cdot \Delta F_i \in \mathbb{R}^d. \tag{5}$$

The full VDE output consists of three components: (1) the global attended difference vector $\mathbf{v}_{\Delta}$, (2) the patch-level difference features ΔF for fine-grained analysis, and (3) the change magnitude map M for interpretability.

3.3 Action-Aware Critic Head

The Critic Head integrates visual difference features with action and task context to produce multi-task predictions. We encode the action a_t and task instruction l using the text encoder of the backbone MLLM:

$$\mathbf{h}_a = \text{TextEnc}(a_t), \quad \mathbf{h}_l = \text{TextEnc}(l). \tag{6}$$

The fused representation is obtained via cross-attention, where the patch-level difference features serve as queries and the textual context provides keys and values, combined with spatial self-attention over ΔF:

$$\begin{aligned} \mathbf{z}_{\text{cross}} &= \text{Pool}\big(\text{CrossAttn}(Q{=}\Delta F,\ K{=}[\mathbf{h}_a;\mathbf{h}_l],\ V{=}[\mathbf{h}_a;\mathbf{h}_l])\big), \\ \mathbf{z} &= \mathbf{z}_{\text{cross}} + \text{Pool}\big(\text{SpatialAttn}(\Delta F)\big), \end{aligned} \tag{7}$$

where both branches are projected to the same hidden dimension and $\text{Pool}(\cdot)$ denotes change-magnitude-weighted pooling using α from Sec. 3.2. The multi-task outputs are:

$$\hat{y}_{\text{suc}} = \sigma(\text{MLP}_{\text{suc}}(\mathbf{z})), \tag{8}$$

$$\hat{y}_{\text{prog}} = \tanh(\text{MLP}_{\text{prog}}(\mathbf{z})), \tag{9}$$

$$\hat{y}_{\text{err}} = \text{softmax}(\text{MLP}_{\text{err}}(\mathbf{z})). \tag{10}$$

When $\hat{y}_{\text{suc}} < \gamma$, downstream agents may optionally use $\mathbf{z}$ and the predicted error type to drive retry, backtracking, or recovery hooks. In this work we focus on the core accept/reject verification; recovery policy design is outside the main quantitative scope.

3.4 Critic-Guided Agent Loop

VisCritic integrates with existing GUI agents through two complementary mechanisms:

Best-of-N Selection (Pre-Execution, Optional). When a visual state predictor is available (*e.g.*, MobileDreamer [2]), the agent generates N candidate actions and VisCritic scores each based on the predicted post-action state $\hat{s}_{t+1}^{(a)}$, rendered as simplified layout images. VisCritic selects the highest-scoring action (details and a text-only fallback analysis are in the supplementary material):

$$a_t^* = \arg \max_{a \in \{a_t^{(1)}, \ldots, a_t^{(N)}\}} \mathcal{C}(s_t, a, \hat{s}_{t+1}^{(a)}, l)_{\text{suc}}. \tag{11}$$

Post-Execution Verification. After executing the selected action, VisCritic verifies the actual outcome by comparing s_t and the real s_{t+1}. This is the primary verification mode and does not require any state prediction—it operates on the *actual* visual outcome. If $\hat{y}_{\text{suc}} < \gamma$, the agent can enter an optional recovery policy that retries, backtracks, or requests a recovery suggestion. In our interactive experiments, low-confidence steps are handled by the same fixed retry/recovery hook across compared settings; thus task-SR gains should be interpreted as the effect of VisCritic's verification signal under this fixed integration protocol, rather than as an isolated evaluation of recovery-policy design. Post-execution verification is the recommended default mode, as it uses actual post-execution screenshots and avoids learned state prediction.

3.5 Training Pipeline

Annotation-Free Data Construction. A key advantage of VisCritic is that critic-training data can be automatically constructed from existing GUI trajectory datasets without additional human labels for critic training. As illustrated in Fig. 3, we define five types of training samples:

Positive samples are drawn from successful trajectories, where consecutive tuples (s_t, a_t, s_{t+1}) are assigned a weak positive proxy label $y_{\text{suc}} = 1$.

Negative samples are generated through four perturbation strategies:

1. **Action mismatch**: Replace a_t with a random action a' from the same trajectory while keeping s_t and s_{t+1} fixed, creating cases where the visual change does not correspond to the described action.
2. **State mismatch**: Replace s_{t+1} with a screenshot from a different trajectory, creating cases where the visual outcome is unexpected.
3. **No-op detection**: Pair s_t with $s_{t+1} \approx s_t$ (SSIM > 0.98) to train detection of ineffective actions.
4. **Failed trajectory sampling**: Draw tuples from failed trajectories. Steps after the estimated divergence point are assigned a weak negative proxy label $y_{\text{suc}} = 0$; the divergence point is estimated by comparing against the closest successful trajectory.

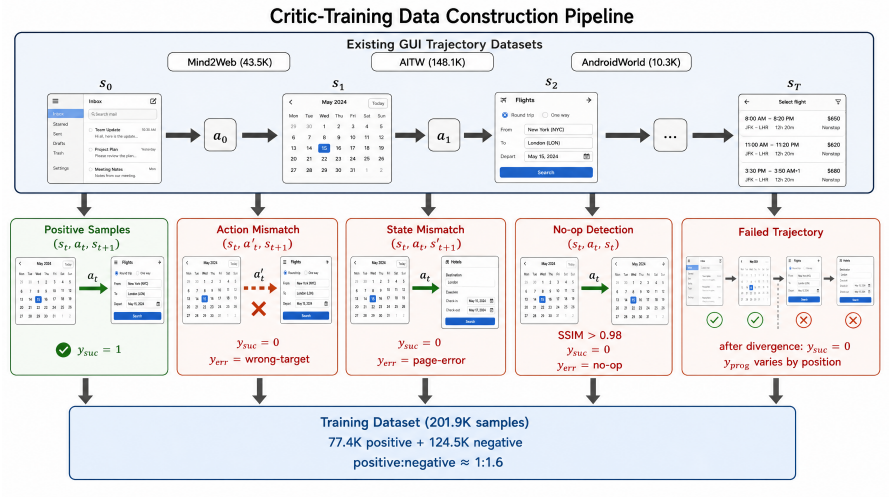

Fig. 3: Critic-training data construction pipeline. From existing GUI trajectory datasets, we automatically generate five types of training samples: positive samples from successful trajectories, and four types of negatives (action mismatch, state mismatch, no-op, failed trajectory) without additional human labels for critic training.

Error-type labels (y_{err}) are derived deterministically from the perturbation strategy (81.4% agreement with 500 manually annotated cases). **Progress scores** are trajectory-derived heuristic labels: $y_{\text{prog}} = (t{+}1)/T$ for successful steps, and $y_{\text{prog}} = -k/K$ after the divergence point in failed trajectories. Details are in the supplementary material.

Multi-Phase Training. Phase 1: Contrastive Pre-training. The VDE is pre-trained with an InfoNCE loss [5, 24]:

$$\mathcal{L}_{\text{contrast}} = -\log \frac{\exp(\text{sim}(\mathbf{v}_\Delta, \mathbf{h}_a^+)/\tau_c)}{\sum_j \exp(\text{sim}(\mathbf{v}_\Delta, \mathbf{h}_a^{(j)})/\tau_c)}, \tag{12}$$

where $\mathbf{h}_a^+$ is the correct action feature. For samples with a reliable action-region mask, an auxiliary localization loss (KL divergence) encourages attention α to concentrate on UI elements affected by the action, using a binary mask $\mathbf{m}$ from action coordinates:

$$\mathcal{L}_{\text{loc}} = \sum_{i=1}^{P} \hat{\alpha}_i \log \frac{\hat{\alpha}_i}{\alpha_i}, \tag{13}$$

with $\hat{\alpha}_i = m_i / \sum_j m_j$ when $\sum_j m_j > 0$. Samples without a reliable action-region mask skip $\mathcal{L}_{\text{loc}}$. The Phase 1 objective is $\mathcal{L}_{\text{P1}} = \mathcal{L}_{\text{contrast}} + 0.3 \cdot \mathcal{L}_{\text{loc}}$ for masked samples. The localization mask is used *only during training*; at inference, attention is computed solely from the learned change magnitude map M.

Phase 2: Multi-Task Fine-Tuning. The full model is trained end-to-end with a multi-task loss [18]:

$$\mathcal{L} = \lambda_1 \mathcal{L}_{\text{BCE}}(\hat{y}_{\text{suc}}, y_{\text{suc}}) + \lambda_2 \mathcal{L}_{\text{MSE}}(\hat{y}_{\text{prog}}, y_{\text{prog}}) + \lambda_3 \mathcal{L}_{\text{CE}}(\hat{y}_{\text{err}}, y_{\text{err}}), \tag{14}$$

where $\lambda_1, \lambda_2, \lambda_3$ are loss weights.

4 Experiments

4.1 Experimental Setup

Benchmarks. We evaluate on five benchmarks: **Mind2Web** [9] (cross-task test split, 252 web tasks, step SR), **AndroidWorld** [29] (116 total tasks; we evaluate on a 34-task application-disjoint held-out split, live emulator), **OSWorld** [36] (369 desktop tasks), **WebArena** [44] (812 web tasks), and **AITW** [30] (715K Android episodes, action accuracy). For AITW, which is an offline benchmark, we report a logged-screenshot consistency diagnostic using recorded pre-/post-action screenshots; these numbers should not be interpreted as live inference-time gains for arbitrary candidate actions.

Base Agents. To demonstrate plug-and-play compatibility, we apply VisCritic to multiple base agents: SeeClick [7], ShowUI [21], CogAgent [15], and an agent based on Qwen2.5-VL [28].

Baselines. We compare against: (1) base agents without any critic, (2) vanilla Best-of-N with self-consistency, (3) GUI-PRA [37] (text-based PRM), (4) Guid-Nav [17] (text PRM with search), (5) BacktrackAgent [35] (text-based error detection), and (6) GUI-Critic-R1 [34] (pre-execution critic). All baselines are re-implemented using publicly released code and evaluated under our unified protocol where applicable, with AITW treated as an offline diagnostic.[1] We note that GUI-Shepherd [4] has not released its training data or model weights. Web-Shepherd [3] releases a web-focused PRM and WebPRM Collection, but its checklist-based reward formulation, annotation protocol, and web-agent setting differ from our screenshot-pair visual critic protocol. A parameter-, data-, and protocol-controlled comparison is therefore non-trivial, so we restrict baselines to methods whose released artifacts permit faithful re-implementation under our unified protocol. For GUI-PRA, we use a Qwen2.5-VL-7B supervisor to match the parameter budget of other baselines. The Pixel-Critic baseline (Tab. 3) uses the same Critic Head architecture but replaces the VDE with raw pixel-level difference maps ($|s_{t+1} - s_t|$ in RGB space), ensuring a fair comparison that isolates the contribution of learned semantic features.

[1] Re-implemented numbers may differ from original publications due to our unified evaluation protocol (screenshot-only input, unified action space). Details in the supplementary material.

Baseline fairness. VisCritic (InternViT-6B + Qwen2.5-7B, ∼13B total) uses a larger backbone than some baselines. To control for model capacity, the Text-Critic in Tab. 3 uses the *same* Qwen2.5-7B backbone and identical Critic Head, differing only in modality; VisCritic's advantage (84.6% vs. 66.2% F1 in Tab. 3) supports improvement from the visual comparison paradigm rather than model scale. In the controlled AndroidWorld critic-quality setting, a smaller SigLIP-400M [40] backbone (∼7.4B total) remains stronger than the text-based critic baselines reported in the same setting (80.3% F1 and 23.1% task SR; see supplementary material).

Implementation Details. VisCritic uses InternViT-6B [6, 10] with LoRA [16] adapters (rank=16) and Qwen2.5-7B text backbone. Training data (∼202K samples) is constructed from Mind2Web (43.5K), AITW (148.1K), and AndroidWorld (10.3K) with ∼1:1.6 positive-to-negative ratio. OSWorld and WebArena are *not* in the training data; results on these benchmarks represent zero-shot cross-benchmark generalization (and additionally cross-platform for OSWorld's desktop environment). All evaluation uses held-out test splits with strict deduplication (details in supplementary material). Phase 1 trains the VDE for 5 epochs (lr=1e-4, λ_{loc}=0.3); Phase 2 fine-tunes end-to-end for 10 epochs (lr=2e-5, λ_1=1.0, λ_2=0.5, λ_3=0.5). We set γ=0.5; unless otherwise stated, main results use post-execution verification, while N=3 is used only in optional Best-of-N analyses. All experiments use 8×A100 GPUs.

4.2 Main Results

Tab. 1 presents the main results. We highlight three key observations: (1) Excluding the offline AITW diagnostic, VisCritic improves every base agent on the large majority of interactive evaluated settings, demonstrating its generality as an inference-time plug-in under our tested protocol; in one interactive case (Qwen2.5-VL on OSWorld), a text-based critic marginally outperforms, reflecting scenarios where textual verification may have a natural advantage in text-heavy desktop environments. (2) The improvement is particularly pronounced on interactive web benchmarks (WebArena, Mind2Web), supporting that step-level visual verification is most valuable when error accumulation is severe. (3) Among benchmark–agent combinations where released text-based baselines could be faithfully re-implemented under our unified protocol, VisCritic outperforms text-based critic baselines in the majority of settings, supporting that visual state comparison generally provides a reliable verification signal complementary to textual reasoning about GUI changes. Notably, the improvement generally correlates with base agent capability—Qwen2.5-VL gains +5.4 pts in interactive average and ShowUI gains +4.6 pts—though the relationship is not strictly monotonic (CogAgent gains +2.9 pts, possibly due to its dual-encoder architecture limiting plug-and-play integration).

Statistical significance. All results are averaged over 3 independent runs; standard deviations range from 0.3 pts (AITW, largest offline diagnostic set) to

Table 1: Performance (%) across benchmarks (task SR for AndroidWorld/OSWorld/ WebArena, step SR for Mind2Web, action accuracy for AITW). AITW is an offline logged-screenshot consistency diagnostic rather than a live interaction setting. "Interactive Avg." is the arithmetic mean over Mind2Web, AndroidWorld, OSWorld, and WebArena only. Best in **bold**. †Not all baselines shown; some require agent-specific hooks unavailable in released codebases (see supplementary).

Method	Mind2Web	AndroidWorld	OSWorld	WebArena	AITW	Interactive Avg.
SeeClick	33.4	14.7	6.8	12.4	62.1	16.8
+ GUI-PRA	36.2	17.1	8.2	14.6	64.5	19.0
+ BacktrackAgent	36.9	17.9	9.0	15.3	64.2	19.8
+ GUI-Critic-R1	37.2	18.3	9.3	15.7	**64.8**	20.1
+ **VisCritic (Ours)**	**37.6**	**19.1**	**9.5**	**16.4**	64.5	**20.7**
ShowUI	43.5	19.8	10.3	17.6	66.7	22.8
+ GUI-PRA	46.2	22.1	12.0	19.7	68.8	25.0
+ **VisCritic (Ours)**	**48.9**	**24.8**	**12.8**	**23.2**	**69.8**	**27.4**
CogAgent†	41.2	17.5	8.9	15.1	64.8	20.7
+ **VisCritic (Ours)**	**44.7**	**20.8**	**10.6**	**18.4**	**67.1**	**23.6**
Qwen2.5-VL Agent	47.3	24.6	14.2	22.8	70.4	27.2
+ GUI-PRA	50.1	27.0	16.3	24.9	72.5	29.6
+ GuidNav	51.2	27.8	17.0	26.1	73.0	30.5
+ GUI-Critic-R1	51.8	28.2	**17.4**	26.5	73.3	31.0
+ **VisCritic (Ours)**	**54.1**	**29.8**	17.1	**29.4**	**74.2**	**32.6**

Table 2: Action verification quality on AndroidWorld (Qwen2.5-VL base). We report precision, recall, and F1 (%) for action success prediction.

Method	Precision	Recall	F1
GUI-PRA (Text)	73.2	67.8	70.4
BacktrackAgent (Text)	75.6	64.3	69.5
VisCritic (Visual)	**87.1**	**83.4**	**85.2**

1.4 pts (AndroidWorld, smallest held-out split). On Mind2Web, WebArena, and AITW, improvements over the best text-based baseline are statistically significant ($p < 0.01$, paired bootstrap). On OSWorld, $p < 0.05$. On AndroidWorld, the improvement is directionally consistent but does not reach conventional significance ($p \approx 0.08$) due to the small held-out split (34 tasks); a larger evaluation pool would provide stronger statistical power, and the per-task win rate (VisCritic wins on 20/34 tasks vs. best baseline) is directionally consistent. We use paired bootstrap with task/episode-level resampling where applicable; p-values are nominal and uncorrected for multiple comparisons.

4.3 Critic Quality Analysis

Tab. 2 evaluates the intrinsic quality of action verification.

Table 3: Visual vs. textual critic comparison: action verification F1 (%) by action category on held-out AndroidWorld + Mind2Web.

Critic	Visual-Only	Semantic	No-Op	Overall
Text-Critic	58.3	74.1	62.7	66.2
Pixel-Critic	71.6	63.8	78.4	70.5
VisCritic	**84.3**	**85.6**	**84.2**	**84.6**

Table 4: Ablation study on VisCritic components. Task success rate (%) on AndroidWorld with ShowUI base agent.

Variant	Success Rate
Full VisCritic	24.8
w/o Change Region Attention	23.4
w/o Multi-task Head (success only)	24.1
w/o Contrastive Pre-training	22.0
w/o Post-execution Verification	21.4
Pixel-level Diff (replace VDE)	21.2

4.4 Visual vs. Textual Critic Comparison

To isolate the benefit of visual state comparison, we conduct a controlled experiment where all critics share identical Critic Head architecture but differ in how they assess UI state change: (1) **Text-Critic** uses textual descriptions of expected changes; (2) **Pixel-Critic** uses raw pixel difference maps; (3) **VisCritic** uses learned VDE semantic features.

Tab. 3 presents the results. Three findings stand out: (1) Text-Critic performs poorly on *visual-only* changes (*e.g.*, button highlight, icon state toggle, color transitions) that are difficult to describe textually, achieving only 58.3% F1 compared to VisCritic's 84.3%. (2) Pixel-Critic struggles with *semantic* changes (*e.g.*, page navigation, content loading) where pixel-level differences are large but uninformative, scoring 63.8% F1 versus VisCritic's 85.6%. (3) VisCritic achieves balanced high performance across all categories, supporting that learned semantic difference features capture both fine-grained visual changes and high-level state transitions effectively. Notably, VisCritic's No-Op detection (84.2%) significantly outperforms Text-Critic (62.7%), as no-op cases exhibit a distinctive minimal-change pattern in the VDE feature space that is difficult to express textually but easy to detect visually.

4.5 Ablation Study

Tab. 4 isolates the contribution of each VisCritic component.

Key observations: (1) Contrastive pre-training is the most impactful training component (-2.8 pts), supporting the importance of learning discriminative visual difference representations before multi-task fine-tuning. (2) Removing post-

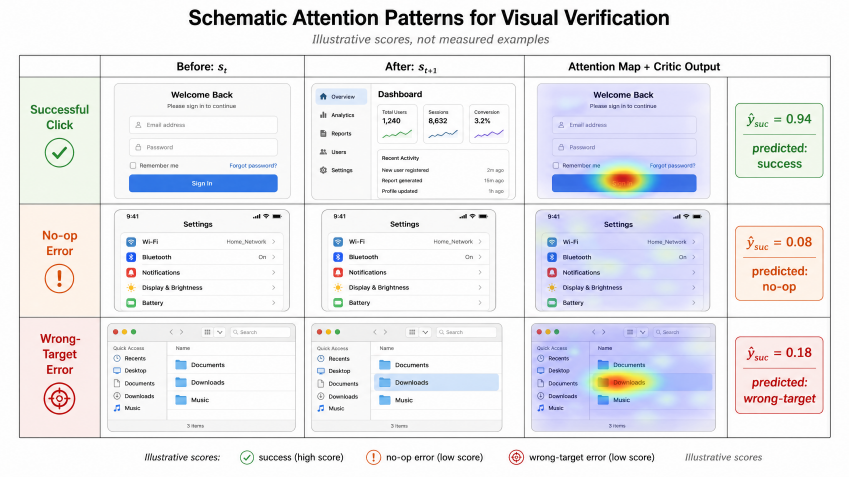

Fig. 4: Conceptual illustration of change region attention patterns. (top) For successful clicks, attention is expected to concentrate on the clicked element ($\hat{y}_{\text{suc}} > 0.9$). (middle) For no-op errors, attention disperses uniformly ($\hat{y}_{\text{suc}} < 0.15$). (bottom) For wrong-target errors, attention focuses on an unintended region ($\hat{y}_{\text{suc}} \approx 0.2$). UI mockups are schematic; category-level verification performance is reported in Tab. 3.

execution verification (−3.4 pts) weakens the fixed recovery loop, showing that verification against actual screenshots is critical under our integration protocol; pre-execution selection alone is insufficient. (3) Replacing VDE with pixel-level differences (−3.6 pts) validates that learned semantic features are essential; raw pixel differences are too sensitive to rendering noise. (4) Change region attention contributes +1.4 pts, focusing the model on relevant UI changes. (5) The multi-task head provides a modest +0.7 pts over success-only prediction; while auxiliary tasks benefit training, the success prediction objective dominates. Additional ablations on backbone choice, Best-of-N candidates, and loss weights are in the supplementary material.

4.6 Qualitative Analysis

Fig. 4 schematically illustrates three characteristic attention patterns consistent with the corresponding verification outcomes. For **successful clicks**, the change region attention concentrates on the clicked element and its visual response ($\hat{y}_{\text{suc}} > 0.9$). For **no-op errors**, the attention disperses uniformly due to the absence of meaningful visual change ($\hat{y}_{\text{suc}} < 0.15$). For **wrong-target errors**, the attention focuses on an unintended changed region ($\hat{y}_{\text{suc}} \approx 0.2$).

4.7 Cross-Agent Transferability and Overhead

As shown in Tab. 1, a single VisCritic model trained once transfers to four architecturally distinct agents without any agent-specific adaptation, supporting

that visual state verification is largely agent-agnostic under our tested integration protocol.

VisCritic adds 6.2B parameters (InternViT-6B with LoRA adapters plus the Critic Head) when the Qwen2.5-7B text backbone is shared with the base agent; for agents using a different LLM backbone (*e.g.*, SeeClick, CogAgent), an additional text encoder is required. While this is a significant overhead, we note that: (1) the critic runs independently and can be deployed on a separate GPU; (2) post-execution mode adds only 66ms per step (15.7% overhead on A100-80GB with FP16, comprising 52ms for the Siamese ViT forward pass and 14ms for the Critic Head), negligible compared to typical 1–3s environment interaction latency; (3) the observed 1.7–6.8% absolute metric improvements may justify the additional compute in accuracy-critical settings. For resource-constrained settings, a SigLIP-400M backbone reduces overhead to 35ms while retaining 80.3% critic F1 (see supplementary material). Additional analyses including progress score evaluation, error-type classification, robustness to dynamic visual changes, and pre- vs. post-execution comparison are provided in the supplementary material.

5 Conclusion

We presented VisCritic, a visual process reward framework for GUI agents that verifies action outcomes through direct visual state comparison. By introducing a Siamese-based Visual Difference Encoder with change region attention, VisCritic captures meaningful UI changes at the semantic level rather than relying on textual descriptions. Our critic-training data construction pipeline enables scalable weak supervision without additional human labels for critic training. Extensive experiments and offline analyses demonstrate that VisCritic serves as an effective plug-and-play enhancement for diverse GUI agents, outperforming evaluated text-based verification approaches on the large majority of comparable benchmark–agent combinations. The change-region attention maps further offer a qualitative view of which visual changes influence verification.

Limitations and future work. VisCritic inherits the visual understanding limitations of its backbone ViT encoder, particularly for fine-grained text changes or subtle UI state transitions (*e.g.*, a counter incrementing from "3" to "4"). Task-irrelevant visual dynamics (animations, ads, asynchronous loads) can degrade verification accuracy; while the learned semantic features and change region attention provide meaningful robustness, extreme viewport and resolution changes remain challenging. The critic-training data construction pipeline relies on heuristic weak labels, including divergence point estimation via trajectory alignment, which may inject label noise (sensitivity analysis in the supplementary material). Recovery policies, while architecturally compatible with VisCritic outputs, have not been quantitatively evaluated in isolation.

Several promising directions remain for future work: (1) **Lightweight variants**: while our default InternViT-6B backbone achieves the best accuracy, the SigLIP-400M variant retains 94% of the critic F1 at 15× fewer vision parame-

ters, making deployment practical on consumer GPUs; exploring distillation or architecture-specific optimizations could further reduce the footprint. (2) **Hybrid verification**: combining VisCritic with deterministic validators [11] for action types with programmatically verifiable outcomes (*e.g.*, text input) could improve coverage on such action types, as the two approaches are naturally complementary. (3) **Integration with tree search**: using VisCritic as a learned value function within MCTS frameworks [32, 43] could provide more grounded state assessments than text-based evaluators. (4) **Multi-modal process supervision**: combining VisCritic's visual signal with textual PRM scores [3, 4] for fused multi-modal verification.

References

1. Bertinetto, L., Valmadre, J., Henriques, J.F., Vedaldi, A., Torr, P.H.S.: Fully-convolutional siamese networks for object tracking. In: ECCV Workshops (2016)
2. Cao, Y., Zhong, Y., Zeng, Z., Zheng, L., Huang, J., Qiu, H., Shi, P., Mao, W., Wan Guanglu: MobileDreamer: Generative sketch world model for GUI agent. arXiv preprint arXiv:2601.04035 (2026)
3. Chae, H., Kim, S., Cho, J., Kim, S., Moon, S., Hwangbo, G., Lim, D., Kim, M., Hwang, Y., Gwak, M., Choi, D., Kang, M., Im, G., Cho, B., Kim, H., Han, J., Kwon, T., Kim, M., Kwak, B.w., Kang, D., Yeo, J.: Web-Shepherd: Advancing PRMs for reinforcing web agents. In: Adv. Neural Inform. Process. Syst. (2025)
4. Chen, C., Ji, K., Zhong, H., Zhu, M., Li, A., Gan, G., Huang, Z., Zou, C., Liu, J., Chen, J., Chen, H., Shen, C.: GUI-Shepherd: Reliable process reward and verification for long-sequence GUI tasks. arXiv preprint arXiv:2509.23738 (2025)
5. Chen, T., Kornblith, S., Norouzi, M., Hinton, G.: A simple framework for contrastive learning of visual representations. In: Int. Conf. Mach. Learn. (2020)
6. Chen, Z., Wang, W., Cao, Y., Liu, Y., Gao, Z., Cui, E., Zhu, J., Ye, S., Tian, H., Liu, Z., Gu, L., Wang, X., Li, Q., Ren, Y., Chen, Z., Luo, J., Wang, J., Jiang, T., Wang, B., He, C., Shi, B., Zhang, X., Lv, H., Wang, Y., Shao, W., Chu, P., Tu, Z., He, T., Wu, Z., Deng, H., Ge, J., Chen, K., Zhang, K., Wang, L., Dou, M., Lu, L., Zhu, X., Lu, T., Lin, D., Qiao, Y., Dai, J., Wang, W.: Expanding performance boundaries of open-source multimodal models with model, data, and test-time scaling. arXiv preprint arXiv:2412.05271 (2024)
7. Cheng, K., Sun, Q., Chu, Y., Xu, F., YanTao, L., Zhang, J., Wu, Z.: SeeClick: Harnessing GUI grounding for advanced visual GUI agents. In: ACL (2024)
8. Daudt, R.C., Saux, B.L., Boulch, A.: Fully convolutional siamese networks for change detection. In: IEEE International Conference on Image Processing (ICIP) (2018)
9. Deng, X., Gu, Y., Zheng, B., Chen, S., Stevens, S., Wang, B., Sun, H., Su, Y.: Mind2Web: Towards a generalist agent for the web. In: Adv. Neural Inform. Process. Syst. (2023)
10. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., Houlsby, N.: An image is worth 16x16 words: Transformers for image recognition at scale. In: Int. Conf. Learn. Represent. (2021)
11. Favreau, P.L., Lo, J.P., Guiguet, C., Simon-Meunier, C., Dehandschoewercker, N., Roush, A.G., Goldfeder, J., Shwartz-Ziv, R.: Do multi-agents dream of electric

screens? achieving perfect accuracy on AndroidWorld through task decomposition. arXiv preprint arXiv:2602.07787 (2026)
12. Gou, B., Wang, R., Zheng, B., Xie, Y., Chang, C., Shu, Y., Sun, H., Su, Y.: Navigating the digital world as humans do: Universal visual grounding for GUI agents. In: Int. Conf. Learn. Represent. (2025)
13. Guan, Y., Yu, R., Zhang, J., Wang, L., Zhang, C., Li, L., Qiao, B., Qin, S., Huang, H., Yang, F., Zhao, P., Wutschitz, L., Kessler, S., Inan, H.A., Sim, R., Rajmohan, S., Lin, Q., Zhang, D.: Computer-using world model. arXiv preprint arXiv:2602.17365 (2026)
14. Gur, I., Furuta, H., Huang, A., Safdari, M., Matsuo, Y., Eck, D., Faust, A.: A real-world WebAgent with planning, long context understanding, and program synthesis. In: Int. Conf. Learn. Represent. (2024)
15. Hong, W., Wang, W., Lv, Q., Xu, J., Yu, W., Ji, J., Wang, Y., Wang, Z., Dong, Y., Ding, M., Tang, J.: CogAgent: A visual language model for GUI agents. In: IEEE Conf. Comput. Vis. Pattern Recog. (2024)
16. Hu, E.J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., Chen, W.: LoRA: Low-rank adaptation of large language models. In: Int. Conf. Learn. Represent. (2022)
17. Hu, Z., Xiong, S., Zhang, Y., Ng, S.K., Luu, A.T., An, B., Yan, S., Hooi, B.: Guiding VLM agents with process rewards at inference time for GUI navigation. arXiv preprint arXiv:2504.16073 (2025)
18. Kendall, A., Gal, Y., Cipolla, R.: Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In: IEEE Conf. Comput. Vis. Pattern Recog. (2018)
19. Liao, Y.H., Mahmood, R., Fidler, S., Acuna, D.: Can large vision-language models correct semantic grounding errors by themselves? In: IEEE Conf. Comput. Vis. Pattern Recog. (2025)
20. Lightman, H., Kosaraju, V., Burda, Y., Edwards, H., Baker, B., Lee, T., Leike, J., Schulman, J., Sutskever, I., Cobbe, K.: Let's verify step by step. In: Int. Conf. Learn. Represent. (2024)
21. Lin, K.Q., Li, L., Gao, D., Yang, Z., Wu, S., Bai, Z., Lei, S.W., Wang, L., Shou, M.Z.: ShowUI: One vision-language-action model for GUI visual agent. In: IEEE Conf. Comput. Vis. Pattern Recog. (2025)
22. Liu, H., Li, C., Wu, Q., Lee, Y.J.: Visual instruction tuning. In: Adv. Neural Inform. Process. Syst. (2023)
23. Nong, S., Tang, X., Xu, J., Zhou, S., Chen, J., Jiang, T., Xu, W.: CRAFT-GUI: Curriculum-reinforced agent for GUI tasks. arXiv preprint arXiv:2508.11360 (2025)
24. van den Oord, A., Li, Y., Vinyals, O.: Representation learning with contrastive predictive coding. arXiv preprint arXiv:1807.03748 (2018)
25. OpenAI: GPT-4 technical report. arXiv preprint arXiv:2303.08774 (2023)
26. Qian, J.: Visual inception: Compromising long-term planning in agentic recommenders via multimodal memory poisoning. arXiv preprint arXiv:2604.16966 (2026)
27. Qian, J., Kang, Z.: Penny wise, pixel foolish: Bypassing price constraints in multimodal agents via visual adversarial perturbations. arXiv preprint arXiv:2604.16515 (2026)
28. Qwen Team: Qwen2.5-VL technical report. arXiv preprint arXiv:2502.13923 (2025)
29. Rawles, C., Clinckemaillie, S., Chang, Y., Waltz, J., Lau, G., Fair, M., Li, A., Bishop, W., Li, W., Campbell-Ajala, F., Toyama, D., Berry, R., Tyamagundlu, D., Lillicrap, T., Riva, O.: AndroidWorld: A dynamic benchmarking environment for autonomous agents. In: Int. Conf. Learn. Represent. (2025)

30. Rawles, C., Li, A., Rodriguez, D., Riva, O., Lillicrap, T.: AndroidInTheWild: A large-scale dataset for android device control. In: Adv. Neural Inform. Process. Syst. (2023)
31. Shi, W., Zhang, M., Zhang, R., Chen, S., Zhan, Z.: Change detection based on artificial intelligence: State-of-the-art and challenges. Remote Sensing **12**(10), 1688 (2020)
32. Tang, S., Chen, R., Lan, T.: Agent Alpha: Tree search unifying generation, exploration and evaluation for computer-use agents. arXiv preprint arXiv:2602.02995 (2026)
33. Wang, J., Xu, H., Ye, J., Yan, M., Shen, W., Zhang, J., Huang, F., Sang, J.: Mobile-Agent: Autonomous multi-modal mobile device agent with visual perception. arXiv preprint arXiv:2401.16158 (2024)
34. Wanyan, Y., Zhang, X., Xu, H., Liu, H., Wang, J., Ye, J., Kou, Y., Yan, M., Huang, F., Yang, X., Dong, W., Xu, C.: Look before you leap: A GUI-critic-R1 model for pre-operative error diagnosis in GUI automation. In: Adv. Neural Inform. Process. Syst. (2025)
35. Wu, Q., Gao, P., Liu, W., Luan, J.: BacktrackAgent: Enhancing GUI agent with error detection and backtracking mechanism. In: EMNLP (2025)
36. 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., Yu, T.: OSWorld: Benchmarking multimodal agents for open-ended tasks in real computer environments. In: Adv. Neural Inform. Process. Syst. (2024)
37. Xiong, T., Hu, X., Chen, Y., Liu, Y., Wu, C., Gao, P., Liu, W., Luan, J., Zhang, S.: GUI-PRA: Process reward agent for GUI tasks. arXiv preprint arXiv:2509.23263 (2025)
38. Yu, X., Peng, B., Galley, M., Cheng, H., Wu, Q., Kulkarni, J., Nath, S., Yu, Z., Gao, J.: Dyna-Mind: Learning to simulate from experience for better AI agents. arXiv preprint arXiv:2510.09577 (2025)
39. Yuan, X., Zhang, J., Li, K., Cai, Z., Yao, L., Chen, J., Wang, E., Hou, Q., Chen, J., Jiang, P.T., Li, B.: SE-GUI: Enhancing visual grounding for GUI agents via self-evolutionary reinforcement learning. In: Adv. Neural Inform. Process. Syst. (2025)
40. Zhai, X., Mustafa, B., Kolesnikov, A., Beyer, L.: Sigmoid loss for language image pre-training. In: Int. Conf. Comput. Vis. (2023)
41. Zhang, C., Yang, Z., Liu, J., Li, Y., Han, Y., Chen, X., Huang, Z., Fu, B., Yu, G.: AppAgent: Multimodal agents as smartphone users. In: Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems (2025). https://doi.org/10.1145/3706598.3713600
42. Zheng, C., Mo, X., Ma, X., Lin, Q., Zhao, Y., Zhu, J., Lou, X., Wang, J., Wang, Z., Liu, W., Zhang, Z., Yu, Y., Zhang, W.: Adaptive milestone reward for GUI agents. arXiv preprint arXiv:2602.11524 (2026)
43. Zhou, A., Yan, K., Shlapentokh-Rothman, M., Wang, H., Wang, Y.X.: Language agent tree search unifies reasoning, acting, and planning in language models. In: Int. Conf. Mach. Learn. (2024)
44. Zhou, S., Xu, F.F., Zhu, H., Zhou, X., Lo, R., Sridhar, A., Cheng, X., Ou, T., Bisk, Y., Fried, D., Alon, U., Neubig, G.: WebArena: A realistic web environment for building autonomous agents. In: Int. Conf. Learn. Represent. (2024)
45. Zhou, Y., Dai, S., Wang, S., Zhou, K., Jia, Q., Xu, J.: GUI-G1: Understanding R1-zero-like training for visual grounding in GUI agents. In: Adv. Neural Inform. Process. Syst. (2025)